Universidad de Valladolid

Escuela de Doctorado Universidad de Valladolid

PROGRAMA DE DOCTORADO EN INFORMÁTICA

TESIS DOCTORAL:

Improving efficiency of Trajectory-Based Operations for Air Traffic Management using Deep Learning

Presentada por Jorge Silvestre Vilches
para optar al grado de Doctor
por la Universidad de Valladolid

Dirigida por:

Miguel Ángel Martínez Prieto

Aníbal Bregón Bregón

Improving efficiency of Trajectory-Based Operations for Air Traffic Management using Deep Learning

Jorge Silvestre Vilches

July 21, 2025

Acknoledgements

This Ph.D. Thesis was funded through the 2019 Call for Predoctoral Contracts by the Universidad de Valladolid and Banco Santander, whose support is gratefully acknowledged. Partial support for this research was also provided by the Spanish Ministerio de Ciencia e Innovación under grants PID2020-114635RB-I00 and PID2021-126659OB-I00. The author extends sincere gratitude to Boeing Research and Technology Europe (BR&T-Europe) for granting access to their ADAPT platform, which significantly contributed to the development of this work.

Abstract

Trajectory-Based Operations (TBO) are the cornerstone of contemporary Air Traffic Management (ATM) systems, aiming to enhance operational efficiency, predictability, and safety through the strategic management of flight trajectories. TBO define flight trajectories as 4D-trajectories, where each point in the flight path is defined through its latitude, longitude, altitude and time. Hence, the use of airspace capacity can be optimized by strategically planning the flights not only from the perspective of the aircraft position, but also of the time. 4D trajectories can be enriched with additional flight information to provide a more holistic representation of flights. However, this information is scattered and harmonizing it in an integrated representation is yet to be achieved.

This Thesis addresses the challenge of leveraging heterogeneous data sources to generate high-quality 4D trajectories, and explores their potential to improve ATM operations through a data-driven approach based on deep learning. We first design a conceptual data model tailored to represent the most critical elements of ATM flight operations. This model encompasses multiple domains, including surveillance data, flight plans, meteorological information, and airport infrastructures, ensuring a unified representation of the foundations of ATM. A data integration architecture is developed to process data from heterogeneous sources and reconstruct enriched 4D trajectories conforming to the defined data model. The data quality is thoroughly evaluated to ensure that the data is complete, precise and reliable, with special attention to the position and time where it is missing or incorrect.

The Thesis further investigates the applicability of these enriched 4D trajectories in addressing critical operational challenges within ATM, and does so by proposing specific deep learning architectures that model 4D trajectories as time series. Two primary use cases are identified: the estimation of arrival times (ETA) for en-route flights and the prediction of flight trajectories, since both represent core elements of TBO and are vital for improving air traffic flow management, reducing delays, and optimizing resource allocation in congested airspace. We focus on flights arriving at the Madrid Barajas-Adolfo Suárez Airport (Spain) as our case study to exemplify and validate our proposal. The time of arrival at the destination airport for en-route flights can be reliably estimated based on the current state of the flight, together with other factors influencing over the flight, such as weather and schedule data. Our approach based on Long Short-Term Memory networks (LSTM) has improved the current state of the art, providing longer term and more accurate predictions, at any point in the trajectory, than existing methods with a mean error of 2.67 minutes versus the 3.67 minutes of the state of the art. A similar approach can be applied to predict the trajectory of a flight, as the future positions can be predicted based on the past positions of the flight. The state of the art methods provide results with a mean error of 0.005 and 0.01 degrees (0.55 and 1.11 kilometres) for latitude and longitude predictions, respectively, but does so by focusing on a single route (repetitive flights between two airports).

In contrast, our solution apply Temporal Fusion Transformers to achieve a similar precision at predicting the 2D position for all flights arriving at a specific airport, regardless of their origin.

The outcomes of this Thesis not only corroborate our research hypothesis and fulfils our objectives, but also contribute to advancing the state of the art by demonstrating how enriched 4D trajectories, derived from a comprehensive data integration pipeline, can serve as the foundation for innovative, data-driven solutions for critical Trajectory Based Operations in the increasingly complex context of the global Air Traffic Management.

Keywords: Air Traffic Management, Trajectory Based Operations, Deep Learning, Estimated Time of Arrival, Trajectory prediction, Long Short-Term Memory, Temporal Fusion Transformers, OpenSky, ADS-B

Contents

Part I

Introduction

Chapter 1

Introduction

Nowadays, air traffic management has become one of the most critical global systems. The relevance of air traffic operations in the economy, society and geopolitics is manifest. In 2023, the International Civil Aviation Organization (ICAO) numbered in 4,319 million of passengers on a total of 35.3 million flights in their latest annual report "The World of Air Transport" in 2023 [66].Aircraft have become one of the main pillars of modern transportation systems, primarily due to their capacity to cover long distances efficiently, both in terms of time and cost. While the air transportation was hit heavily by the COVID-19 pandemic, which reduced the global air traffic to less than a third with respect to year 2019, in 2024, just four years later, it is expected that the sector will finally overcome the situation and surpass the pre-pandemic numbers. Moreover, the inter annual statistics shows that the growth of the sector is accelerating vigorously.

In this context, the European airspace was one of the busiest in the world. In 2024, EUROCONTROL reported 10.7 million flights transporting 1,280 million passengers [34], that is, almost a third of the global air traffic with respect to the latest consolidated statistics by ICAO. This means that around 34,350 flights departed from European airports every day, and 3.5 million people embarked on those flights daily. The volume of operations is one of the main reasons behind the European airspace being one of the most complex in the world. In addition, the European airspace is highly fragmented, since the sovereignty over the airspace belongs to each state. The control over the air traffic on each of these airspaces resides in the national authorities, that are responsible of exerting such control, providing different air traffic services, and defining specific rules applicable to aircraft crossing its airspace. These activities are only a part of what is called Air Traffic Management (ATM). ATM comprises all the systems, procedures and services involved in ensuring the safe and efficient movement of aircraft in the airspace and at the airports. To this end, the ATM aims to maintain safety while optimizing the air traffic flows and minimizing delays. However, the fragmented nature of European airspace and the sheer volume of flights and passengers make this task increasingly challenging.

In 2004, the European Union started an initiative to solve this problem: the Single European Sky (SES)[1]. SES aimed to transform the European airspace to become more safe and promote the homogenization of the Air Traffic Management across Europe, with a particular focus on three pillars: safety, capacity and efficiency. In addition to the consolidation of standardized protocols and services, the materialization of these objectives had to be supported by technology. The

[1]Single European Sky: `https://transport.ec.europa.eu/transport-modes/air/single-european-sky_en`. Visited: 2025-07.

SESAR programme (Single European Sky ATM Research)[2] was established to promote the modernization and harmonization of the European ATM system through innovative solutions, such as information standardization, data sharing platforms, and advanced decision support tools. In this context, the concept of *Trajectory Based Operations* (TBO) [56] has become central for coordinating and structuring modern Air Traffic Management. TBO goes beyond decision making based on flight plans by introducing the concept of *4D trajectory*, which explicitly incorporates time into the aircraft's three-dimensional flight path (latitude, longitude, and altitude) [32]. Trajectories are negotiated and agreed upon by all involved stakeholders (airlines, air navigation service providers, airports, etc.) to enable a more dynamic and efficient allocation of airspace and airport resources. This detailed and data-rich description of each flight enhances advanced management tasks, such as airspace capacity management, improved tactical planning, and more robust safety assessments.

A key enabler for the success of these initiatives is data. As stated by ICAO in 2005, "the ATM community will depend extensively on the provision of timely, relevant, accurate, accredited and quality-assured information to collaborate and make informed decisions. Sharing information on a system-wide basis will allow the ATM community to conduct its business and operations in a safe and efficient manner" [64]. Over the last two decades, significant efforts have been made to improve the collection, monitoring and sharing of operational data. For instance, in 2017 the installation of Automatic Dependant Surveillance – Broadcast (ADS-B) systems became mandatory in all new aircraft, and in 2020 was mandatory for all aircraft flying in European airspace. ADS-B [90] is a surveillance technology that takes advantage of the aircraft's capabilities to determine its position as well as other important flight parameters (altitude, speed, bearing, etc.), and broadcast this information to enhance the situational awareness of any aircraft or control authority within the range of emission, and characterize in near real-time the state of the airspace during operations. Another example is the centralization of the flight plan management in the Initial Flight Plan Processing System (IFPS) and Enhanced Tactical Flow Management System (ETFMS). All flights must file in advance a flight plan describing the relevant information about the flight: the scheduled times, the operating airline, the characteristics of the aircraft, the planned trajectory, etcetera. Flight plans are filed up to one week before the flight, and the information is shared with all involved stakeholders through the aforementioned systems. Therefore, strategic decisions and resource allocation can be performed well in advance to enhance the efficiency and predictability of the air traffic operations, and the real-time sharing right before and during the flight allows to maximize the safety of these operations. Any other aspect influencing air traffic operations is being slowly integrated in the monitoring, information sharing and decision making processes. A prominent example of these influencing factors is weather conditions, which have a significant impact on the performance of flights. Many efforts have been devoted to ensure the detailed description of current weather both at the airports and throughout the routes, and the increasingly accuracy of forecast models contributes to enhance the predictability and the efficiency of flights.

The availability of large and detailed datasets describing air traffic operations has brought significant interest in data-driven methods. Traditionally, Air Traffic Management has relied on simulations, mathematical modeling, and expert-driven assessments to predict and manage aircraft trajectories. While these approaches have been effective, the explosive growth in traffic volume and the increasing complexity of airspace structures have challenged their scalability and

[2]Single European Sky ATM Research: `https://transport.ec.europa.eu/transport-modes/air/single-european-sky/sesar-project_en`. Visited: 2025-07.

accuracy. Many data-driven methods leverage historical data describing the past states of the aircraft and the context in which the flight takes place (weather conditions, airspace congestion, etc.), to accurately predict the future state of the airspace. By learning statistical patterns and dependencies from these datasets, data-driven models can produce accurate and dynamically updated trajectory predictions. The introduction of data-driven approaches aligns naturally with the goals of Trajectory Based Operations (TBO), as it supports more reliable planning and allocation of airspace and airport resources based on actual operational data.

Despite all these efforts, the envisioned modern air traffic management still faces important challenges to be fully materialized. Although large amounts of data are continuously generated and shared among all involved stakeholders in the industry, there is no centralized source of data that aggregates and harmonizes this information at the scale and granularity required for research and innovation. As a result, most of the available data is scattered across isolated data silos, which are difficult to integrate due to differences in format, coverage, granularity and quality standards. In addition, access to these datasets is generally limited for independent research, either because of prohibitive fees or strict confidentiality and security constraints imposed by providers. Even when the data is accessible, it often comes at a large scale and is affected by missing data, noise and errors, which can compromise the quality and reliability of downstream systems and applications that depend on it. These challenges limit the potential of data-driven methods to support key ATM tasks, such as trajectory prediction and arrival time estimation, and highlight the need for robust techniques capable of dealing with incomplete and heterogeneous data, while still delivering accurate results.

This Thesis contributes to address these challenges by focusing on two main objectives. First, we design and implement a data integration pipeline capable of automatically collecting, harmonizing, and ensuring the quality of heterogeneous data sources relevant to air traffic management. We begin by defining a conceptual model that captures the data required to meet the needs of data-driven solutions to ATM challenges. Then, it is necessary to identify the appropriate data sources and assess their suitability to feed the pipeline. To achieve this, we characterize the available data and evaluate its quality in order to propose corrective measures and establish continuous quality monitoring throughout the pipeline. In particular, we focus on integrating surveillance data, flight plans, and weather information.

Second, we leverage the resulting integrated data to apply data-driven methods that support critical ATM operations, such as arrival time estimation and trajectory prediction, through the application of machine learning and deep learning techniques. To this end, we propose novel solutions based on deep learning methods for two use cases, and exemplify their performance on real-world data from inbound flights at the Madrid Barajas-Adolfo Suárez airport (Spain). The first use case addresses the estimation of the time of arrival (ETA) for en-route flights, which, which is one of the most critical tasks in air traffic management, given the high density of aircraft and the dynamic conditions around airports. The precise prediction of arrival times is crucial for optimizing airspace organization and resource allocation both at the airport and along the flight path, and ensuring the safety of landing operations thanks to the enhanced predictability of the flight. The second use case involves the prediction of an aircraft's trajectory during flight. Accurate trajectory predictions can help air traffic controllers to manage concurrent flights within crowded airspace, particularly near airports, by determining their most probable flight path in the short-term future. Both use cases leverage the concept of 4D trajectories, and rely on data-driven methods specialized in sequential data processing. By integrating heterogeneous sources of surveillance data, flight plans and weather information into a unified data model, this Thesis aims to demonstrate how deep learning approaches can effectively support critical ATM tasks, ultimately contributing to a safer and more efficient air traffic management system.

1.1 Objectives

In the previous section, we have discussed the value of an information system that supplies with data to support air traffic operations. These arguments lead us into the following research hypothesis (RH) that sets the goals we aim to achieve during this Thesis:

RH-1. The integration of data from heterogeneous sources related to Air Traffic Management through a data transformation workflow to produce enriched 4D trajectory data according to an unified data model, can improve the operational efficiency of Air Traffic Management operations by leveraging data driven approaches based on machine learning.

In order to facilitate the organization, planning, and progress assessment of this Thesis, we have identified the following research objectives (RO) derived from the defined research hypothesis:

RO-01. Design a conceptual model that describes the main concepts involved in ATM operations (surveillance data, flight plans, weather and infrastructure, among others) and relevant for the scope of this Thesis, and whose information can be gathered by automatic means and extracted from heterogeneous data sources, preferably non-commercial.

RO-02. Develop a data integration architecture to construct 4D trajectories based on the available data sources, that satisfies the data model defined in RO-01.

RO-03. Identify and explore several use cases for the data sets resulting from the RO-02, whose efficiency can be improved by means of deep learning approaches that leverage the 4D trajectory representation produced by our data integration architecture.

- **RO-03.1.** Estimated time of arrival (ETA) at the destination airport for en-route flights.
- **RO-03.2.** Trajectory prediction for en-route flights.

Fulfilling these objectives should allow us to validate the research hypothesis stated above. In particular, the expected result of the objectives RO-01 and RO-02 is an information system that covers ATM operations focused on commercial flights. This has several implications with respect to the scope of the defined conceptual model, and the selection of data sources that will eventually feed this information system. We have made this decision to narrow down the potential scope of the Thesis, given the complexity and diversity of ATM operations.

A set of deliverables have been proposed as a result for each of the defined objectives, which can be divided in two groups:

- On the one hand, we will communicate our research results to adequate forums in order to validate them. The chosen forums consist on specialist conferences, given the potential interest of our investigations for such venues, and research conferences or journals dedicated to Data Science applications.
- On the other hand, all the developed software materials will be made public in a Github repository: `https://github.com/JorgeSilvestre/silvestre-thesis`.

1.2 Methodology

The development of this Thesis follows a data-driven approach to tackle specific challenges in Air Traffic Management. As such, it aligns with the core principles of Data Science, which include the acquisition, transformation, analysis, and the application of data to support predictive tasks. Given the exploratory and iterative nature of data science projects, especially in complex domains such as Air Traffic Management, it is crucial to adopt a methodological framework that provides both structure and flexibility. On the one hand, the methodology must provide a systematic way to plan, execute, and document the different phases of the research. On the other hand, it must be capable of incorporating ongoing learning, feedback, and revisions that occur as data is explored and models are refined.

For this purpose, this work adopts the CRISP-DM methodology (Cross-Industry Standard Process for Data Mining) [119], which has proven to be particularly suitable for applied research projects involving real-world data. Its phased structure supports clear project definition and reproducibility, and its iterative nature aligns well with the trial-and-error process inherent to model development and evaluation.

We considered other methodologies for data-intensive projects were considered, such as KDD (Knowledge Discovery in Databases) or SEMMA (Sample, Explore, Modify, Model, Assess). To assess their suitability for our project, we reviewed different comparisons between these methodologies [10, 81, 97], and concluded that CRISP-DM was the best fit for our purposes for several reasons:

- CRISP-DM provides a well-defined sequence of phases that support the project from its formulation to the validation of results, which ensures the traceability of the progresses.

- The cyclical nature of CRISP-DM allows for frequent iteration between phases, so the discovery of new patterns or problems in the data can feed back into earlier stages.

- CRISP-DM is a widely adopted methodology in data science projects in industry and academia [119]. It has been successfully applied in research projects involving predictive models in domains similar to ATM [77, 72]. Moreover, it has been generalized to enable its use in a more general engineering environment [54].

In contrast, the KDD process lacks a standardized operational structure and focuses mainly on knowledge discovery, with less emphasis on business understanding and deployment. SEMMA, on the other hand, is more focused on the modeling stage and assumes that data preprocessing has already been completed, making it less suitable for projects where raw and heterogeneous data sources must be integrated and transformed from scratch.

The CRISP-DM methodology is structured into six iterative phases. In the following, we discuss how each of these phases is implemented in this Thesis.

a. Business understanding. In this phase, we define the strategic goals of the research and identify how data science techniques can contribute to improving specific operational aspects of Air Traffic Management (ATM). To this end, we carry out a review of the relevant literature (both the research on this area and technical documents published by ATM organizations) to understand the ATM operational context, the existing challenges and the proposed solutions.

This phase contributes to RO-01 by defining the core concepts and operational aspects that will be modeled later, and to RO-03 by identifying the relevant use cases where deep learn approaches can be applied.

b. Data understanding. This phase involves the exploration and assessment of the available data sources. We evaluate their reliability, completeness, and limitations, and we identify relevant attributes that could be integrated to satisfy the domain conceptual model implemented in RO-01. In particular, we must identify suitable flight plans, surveillance and weather data sources, and assess their contribution to fulfill the information requirements of the Thesis.

This phase contributes to RO-01 by learning how the defined concepts map to real world data, and to RO-02 by determining how to extract and integrate the data in the identified heterogeneous data sources.

c. Data preparation. This phase comprises the design and implementation of the data integration architecture. It includes data cleaning, normalization, temporal and spatial alignment of heterogeneous sources, and the construction of the 4D trajectory representations. This stage also covers the adaptation of the data to the requirements of the models, including operations such as windowing, normalization and feature extraction.

This phase addresses RO-02 by defining how the data must be integrated and prepared, and RO-03 by enabling the data to be used in the training and evaluation of deep learning models.

d. Modeling. In this phase, predictive models are developed using deep learning techniques adapted to time series data. We focus on the two use cases described in previous stages: estimated time of arrival prediction (sequence-to-value problem), and trajectory prediction (sequence-to-sequence problem) tasks, exploring different architectures to determine the most suitable methods for these tasks.

This phase contributes to objectives RO-03.1 and RO-03.2. (and therefore to RO-03) by designing and training specialized models using the data generated in previous stages of the process and conducting an optimization process of the hyperparameters of the models.

e. Evaluation. We assess the models using quantitative metrics relevant to each use case, both standard (MAE, RMSE) and ad-hoc metrics. The results are analyzed to identify the method with the best performance for each of the tackled challenges. This phase contributes to RO-03 by validating the effectiveness of the developed models.

f. Deployment. Although full operational deployment is outside the scope of this Thesis, a research-level deployment is conducted through the development of software artifacts and experimentation pipelines. These can be reused for future studies or to reproduce the results of our research.

These six stages are carried out iteratively to progressively refine the accomplished research results. Therefore, CRISP-DM allows to incorporate new insights as they emerge during the research lifecycle, and it enables retroactive refinement: as we gain a deeper knowledge about the data, the methods or the domain, it often becomes necessary that upstream phases are adjusted. To this end, the use of versioned data processing scripts, model tracking tools, and experimental notebooks will be essential to manage this iterative development effectively and reproducibly.

In summary, the application of CRISP-DM in this Thesis has not been strictly sequential but has evolved through incremental cycles of development, evaluation, and adjustment. This adaptive structure has enabled the project to respond to complexity while maintaining coherence across its different stages, ultimately supporting a robust and extensible research process.

However, CRISP-DM does not explicitly define when a task or objective is considered complete. In this Thesis, completion criteria are defined by combining a scientific validation approach, based on the testing of our research hypothesis RH-01 and empirical evaluation, with an engineering perspective, focused on iterative refinement until satisfactory performance is achieved.

- The conceptual and architectural objectives (RO-01 and RO-02) are completed when the designed models and systems meet their functional requirements and demonstrate usability across different data sources.

- The use cases in RO-03.1 and RO-03.2 are evaluated through quantitative performance metrics and comparisons with existing approaches, to determine whether they improve the state of the art.

This combined framework allows for iteration between phases when needed, and ensures that each result contributes to validating the main hypothesis.

1.3 Thesis Planning

The Thesis was planned for completion within a three-year period, starting from enrolment in the doctoral programme during the 2019–2020 academic year. Accordingly, the expected completion date for the Thesis was set for February 2023. However, it became necessary to extend the original timeline by one and a half years due to delays caused by several problems in the access and processing of the data. In particular, we lost access to the data sources that were used at the moment, which forced us to find alternative data sources to replace the initial sources, and repeat the study of the characteristics and quality of the new data. The Thesis planning, updated at the end of the last year of the Thesis, is shown in Figure 1.1.

The objectives are interdependent, requiring at least initial results from one before progressing to the next. For instance, we can not design a data integration process without first defining the domain model, or characterizing the data sources. Progress will therefore be cascaded to some extent. However, we have planned the work using an iterative and incremental approach: in each iteration, new partial results will be incorporated into the consolidated results; in this way, progresses on all objectives will feed back to the others and provide a more solid foundation to future advances.

Similarly, we have defined a communication plan to validate our results in appropriate forums, such as specialist symposia or scientific conferences. This will help us enable us to identify and correct any misunderstandings or errors in our approach at an early stage. Later on, we intend to make contributions to high impact journals as we reach more mature results. Simultaneously, the accumulated knowledge during the Thesis will be consolidated into the corresponding dissertation, which will be developed throughout the Thesis and finalized at the end of the doctoral program with due conclusions.

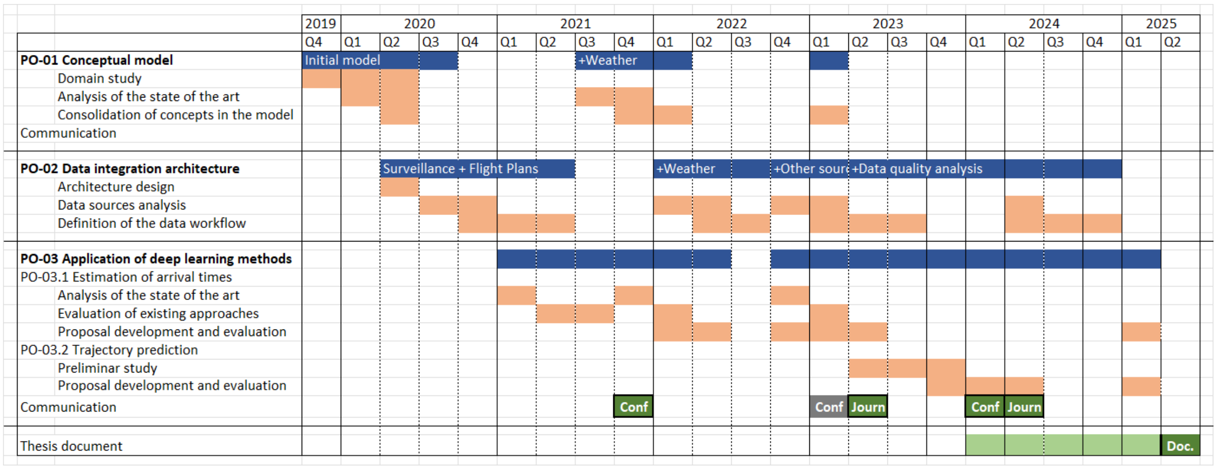

Figure 1.1: Gantt diagram of the Thesis.

In the following, we describe the planned tasks to achieve the defined objectives. To address the OP-01, we will first conduct a comprehensive study of the aviation and Air Traffic Management domains, since there is a substantial amount of specialized knowledge we need to be familiar with before modelling some of their key concepts. A thorough review of the state of the art will be of great help to guide our understanding of this field. Once we have gained sufficient confidence in this domain, we will propose a first conceptual model focusing on surveillance data and flight plans. This first iteration will later be expanded to include weather information as the research progresses.

After defining the data model, we will proceed with the data extraction and processing tasks necessary to fulfil RO-02. This will involve identifying and characterizing the available data sources and assessing their suitability for integration and their usage in this Thesis, with a special attention to the quality of the data. We will design a data workflow to integrate these heterogeneous data sources into our envisioned enriched 4D trajectories. This workflow will provide us with reference datasets to be leveraged in the use cases studied under the next objective.

The tasks for RO-03.1 and RO-03.2 will follow a similar methodology. First, we will conduct a review of the state of the art to identify and learn about the most relevant methods for the defined tasks. Next, we will characterize existing solutions based either on the reviewed literature, or by developing similar methods to serve as benchmarks for our own proposals. These proposals will be built upon the datasets defined in the earlier stages of the Thesis. Finally, we will evaluate our solutions to assess their performance and value compared to the state of the art.

1.4 Contributions

The main results of the research have been published throughout the realization of the Thesis to validate their quality and scientific interest. To this end, we submitted our work to specialized conferences, where it could be reviewed by experts in both the domain and the techniques implied in our research. Given the applied nature of our research, we focused on the results derived from the RO-03, where several use cases are explored to assess the usefulness of the constructed system information. However, these results build upon the results derived from the other two objectives, since these results are described thoroughly in each of the published papers. In the following, we describe the different publications, their alignment with the obtained results, and how they contribute to the satisfaction of the objectives of the Thesis.

According to the Thesis plan described in Section 1.3, we started working towards the RO-03.1, that is, the estimation of the time of arrival. We submitted our initial results to the OpenSky Symposium in 2021 in the paper :

> Silvestre, J., Santiago, M., Bregon, A., Martínez-Prieto, M.Á., Álvarez-Esteban, P.C. "On the Use of Deep Neural Networks to Improve Flights Estimated Time of Arrival Predictions". In: Engineering Proceedings 13.1 (2021), p. 3. issn: 2673-4591. [101]

This paper described an initial version of the data processing workflow that integrated surveillance and flight plan data to construct 4D trajectories with the aircraft reported positions. Using these data, we trained several models based on LSTM (Long-Short Term Memory) networks [53], an advanced type of recurrent neural network that can process long sequences of data, to predict the estimated time of arrival of an aircraft at any point of the flight. These neural networks are explained in more detail in Chapter 3. These models used sequences of surveillance datapoints to characterize the state of the flight at that moment, and estimate when the aircraft will arrive at the destination airport based on historical data. We evaluated our approach on a sample of flights arriving at the Madrid Barajas-Adolfo Suárez airport (LEMD) in Spain during three weeks. The presentation of this work allowed us to receive valuable feedback from domain experts, and the analysis of the obtained results helped us to accomplish a deeper understanding of the data and the prediction task.

This line of work culminated in 2024 with the publication of a paper in the Journal of Supercomputing (Springer).

> Silvestre, J., Martínez-Prieto, M.Á., Bregon, A., Álvarez-Esteban, P.C. "A Deep Learning-Based Approach for Predicting in-Flight Estimated Time of Arrival". In: The Journal of Supercomputing 80.12 (Aug. 2024), pp. 17212–17246. issn: 1573-0484 [102]

In this paper, we constructed a more ambitious proposal, with important novelties in several fronts. First, we developed a larger battery of models to fully assess the performance of several approaches on predicting flights ETA. In particular, we evaluated several architectures based on LSTM, and several machine learning models, such as Random Forests [15], AdaBoost [43] and Gradient Boosting Machines [44], which constituted the state of the art for this task. Second, we incorporated the analysis of the weather conditions at the airport. This implied that new data sources had to be included into the data workflow, and then integrated with the rest of the data. Third, we could delve into the data cleaning process thanks to our deeper knowledge of the data, and propose solutions to data problems that were not tackled before. Among them, the main problems were the delayed vectors and the errors in the reported positions. This fact led us to develop a method to reorder the positions in a trajectory to overcome the assignment of incorrect time stamps to the reported positions, as well as an outlier detection mechanism. We also designed a battery of data quality metrics to monitor the process, albeit they were not included in this paper. As a result, we managed to obtain high quality 4D trajectories, enriched with flight plan and weather data that could be used as features to improve the performance of the models. Fourth, we conducted a thorough experimentation based on a larger sample of data than before, which grew from three weeks to 9 months of data. The results accomplished by our proposal based on LSTM networks outperformed all comparable approaches in the state of the art, as explained in Chapter 7.

We leveraged the acquired knowledge about the data and the used deep learning architectures to tackle a different problem, the prediction of future positions in a trajectory. This problem poses more complexity than ETA prediction, since the goal is predicting entire sequences of data instead of estimating a scalar value. We started by exploring the task using architectures based on LSTM that were similar to those applied for ETA prediction. We presented our initial results in the 39th ACM/SIGAPP Symposium on Applied Computing, with the short paper:

> Silvestre, J., Mielgo, P., Bregon, A., Martínez-Prieto, M.Á., Álvarez-Esteban, P.C. “Towards Aircraft Trajectory Prediction Using LSTM Networks”. In: Procs. 39th ACM/SIGAPP Symposium on Applied Computing. SAC ’24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1059–1060. [104]

In this paper, we trained several models with a LSTM-based architecture to predict the position of a flying aircraft two and a half minutes in the future, based on its flown trajectory so far. However, the results did not improve the state of the art, and we decided to explore other architectures that could improve these preliminary results. As a result, we published the following paper in the IEEE Access journal (IEEE) in 2024:

> Silvestre, J., Mielgo, P., Bregon, A., Martínez-Prieto, M.Á., Álvarez-Esteban, P.C. “Multi-Route Aircraft Trajectory Prediction Using Temporal Fusion Transformers”. In: IEEE Access (2024), pp. 1–1. issn: 2169-3536. [103]

In this paper, we investigated the application of the TFT architecture to the trajectory prediction task. Temporal Fusion Transformers (TFT) [74] is an architecture based on the Transformer architecture [114]. Unlike LSTM networks, which use recurrence to construct very deep networks to be capable of processing sequences of data, the Transformer architecture implements attention, a much more lightweight mechanism to extract the most important information from a sequence of data. TFT conciliates these two approaches, in combination with a special treatment of the different inputs, was an attractive choice to process 4D trajectories. For this paper, we designed an exhaustive experimentation on flights arriving at LEMD during nine months, similar to the dataset used in [102], albeit in this case it was more challenging due to the higher complexity of the resulting models. The accomplished results were satisfactory and on-par with the state of the art for trajectory prediction, since the trained models were capable of making accurate predictions for a wide variety of routes, in contrast with the state of the art, which only tackled single routes (that is, flights between a pair of airports).

During the work in these papers, the data workflow was revised and improved by introducing the abstraction of the trajectory. Until the definition of this abstraction, there was not an integrated representation of trajectories: the data was disperse among several data collections, and had to be integrated on the fly to work with individual trajectories. We implemented the notion of trajectory to represent the richness of the 4D trajectory concept, which comprises bot the sequence of positions with its associated timestamp, and a set of metadata that are key for fully understand the constructed trajectories. In this way, the parts of the data workflow related to trajectories were rebuilt upon this concept for bringing a more semantic way of working with the produced data. In addition, we defined a new set of specific metrics to assess the quality of the trajectories, beyond the data quality metrics defined before.

In conclusion, these contributions helped us to validate our work and our results in two ways: First, the OP-03 was completed thanks to the satisfaction of its two sub-objectives: two proposals were constructed that accomplished state of the art results for ETA prediction and trajectory prediction. To this end, novel architectures were applied to each of these problems. Second, these approaches combined features from different data sources to make accurate predictions. The developed information system, which was designed based on the constructed conceptual model as a result of the RO-01, was capable of satisfying the data requirements of these approaches, validating the fact that this information system comprises the main concepts involved in ATM operations, and effectively provides such data in an appropriate and timely form, so it can be consumed by deep learning models like the ones developed in this Thesis, as stated by the RO-02.

1.5 Document structure

The document is organized as follows. Chapter 2 presents a comprehensive review of key concepts in Air Traffic Management to facilitate the understanding of the subsequent chapters. In Chapter 3 introduces the relevant techniques for this work, with a particular focus on the deep learning architectures employed in later chapters. Chapter 4 introduces the conceptual model defined to structure data involved in flight operations and discusses the available data sources to meet the data requirements of the model. Chapter 5 presents the data transformation workflow, which includes extracting the data from the original sources, addressing known data quality issues, and integrating the information into a 4D trajectory representation. These trajectories are analyzed in Chapter 6, where different approaches are proposed to enhance data quality. Chapters 7 and 8 explore the use of the generated data to develop data-driven methods based on deep learning to support flight operations and improve the efficiency of ATM. Specifically, Chapter 7 focuses on the estimation of arrival times, while 8 addresses the prediction of flight trajectories. Finally, 9 summarizes the main conclusions of the work.

Chapter 2

Air Traffic Management

Air traffic management is a complex domain where numerous agents, organizations, standards and events interact under strict rules and protocols. While the management of air traffic within a given airspace is responsibility of the owner states, recent decades have seen significant efforts to harmonize and standardize practices across air spaces worldwide. These efforts have been led by International Civil Aviation Organization (ICAO), a global organization that works under the umbrella of the United Nations, alongside other transnational organizations such as EUROCONTROL in Europe and the Federal Aviation Administration (FAA) in the United States. The outcomes of this standardization process are regularly published and updated in the ICAO Documents, which define key concepts, rules, and procedures for air traffic operations. Although the adoption of these regulations by individual states is not mandatory, they are widely accepted and typically implemented within national and regional air traffic management frameworks.

More formally, Air Traffic Management (ATM) is defined by ICAO as "the dynamic, integrated management of air traffic and airspace including air traffic services, airspace management and air traffic flow management (safely, economically and efficiently) through the provision of facilities and seamless services in collaboration with all parties and involving airborne and ground-based functions" [62]. This definition is derived from the operational concept of ATM established in the Document 9854 [64], which defines a vision for the future ATM based on five guiding principles:

- Safety: Achieving safe operations is the main priority of modern ATM, and any initiative in ATM is subordinated to ensure safety in all contexts.
- Humans: All operations ultimately rely on humans to be successful, since humans are responsible of managing, monitoring and take control, so human factors must play a key role in the organization of air traffic operations.
- Technology: The management of air traffic relies on information provided by automatic systems like surveillance, navigation and communication systems. It is required that these systems work in an integrated, interoperable and robust way to ensure the homogeneity of ATM across different regions and traffic flows.
- Collaboration: The homogenization of technologies, information and processes would enable strategic and tactical collaboration between agents and organizations involved in ATM, to ensure optimal conditions in terms of safety and efficiency.

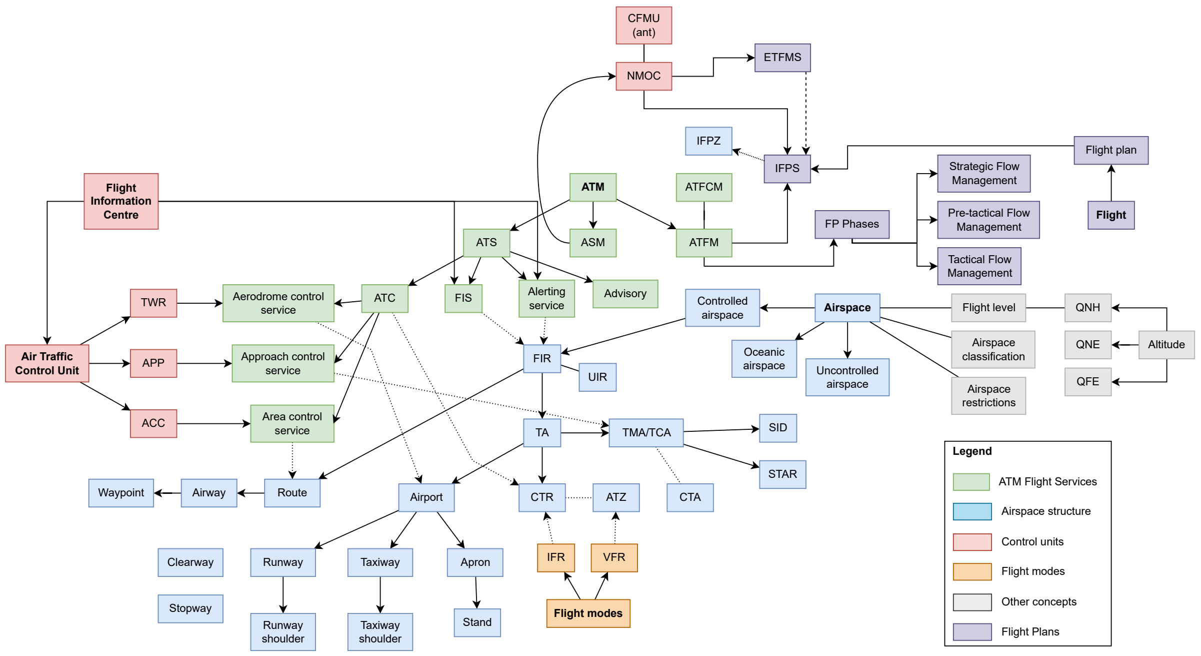

Figure 2.1: Concept map that brings together all the concepts covered in this section.

- Continuity: The requirements of safety and efficiency enforce the maximum continuity of ATM services, and account for measures to operate even in adverse conditions (natural disasters, security threats, etc).

These guiding principles are materialized through different key components, which are thoroughly elaborated in the different ICAO documents. Seven interdependent key components are defined [64]:

1. Airspace organization and management: Stablish airspace structures to ensure that the different types of air traffic, volume of traffic and levels of service are satisfied adequately.

2. Aerodrome operations: Manage ground infrastructures (runways, taxiways, ground services, lighting) and provide effective surface guidance to maximize efficiency and safety in all weather conditions at the aerodrome.

3. Demand and capacity balancing: Strategically evaluate system-wide traffic flows and capacities (specially at the aerodromes) to reduce the risk of conflicts and increase the efficiency of the air traffic.

4. Traffic synchronization: Tactically analyze air traffic to maintain a safe, orderly and efficient traffic flow, based on the concept of 4D trajectory operations

5. Conflict management: In terms of the previous two aspects, and separation provision (between two aircraft, and between aircraft and other obstacles) and collision avoidance.

6. Airspace user operations: Optimize 4D trajectory planning based on aircraft performance, flight conditions and other ATM resources, enabling tactical and strategic situational awareness and conflict management.

7. ATM service delivery management: Ensure a continuous and seamless delivery of ATM services during the entire flight and across all agents involved.

Delving into all these aspects is beyond the scope of the Thesis. Instead, in this chapter, our goal is providing a succinct description of some aspects of ATM that are relevant to our research, since a basic understanding is required to address our research objectives, and for the reader to comprehend the results explained in later chapters. Figure 2.1 provides an overview of the concepts covered in the following sections, divided in several groups and color-coded accordingly (see the legend in the diagram).

2.1 Services of Air Traffic Management

As defined by ICAO [62], air traffic management is "the dynamic, integrated management of air traffic and airspace including air traffic services, airspace management and air traffic flow management". These three aspects correspond to the three components into which ATM is divided: Air Traffic Services (ATS), Air Traffic Flow Management (ATFM) and Air Space Management (ASM) [62]. Each of these components has a specific role and scope of application. In this section, we provide an overview of the responsibilities and functions of each of these services.

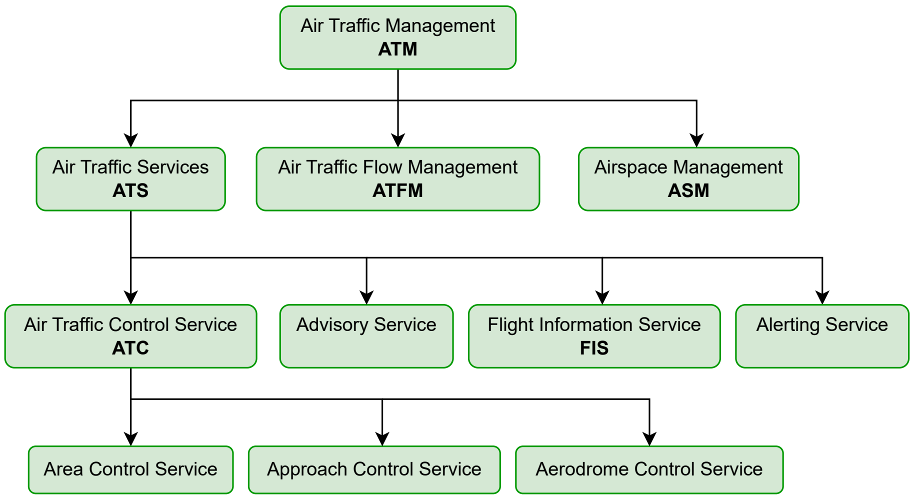

Figure 2.2: Hierarchy of ATM services [62].

2.1.1 Air Traffic Services (ATS)

Air Traffic Services (ATS) comprise a variety of services to support air traffic operations and management, including the Flight Information Service (FIS), the Air Traffic Advisory Service, the Air Traffic Control (ATC) and the Alerting Service. ATC is further divided into three specific services, depending on the area in which this service is to be provided: the aerodrome, the area near the aerodrome or the en-route airspace. In the following sections, a succinct description of these services is provided.

The Air Traffic Services as a whole pursue five objectives [57]:

1. Prevent collisions between aircraft.

2. Prevent collisions between an aircraft in the Terminal Manoeuvring Area (TMA) and obstructions in that area.

3. Expedite and maintain an orderly flow of air traffic.

4. Provide advice and information useful for the safe and efficient conduct of flights.

5. Notify appropriate organizations regarding aircraft in need of search and rescue aid.

However, these objectives are not addressed equally by the different ATS services. Each service has the responsibility over specific objectives, and is provided by specific Control Units in different areas of the airspace [57]. In particular, the Air Traffic Control (ATC) service pursues objectives 1, 2 and 3, while the objectives 4 and 5 are responsibility of the Flight Information Service (FIS) and the Alerting Service, respectively. Finally, the Advisory Service aims to ensure separation between aircraft (objective 1) when none of the remainder services are available.

With respect to the units providing these services, flight information centres (FIC) provide FIS and alerting service within FIRs (that is, within controlled airspace), unless the responsibility over the flight is assigned to an air traffic control unit with those capabilities. Air traffic control units provide air traffic control, flight information and alerting services in specific parts of the airspace, depending on their type. The Table 2.1 shows a summary of the correspondence between services, objectives and control units.

	Objective					Control Unit			
Service	1	2	3	4	5	FIC	ACC	APP	TWR
FIS				x		x	x	x	x
ATC/ACS	x		x				x	x	x
ATC/ApCS	x		x					x	x
ATC/AeCS	x	x	x						x
Alerting					x	x	x	x	x
Advisory	x				x				

Table 2.1: Correspondence between the air traffic services with their assigned ATS objectives and the units responsible of providing those services.

Air Traffic Control (ATC)

The main purposes of Air Traffic Control (ATC) are avoiding collisions and ensuring an orderly flow of the air traffic [57]. The application of ATC depends on the class of the airspace where the service is available (airspace classes are defined in Section 2.4), and the applicable flight rules (see 2.5.1): ATC is applicable to flights under instrumental rules in airspace classes A, B, C, D and E; to flights under visual rules in airspace classes B, C and D; and to all air traffic at controlled aerodromes. Each ATC sub-service focuses on the realization of these objectives in different airspaces:

- Area control service addresses objectives 1 and 3 by providing air traffic control service for controlled flights in the parts of the flight that are not associated with arrival or departure.

This service can be provided by either an Area Control Centre (ACC), or the unit providing approach control service in a control zone or in a control area (APP).

- Approach control service also addresses objectives 1 and 3 by providing the air traffic control service for the parts of controlled flights that are associated with arrival or departure. Approach control service is provided by the aerodrome control tower (TWR) or an area control centre.

- Aerodrome control service addresses the three objectives of ATC by providing air traffic control service for all traffic near to the aerodrome, and is provided by the Tower of the aerodrome (TWR).

Any of the control units above are responsible of providing ATS for flights in their scope of responsibility. Whenever none of these units are available, ATS services are to be provided by the corresponding FIS unit.

Regardless of their control scope, all ATC units have the following responsibilities [57]:

- Be provided with information of the intended movement of each aircraft and any variation on that intent, and with information on the actual progress of each aircraft.

- Determine the relative positions of known aircraft to each other using the received information.

- Issue clearances and information for the purpose of preventing collision between aircraft, expediting and maintaining an orderly flow of traffic.

- Coordinate clearances as necessary with other units: whenever an aircraft might otherwise conflict with traffic operated under the control of such other units, or before transferring control of an aircraft to such other units.

Flight Information Services (FIS)

According to ICAO Document 4444 [57], the Flight Information Service (FIS) is "a service provided for the purpose of giving advice and information useful for the safe and efficient conduct of flights". FIS is provided by flight information centres (FIC) within its scope of responsibility. However, this function may be assumed by the control units that provide ATC, if applicable, which effectively replace the information centres mentioned above. That is, ATC units have precedence over information centres, whenever available. In any case, FIS must be provided to all aircraft to which that information is relevant within the scope of the information service.

Flight Information Services shall provide the following information to all agents concerned by air traffic:

- Weather information published as Significant Meteorological Information (SIGMET) and Airmen's Meteorological Information (AIRMET).

- Volcanic activity and emissions, and radioactive materials or toxic chemicals into the atmosphere.

- Changes in the availability of radio navigation services.

- Changes in condition of aerodromes and associated facilities.
- The presence of unmanned free balloons.
- Any other aspect likely to affect safety.

The flights must also be provided with:

- Weather conditions reported or forecast at departure, destination and alternate aerodromes.
- Collision hazards within airspace classes C, D, E, F, and G.
- For flights over water areas, any available information such as radio callsign, position, true track, speed, etc. of surface vessels in the area.

Alerting service and advisory service

The Alerting Service shall be provided by flight information centres or area control centres for all aircraft to which ATC service is available, and, in so far as practicable, to all other aircraft having filed a filed plan or otherwise known to the ATS. It collects all information relevant to a state of emergency of an aircraft operating within the FIR or control area concerned and forwards such information to the appropriate rescue coordination centre.

The advisory service is provided to ensure separation between aircraft which are operating on IFR flight plans, and is responsibility of the flight information centre in the area.

2.1.2 Air Traffic Flow Management (ATFM)

ICAO defines Air Traffic Flow Management (ATFM) as “the service whose aim is contributing to a safe, orderly and expeditious flow of air traffic and optimizing the use of ATC, so the air traffic demand does not exceed the airspace capacity and the safety is not compromised” [61]. To this end, ATFM exerts flow control on current and planned flights, that is, adopts measures to adjust the flow of traffic into a given airspace, along a given route, or bound for a given aerodrome, so as to ensure the effective utilization of the airspace. In Europe, the Network Manager Operations Centre (NMOC) is responsible of providing ATFM under all of these circumnstances.

ATFM is carried out in three phases [62, 22]:

1. Strategic planning, if the action is made more than one day before the day on which it will take effect. Strategic planning is normally carried out well in advance, typically two to six months ahead, although it can occur until seven days before the day of operation. In this phase, the demand is adjusted to match the expected capacity.
2. Pre-tactical planning, if the action is to be taken between one and seven days before the day on which it will take effect. The strategic planning is refined according to the short-term estimations of capacity and usage. This is the phase in which flight plans are submitted (and updated if necessary) to the Initial Flight Plan Processing System (IFPS), which is the information system where flight plans are communicated by the airlines up to seven days before the day of the flight.
3. Tactical operations, if the action is taken on the day on which it will take effect. The necessary resources are allocated to service the flights of that day according to the actual capacity and air traffic during that day.

2.1.3 Air Space Management (ASM)

Air Space Management (ASM) has the responsibility of planning and publishing the management of airspace, attending to its division in air routes, civil and military control routes and airports' control areas [22]. ASM aims to achieve the optimal configuration of the air space that ensures the satisfaction of the users' needs and the safety of the operations.

2.2 Airspace structure

In the previous section we explained that different services and units operate in different parts of the airspace such as near to the aerodrome or during the flight. In this section, the definition and classification of such areas is formalized to provide a deeper understanding of the organization of the air traffic services.

According to ICAO [62], an airspace is "a portion of the atmosphere controlled by a country above its territory (land or water)". An airspace can be further divided into smaller airspaces depending on the regulations and services that are to be provided in the sub-areas within that airspace. Therefore, we can refine the definition of airspace as the space of the air delimited by three dimensions: height, width and depth (and which may not start from the ground), and characterized by specific flight regulations and provision of services.

We can establish a first coarse classification of airspaces into two types [62], depending on the services that are available:

- Controlled airspace has defined dimensions within which the ATC service is provided to IFR flights in general, and to VFR flights in accordance with the airspace classification. In particular, controlled airspace is classified into A, B, C, D or E class (see Section 2.4).

- Uncontrolled airspace is where air traffic control does not constitute an executive authority, although it may act in an advisory manner. Uncontrolled airspace corresponds with the classes F and G.

2.2.1 Structure of controlled airspace

Controlled airspace is commonly structured into three hierarchical levels. However, this hierarchy is not necessarily exhaustive: lower-level airspaces are typically nested within higher-level divisions, but not all upper-level airspaces are subdivided. Some portions of the airspace may exist exclusively at a higher level without being assigned to any lower-tier class. These three levels are Flight Information Region (FIR), control areas and control zones, and controlled aerodromes. Note that these are aligned with the control units explained above.

There exists additional divisions of the airspace that do not relate as closely to the services that are provided on them, but are defined to give better support to flight operations. The services available in each subdivision is described in Table 2.2. All of these concepts are published through AIPs (Aeronautical Information Publication) by the authorities managing each airspace. In the following sections we succinctly describe all these airspace divisions.

2.2.1.1 FIR and UIR

A Flight Information Region (FIR) is a portion of the airspace within which the flight information service and the alerting service are provided. The global airspace is divided into 9 FIR, illustrated

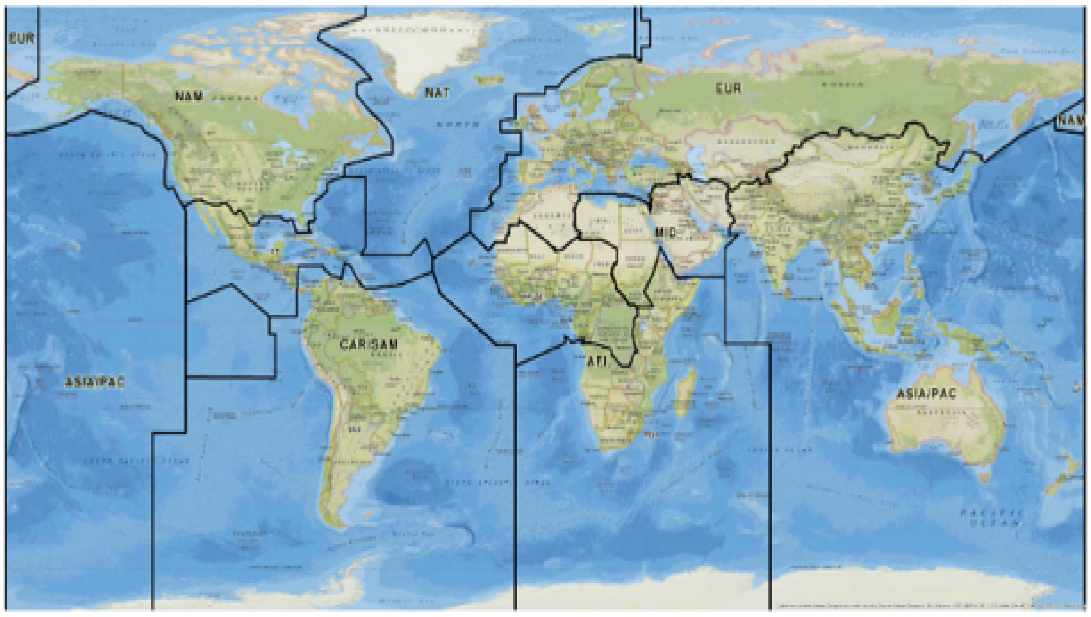

Figure 2.3: Division of the global airspace in nine FIR (ICAO).

in Figure 2.3. These FIR are further subdivided horizontally into smaller FIRs, attending to national airspace borders or managerial criteria.

Usually FIRs are divided vertically; if so, the lower portion remains named as FIR, whereas the airspace above is named Upper Information Region (UIR). In these cases, the FIR extends from the ground above up to the flight level FL 245 (24500 feet), while the UIR extends from FL245 up to infinity or an upper limit (for instance, FL460 for the Canary Islands airspace).

2.2.1.2 ATS Route/En-route

An ATS Route is a specified route designed for channelling the flow of traffic as necessary for the provision of ATS. Within en-route airspace, the main routes for aircraft consist on airways, usually with a width of 5 NM either side of the centre-line, and placed at an specific altitude, or Upper Air Routes (UAR), which are airways designated to be used by flight at high altitudes (generally higher than the usual altitudes of general aviation). The airways follow straight lines between “significant points”, also widely referred to as “waypoints”. Each ATS route establishes a protected airspace along them, as well as a safe spacing between adjacent ATS routes. Depending on their purpose, several types of airways can be identified: airway, advisory route (where only advisory service is available), controlled or uncontrolled route (if it crosses a portion of uncontrolled airspace), etc. Terminal routes are particularly relevant, since they contain the departing and arriving traffic from and at an airport. Depending on their usage, they can be divided in arrival or departure routes.

2.2.1.3 TA (Terminal Airspace)

Terminal Airspace is a generic term describing the controlled airspace surrounding an airport where air traffic services are provided and all traffic operating along Terminal Routes is contained.

Thus, it encompasses TMA, CTA, CTR and ATZ airspaces, as illustrated in Figure 2.4.

A Terminal Manoeuvring Area (TMA), also known as Terminal Control Area, is the portion of controlled airspace surrounding a major airport with a high volume of traffic, and in which a large number of airways or departures and arrivals converge [62]. The TMA is designed to handle aircraft arriving and departing flights (excluding aprons), and therefore the TMA is the airspace in which the approach control service is provided.

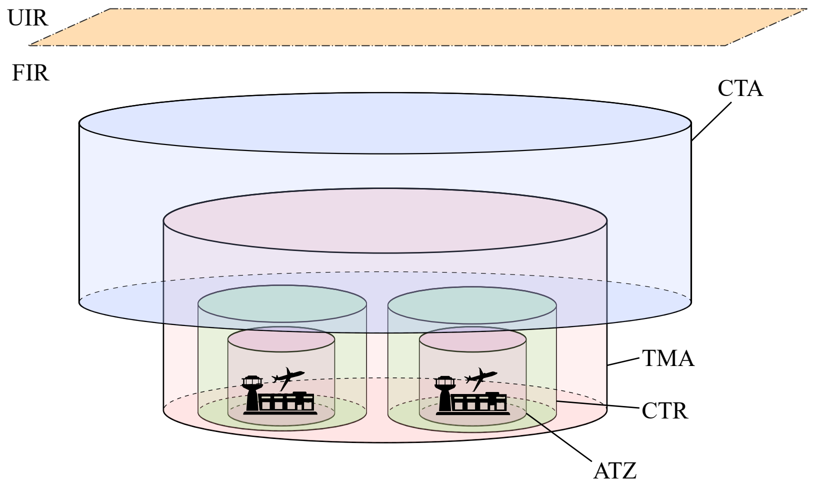

Figure 2.4: Airspace structure. Each airport can have an ATZ and a CTR, and major airports can have a TMA that can also comprise other minor airports. The CTA is connected with the terminations of airways, and enables incoming air traffic to reach the TMA.

The Controlled Traffic Area (CTA) is the portion of controlled airspace that exists in the vicinity of an airport (or several airports), which extends from a lower level (which cannot be the surface, and has to be higher than 200m/700ft) to a specified upper level. The CTA is usually situated on top of the Controlled Traffic Region (CTR), and provides protection to aircraft climbing out from the airport. It can contain the TMA, and a part of the airways converging at that TMA. A CTA can be shared by several airports, in which case its lower limit sits over all of their CTR (in case they exist). In Spain, the lower limit of a CTA is set at 300m/1000ft.

The Controlled Traffic Region (CTR) is the portion of controlled airspace around an airport (or multiple airports that are close together) which extends from the surface up to a specified upper limit, in a way that it does not overlap with a CTA, in case a CTA has been defined. This zone is established to protect air traffic operating to and from the airport (both IFR and VFR). CTR are usually defined with the shape of a cylinder, but they can be designed as irregular shapes depending on the geographical features of the terrain surrounding the airport.

A CTR is defined to have a maximum radius of 9.3 km (5 NM) in any direction from which approaches can be made, and covers from the ground up to the start of the CTA or the TMA, and lower than the vertical limit of the Aerodrome Traffic Zone (ATZ), if it exists [57].

Finally, the Aerodrome Traffic Zone (ATZ) is the portion of controlled airspace around an airport and under the supervision of a control tower (TWR). The ATZ starts from the surface up to 1000-3000 feet as their upper limit. The purpose of the ATZ is to enable the control tower to provide support to operations of VFR flights around the airport (takeoffs, landings and traffic circulation). The ATZ is not present in all airports: when IFR traffic is also present, the ATZ is usually included in the Controlled Traffic Region (CTR).

ICAO defines a maximum lateral extension of 25 NM for these zones. In Spain, ATZ are defined as cylinders with maximum dimensions of 8 km of radius and altitude of 900 m.

Service	FIS	ATC			Alerting	Advisory
		ACS	ApCS	AeCS		
FIR/UIR	x				x	x
ATS route	x	x			x	x
TA	x				x	
TMA	x		x		x	
CTA	x		x		x	
CTR	x				x	
ATZ	x				x	
Airport	x			x	x	

Table 2.2: Services provided within each of the airspace subdivisions.

2.2.2 Airports

Airports (or aerodromes) are facilities designed to support landing and takeoff operations, which are equipped with the necessary installations, services and equipment to ensure the effectiveness and safety of these operations. Airports are complex systems, so in this section we will only cover several of the main parts and concepts related to aircraft management, in particular those related to the *movement area*: aprons, taxiways and runways, and also the significant points around the airport aimed to control air traffic procedures around the airport.

The aprons are designated areas intended to accommodate aircraft on the ground for activities like loading or unloading passengers, cargo handling, refueling, parking or maintenance (de-icing, revisions, etc). Taxiways, on the other hand, are one-way “streets” that connect aprons to the runways of an airport. Aircraft taxiing is subject to approval of the control tower or ground controllers. There are several types of taxiways, depending on their purpose and the elements they connect, although we will not delve in such considerations.

More important, physical runways are defined areas on the aerodrome prepared for the landing and take-off of aircraft. An aerodrome may contain one or more runways, each with well-defined characteristics in terms of longitude, width, direction or defined landmarks, among other aspects. Each physical runway corresponds with two logical runways, depending on the direction in which the aircraft travel along it. For example, on an east-west runway, aircraft can approach from the east (heading west and travelling down the runway in that direction), or from the west (in which case the aircraft will travel down the runway towards the east). This fact leads into the concept of runway configurations. The orientation of a logical runway is defined by the bearing of the aircraft that is going to use it. In the example, there would be a west configuration, and an east configuration, respectively.

The configurations of an airport are defined as the possible combinations of all its logical runways. For example, an airport with a single runway has two possible configurations, one for each logical runway (note that these configurations are mutually exclusive, since both can not be active at the same time). An airport with two runways will have a maximum of four configurations, since the four logical runways can be arranged in four different combinations. However, some of these theoretical configurations may eventually be discarded if they compromise the safety of operations or are not feasible (e.g. the orography around the airport, the prevalence of winds in some direction, or the location of the airport). Runways at an airport can be arranged in different patterns (see Figure 2.5), which also influences the available configurations.

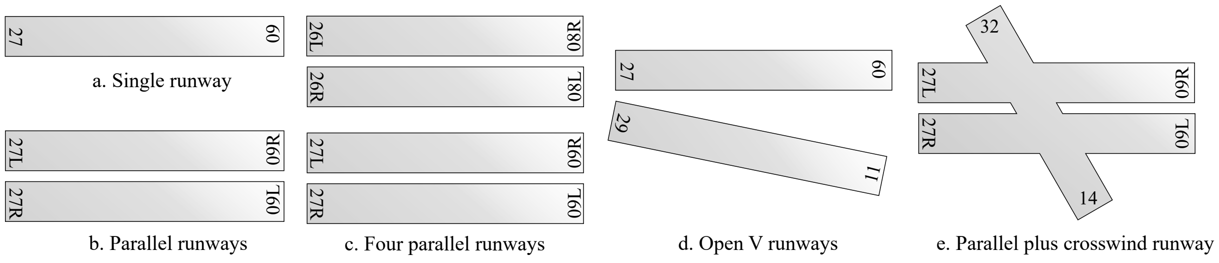

Figure 2.5: Different arrangements of physical runways.

Each logical runway is named according to a standard naming convention, consisting on two digits indicating its orientation with respect to the north (in tens of degrees and rounded to the nearest whole number) plus various suffixes to distinguish it from the other runways at the airport, if required. Going back to the runway in our previous example, the two logical runways would be named as 09 and 27, since their bearings are 90º and 270º, respectively. If multiple logical runways share the same orientation, a letter L, R or C (for left, right and center, respectively) may be appended to the number depending on the position of the runway with respect to the others. Lets suppose we have a second east-west runway at the airport. Both runways share the same orientation, so their number would be equal. The logical runways would be named as 09L and 18R for one physical runway, and 09R and 18L for the other. Additional rules are defined to more specific (and rare) cases. For instance, for the case of four parallel runways, they are named as two pairs of two parallel runways, and the numbers in one of these pairs is increased by 1.

Consequently, configurations are named after the logical runways that define them. In our two parallel runway airport example, the four configurations would then be 09R-18R, 09R-18L, 09L-18R and 09L-18R. In the next section, the Adolfo Suárez Madrid-Barajas airport is used to illustrate these concepts with a real-world example.

Airports also have associated standard procedures to orderly control the flow of air traffic arriving at or departing from the airport: the Standard Instrument Departure (SID) and Standard Terminal Arrival Routes (STAR) routes, respectively. SIDs and STARs provide a predefined procedure under IFR rules that establishes the flight path, speed patterns and altitudes that should satisfy any aircraft following the procedure. The SID or STAR that a flight should follow can be defined in the flight plan, and changed tactically by ATC if the situation requires so before the procedure is started. A SID links the aerodrome or a specific runway, with a specific significant point around the airport, such as a designated ATS route or any other reference point the aircraft has to fly by to leave the aerodrome. Conversely, a STAR defines the trajectory between an entry point to the airport TMA and the designated runway on which the aircraft must land.

There is one standard procedure more that is worth mentioning: holding procedures. With this procedure, the aircraft adopts a waiting position, in which it flies in circles over a holding point following a predefined pattern, as shown in Figure 2.6. This procedure can be ordered by ATC when an arriving aircraft can not land at the airport due to several reasons: all the runways are occupied, the weather conditions do not allow to perform the landing procedure, etcetera. In such situations, the aircraft must wait until the authorization to land is finally issued. Holding procedures provide a safe area to wait following a predefined flight path in the form of a descendent spiral pattern.

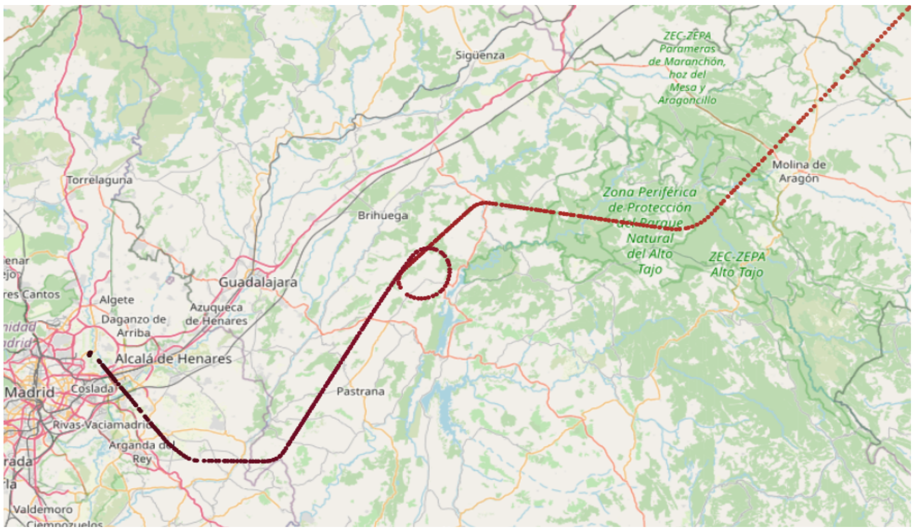

Figure 2.6: Example of a holding procedure. The aircraft changes direction when entering the TMA of the airport, and is instructed to wait at a designated position following a holding maneuver. When the pilot is authorized to land, the aircraft gets out of this position and heads towards the designated runway.

2.2.3 Description of Spanish airspace

To illustrate the previous concepts, we can use the airspace of Spain as example. Spain is situated in the FIR EUR, and has its airspace divided in three smaller FIR: Madrid, Barcelona and Canary Islands (with their corresponding UIR). Due to its size and volume of air traffic, the south region of the FIR of Madrid is delegated into another control centre situated in Seville. The three FIR are shown in Figure 2.7. Within these FIR, there are 12 TMA; every large and busy airport in Spain has its own TMA, although many are shared with smaller airports that are close to the main airport. Outside of these TMA, eight other airports have their own CTA. Figure 2.8 shows the location and boundaries of all these airspaces. A more exhaustive description of Spanish airspace can be consulted in [29].

We will focus now on the Adolfo Suárez Madrid-Barajas airport (ICAO code: LEMD, IATA code: MAD), managed by AENA and located 13 kilometres from Madrid. The LEMD airport is situated at 609 meters over the sea level (1998 feet). The Adolfo Suárez Madrid-Barajas airport is the busiest airport in Spain, and was the fifth in Europe and the fifteenth busiest airport in the world in 2023.

The airport contains four terminals and four physical runways arranged as two crosswind pairs of parallel runways (see Figure 2.9), resulting in eight logical runways: 14L, 14R, 32R,

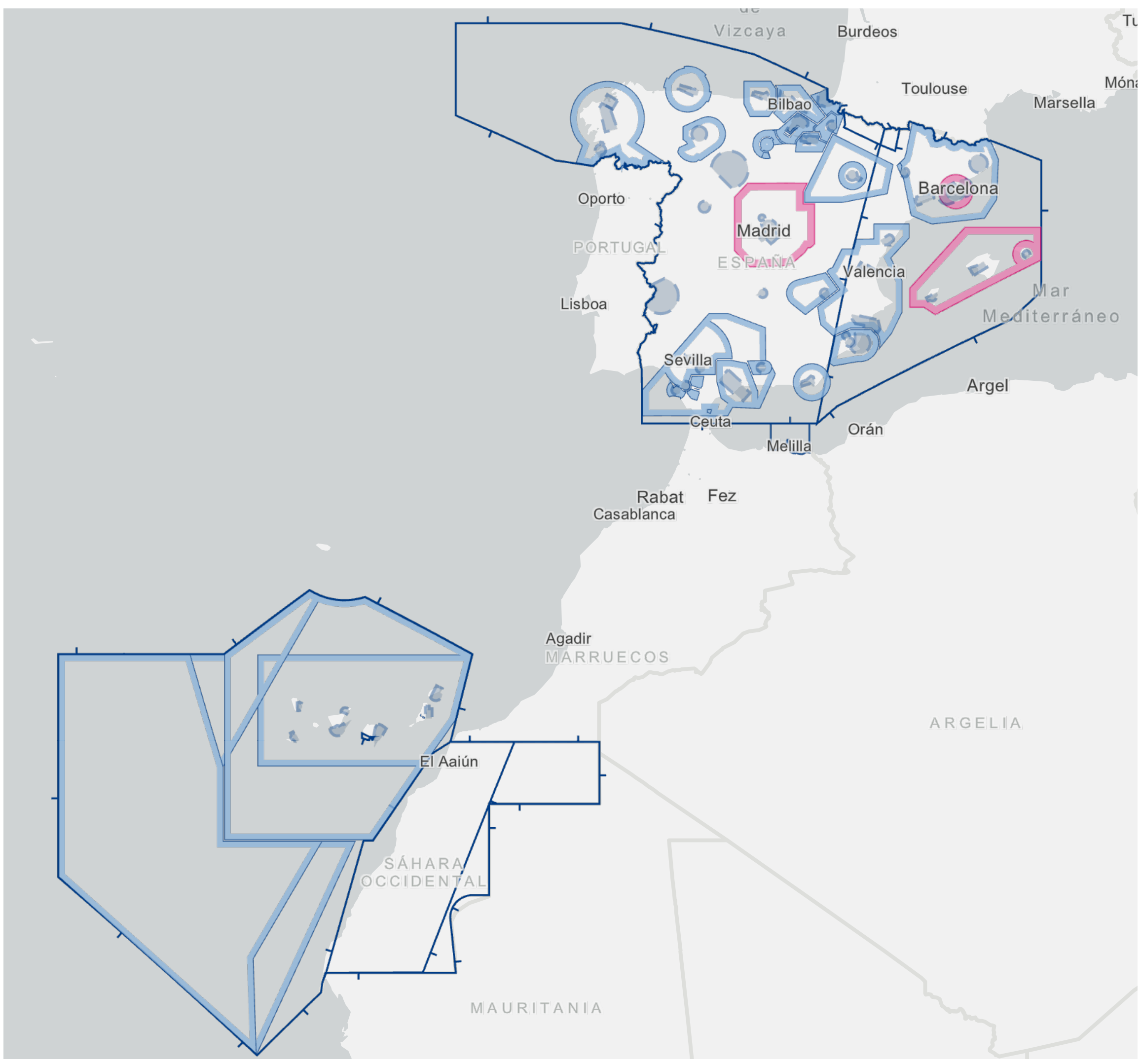

Figure 2.7: Structure of the Spanish airspace. The thin, dark blue lines indicate the limits of the three FIR in Spain: Madrid, Barcelona and Canary Islands. The shadowed, blue lines indicate the limits of the defined TMA. The areas in gray describe the CTR around the main airports. Source: Insignia (ENAIRE).

32L, 18L, 18R, 36R, and 36L. That is, two physical runways north-south and two southeast-northwest. However, only two configurations are commonly used: the north configuration (32L/32R-36L/36R) and the south configuration (18L/18R-14L/14R) due to the wind conditions that are prevalent in the zone. A total of 8 SID and 4 STAR are defined to leave and arrive to the airport, respectively, and each STAR is associated with several holding points for incoming flights.

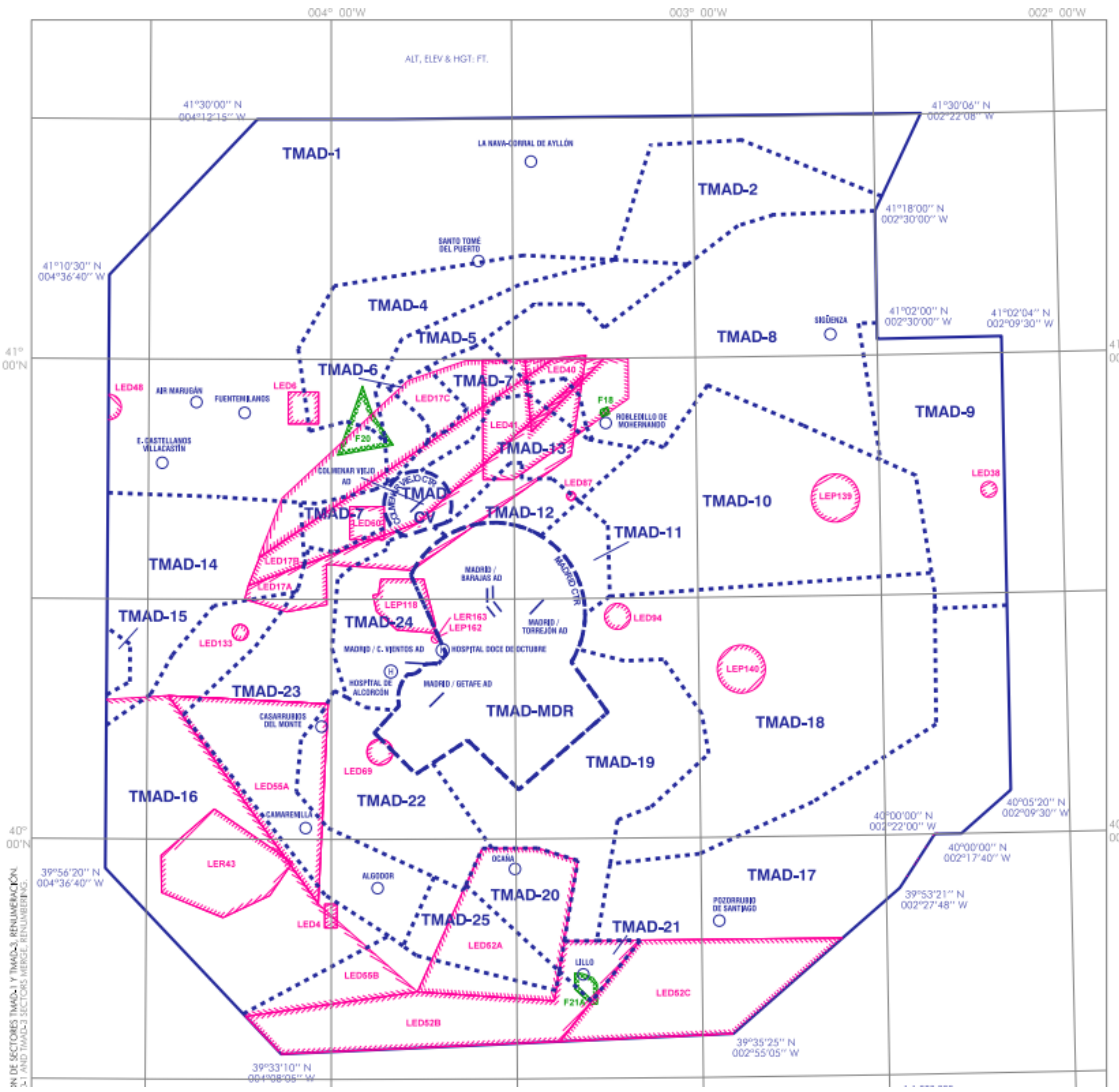

Figure 2.8: Structure of the TMA of the LEMD airport, comprising several airports (LEMD, Torrejón, Getafe...). Source: Enaire [31].

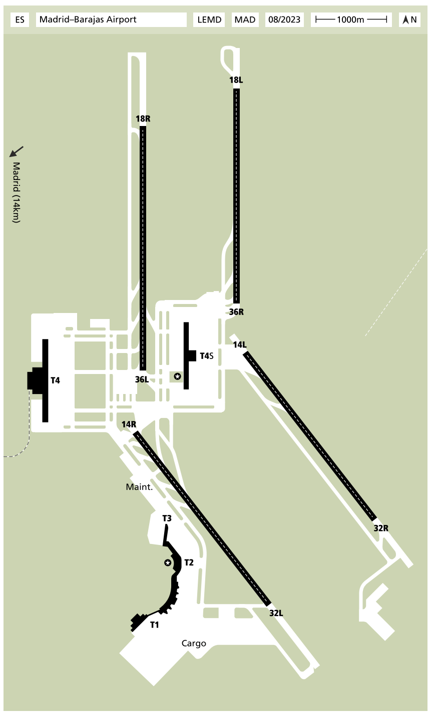

Figure 2.9: Runway arrangement of the Adolfo Suárez Madrid-Barajas airport. Source: Wikipedia (uncredited).

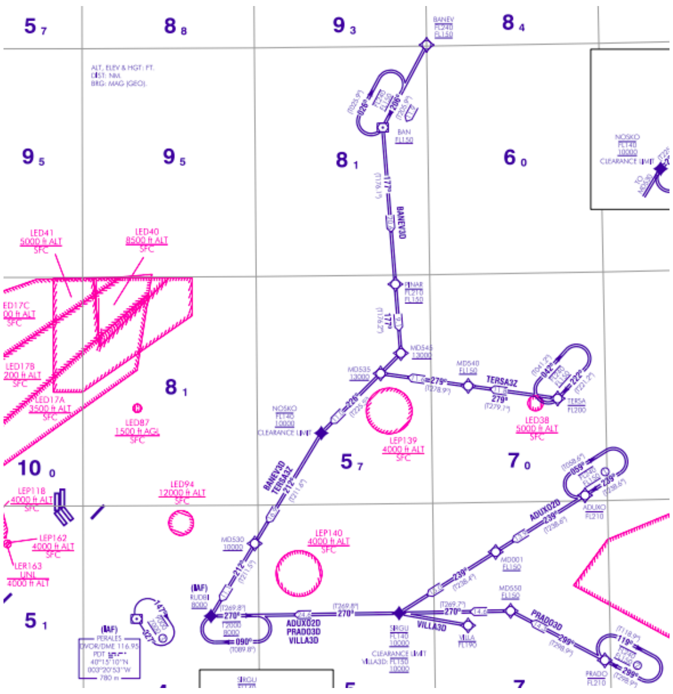

Figure 2.10: Examples of holding positions near the Adolfo Suárez Madrid-Barajas airport. Source: Enaire [30].

2.3 Altitude and flight levels

In previous sections, we have seen several criteria to divide the airspace, and in most of then, the altitude is also used to define their boundaries. Altitude is also key, for obvious reasons, to plan and carry out flight operations. While this concept can be understood intuitively, there are some relevant aspects to consider to use it in air traffic management.

In aviation, the vertical position of aircraft or other entities (such as significant points, or places on the surface of the Earth), can correspond with three different concepts. The *altitude*, as defined by ICAO [62], is the vertical distance of any object above the surface of the Earth to the mean sea level (MSL). The *elevation* is similar, but it is used to describe objects directly on the surface of the Earth, and generally it refers to their point of maximum height. The definition of *height* is more generic, and it indicates the vertical distance with respect to an specific datum, and not necessarily the MSL.

The altitude of an aircraft is measured using the altimeter installed in the cabin or the on-board GPS systems. The altitude calculation methods differ, and so do the resulting values for the altitude of an aircraft. Lets see how they are different:

- The geometric altitude is that given by the GPS system of the aircraft. Defines the altitude with respect to the reference system WGS-84, which defines a geoid that approximately describe the shape of the Earth. This approximation introduces a deviation between -100 and 70 metres with respect to the real altitude over the MSL.

- The barometric altitude is obtained directly from the altimeter of the aircraft, and is calculated based on the pressure difference with respect to a reference pressure. This reference pressure is manually set by the pilot, and must be adjusted in different situations during the flight.

Therefore, the geometric altitude is independent of the atmospheric pressure, but the barometric altitude varies depending on the pressure setting defined by the pilot. In aviation, there are two main pressure settings that are used in most airspaces, as shown in Figure 2.11:

- The QNH corresponds to the sea level pressure at the position of the aircraft, and is generally communicated by the ATC. It is constant within the zone from which it is defined, but it can change from one zone to another, since it depends on the meteorology among other factors.

- The QNE (or Standard altimeter setting) defines a reference pressure set at the Standard Pressure of 1,013.25 millibars/29.92 inches Hg. The calculated altitude is variable depending on the weather conditions (and therefore it might be incorrect), but since the same reference pressure is used by all aircraft within an area, the altimeters of all aircraft within that area will be off in the same amount.

There exists a third setting: QFE, which sets the pressure according to the pressure level at the airport, so the altitude at the ground of the airport is 0, instead of the MSL. In contrast, when it is set to QNH, the altimeter should inform the MSE elevation of the airport. However, this configuration is now generally in disuse.

The airspace is structured vertically, using the barometric altitude, in flight levels. A flight level or “pressure altitude” is a surface of constant atmospheric pressure which is related to a specific pressure datum (generally QNE), and is separated from other such surfaces by specific pressure intervals. The flight levels are defined to ensure a minimum vertical separation between aircraft, since two aircraft are not allowed to share a flight level within a radius in terms of horizontal separation. Flight levels are named using the two letters 'FL' with the altitude (with standard pressure or QNE) in hundreds of feet, rounded down. For example, an aircraft flying at 25,500 feet QNE is flying at the FL255.

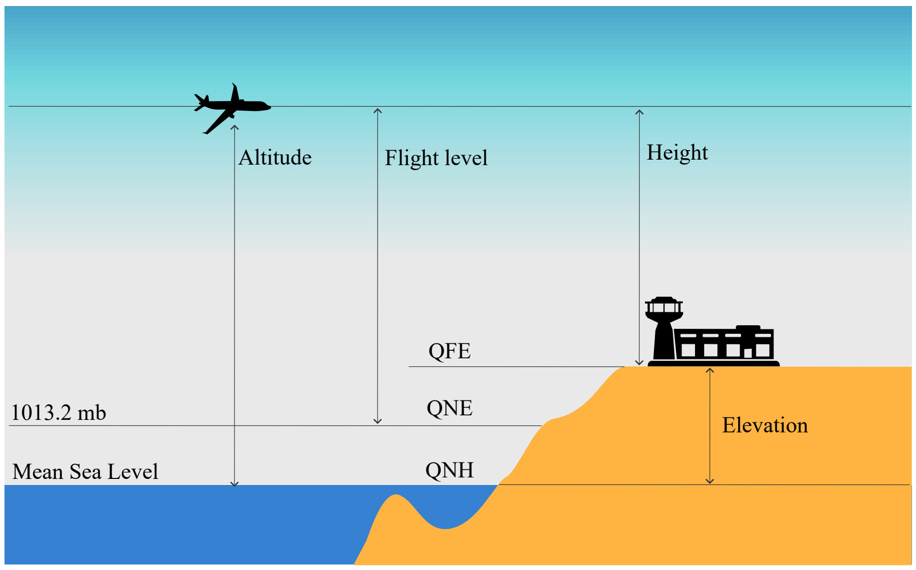

Figure 2.11: Altitude settings.

Both pressure settings are used during the flight. The standards dictate that QNH shall be used under the transition altitude, and any aircraft shall adjust its altimeter to QNE when flying above the transition altitude. The transition altitude is set at 18,000 feet (FL180) using QNH in the United States; in Europe, it varies from airport to airport, and the pressure setting depends on the intent of the aircraft: QNE is used for climbing, and QNH when descending. The transition flight level is defined as the last flight level available that is above the transition altitude.

2.4 Airspace classification

ICAO defines different airspaces of defined dimensions [57], alphabetically designated, within which specific types of flights may operate and for which air traffic services and rules of operation are specified. The characteristics of the different classes are summarized in Table 2.3. Class A is exclusive for IFR flights, which are provided with ATC and separation services. In the rest of airspace classes both IFR and VFR flights are permitted. The classes are organized in decreasing restrictiveness. Flights with IFR in airspace of classes B, C, D and E are subject to ATC, and the difference resides in the vicinity of an airport. Depending on the size and the air traffic density, the airspace around the airport can be of class B, C or D, as shown in Figure 2.12.

Additionally, different restrictions can be applied, temporally or permanently, to portions of airspace. There are three types of restrictions:

Class	Type of flight	Service Provided	Communication	ATC clearance
A	IFR	ATC	Continuous, two ways	Yes
B	IFR	ATC	Continuous, two ways	Yes
	VFR	ATC	Continuous, two ways	Yes
C	IFR	ATC	Continuous, two ways	Yes
	VFR	Traffic info	Continuous, two ways	Yes
D	IFR	ATC	Continuous, two ways	Yes
	VFR	Traffic info	Continuous, two ways	Yes
E	IFR	ATC	Continuous, two ways	Yes
	VFR	Traffic info	No	No
F	IFR	Advisory ATC, FIS	Continuous, two ways	No
	VFR	FIS	No	No
G	IFR	FIS	Continuous, two ways	No
	VFR	FIS	No	No

Table 2.3: Summary of characteristics of the defined airspace classes.

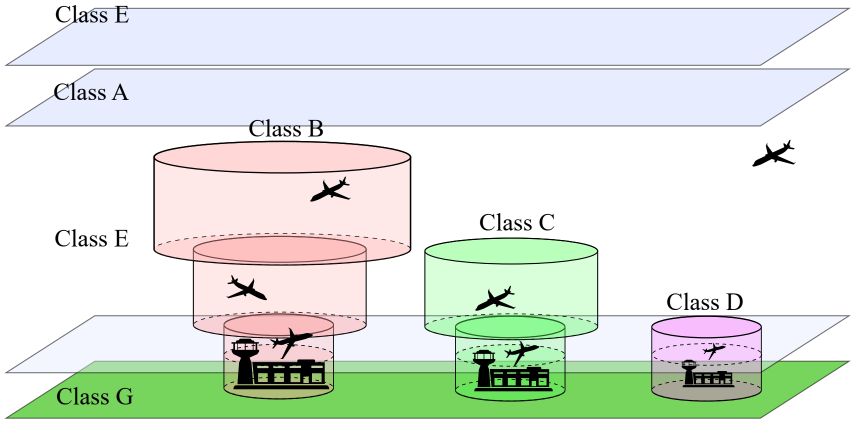

Figure 2.12: Illustration of the airspace classes.

Danger zone. A dangerous zone (D) is an airspace of defined limits in which dangerous activities for aircraft may develop. Typical activities developed might be: test flights, parachuting, rocket-launching... Normally, there are specific periods of time when dangerous activities take place; then, the dangerous zone is said to be 'active'. If this zone is not active, then this area can be considered as disabled and there is no danger.

Restricted zone. A restricted zone (R) is an airspace of defined limits in which dangerous activities for aircraft may develop. Typical restricted zones might be zones with: training flights, military training, sensitive fauna...

Prohibited zone. A prohibited zone (P) is an airspace of defined limits in which flying is totally prohibited (except for some authorized military and government use). Civil flights are not allowed in these areas, except special authorization. Normally, there are specific periods of time when dangerous activities take place; then, the restricted zone is said to be 'active' and it must not be crossed unless complying with published requirements.

2.5 Flights

Flights are defined by ICAO as "a planned operation of an aircraft from a specific point of departure to a specific point of destination, including intermediate segments if applicable." [62]. The definition by EUROCONTROL focuses more on the actual trajectory: "A flight is an aircraft movement identified by a unique Aircraft Identification (ACID) and flight plan, covering the period from off-block at the departure aerodrome until on-block at the arrival aerodrome" [38]. Therefore, a flight is characterized by two main aspects: the flight plan, which include the applicable flight rules, and the actual trajectory, which can take place in different stages. This section provides a succinct overview of these concepts.

2.5.1 Flight rules

The operation of an aircraft either in flight or on the movement area of an aerodrome shall be in compliance with the general rules [60] and, in addition, when in flight, either with the visual flight rules or the instrument flight rules.

Visual Flight Rules (VFR) are the flight rules applicable to operations by aircraft in visual meteorological conditions, that is, conditions in which it is possible to fly solely by visual reference. These conditions allow for a non-instrumentally aided navigation ensured that particular visibility conditions are met, and the pilot can guide the flight based on visual contact with other airspace users and determine the route guided by geographical landmarks.

Visual conditions depend on the situation, although certain minima are defined internationally [60]. For instance, the pilot must have a visibility of at least 8 km, and there must be no clouds within 1500 metres horizontally, or closer than 1000 feet in vertical. The pilot must also maintain a distance of 1000 feet above any obstacle in the ground over congested areas, or 500 feet when flying over uncongested areas.

Instrument Flight Rules (IFR) are applicable to properly equipped aircraft to fly under instrument meteorological conditions. These conditions are meteorological conditions expressed in terms of visibility, distance from cloud and ceiling, which are less restrictive than the minima defined for visual meteorological conditions.

An IFR flight operating outside controlled airspace and required by the appropriate ATS authority to submit a flight plan and maintain an air-ground voice communication with the air traffic services unit providing flight information service, shall report its time and level of passing each designated compulsory reporting point.

2.5.2 Flight stages

Flights develop in several stages, defined with their own goals, regulations and procedures. There exist different criteria in which a flight can be divided: IATA defines up to 18 phases, while ADREP [55], a taxonomy maintained by ICAO, reduces them to nine. However, it is possible to match the steps of one to another and it is only a matter of the description granularity. We will focus on the division proposed by ICAO from now on, illustrated in the Figure 2.13, which consist on:

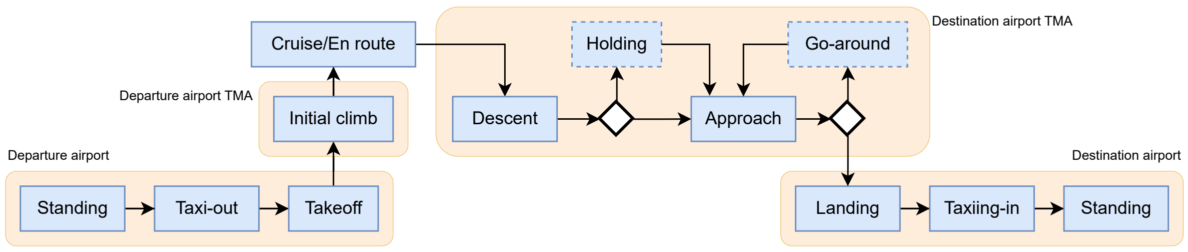

Figure 2.13: Flight stages defined by ICAO.

- Standing: Includes all operations before the aircraft engine is turned on and starts moving: flight planning, passenger and cargo onloading, aircraft preparation, etcetera. The moment in which the aircraft is authorized to start movement is called off-block time.
- Taxi-out: Corresponds with the movement on the surface of the aircraft, from the gate at the terminal in which it was parked, to the runway where the the takeoff is going to be performed.
- Takeoff: Is the phase where the aircraft travels on the runway and elevates from the ground, maneuvering to head towards the designated exit point from the airport's TMA.
- Initial climb: This phase starts once the aircraft reaches an altitude of 1000 feet/300 metres above runway elevation, and ends when the designated cruise altitude is reached.
- Cruise or en-route: In this phase, once the cruise altitude is exceeded, the aircraft climbs up to the designated altitude and heads towards the flight destination and the altitude, bearing and speed are mostly constant.
- Descent: This phase starts when the aircraft leaves its cruise stance and starts descending towards the destination TMA. During this phase, a holding procedure may be issued in case the aircraft cannot land at the airport.
- Approach: Once the aircraft have been authorized to land at a particular runway, the aircraft starts manoeuvring to align with the designated runway and attempt to land on it. If the landing cannot be performed (for example, due to winds or any problem at the runway), a go-around procedure may be issued to allow the aircraft to abort the landing and ascend again and move away from the airport to try again the landing operation or divert to another airport.
- Landing: During landing, the aircraft slowly descends with the purpose of landing smoothly at the designated runway, and decrease its speed once all its tires are in contact with the ground.
- Taxi-in: It corresponds with the movement of the aircraft, once it has reached the taxiing speed, from the landing runway to the gate of the terminal, until it parks there and stops the engines.
- Standing: During this last phase, the cargo, passengers and crew are offloaded.

2.5.3 Flight plans

A flight plan describes the main flight information to allow for strategic decisions related to route planning, resource allocation and safety assessment. The airlines are responsible of filing the flight plans for their scheduled flights prior to the flight start and submitting them to the IFPS. The Initial Flight Plan Processing System (IFPS) is a centralized service of the Network Manager Operations Centre (NMOC) to manage the initial processing and distribution of the flight plans in European airspace during the strategic planning phase. These flight plans are filed up to 5-7 days before the actual flight starts (i.e. the estimated off-block time, or EOBT), and include scheduling information such as the planned departure and arrival times, the airline that operates the flight, the origin and destination airports, and so on. Between the filing of the flight plan and the departure, these flight plans can be amended to accommodate possible changes in the schedule or the availability of resources. Twenty hours before EOBT, flight plan data is transferred to the ETFMS. The Enhanced Tactical Flow Management System (ETFMS) provides enhanced tactical data to all operational stakeholders, and facilitates improvements in flight management from the pre-planning stage to the arrival of the flight. Thus, ETFMS data describe the pre-tactical and execution phases of a flight, and goes beyond schedule information by providing data describing the actual flight, such as the effective times of the takeoff or landing, or the events taking place during the flight.

Any commercial flight in European airspace must file a flight plan and inform of any modification at any moment during the planning. The IFPS and ETFMS systems enable the real-time information sharing about flight plans among all stakeholders, as part of the System Wide Information Management (SWIM) infrastructure, to ensure the aforementioned conditions of efficiency and safety. Traditionally, flight plans consisted on a encoded text description indicating the main characteristics of the flight and the temporal and spatial milestones in its path. These descriptions were physically printed, and placed in front of the controllers responsible of ATC along the route of the aircraft. Nowadays, the flight plans contain much more information to allow for a more detailed description of the flights and enhance the predictability of the airspace. A complete description of the data shared through the IFPS and ETFMS systems is included in Chapter 4.

2.5.4 4D trajectories

4D trajectories are a key concept in the modern air traffic management. 4D trajectories enable a holistic representation of flight operations that integrate all relevant aspects into a single integrated and consistent form. The simplest conceptualization of 4D trajectories is as sequences of 3D positions, described by the latitude, longitude and altitude of the aircraft according to a geographical reference system, with their associated timestamp. The introduction of time in the trajectory description allows for an optimized airspace capacity management and safety assurance, in contrast with more traditional descriptions based on the waypoints included in the flight plan. This concept is depicted in Figure 2.14, with an example of semantic annotation by defining the flight phase at each point.

However, the concept of 4D trajectories has evolved to provide effective support to modern trajectory-based operations, depending on the airspace and the corresponding regulations and restrictions. The raw position information can be interpreted and enriched with semantics, such as flight phases, and other valuable information: weather, applicable flight rules, flight and aircraft types, airports of departure and destination, scheduled departure and arrival times,

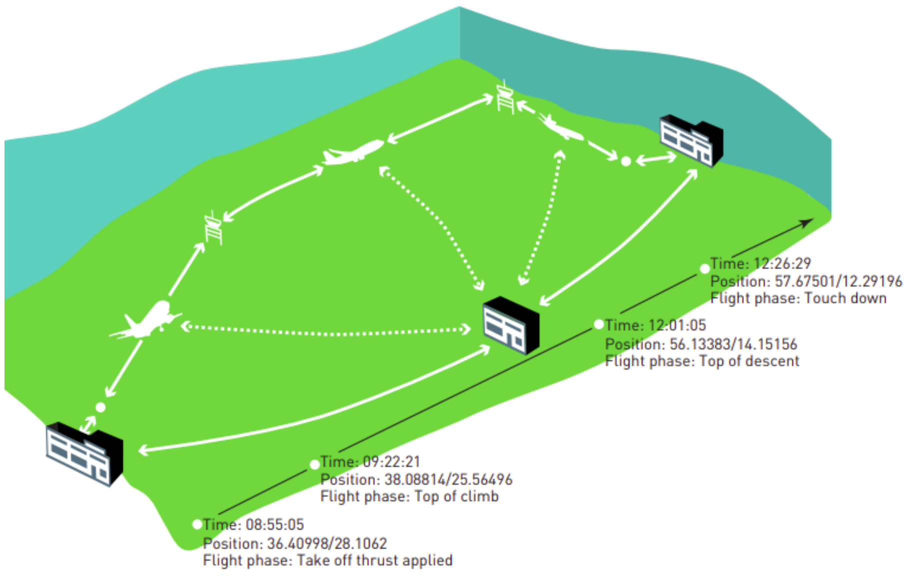
Time: 12:26:29
Position: 57.67501/12.29196
Flight phase: Touch down
Time: 12:01:05
Position: 56.13383/14.15156
Flight phase: Top of descent
Time: 09:22:21
Position: 38.08814/25.56496
Flight phase: Top of climb
Time: 08:55:05
Position: 36.40998/28.1062
Flight phase: Take off thrust applied

Figure 2.14: Diagram of a 4D trajectory, combining 3D position and time information.

etcetera [39]. The aforementioned regulations include separation criteria, capacity limitations at the airports, and other considerations that may affect the planning of flights. In particular, the vision of 4D trajectories in the context of European airspace was established in early 2010s as “business trajectories” for civil flights [40]. The detailed and unified representation of trajectories helps the required information sharing between all stakeholders implied in the ATM tasks. First, trajectories are defined by the airlines during the planning stage of the flight, and shared to the rest of the stakeholders. This initial definition and any subsequent changes in the flight plan are shared in real-time through the SWIM.

Chapter 3

Background

Modern Air Traffic Management relies on automated systems to continuously monitor the airspace and the traffic operating within it. These systems rely on a variety of sensors to assess key factors affecting the performance of the ATM elements, such as aircraft states and position, weather conditions, quality of communication channels and so on. Therefore, a lot of data is produced as a result of the operations related to air traffic and other areas in ATM.

Traditionally, many of these operations, such as estimation of the arrival time or trajectory prediction, have been tackled using approaches based on physical simulations using deterministic models [85]. These models take into account the different factors that have effects on the performance of aircraft, such as the aforementioned weather conditions, the flight characteristics or the performance profiles of the aircraft's makemodel, to produce a particular measurement to guide or evaluate the operations of an aircraft. However, the variety of these factors make it difficult to account for all of them, and are prone to significant deviations in presence of inaccuracies or errors in the defined parameters, which also require to be known in advance to the operation. They are also complex and costly to develop, and specific for a particular aircraft model. Therefore, other approaches have been explored in later decades to complement these effective yet often impractical methods. Data driven methods are one of the most prominent candidates to fulfil this role.

Many data driven methods leverage historical data to model complex problems based on observed patterns and previous experience. Among them, machine learning has emerged as a prominent approach for modelling real-world events based on data, due to the development of advanced algorithms and the notable increase in the availability of computational resources. Machine learning relies on automatic systems that "learn" from past observations, each characterized by a set of dimensions or *features*. The learning process consists of optimizing a function that approximates the desired outputs for a particular set of inputs. During training, past observations are fed into the model, and the internal parameters that define the function (the *weights*) are adjusted iteratively to minimize the prediction error on these examples.

Deep learning is one of the aforementioned advanced machine learning algorithms that are currently succeeding in tackling the most complex tasks. Deep learning builds upon the concept of *neural networks*, a set of smaller, inter-connected components that allow to define more complex functions than simpler machine learning models. These networks typically consist of an input layer, responsible of taking the input into the network; an output layer that consolidates the outputs generated by the network; and one or more hidden layers, which process the inputs by applying several transformations in order to produce the desired outputs. In a typical neural network, the information is passed from one layer to the next layer only (in such case they are called feedforward neural networks). There exist other types of networks that define different connections, such as recurrent networks, in which the elements within the same layer can be connected, or residual networks, which implement skip connections that allow the information to bypass some layers.

The approach (either machine learning or deep learning) and algorithms chosen depend on the complexity of the task and the nature of the data. In ATM, there is a prevalence of sequential data, like data describing the state of a particular real-world element over time. In this chapter, we discuss these topics in order to identify the most suitable approach to fulfil the objectives of this Thesis.

3.1 Sequential data

Sequential data consist of a set of individual elements, or data points, that are arranged in an ordered fashion and a relation of dependence exists between the individual elements in the sequence [26]. Thus, individual elements are interpretable in the context of the other elements, more than on their own. Moreover, in sequential data, the progression of the values across the sequence is often more relevant than values alone. Sequential data have dimensions, which are determined by the length of the sequence (the number of elements) and the number of features that describe each individual element in the sequence. Each element can be described by one or more features with a varied nature, either numeric or categorical.

Time series are a prominent example of sequential data. In time series, each individual element has an associated timestamp, which is often unique within the sequence of data, in addition to the features comprising each element. In general, sequential data of time series correspond to periodic measurements of a particular event or metric, with a defined rate (e.g. daily, hourly, etc.). In time series, the order of the elements is determined not (only) by their position within the sequence, but mainly by their order in the time dimension. The time dimension enables further analysis of the sequences, introducing additional notions such as the time distance between the elements, or the consistency of these distances between adjacent elements.

Common examples of time series are data generated by the monitoring of physical or industrial processes, where a sensor regularly takes readings of a particular metric to characterize different dimensions of the monitored process. In the context of Air Traffic Management, 4D trajectory data can easily be interpreted as a time series, albeit a particularly complex one. In Chapter 2, we introduced some of the dimensions that characterize the state of an operating flight, such as location, altitude, or speed. That is, 4D trajectory data constitute a *wide* sequence, since each element encompasses numerous features, and the data sequences are typically very long. Current monitoring techniques, such as ADS-B (which will be covered in the next chapter), generate new readings several times per second. In flights several hours long, the number of elements in the sequence is obviously considerable. Moreover, the data sequences as a whole can be associated descriptive properties, which are constant for all the elements in the sequence, such as the identity of the flight, the departure and destination airports, and so on. These properties are often called static features.

Many tasks related to sequential data require some form of prediction, where a sequence of data can be utilized to derive a value or another sequence, generally the next elements that would continue the sequence. These tasks are known as many-to-one and many-to-many (or sequence to value and sequence to sequence problems), respectively. In ATM, there exist examples of both scenarios. On the one hand, an example of sequence to value prediction problem is the estimation of the arrival time of an aircraft, where based on the sequence of the latest states of the flight, we try to determine the remaining time until it arrives at the destination airport. On the other hand, trajectory prediction requires to forecast several future states of an aircraft, which should also be coherent; that is, these predicted states should also form another sequence of data. In time series, the time dimension is especially relevant, since it is key to fully apprehend the progression in the values of the features as it might not be linear if the time distance between the elements is not constant.

3.1.1 Time series prediction strategies

In sequence to sequence prediction problems, multiple approaches or strategies can be followed to produce multiple elements based on a single sequence. Ben Taieb et al. [13] identified five strategies to make multi-step predictions in time series, which are illustrated in Figure 3.1. Each strategy is characterized by two aspects: whether a single or multiple models are used to make predictions, and the length of the output of the models.

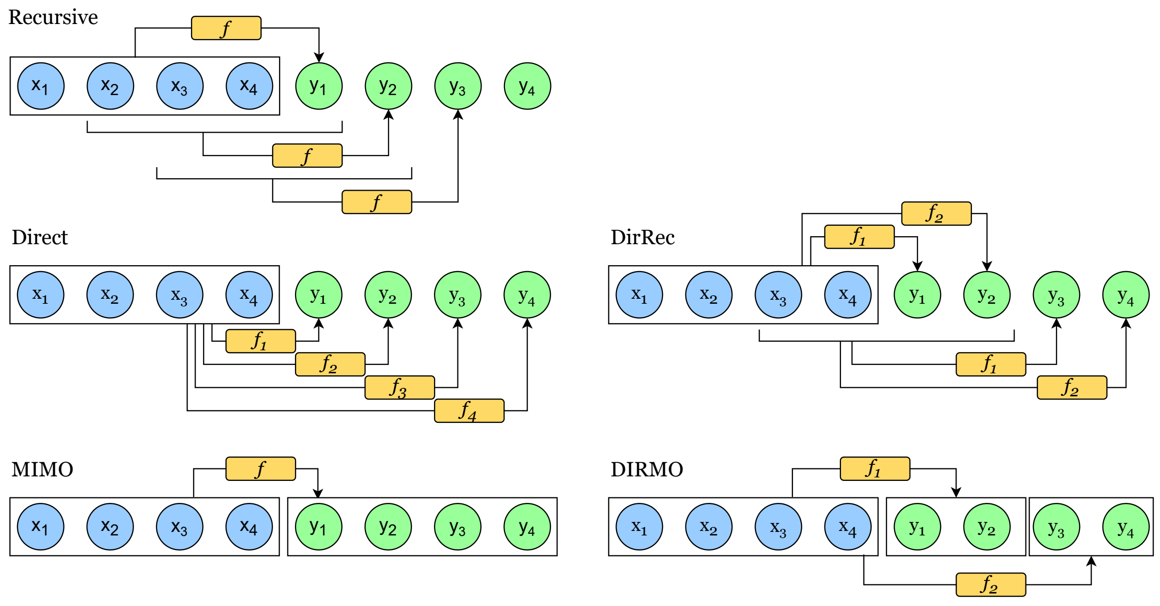

Figure 3.1: Graphical description of the defined prediction strategies. In all examples, the models take four elements as inputs.

The *Recursive* strategy defines a single-output, one-step ahead model to predict the next timestep. In order to predict further timesteps, the previous predictions are appended to the input window; therefore, the prediction error is propagated through successive predictions.

$$y_{m+1} = f(x_1, ..., x_m) \tag{3.1}$$

The *Direct* strategy defines a single-output model specifically for each time horizon that should be predicted. This method avoids error propagation, but implies a conditional independence between successive predictions:

$$(y_{m+1}, ..., y_{m+p}) = (f_1(x_1, ..., x_m), ..., f_p(x_1, ..., x_m)) \tag{3.2}$$

The *DirRec* (Direct-Recursive) strategy combines the previous two strategies. DirRec consists of predicting several timesteps using a direct strategy, and appending these predictions to the input sequence, to predict the next set of timesteps recursively.

$$\begin{aligned}(y_{m+1}, ..., y_{m+p}) &= (f_1(x_1, ..., x_m), ..., f_p(x_1, ..., x_m)) \\ (y_{m+1+p}, ..., y_{m+2p}) &= (f_1(x_{1+p}, ..., x_{m+p}), ..., f_p(x_{1+p}, ..., x_{m+p}))\end{aligned} \tag{3.3}$$

The remaining two strategies define multi-output models that predict more than one timestep at a time. The MIMO (Multi-input multi-output) strategy defines a multi-output model to predict the next p timesteps. This approach conserves the stochastic dependence between the elements in the predicted sequence and avoids the error propagation, but has a reduced flexibility given that only one model is used for all time horizons:

$$(y_{m+1}, ..., y_{m+p}) = f(x_1, ..., x_m) \tag{3.4}$$

Finally, DIRMO combines Direct and MIMO strategies: multiple multi-output models are defined for different blocks, each of them calculated in a MIMO manner.

The choice of the best strategy depends on the capabilities of the chosen models and the needs of the problem. The strategies with a recursive nature (Recursive and DirRec) can pose some risks due to error accumulation: the error in the first outputs influences the inputs for later predictions, and in some cases the accumulated error can affect the performance of the resulting model. The MIMO strategy, however, is more complex and difficult to train, and may be more susceptible to noise in the data. Finally, the outputs in DirRec and DIRMO approaches are independent of each other, ignoring the sequential nature of the desired output, and their performance degrades as the prediction moves further out in time.

3.1.2 Application of deep learning approaches to sequence processing

In general, sequences can be of different lengths: for example, different flights have different durations, even if they share the same origin and destination airports. This can constitute an obstacle to apply many of the existing deep learning approaches to sequential data processing. Most of these methods assume that the input sequences are of fixed length. Additionally, time series processing methods also assume the sequences to be homogeneous and evenly distributed from the time perspective. That is, the sequences are continuous –there are no gaps or skips in the sequence of elements–, and the periodicity between elements is consistent, that is, the time between successive elements is constant for all elements. Finally, several methods require the data to be numerical, o restricted to a particular range of values, albeit it is clear that data will not always fit this description.

There exist multiple approaches to face situations where these requirements are not met. For example, variable length in the sequences can be resolved by shortening or filling up the sequences to accommodate them to a defined length. Inconsistencies in the time dimension can also be approached in a similar fashion. In the following, some of these techniques are briefly presented.

3.1.2.1 Windowing

Depending on the target use case, it can be required that long sequences with variable length are converted into fixed-length sequences. For instance, several deep learning architectures require that inputs have constant length, while 4D trajectories comprise a variable number of state vectors.In these cases, each sequence can be broken down into one or more sub-sequences of the required length, so each sub-sequence can be processed individually and interpreted as a sequence itself.

Multiple strategies can be adopted to construct these sub-sequences. A typical one is to apply a sliding window on the original sequence. A window is defined as a consecutive interval of lb elements from a sequence (hence having a length of lb). The windowing operation through a sliding window is defined by two parameters: the window size, often referred as lookback, and the step size, which indicates the number of elements skipped between the start of two consecutive windows. A step size of 2 skips one element between windows, while a step size of 1 does not skip any elements. More formally, given a sequence of n elements, $(x_1, x_2, ..., x_n)$, we can generate $\lfloor (n - lookback)/step \rfloor$ sub-sequences of length lb where each window is a subsequence $w_t = (x_{t \cdot step-1}, ..., x_{t \cdot step-1+lb})$ of the original sequence, as illustrated in Figure 3.2. In the example, the windows have a length of 3, and each new window skips one element at a time. As a side note, if the lookback and the step size have the same value, the resulting sub-sequences are disjoint.

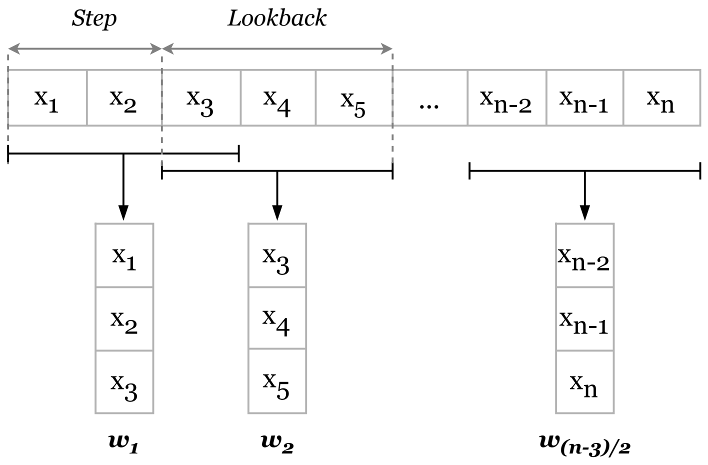

Figure 3.2: Example of the windowing operation on a sequence with n elements with a lookback of 3 and a step size of 2.

3.1.2.2 Normalization

Most machine learning methods require numeric data, so often it is necessary to preprocess the data to apply the corresponding transformations. Sequential data is no different, and therefore the standard transformations can also be applied on them.

Some common transformations that are required to process 4D trajectory data are the encoding and the scaling of the values. Encoding is the operation of transforming categorical data (generally features that are not numerical) into numerical values. Some techniques to this are to replace each categorical value with a particular numerical value according to a replacement mapping. Scaling consists of transforming numerical values through a mathematical expression to apply an invertible transformation (so that the original values can be recovered later), to make the values appropriate for better interpretation by the model. Some examples of these transformations are scaling the values to a particular range (such as $[0, 1]$) to compress all features into this range and neutralize the scale differences between the different features; or transforming the data distribution by centering it at 0 to have more symmetrical distributions.

Ultimately, these transformations depend on the requirements of the model. Later we will introduce Recurrent Neural Networks as a means to process sequences of data; these networks can only use numerical data, and is common practice to compress the range of all features into $[0, 1]$ to ensure a better behaviour from the network.

3.2 Recurrent neural networks

In recent years, methods based on Recurrent Neural Networks (RNN) have shown good performance in time series modeling tasks, provided that they are able to capture temporal dependencies in sequential data [52]. In contrast with traditional feed-forward neural networks, in which each layer passes information only to the next layer, recurrent layers present cycles within them, that is, they have links between the neurons in the layer. This fact enables RNN to have "memory" from the elements that have already been processed. A simple recurrent layer contains a single recurrent neuron that expects a sequence of elements as input. Recurrent neurons contain a *cell state* that changes after processing each input element, and is used to process the next input element. The layer iterates on every element in the sequence: at each step, the cell state is propagated from the previous iteration and used to process the next element in the sequence. Thus, the layer has "memory" of the elements that were processed before in the input sequence. When all elements in the sequence have been processed, the final output is passed to the next layer.

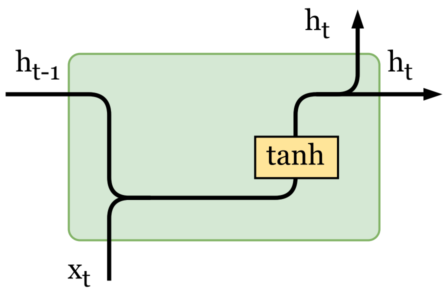

Figure 3.3: RNN unit internal structure.

This structure causes RNNs to form very deep structures that increase the risk of the vanishing gradient problem [52], particularly when analyzing sequences with long-distance dependencies. Vanishing gradient problem consists of the error becoming too small or zero during the back-propagation step in model training. If the errors are zero, the parameters of the model are not updated (the value of the weights does not change); this translates into a part of the model is not "learning" correctly, or taking a lot of time to learn long-term dependencies in long sequences.

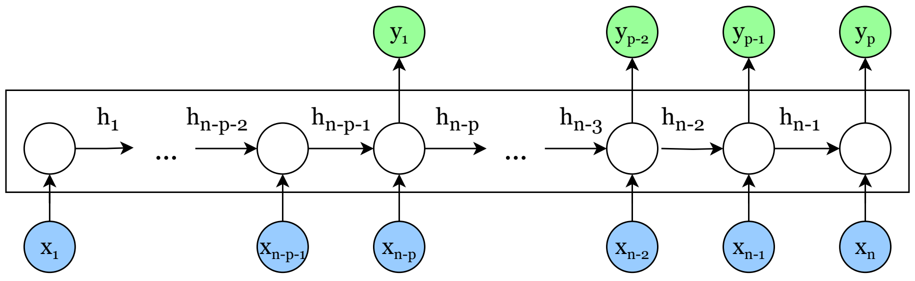

Figure 3.4: Example of a RNN that takes a sequence $\boldsymbol{x} = (x_1, ..., x_n)$ of size n and outputs a sequence $\boldsymbol{y} = (y_1, ..., y_p)$ with size p.

This problem can be handled using different techniques, but Long Short-Term Memory networks in particular have attracted much attention in recent years.

3.2.1 Long-Short Term Memory networks

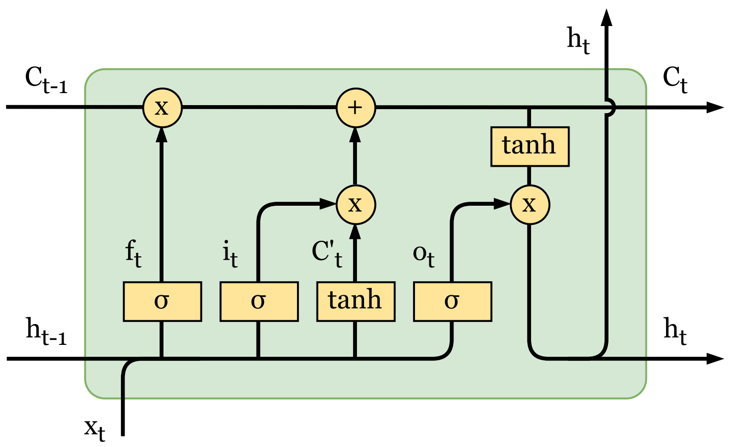

Figure 3.5: LSTM unit internal structure.

Long Short-Term Memory (LSTM) [53] is a gated recurrent network architecture that ensures error propagation even in deep recurrent layers, allowing the model to have “long-term” memory without the loss of “short-term” memory shown by traditional RNN.

Inside a LSTM cell (see Figure 3.5), three multiplicative units are defined that act as gates with different purposes and regulate the flow of information across the layer: the forget, input and output gates. This information consists of the cell state (the long-term memory mechanism), and the hidden state, which is the output of the unit to be passed to the next iteration, or the next layer if the iteration has finished. At each step, the unit receives three values: the element itself, the cell state, and the hidden state of the previous iteration.

Let us exemplify the process for an element, x_t, in an input sequence with length lb: $X = (x_1, ..., x_{t-1}, x_t, x_{t+1}, ..., x_{lb})$, using the cell state and the hidden state from the previous iteration, C_{t-1} and h_{t-1}, respectively. First of all, the element x_t is concatenated with the hidden state h_{t-1}: $V_t = [h_{t-1}, x_t]$, to be used as input to the three gates. Thus, each element is processed in light of the output of the previous iteration. The forget gate f_t determines the influence of the propagated cell state on the processing of the current element.

The input gate i_t controls how much information from the input element will contribute to the cell state. In practice, it protects the cell state from perturbations and irrelevant elements in the input sequence. This gate comprises two parts: a sigmoid function that controls which parts of the cell state will be updated, and a hyperbolic tangent function (tanh) that generates a candidate cell state to update the previous one.

$$
\begin{aligned}
&\text{Forget gate:} && f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
&\text{Input gate:} && i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)
\end{aligned}
$$

The cell state is updated with the results of the forget and input gates: the first determines which parts of the cell state should be forgotten, and the second determines which values from the candidate cell state should be added to the definitive cell state. This cell state is propagated directly to the next iteration of the recurrent layer.

Finally, the output gate outputs the most relevant parts of the cell state, once it has been updated. This helps to filter the information to be passed on to the next iteration, avoiding the propagation of irrelevant information from the current hidden state in light of the current element. The hidden state is propagated to the next iteration in the recurrent layer, and it can also be passed to the next layer, depending on the specified output size of the layer. It is worth noting that, during this process, the hidden state and the output are updated separately, which helps to ensure long-term memory.

$$
\begin{aligned}
&\text{Candidate cell state:} && C'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \\
&\text{Cell state:} && C_t = f_t \odot C_{t-1} + i \odot C'_t \\
&\text{Hidden state:} && h_t = O_t \odot \tanh(C_t) \\
&\text{Output gate:} && o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)
\end{aligned}
$$

LSTM networks have already proven to be effective in dealing with air traffic data in different problems in the ATM field. In [14], LSTMs were combined with convolutional neural networks to predict the aircraft type using ADS-B data. Different aircraft types present particular patterns in their displacement, which can be identified in sequences of surveillance data. LSTM can analyze the progression of the latitude and longitude values, among other features, to classify the flight according to the aircraft type. Beyond RTA prediction, other regression problems have also been tackled with the use of LSTM. Shi et al. [99] proposed an architecture for trajectory prediction based on ADS-B data. Their results showed that the last reported positions of an aircraft, among other features of interest for this task, allow accurate prediction of its future positions.

3.2.2 Gated Recurrent Units

Gated Recurrent Units, or GRU [20], offer a simplified alternative to LSTM, reducing computational complexity while preserving sequence modeling capabilities. The architecture of GRU reduced the number of gates in LSTM networks by merging the forget and input gates into a single "update gate" (see Figure 3.6). This simplification often leads to small improvements in training times since the architecture has fewer parameters to train, although the performance difference between GRU and LSTM units largely depends on the nature of the problem. Instead, the input sequence is encoded into a sequence of vectors, and a subset of these vectors is selected during decoding.

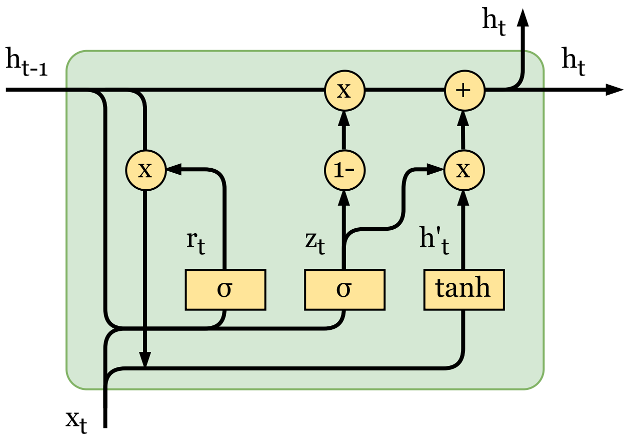

Figure 3.6: GRU unit internal structure.

We can illustrate the processing of a GRU layer using the same example as for LSTM networks. GRU units take only two inputs in each iteration: an element x_t, and the hidden state from the previous iteration h_{t-1}. In this case, the hidden and the cell states are not propagated separately.

On the one hand, the element x_t and the propagated state are concatenated and passed through the reset gate r_t, whose role is to determine which parts of the hidden state are relevant for the current iteration. Again, a sigmoid function is used with that purpose. The modified hidden state is concatenated with x_t to calculate the candidate state of the cell, h'_t.

On the other hand, the concatenation of x_t and h_{t-1} is also passed to the update gate z_t, which decides, in a single step, which parts of the input element will contribute to the hidden state, and which parts of the candidate state will modify the current cell state.

Reset gate: $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$

Update gate: $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$

Candidate cell state: $h'_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$

Cell state: $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t$

GRU networks have been used less than LSTM, since they are significantly more recent. However, the development of attention-based architectures may be an even stronger argument against GRU gaining attraction, especially since attention mechanisms have begun to displace recurrent networks in sequence processing research. In the following sections, we will introduce these mechanisms, and motivate why they have attracted such interest.

3.3 Transformer-based architectures

Attention mechanisms [11] help models to discover implicit patterns in sequences of data by identifying which elements are more important for performing a task. In this way, the model can modulate the importance of each element in the sequence by assigning weights to them, and focus more on the most relevant elements. Attention mechanisms became most prominent with the publication of the Transformers architecture [114], which implied a breakthrough in sequence to sequence problems.

3.3.1 Attention

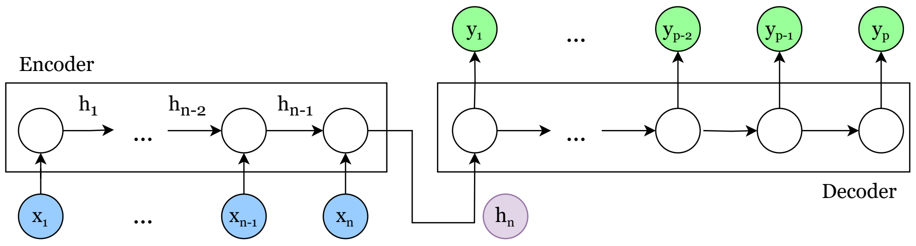

Figure 3.7: Example of an encoder-decoder architecture.

Bahdanau et al. [11] proposed attention mechanisms to improve the performance of encoder-decoder architectures for long and complex sequences. Encoder-decoder architectures consist of two separate models (the encoder and the decoder) with two different roles: the encoder takes an input sequence, and encodes it into a fixed-length sequence; that is, the encoder summarizes the input sequence. Then, the decoder processes this sequence to produce a new sequence. One of the first applications of such architecture was automatic translation [20], where a sentence in the source language is encoded into a fixed length representation, which is later used by the decoder to produce the corresponding sentence, a word at a time, in the target language. However, encoder-decoder models based on recurrence have a few shortcomings that attention mechanisms try to alleviate to some degree. Without attention, the decoder does not have access to all information in the input sequence, given the summarized representation as a fixed-length sequence (which in principle is much shorter than the input sequence) that is generated by the encoder.

Attention is constructed upon three concepts: a set of scores e_{ij} according to an alignment model a, a matrix of weights α_{ij} and a context vector c_i:

$$e_{ij} = a(s_{i-1}, h_j) \tag{3.5}$$

$$\alpha_{ij} = \mathrm{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{3.6}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{3.7}$$

The alignment model a evaluates the strength of the match between the inputs in positions i and j (in the original paper, this model is implemented as a feedforward neural network).

h_j is the j-th annotation generated by the encoder, and s_{i-1} is the hidden state of the decoder right before generating the i-th element in the output sequence, y_i. The weights α_{ij} describe the probability that the target element, y_i, is aligned to a source element, x_j. That is, the importance of the annotation h_j with respect to the previous hidden state (of the decoder) s_{i-1} in deciding the next state s_i and generating the output element y_i. The softmax function ensures that $\alpha_{q,k_i} \in [0, 1]$ and $\sum_i \alpha_{q,k_i} = 1$ for the calculated weights. Effectively, these parts provide an attention mechanism to the decoder, since the decoder can decide which parts of the input sequence should be given more importance, and thus the encoder can spread the information from the input sequence throughout the sequence of annotations instead of having to encode it all into a fixed-length vector.

Finally, the context vector c_i depends on a sequence of annotations $(h_1, ..., h_{T_x})$ to which the encoder maps the input sequence. These annotations are generated by the encoder: each annotation h_i corresponds with the hidden state of the encoder after processing the i-th element in the sequence, and it contains information about the whole input sequence, with a particular focus on the part surrounding the i-th element in the sequence. The context vector is used by the encoder to process new elements in the sequence in light of the previously processed elements.

Several variants to this attention mechanism have been proposed, with self-attention and multi-head attention being two of the most prominent. These concepts were introduced as parts of the Transformer architecture, and thus they are covered in the following section.

3.3.2 Transformer

The Transformer architecture [114] replaces recurrence with attention mechanisms in deep neural networks, achieving great success in tasks such as machine translation. With the removal of recurrence, which enforces a sequential processing of the input sequences, Transformer allows for parallelization during the training process, although it loses the notion of the order of the elements within the input sequence. This fact is overcome by adding a component responsible of position-encoding the elements in the input sequence. Hence, Transformer provides a parallelizable, high-performance alternative to traditional recurrent networks for sequence analysis. The building block of this architecture is shown in Figure 3.9.

Transformer relies on three key concepts: self-attention, multi-head attention and the positional encoding of the elements in the input sequence. Self-attention takes into consideration the importance of each element in the context of the rest of the elements in the input sequence, rather than using a predefined set of keys. Therefore, queries and keys in self-attention are the same. Multi-head attention (see Figure 3.8) allows for different interpretations of the input sequence by projecting the query vectors into different representation subspaces. Finally, positional encoding allows the model to account for the order of the elements in the sequence, which was lost after removing the recurrence. In the following, we briefly review these three components.

Transformer generalizes the concepts presented in [11] and makes an analogy to searching information in a database: these concepts are defined as queries Q (which are the previous decoder outputs, s_{t-1}), values V (the encoded inputs, h_i), and keys K (equal to the values). The vectors in Q and K have a dimension d_k, while the keys have a dimension d_v, which corresponds with the length of the encoded representation of the inputs. Then, attention is defined as:

$$\text{Attention}(Q, K, V) = A(Q, K)V = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.8}$$

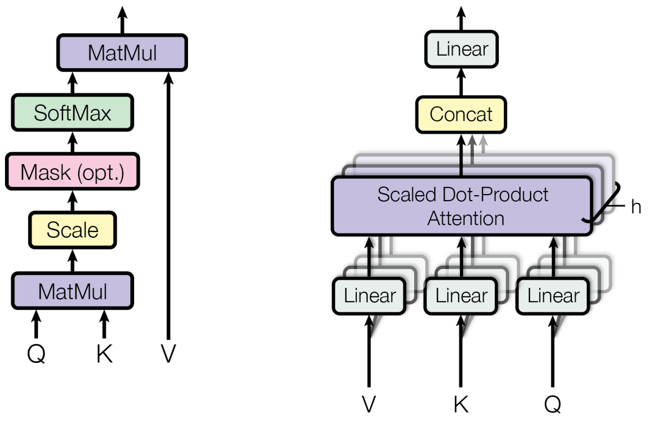

Figure 3.8: Single head (left) and multi-head (right) attention [114].

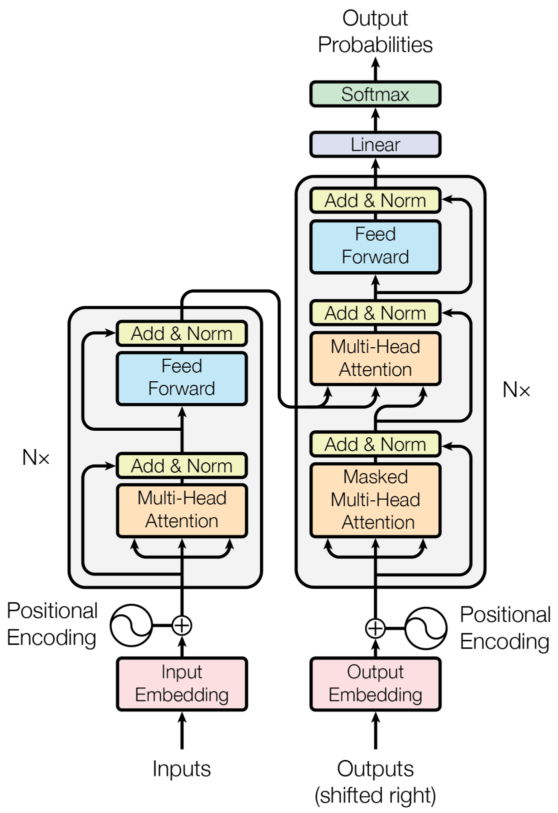

Figure 3.9: Transformer architecture [114]. The encoder (left) encodes the input sequence through multiple attention blocks, and passes the results to the decoder. The decoder produces the output by feeding the previous outputs, or a placeholder in the first iteration.

Where QK^T is the dot product of the query and key vectors, scaled by a factor of $(\sqrt{d_k})^{-1}$ to reduce the saturation effect of the softmax function for large values of d_k. The result of the softmax (equivalent to the Equation 3.6 presented in the previous section) is used to weight the values (Equation 3.5) through the dot product. By avoiding recurrence, the sequential nature of the operations disappears and enables the computation of all vectors in parallel as matrices.

Multi-head attention further modifies attention mechanisms by making multiple projections of the input, $H_h = A(Q, K)$ (*heads*). In this way, each head can focus on different features at different positions in the input sequence by transforming it into different representation subspaces:

$$\text{MultiHead}(Q, K, V) = [H_1, ..., H_{m_H}]\, W_H \tag{3.9}$$

$$H_h = \text{Attention}(QW_Q^{(h)}, KW_K^{(h)}, VW_V^{(h)}) \tag{3.10}$$

Where $W_K^{(h)}, W_Q^{(h)} \in \mathbb{R}^{d_{model} \times d_{attn}}$, and $W_V^{(h)} \in \mathbb{R}^{d_{model} \times d_V}$ are head specific weights for keys, queries and values, and $W_H \in \mathbb{R}^{(m_h \cdot d_V) \times d_{model}}$ linearly combines outputs concatenated from all heads H_h.

The results of all attention heads are concatenated and projected into a final dimension d_v, equivalent to that used in single-head attention.

3.3.3 Temporal Fusion Transformer

Temporal Fusion Transformer (TFT) [75] introduced several novelties that make this approach appropriate to work with time series, instead of simple sequences of data. In particular, TFT distinguishes between different types of inputs, including contextual information; is able to make multi-step predictions; and optimizes the performance of the process by applying specific variable selection for all types of inputs. In the following, we review the main characteristics of this architecture, and discuss its suitability for our project.

Temporal Fusion Transformer tackles the problem of multi-horizon forecasting problem, which is defined as the prediction of variables of interest at multiple future time steps, similar to the MIMO strategy discussed in Section 3.1.1. For this purpose, the architecture of TFT combines recurrent layers to learn temporal relations between variables at different scales, and self-attention layers to account for long-term dependencies. Moreover, different types of variables are processed separately, in contrast with previous works that face this task like deep neural networks with attention mechanisms. These approaches fail to consider the different types of inputs with decisions such as considering all exogenous inputs known into the future, or treat static variables as sequential variables by replicating their values on each example. These decisions have an impact on the computation cost and the effectiveness of the resulting model.

The Figure 3.10 describes the forecast process. In TFT, each example $i \in I$ is characterized by the time-dependent input features and the target features for each time step $t \in [0, T_i]$, and a set of static covariates. The time-dependent input features, $\boldsymbol{\chi}_{i,t} = [\boldsymbol{z}_{i,t}^T, \boldsymbol{x}_{i,t}^T]^T \in \mathbb{R}^{m_\chi}$ are divided in observed inputs, $\boldsymbol{z}_{i,t}^T \in \mathbb{R}^{m_z}$, which are unknown in advance, and known inputs, $\boldsymbol{x}_{i,t}^T \in \mathbb{R}^{m_x}$, whose value is known for the forecast time period. The scalar targets, $y_{i,t} \in \mathbb{R}$ describe the past (known) values of the variable to be predicted. Finally, the set of static covariates $\boldsymbol{s}_i \in \mathbb{R}^{m_s}$ are time-independent and constant for all time steps.

Then, for each example i, TFT outputs forecasts for τ_{max} time-steps, based on all past information that is incorporated within a finite look-back window k (including the forecast start

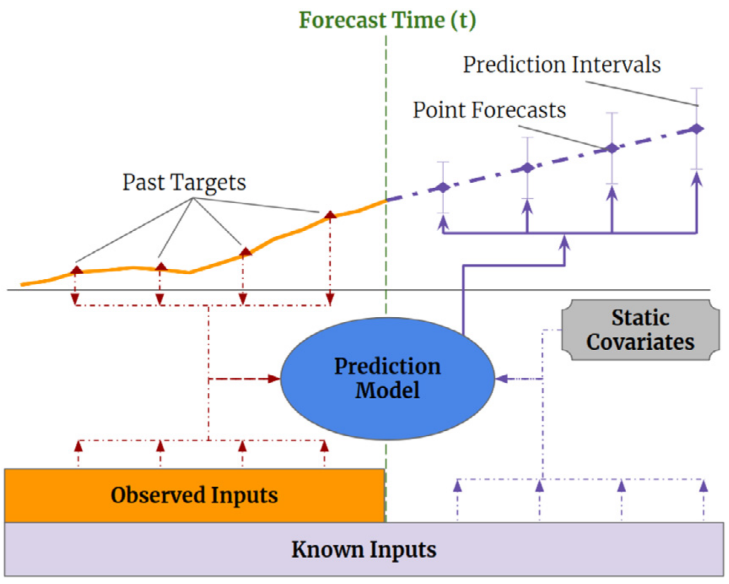

Figure 3.10: Diagram of the TFT approach to multi-horizon forecast.

time t): $y_{i,t-k:t} = \{y_{i,t-k}, ..., y_{i,t}\}$, and past and future inputs across the entire range: $\boldsymbol{x}_{i,t-k:t+\tau} = \{\boldsymbol{x}_{i,t-k}, ..., \boldsymbol{x}_{i,t}, ..., \boldsymbol{x}_{i,t+\tau}\}$.

The architecture of TFT is represented in Figure 3.11. In the following, we briefly review its main components and their roles.

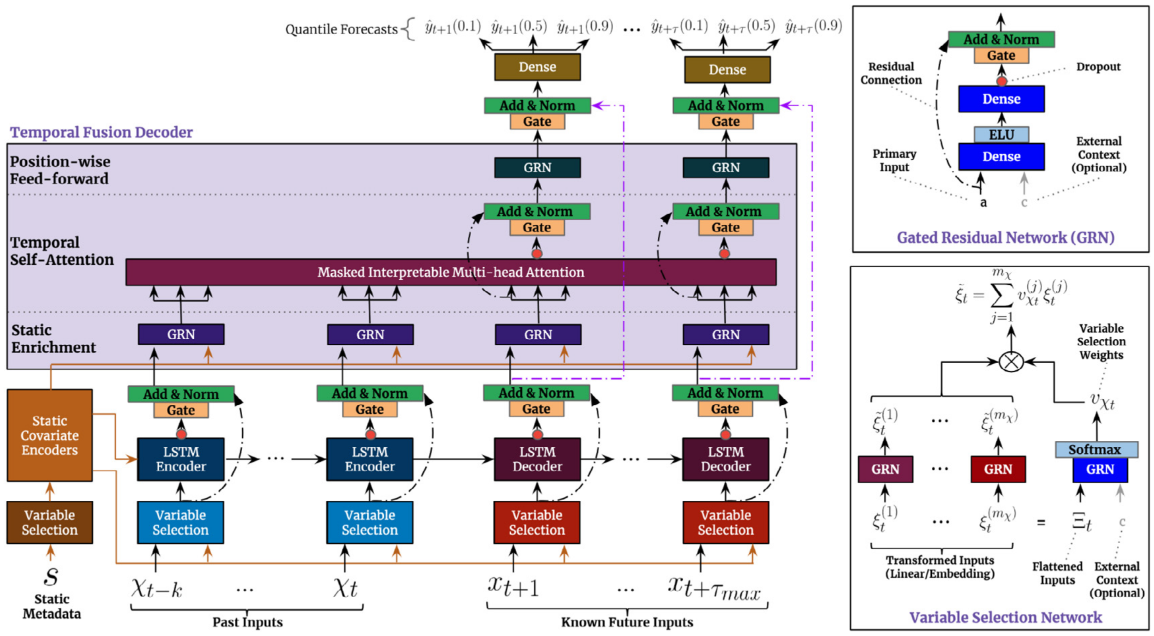

Figure 3.11: Structure of Temporal Fusion Transformer architecture [74]

Gating mechanisms. Gating mechanisms allow the model to decide whether each exogenous variable is relevant, and the required complexity of the relationship between each variable and the target, that is, if it should be modelled whether as a linear or as a non-linear relationship. In particular, TFT implements Gated Residual Networks (GRN), which apply non-linear processing when needed, and fall back to linear processing otherwise. As seen in the top right corner of the Figure 3.11, a GRN first processes an input **a** and an optional encoded context vector **c** through two layers, with an intermediate ELU (Exponential Linear Unit) layer as activation function. ELU acts as the identity function if the processed input after the first layer is $\ll 0$, and as a linear layer (with a constant output) if it is $\gg 0$.

Then, a GLU (Gated Linear Unit) is used as a gating mechanism to regulate the contribution of the result through the residual connection, or even skip it altogether. Finally, the result is normalized by means of a normalization layer.

Variable selection networks. TFT applies instance-wise variable selection on both static covariates and time dependent covariates, in order to ignore irrelevant or noisy inputs and pay attention to significant inputs only. To this end, specific variable selection networks are defined for static metadata, past inputs and future inputs. These networks are represented in the bottom right corner of the Figure 3.11. First, categorical and continuous variables are transformed using entity embeddings and linear transformations, respectively. Then, each variable at a time t is passed to a GRN, and combined with the result of operating the concatenation of all past inputs at that time t with the context vector c_s through another GRN.

Static covariate encoders. TFT integrates information from static metadata using separate GRN encoders to produce four different context vectors: temporal variable selection, c_s; local processing of temporal features, c_c, c_h; and enriching of temporal features with static information, c_e. As shown in the Figure 3.11, static covariates are encoded using LSTM networks, although it is noted that other approaches can be implemented.

Interpretable multi-head attention. TFT employs a self-attention mechanism to learn long-term relationships across different time-steps. In general, attention mechanisms scale values $\boldsymbol{V} \in \mathbb{R}^{N \times d_V}$ based on relationships between keys $\boldsymbol{K} \in \mathbb{R}^{N \times d_{\text{attn}}}$ and queries $\boldsymbol{Q} \in \mathbb{R}^{N \times d_{\text{attn}}}$:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = A(\boldsymbol{Q}, \boldsymbol{K})\boldsymbol{V} \tag{3.11}$$

Multi-head attention employs different heads for different representation subspaces:

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = [\boldsymbol{H}_1, ..., \boldsymbol{H}_{m_H}]\, \boldsymbol{W}_H \tag{3.12}$$

$$H_h = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_Q^{(h)}, \boldsymbol{K}\boldsymbol{W}_K^{(h)}, \boldsymbol{V}\boldsymbol{W}_V^{(h)}) \tag{3.13}$$

Where $\boldsymbol{W}_K^{(h)}, W_Q^{(h)} \in \mathbb{R}^{d_{model} \times d_{attn}}$, and $\boldsymbol{W}_V^{(h)} \in \mathbb{R}^{d_{model} \times d_V}$ are head specific weights for keys, queries and values, and $\boldsymbol{W}_H \in \mathbb{R}^{(m_h \cdot d_V) \times d_{model}}$ linearly combines outputs concatenated from all heads $\boldsymbol{H}_h$.

Temporal fusion decoder. The Temporal Fusion Decoder (depicted in purple in the Figure 3.11) is composed of three types of layers with specific roles. First, the input sequence is enriched with positional encoding using a sequence to sequence model (such as LSTM) to improve significant points detection and provide a better interpretation of the context of each element. Static covariates are also incorporated to the sequence. The static covariates are also incorporated by means of a GRN and the context vector c_e. The resulting sequence is applied self-attention at each forecast time. Finally, an additional GRN layer applies additional non-linear processing to the outputs of the attention layer.

Part II

Data workflow

Chapter 4

Flights information system

The automation of information management has motivated significant efforts to standardize data processes and data representation, both conceptually and practically. Conceptual models and ontologies and have a relevant role in such efforts.

4.1 Background

The automation of information management has motivated significant efforts to standardize data processes and data representation, both conceptually and practically. Conceptual models and ontologies and have a relevant role in such efforts.

4.1.1 Information exchange models

Prior to the ongoing initiatives aimed at building a unified and comprehensive perspective across all stakeholders in Air Traffic Management, automation efforts were undertaken separately by each organization, often limited to their individual operational domains. In consequence, different exchange models where developed separately according to the needs of each business area. These exchange models define an information model (usually as UML models), and an information schema based on XML. Below we list some of the most relevant exchange models in ATM:

- Aeronautical Information Exchange Model (AIXM) [36] is a global specification for the encoding and the distribution of digital aeronautical information. To this end, AIXM defines a logical data model and an XML schema: Aeronautical Information Conceptual Model (AICM) + XML Exchange Model.

- Flight Information Exchange Model (FIXM) [41] is an exchange model based on XML capturing Flight and Flow information.

- ICAO Meteorological Information Exchange Model (IWXXM) [65] is an information model designed for the operational exchange of meteorological information for ATM purposes. This model includes all of the information required by ICAO Annex 3 [61], including reports such as METAR/SPECI, TAF, SIGMET, AIRMET, Tropical Cyclone Advisory, Volcanic Ash Advisory and Space Weather Advisory.

- Weather Information Exchange Model (WXXM) is developed by EUROCONTROL and FAA, and extends IWXXM with further weather information not covered by the latter.

However, the different points of view and the separate developments can lead into discrepancies between information models in how different concepts are related, or even in the meaning of the same concept. In the current context of integrated information, ATM Information Reference Model (AIRM) has become the most significant effort to reconcile these interpretations and provide an holistic information system for the ATM.

4.1.2 ATM Information Reference Model (AIRM)

ATM Information Reference Model (AIRM) [35] consolidates a global standard vocabulary for ATM, based on the ICAO Annexes and Documents, and different information exchange models used at a global scale. In order to ease the adaptation of idiosyncratic exchange models to this standardized vocabulary, several mappings have been established between the AIRM and the models defined in these exchange models. In this way, AIRM enables semantic interoperability at a global scale by ensuring that the concepts used by all agents involved in ATM have an unique, unambiguous meaning.

The scope of AIRM is ATM-wide, that is, it encompasses all the concepts related to air traffic management on any of its areas: airspace and base (ground) infrastructures, air traffic operations and surveillance, flight information and meteorology, among others. AIRM comprises three components, which are described using UML (Unified Modeling Language):

- A *contextual model*, providing the terminology, abbreviations, standards and classification scheme that characterize the concepts behind ATM operations. This component gathers business terms from international organizations, such as ICAO, ISO and WMO.
- A *conceptual model*, which serves as a reference model for operational experts.
- A *logical model*, which serves as a reference model for technical agents and developers.

However, AIRM does not provide with an information exchange model that defines how information is actually encoded, so it still needs to rely on the exchange models described before for its implementation on specific domains, and is the reason why individual exchange models still are relevant today.

4.2 Conceptual model

Based on the AIRM, we have designed a specific conceptual model that represents the concepts relevant to this Thesis. However, in prevision of future extensions, and to homologate our model to the existing approaches, we have mapped the concepts in our model to those existing in AIRM (and, by extension, to the specific exchange models applicable to each of the ATM areas).

Our conceptual model is shown in Figure 4.1. In gray, we have indicated the entities that will be included as part of our future work, but are currently not defined in the conceptual model since they are not used later in the data workflow. Since our focus are flown 4D trajectories, the Flight entity is the central concept in our model. We define a flight as the trajectory followed by an aircraft between the off-block time, where it starts moving at the departure airport, and the on-block time at the destination airport; that is, when the aircraft is parked after landing and stops moving at the terminal gate, with no stops in between. This notion of Flight is analogous to the concept of *flight leg* that is common in ATM, since commercial flights can make scales (short duration stops) at intermediate airports. In these cases, a single flight comprises several legs, each with its own arrival and departure times, flight plan, or even aircraft and operating carrier. Considering these two concepts introduced additional layers of complexity in our model, so we decided to make it simpler by focusing on legs only, and renaming them as Flights (instead of Flight legs) for clarity. Therefore, a multi-leg flight would be represented as multiple, independent flights in our model.

The involved concepts are specified in Tables 4.1-4.8. A succinct definition of each concept and its descriptive attributes is provided in each table, based on the corresponding AIRM concept.

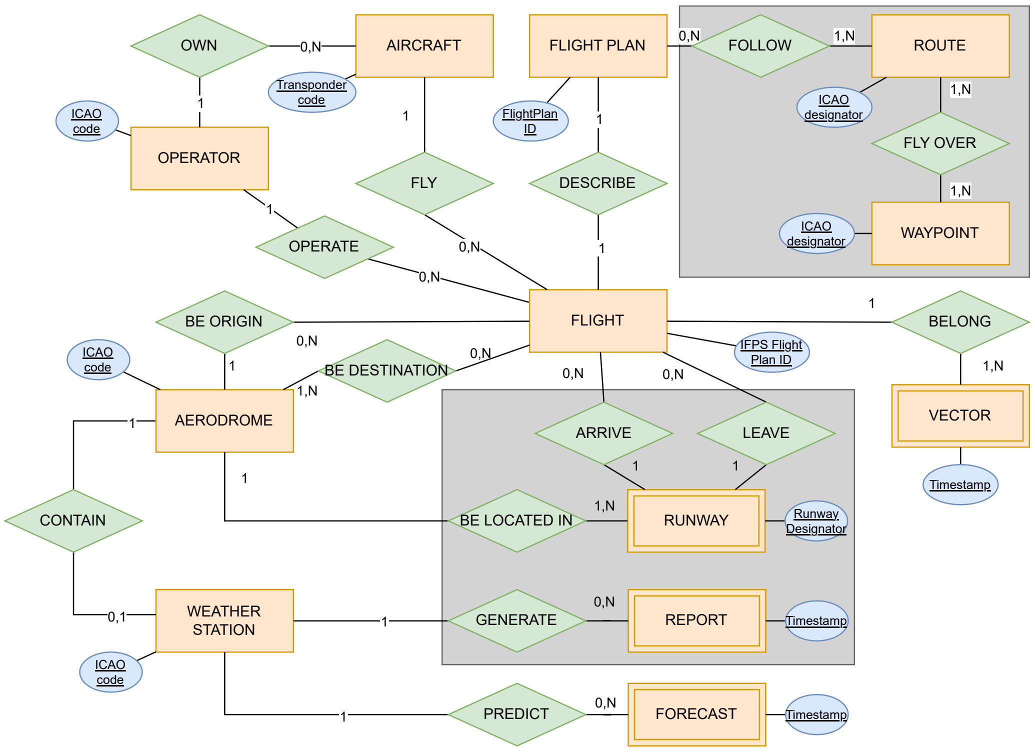

Figure 4.1: Conceptual model

AERODROME	AIRM: Aerodrome	
A defined area on land or water (including any buildings, installations and equipment) intended to be used either wholly or in part for the arrival, departure and surface movement of Aircraft.		
Attribute	AIRM concept	Definition
ICAO code	designator	Four-letter identifier assigned by ICAO.
IATA code	IATAdesignator	Three-letter identifier assigned by IATA.
Latitude	aerodromeReferencePoint.lat	The latitude coordinate in WGS84.
Longitude	aerodromeReferencePoint.lon	The longitude coordinate in WGS84.
Altitude	fieldElevation	The vertical distance in feet above Mean Sea Level (MSL) of the highest point of the Landing area.
City	servedCity	The city primarily served by the Aerodrome.
Name	name	The primary official name of an Aerodrome.
Type	type	The type of Aerodrome (AD=Airport, HP=Heliport, LS=Landing site, AH=Airport and heliport).
Transition altitude	transitionAltitude	The altitude in feet at or below which the vertical position of an Aircraft is controlled by reference to altitudes.
Transition level	transitionLevel	The lowest flight level available for use above the transition altitude.

Table 4.1: AERODROME entity description

AIRCRAFT	AIRM: Aircraft	
Attribute	AIRM concept	Definition
ICAO24 Code	aircraftAddress	The unique address (a combination of twenty-four bits) of the aircraft for the purpose of air-ground communications, navigation and surveillance.
Aircraft model	aircraftMakeModelSeries	Aircraft model, make and series information as provided by the manufacturer.
Type	aircraftType	The Aircraft type (LANDPLANE, SEAPLANE, AMPHIBIAN, HELICOPTER, etc.).
Wake turbulence	wakeTurbulenceCategory	A classification of aircraft based on the magnitude of wake turbulence they are expected to generate (LIGHT, MEDIUM, HEAVY, and SUPER).

Table 4.2: AIRCRAFT entity description.

FLIGHT	AIRM: Flight	
The operation of an Aircraft which, in the case of a manned Aircraft, takes place between the time any person boards the Aircraft with the intention of Flight until such time as all such persons have disembarked.		
Attribute	AIRM concept	Definition
callsign	aircraftIdentification. flightNumber	A combination of letters and/or numbers which is used to identify the flight.
flightDate	dateOfDeparture	The date of flight departure.
takeOffTime	airborneTime	The time at which the aircraft leaves the ground and becomes airborne.
landingTime	landingTime	The time at which the aircraft touches down on the runway.
actualTimeOfArrival	actualArrivalTime	The time at which the Aircraft reaches its final parking position and the brakes are set.
flightRules	flightRules	The regulations that govern all aspects of flight operations (VFR, IFR).
codeFlightType	flightType	The classification of the flight, such as scheduled passenger, cargo, general aviation, etc.
ssr	ssrCode	The four-digit Secondary Surveillance Radar (SSR) code assigned to the Aircraft for identification by radar.
status	flightStatus	The operational state of the Flight. (Scheduled, Departed, En-route, Landed, Cancelled)

Table 4.3: FLIGHT entity description.

FLIGHT PLAN	AIRM: Flight (f), FlightPlan (fp)	
Specified information provided to air traffic services units, relative to an intended Flight or portion of a Flight of an Aircraft.		
Attribute	AIRM concept	Definition
ifplId		A unique Flight plan identifier allocated by the IFPS.
flightDate	f.dateOfDeparture	The date of Flight departure.
totalEstimatedTime	f.totalEstimatedElapsedTime	The estimated time required from take-off to arrive at the destination aerodrome.
estimatedOffBlockTime		The estimated time at which the aircraft will commence movement from its parking position with the intention of departing.
estimatedTakeOffTime		The estimated time at which the aircraft will become airborne, calculated as the sum of the estimated off-block time and taxi-out time.
estimatedTimeOfArrival		The time at which it is estimated that the aircraft will arrive at the destination aerodrome.
estimatedLandingTime		The estimated time at which the aircraft will make physical contact with the runway (touchdown).
Flight plan version	fp.operatorFlightPlanVersion	The version of the flight plan provided by the operator.

Table 4.4: FLIGHT PLAN entity description.

FORECAST	AIRM: Forecast (f), SurfaceForecastReport (sfr), MeteorologicalForecast (mf), WeatherCondition (wc), Wind, AirspaceCondition (ac)	
A statement of expected meteorological conditions for a specified time or period, and for a specified area or portion of airspace.		
Attribute	AIRM concept	Definition
timestamp	f.forecastTime	The time for which the weather conditions are forecast.
temperature	wc.temperature	The forecast air temperature at the aerodrome surface, in Celsius degrees.
dewPointTemperature	wc.dewPointTemperature	The forecast dew point temperature at the aerodrome surface, in Celsius degrees.
windDirection	Wind.windDirection	The forecast direction from which the wind is blowing, in degrees.
windSpeed	Wind.windSpeed	The forecast wind speed at the surface, in knots.
windGustSpeed	Wind.windGustSpeed	The forecast maximum gust wind speed, in knots.
visibility	ac.visibility	The forecast horizontal visibility, in meters or statute miles.
weatherPhenomena	wc.presentWeather	Forecast weather phenomena. (rain, fog, snow, thunderstorms).
cloudAmount	wc.cloudAmount	The forecast total cloud coverage. (SKC, FEW, SCT, BKN, OVC).
cloudBase	wc.cloudBase	The height of the base of the lowest significant cloud layer, in feet AGL.
cloudType	wc.cloudType	The type of cloud layer. (Cumulonimbus, Towering Cumulus).
forecastType	sfr.type	The type of forecast issued. (TAF, trend, probabilistic).
reportValidityStart	mf.validPeriod.start	Start of the forecast validity period.
reportValidityEnd	mf.validPeriod.end	End of the forecast validity period.
forecastProbability	mf.probability	Probability of occurrence, if applicable. (30, 40).

Table 4.5: FORECAST entity description.

OPERATOR	AIRM: Aircraft Operator	
Any air transport enterprise (airline) offering or operating a Scheduled international air service		
Attribute	AIRM concept	Definition
ICAO code	designatorICAO	The identifier of the AircraftOperator as assigned by ICAO.
IATA code	designatorIATA	The identifier of the AircraftOperator as assigned by IATA.
Name	name	The full official name of the organization.
Callsign code	designator	A distinguishing label, term, abbreviation or acronym used to identify an organization, authority, agency or unit.

Table 4.6: OPERATOR entity description

VECTOR	AIRM: TrajectoryPoint	
A four dimensional Point used in defining the Flight path of an Aircraft, enriched with additional values indicating the state of the aircraft.		
ICAO24 Code	icao24	ICAO 24-bit ID of the transponder (hexadecimal).
Ground speed	velocity	Ground speed en m/s; velocidad sobre el suelo del avión en el momento del estado.
Track	true_track	Bearing in decimal degrees clockwise from north.
Vertical rate	vertical_rate	Ascent or descent rate, in meters per second..
Latitude	lat	WGS-84 latitude in decimal degrees.
Longitude	lon	WGS-84 longitude in decimal degrees.
Barometric altitude	baro_altitude	Barometric altitude in feet.
Geometric altitude	geo_altitude	Geometric altitude (over WSG84 ellipsoid) in meters.
On ground flag	on_ground	Boolean value which indicates if the position was retrieved from a surface position report.
Squawk (SSR)	squawk	The 4-number transponder code (squawk) assigned by ATC.
Callsign	callsign	Callsign of the flight.
Time position	time_position	Unix timestamp (seconds) for the last position update.
Last contact	last_contact	Unix timestamp (seconds) for the last update in general.

Table 4.7: VECTOR entity description with respect to the State Vector concept from OpenSky.

WEATHER STATION	AIRM: WeatherInformationSource (wis)	
Attribute	AIRM concept	Definition
Station ID	wis.identifier	Unique identifier of the station assigned by ICAO
Latitude	Point.latitude	WGS-84 latitude in decimal degrees.
Longitude	Point.longitude	WGS-84 longitude in decimal degrees.
Elevation	Point.altitude	Vertical distance to the MSL in meters.
StationType	stationType	The type of the station. (AWOS, ASOS, METAR, etc).

Table 4.8: WEATHER STATION entity description.

4.3 Data sources

In this section we review the data sources that feed the designed data workflow. Given the information requirements defined in our conceptual model, we have to gather data describing the flights, the flown trajectories, the airports and the operators, and the weather conditions at the airports. In particular, we have chosen open data sources when possible, although the lack of open sources for some of the data categories or the quality requirements of our project have determined the use of different data sources, either restricted or paid.

4.3.1 Surveillance data

The purpose of surveillance is to monitor and describe the state of the airspace (or of the ground facilities within an aerodrome) to enforce the control of the operations taking place in an area of control. This kind of situational awareness requires to know the position of all vehicles within the area, in order to avoid risk situations and ensure the safety of the operations. This task is ultimately conducted by humans (air traffic controllers and pilots), but nowadays surveillance relies on many automatic systems to support human agents to accomplish its goals.

Surveillance systems generate data as a result of the monitoring of the air and ground operations. Surveillance data is therefore characterized by position and, in general, by the identification of the sender of the data. These data depend on the nature of the system providing them: the position can be expressed in terms of its coordinates (latitude, longitude, and altitude), or be relative to the receiver (such is the case of the secondary radar). The same applies for the identification data: a vehicle emitting surveillance data can be identified using different criteria, being the transponder code of an ADS-B emitter or the callsign of the flight are two common examples of aircraft identifiers. In addition, surveillance data can include other data, such as speed, flight level, current intent, or any other datum that may be valuable to characterize the current or future states of the airspace.

Several surveillance systems and technologies are combined in order to provide a detailed picture of the airspace (primary and secondary radars, multilateration, ADS-B, ADS-C), but in recent years ADS-B has become the keystone of surveillance after it became mandatory in European and North American airspaces. In the next section, we provide a succinct description of this technology.

4.3.1.1 ADS-B

Automatic Dependant Surveillance – Broadcast (ADS-B) [90] is a cooperative, GPS-based surveillance technology that has consolidated as a standard since 2020 in the main airspaces in the world. ADS-B plays a similar role to the traditional SSR (Secondary Surveillance Radar) for surveillance purposes, although ADS-B provides a much higher update rate than radar (minimum rate of 2s vs 10-15s), more precise positioning and better coverage with a lower installation cost. The system transmits information autonomous and periodically without involving the pilot or operator. This information is derived directly from the on-board systems such as GNSS (Global Navigation Satellite System) or pressure altimeter.

ADS-B relies on the Mode S transponder to broadcast ADS-B data using the 1090 MHz band. Mode S allows for specific interrogations to aircraft, although ADS-B messages are automatically and continuously broadcast without the need of external interrogation. There are other variants of ADS-B, such as ADS-B UAT (Universal Access Transceiver), which operates in the 978 MHz band and is typically used by aircraft flying below 18,000 feet in the United States under FAA regulations.

ADS-B can function in two modes of operation:

- ADS-B Out: ADS-B Out is responsible for gathering information from on-board systems and transmitting ADS-B messages. Currently, this is the most commonly implemented system in aircraft.

- ADS-B In: Aircraft equipped with ADS-B In can receive ADS-B messages broadcast by other aircraft, which enhances the pilot situational awareness. However, this component is not mandatory, and most aircraft lack ADS-B In capability, relying instead on ATC for traffic information.

ADS-B messages on 1090ES contain 112 bits, which include aircraft identification, downlink metadata and a variable payload depending on the type of the message. ADS-B messages belong to one of 8 different types [80], each providing with identification data, position data, and so on. These message types can be summarized as shown in Table 4.9 [106]. Thus, multiple messages must be captured in order to fully describe the state of the aircraft. Each message can carry additional metadata, such as an estimation of message quality indicators (NIC, NACp) in messages with type code 23-27. The physical structure of ADS-B is shown in Figure 4.2. Each sensor can capture ADS-B messages broadcast by aircraft located within the range of ADS-B transmissions (in blue in the figure): the theoretical range is 320 NM, although in practice the range can be substantially reduced. If the aircraft does not have any receiver in range, all of its ADS-B messages will be lost. On the contrary, if the aircraft is in range of more than one receiver, the same message could be received twice. The travel time to both sensors can be different, and the two copies will be assigned different timestamps. This data duplication problem is further discussed in the next section.

Code	Content description
1-4	Aircraft identification
5-8	Surface position
9-18	Airborne position (with barometric altitude)
19	Airborne velocities
20-22	Airborne position (with GPS altitude)
28	Aircraft status
29	Target state and status information
31	Aircraft operation status
23–27	Reserved for local applications or future extensions

Table 4.9: Description of the types of ADS-B messages.

4.3.1.1.1 ADS-B problems The ADS-B technology has some limitations. Due to its nature, ADS-B requires at least one receiver to be situated within range of the broadcast transmission to be effective. Moreover, the communication is purely unidirectional, so there is no form of

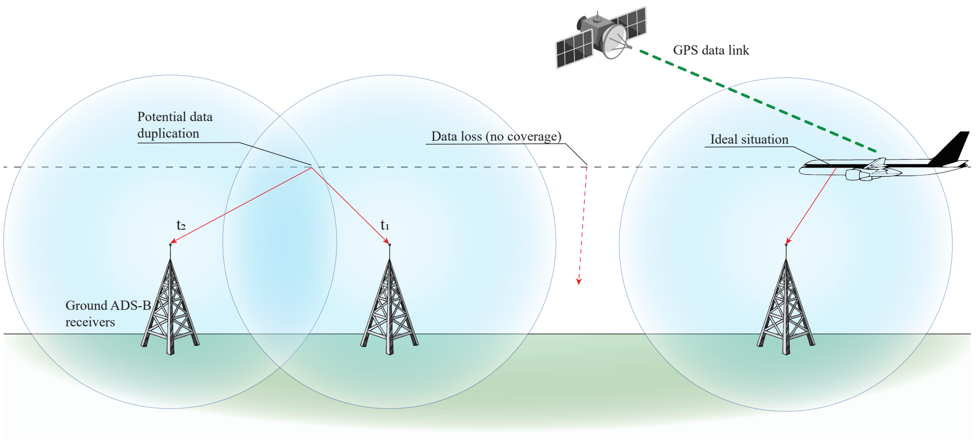

Figure 4.2: ADS-B physical structure. A receiver can only capture ADS-B messages if the emitting aircraft is in range. If the aircraft is in range in more than one receiver, the same message can be captured by each of them, resulting in duplicated data.

persistent link or reception acknowledgment and the sender has no way to determine whether or not the messages are being received. Any message that is not captured by any receiver is lost, since the aircraft does not keep a record of the sent messages. ADS-B Out has a theoretical maximum range of 600 Km (320 NM). In practice, its range is reduced to 450 Km (250 NM), and can be even lower depending on the working conditions [2, 115], the terrain configuration or the Earth's curvature. For example, weather has a significant impact in the quality of the ADS-B transmissions. Moreover, signal degradation is also relevant [110], and the data might result unusable even if a message reaches a receiver. Congestion has also been found to heavily influence the quality of ADS-B data. Alhazbi et al. [2] determined that the availability of receivers in range does not ensure the reception of the emitted messages even at low distances due to packet loss, particularly in conditions of congestion (with multiple aircraft emitting ADS-B messages at the same time).

As indicated before, these factors may cause data losses due to packages being corrupted or not reaching any receiver. When it happens, the theoretical update rate of 1Hz might be compromised, and therefore affect the continuity of the signal from the aircraft. These losses of data can form gaps in the signal that can be punctual or sustained. Given the frequency of ADS-B messages (specially the most critical ones: identification and position), the loss of individual messages might be assumable, but losing several seconds or more of data might be risky in several scenarios. This factor can be alleviated by a dense network of receivers in order to maximize the coverage of the airspace. However, this leads us into a different problem. The complementary situation might also be a problem: the same message can be received by multiple antennae within the emission range. All of their payload would be identical (if the reception was successful), except for their timestamps. Note that the timestamp of each message is set at reception time by the receiving device. This situation must be accounted for when merging data from different receivers; otherwise, the resulting data would tell a story of an aircraft flying over the exact same point twice within a short time span.

There exist other potential problems derived from the dependency of ADS-B on other systems, such as Global Positioning System (GPS) (or other navigation system), Global Navigation

Satellite System (GNSS), or the aircraft on-board systems from which the transponder gathers the necessary data. Ali et al. [6] describe some of these problems, like the complete absence of position data when there is no GPS coverage, the GPS errors or the lack of precision of the GPS. The transmission between the physical interfaces of these systems is also prone to errors. These situations derive in ADS-B transmitting faulty or incorrect data. Factors like the transponder make-model or the aircraft make-model [5], or the heterogeneity of the receivers' hardware and software, also influence the probability of errors in ADS-B data.

ADS-B messages include both geometric and barometric altitudes. The geometric altitude is obtained from GPS, and is subject to the factors described above. The barometric altitude corresponds to the altitude determined by the altimeter according to a configuration set by the pilots, and thus prone to errors and imprecision [7]. These data must be utilized with care, since it can lead into different or even contradictory conclusions. For instance, in [110] it is observed that the geometric altitude overestimates the barometric altitude, while in [7] they observe the opposite. Remind that barometric altitude expresses the altitude with respect to a particular pressure level (specially in the cruise phase), and may not be accurate for the conditions in which the aircraft flights in every moment of the flight, nor the value indicated by the GPS. It is more a conventional altitude agreed by pilots flying the same area.

Finally, ADS-B is vulnerable to intentional attacks due to the lack of cryptography protection or authentication. This allows an attacker to send fake or manipulated ADS-B data to be captured by the receivers and introduced into the surveillance system [93]. This is inherent to the technology, and detecting or avoiding such situations is beyond the scope of this Thesis.

Several metrics have been defined in the literature to evaluate the quality of ADS-B data, according to the characteristics and the theoretical capabilities of this technology. A summary of the metrics defined and the case of application used in each study is provided in 4.10. In the following, we define the most prominent metrics:

- Missing payload: Some of the payload of ADS-B messages can be corrupted or lost during transmission, causing the data to not be available when processed at the receiver.

- Latency: Latency corresponds with the total transmission time between the reading generation by the on-board systems and the effective processing of the data by the receiver. This time includes the generation of ADS-B messages by the aircraft transponder, the travel time from the aircraft to the receiver and the processing the receiver must do to extract the data.

- Refresh/update rate: The update rate corresponds with the frequency the ADS-B messages are generated and sent, if analyzing the emitter, or the frequency the ADS-B messages are captured by the receiver. Since the messages can be lost during transmission by multiple causes, the "perceived" update rate at the receiver is always less or equal than the "initial" update rate at the aircraft.

- Position accuracy: ADS-B gets position information from the GPS system from the aircraft. Therefore, position data is subject to errors and precision problems originated by these systems.

- Data duplication: The architecture of ADS-B makes it possible that the same message is captured twice by different receivers, or even by the same receiver if the signal is divided during transmission and arrives through different paths (multipath propagation).

- Package integrity and precision: ADS-B includes indicators (NIC and NACp) that describe the physical integrity and precision of the data, respectively, which sometimes is degraded under acceptable levels (around 3% of the messages) [110].

- Incorrect position/data jumps: This situation occurs when one or more data points deviate significantly from the adjacent points, specially in position data.

- Packet loss: ADS-B messages may not be received successfully by any receiver, leading into a complete loss of the data. There can be multiple causes for this:

  - Lack of coverage: There is no receivers within the distance of effective communication, or this distance is reduced due to weather conditions or geographical accidents.
  - Distance: Even within the practical distance, it is possible that ADS-B messages are lost with increased probability along the distance.
  - Congestion: Specially in high density areas, like near the airports, the transmission medium or the receiver can be congested and not be able to cope with the data load.

Work	Year	Case study	Period	Sample size
Ali et al. [6]	2014	London City Airport, UK	2011 Jan, 10	26 flights
Ali et al. [5]	2016	London City Airport, UK	2011 Jan, 10; 2011 May, 23	57 flights
Ali et al. [3]	2017	London City Airport, UK	2011 Jan, 10	26 flights
Tabassum et al. [110]	2017	Grand Forks Int., USA	2015 Jun, 15-21	5.2M messages
Verbraak et al. [115]	2017	Receiver in Delft, NE	2015 Sep, 21-27; 2016 Jan, 11-15	Not described
Alhazbi et al. [2]	2019	All OpenSky Network	2015 Apr, 21	304,131 messages

Work	Missing	Latency	Update	Accuracy	Integrity	Duplication	Packet loss	Data jump
Ali et al. [6]	-	x	x	x	x	x	-	-
Ali et al. [5]	x	-	x	x	x	x	-	-
Ali et al. [3]	-	-	x	-	-	-	-	-
Tabassum et al. [110]	x	-	x	x	x	x	x	x
Verbraak et al. [115]	x	x	x	x	x	x	x	-
Alhazbi et al. [2]	x	-	x	-	x	x	x	-

Table 4.10: Previous studies on ADS-B data characteristics and types of analysis performed.

4.3.1.2 OpenSky

OpenSky [94] is an open data initiative launched in 2012. It is based on a participatory sensor network to capture real-time air traffic surveillance data. OpenSky aims at improving the security, reliability and efficiency of the air space usage by providing open access of real-world air traffic control data (mainly ADS-B data) to support the research in this area. Currently, it gathers data from around 7,000 registered receivers (October, 2024) covering most of the airspace on Europe and the U.S.A., as well as parts of South-America and South-Asia. A coarse description of the network's global coverage is shown in Figure 4.3. The OpenSky Network also organizes every year the OpenSky Symposium[1], where aviation specialists and practitioners can communicate their on-going research and developments on the field, with a focus on works using surveillance data, and in particular ADS-B data.

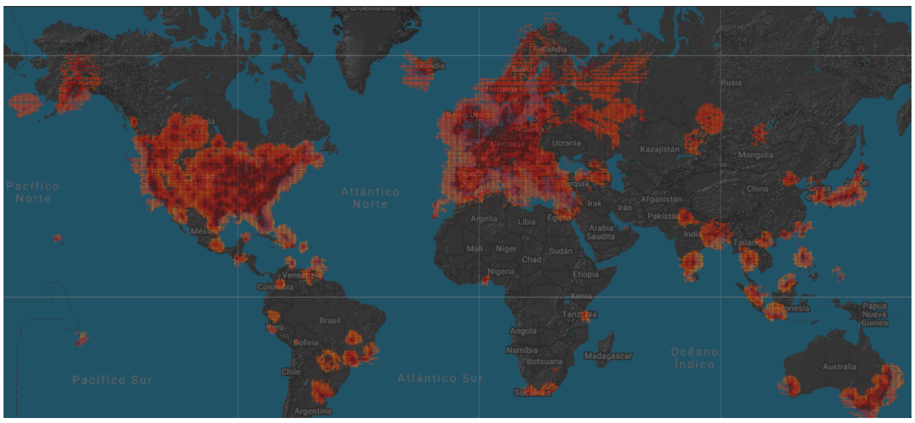

Figure 4.3: Global OpenSky Network coverage (October, 2024).

4.3.1.2.1 Schema and Data processing OpenSky does not (primarily) provide with ADS-B raw data as obtained from the ground ADS-B receiver network. Instead, ADS-B messages are decoded and processed to build state vectors. Each state vector contains the minimum information required by the current standards [59]: i) Aircraft identification: every ADS-B message informs the 24-bit ICAO code of the aircraft's transponder. In some cases, the call sign assigned by the airline is also provided. ii) Position: The aircraft emits ADS-B position messages twice per second, informing its position as latitude, longitude and altitude. iii) Velocity: Both horizontal velocities (north-south and east-west) and vertical rate. iv) Others: Some aircraft broadcast messages with additional data, such as emergency data, priority, technical capabilities or operational models.

The complete relation of the structure of the state vectors is described in Table A.1 (see Appendix A). State vectors are generated every 5 seconds: each state vector aggregates the information of all ADS-B messages received within that time (for a particular aircraft). State vectors include two timestamps: *time_ position*, that indicates the timestamp of the latest ADS-B message with position data used to construct that state vector, and *last_ contact*, which denotes the timestamp of the latest ADS-B with velocity data. If any piece of information is not updated during that time, the last value is considered valid for 15 seconds. That is, old position or velocity

[1]OpenSky Symposium: `https://symposium.opensky-network.org`

values are used for new state vectors if were received less than 15 seconds before. Once that time has elapsed, the corresponding field is left blank. If none of those values are updated for more than 15 seconds, the flight is considered discontinued and no more vectors are generated until new messages are captured.

4.3.2 Flight plan data

While surveillance data describes the actual flight during the tactical phase, in Chapter 2 we stated that flight planning takes place well before that phase and the relevant information is shared among all stakeholders in the form of flight plans. Flight plans include the relevant schedules and metadata to ensure the safe and efficient planning of flight operations, such as when and where the flight is planned to depart and arrive, the entities responsible of the flight, the capabilities of the aircraft performing the flight, etc. All of this data essential for adequately describing any flight, and thus several data sources are available to satisfy these information requirements. In this section, we describe the two data sources that we have considered for including flight plan data in our information system.

4.3.2.1 OpenSky Flights API

In addition to providing surveillance data in the form of state vectors, OpenSky Network also provides flight information derived from this data. For OpenSky, a flight is defined as a continuous sequence of vectors generated by a transponder, identified by its transponder code. The maximum time between two adjacent messages is set to 10 minutes. If this threshold is exceeded, the later message is assumed to belong to a different flight (performed by the same aircraft). Flight data contains aircraft identification information, origin and destination airports (when identifiable) and the timestamps of the first and last ADS-B messages of the flight. The origin and destination are determined using a simple heuristic: choose the closest airport in a radius of 10 Km from the first and last detected messages for that aircraft, respectively. If there are no airports within this radius, this field is left empty.

Flight information is published through the OpenSky REST API[2]. This API can be queried through many endpoints, retrieving flight data for a given time interval and one of the following criteria: i) all flights, ii) by aircraft, iii) by departure or arrival airport. Table A.2 in Appendix A shows the attributes of the response that is shared by all these endpoints.

Since OpenSky flight data is derived from surveillance data, it is prone to provide missing or incorrect information if surveillance data from the complete trajectory is not available. It is also insufficient to identify relevant information beyond the departure and arrival airports, such as the scheduled times for those events, among others. Therefore, this data source does not satisfy our information requirements and does not ensure the required data quality to carry out our research, so it will not be used in this Thesis.

4.3.2.2 Network Manager

The Network Manager[3] (NM) is a platform operated by EUROCONTROL that provides access to flight plan data submitted by airlines to the Network Manager Operations Centre (NMOC) through the Initial Flight Plan Processing System (IFPS). Airlines must file flight plans for their

[2]`https://openskynetwork.github.io/opensky-api/index.html`

[3]`https://www.eurocontrol.int/network-operations`

scheduled flights prior to the flight start to enable strategic decisions related to route planning, resource allocation and safety assessment. These flight plans are filed up to 5-7 days before the actual flight starts (i.e. the estimated off-block time, or EOBT), and include scheduling information such as the planned departure and arrival times, the airline that operates the flight, the origin and destination airports, and so on. Between the filing of the flight plan and the departure, these flight plans can be amended to accommodate possible changes in the schedule or the availability of resources. Twenty hours before EOBT, flight plan data is transferred to the ETFMS. ETFMS data describe the pre-tactical and execution phases of a flight, and goes beyond schedule information by providing data describing the actual flight, such as the effective times of the takeoff or landing, or the events taking place during the flight. Flight data for commercial flights is created mainly by airlines and pilots, since they are responsible of submitting the new flight plans, and the modifications to existing flight plans, to the IFPS and ETFMS systems.

NM publishes these data through two closed APIs: the Flight Plan and Flight Data APIs, as a part of their B2B services.

- Flight plan data provides scheduling information, which may not reflect the actual times if the schedules were not satisfied. In that sense, flight data may be more useful, as it provides the actual times of the start and end of the flight.

- Flight data refers to the pre-tactical phase and the execution phase of the flight, once the flight plan has been submitted from IFPS to ETFMS. This means that the Flight Data API contains flight information about the flight from 24h before the takeoff, during the flight, or after the flight, if the flight plan is updated with a posteriori measurements.

The data provided by Network Manager contains a large amount of fields that describe different aspects related to the flight. For now, we are focusing on a subset of attributes valuable for constructing 4D trajectories using the flight plan schedules to facilitate the management of the data provided by this source. The list of selected attributes of the Flight Plan endpoint is defined in the Table A.7, and in the Table A.11 for the Flight Data endpoint, both tables situated in Appendix A.

4.3.3 Weather data

While weather has a significant influence on ATM operations –specially storms, lightnings and other intense phenomena–, the en-route weather conditions are implicitly taken into account in both the flight plan and the actual trajectory. If adverse conditions are expected to cause a risk situation along the route of the aircraft, its path can be modified in advance or during the flight to avoid facing these phenomena. However, the aircraft must arrive at the destination airport –unless the adverse weather conditions are so extreme that the flight needs to be rerouted to another airport in order to land in safer conditions, so it is necessary to know the actual and forecast weather conditions to adapt the approaching maneuvers and perform a safe landing operation. Therefore, our conceptual model incorporates information on aerodrome weather conditions.

The actual and expected weather conditions at the airports are described by TAF and ASOS systems, respectively, both expressed in METAR format. We describe these concepts in the following sections.

Field	Description	Example	Interpretation
Report type	Whether the report is METAR or SPECI	METAR	
Place	Station identifier that originated the observation	LEMD	Station ID: LEMD
Day+time	Concatenates day of the month, hours and minutes in UTC	280925Z	22nd day in the month, 16:30
Report origin	How the report was generated: manually (empty), automatically (AUTO) or does not contain data (NIL)	AUTO	AUTO
Surface-wind	Direction and speed (in meters per second) of the surface wind	24004MPS	Wind direction 240º Wind speed 4 mps
Visibility	Visibility range	0800 R12/1000U	Prevailing visibility of 800m Visibility for runway 12 of 1000m and increasing (upward)
Weather	A list of observed weather phenomena	DZ FG	Moderate drizzle and fog
Clouds	A list of observed cloud conditions	SCT010 OVC020	Scattered cloud at 300m Overcast at 600m
Temperature	Air temperature and dew point temperature	17/16	Air temperature 17ºC Dew point at 16ºC
Air-pressure	Pressure at the airport ground level	Q1018	QNH defined at 1018 HPa

Table 4.11: Description of the METAR format fields.

4.3.3.1 METAR

The Meteorological Aerodrome Report (METAR) data format is used to distribute reports generated by weather stations located at airports. This format allows to describe both actual or forecast weather conditions at the airport area, given their influence on operations taking place at the airport. METAR reports have a fixed structure defined by:

```
[METAR|SPECI] place day+time [AUTO|NIL] surface-wind visibility weather
clouds temperature air-pressure trend
```

The METAR reports encode all relevant weather information, such as the characteristics of the wind, the visibility and the precipitation, as well as the station identification and the timestamp of the observation (or forecast). The following sequence would be an example of METAR. The definition of each of the element in the report is shown in Table 4.11.

```
METAR LEMD 221630Z 24004MPS 0800 R12/1000U DZ FG
SCT010 OVC020 17/16 Q1018
```

4.3.3.2 ASOS reports

Automated Surface Observing System (ASOS) reports are automatically generated METAR reports that contain actual observations of the current weather at the airport. Routine reports are issued every 60 minutes to provide updated information about the current weather conditions, unless a sudden and significant change is observed at any moment between updates. In this case, a new report can be published to account for the new weather conditions in addition to the scheduled updates. The structure of ASOS reports are identical to that defined for METAR reports in the previous section.

4.3.3.3 TAF reports

Terminal Aerodrome Forecast (TAF) reports are manually-generated messages that describe the forecast weather conditions expected at an airport. Routine TAF reports are generated every 6 hours, and have a defined validity (forecast horizon) of 12-30 hours. TAF reports are structured as follows:

```
TAF place day+time validity general-forecast
temperature-range
[[PROB] BECMG|FM|TEMPO forecast-change]
```

The validity defines the period in which the forecast weather is applicable, unless another report replaces or amends the report. This period can last 30 hours and starts one hour after the issue time of the report. The general forecast describes the expected weather conditions structured as a METAR report, including descriptions of winds, clouds and visibility. The temperature range defines the maximum and minimum temperatures within the validity period, and the times they will occur.

TAF reports can contain a clause with a forecast change. This clause describes one or more scenarios with weather conditions that differ from the weather conditions defined in the base report. The change can be either permanent or temporary. If the weather transitions into different conditions in a short lapse of lapse, the change clause is annotated as FM. However,if the transition is gradual along a larger period of time, then it is denoted as BECMG. These changes define new weather conditions that replace those set in the general forecast, and the time (or the period along the transition will take place, if the change is gradual) they are expected to materialize.

The change can be temporary (TEMPO) and take place only during a particular time period, and return to the previous weather stated in the general forecast after it ends. The previous changes can be qualified with a measure of probability, if the change occurrence is not certain. The probability of the event is indicated as PROB30 or PROB40 (events with a higher probability are included in common TAF reports, and events with lower probability are not reported). The reports that predict a change also have the attributes time_from and time_to, which define the lifespan of the change. BECMG and FM changes will last until the end of the validity of the report, while TEMPO reports have a shorter time of occurrence.

In our data source, TAF reports are already decoded using the pytaf library [4]. The structure of the resulting data is described in Table A.3 (see Appendix A).

[4]Pytaf library: `https://github.com/dmbaturin/pytaf` (visited: july, 2025)

4.3.4 Other

We use two additional small datasets to provide with basic airport and airline data, such as the international identifiers of each airport and airline, or the coordinates of each airport. These datasets are curated by FlightRadar24 [42], and publicly accessible through its web site. The description of these datasets is included in Appendix A.7.

Chapter 5

Data processing

In this chapter, we discuss the extraction and initial cleaning of the data that will be used to accomplish the objectives of this Thesis. First, we explain how the data is structured and extracted from their data sources, described in the previous chapter, and discuss some of the quality problems of the data in their original form. Then, we propose a methodology to solve these data quality problems, and a set of metrics to identify and characterize the presence of these problems in the obtained data. Finally, we evaluate our approach on a representative sample of two weeks of surveillance, flight plan and weather data to assess the quality of the extracted data.

5.1 Description of the data workflow

As defined in the previous chapter, our flights data model requires extracting data from a plethora of heterogeneous data sources and integrating them. The differences in nature, formats and properties, however, pose particular challenges for each of these sources. Moreover, the integration is far from trivial in many cases. Hence, we propose a data workflow in two phases:

- A first step where each source is processed separately and specific solutions are proposed for each of their known defects and problems. This process is monitored using a set of data quality metrics that help us to identify the presence of data problems, and to assess whether our actions have resolved them. Once cleaned, the different data sources are integrated to construct the enriched 4D trajectories we aim to define. This phase is described in Chapter 5.

- A second step, in which we tackle the problems that arise when we consolidate the different sources into 4D trajectories, that could not be discovered until the data is structured according to such representation. In general, these defects are related to the time information, the position, or the consistence of the (evolution of the) data reported by aircraft. A set of quality metrics are proposed in order to assess the defects fount in the constructed trajectories, and characterize the quality of the resulting data sets to evaluate their value for different use cases. This phase is explained in Chapter 6.

Figure 5.1 shows an overview of the data workflow described above, which will be delved into in subsequent chapters. We have defined four levels that describe the maturity of the data:

- Level 0 (L0) corresponds with raw data, that is, data extracted directly from the data sources and placed in our storage. The raw data is stored to ensure traceability to the source itself, and protect us from any changes that can occur in the source repositories.

- Level 1 (L1) corresponds with cleaned data, where the data from the individual data sources have been processed and some of the problems that can be found in them have been addressed. This process is described later in Chapter 5.

- Level 2 (L2) corresponds with integrated data, that is, the constructed 4D trajectories through the integration of the cleaned data. The process of integrating these datasets is explained in Chapter 6.

- Level 3 (L3) contains the processed trajectories. The definition as 4D trajectories arises new problems in the data that are addressed prior to this level. What these problems are, and how they are resolved in our project, is also described in Chapter 6.

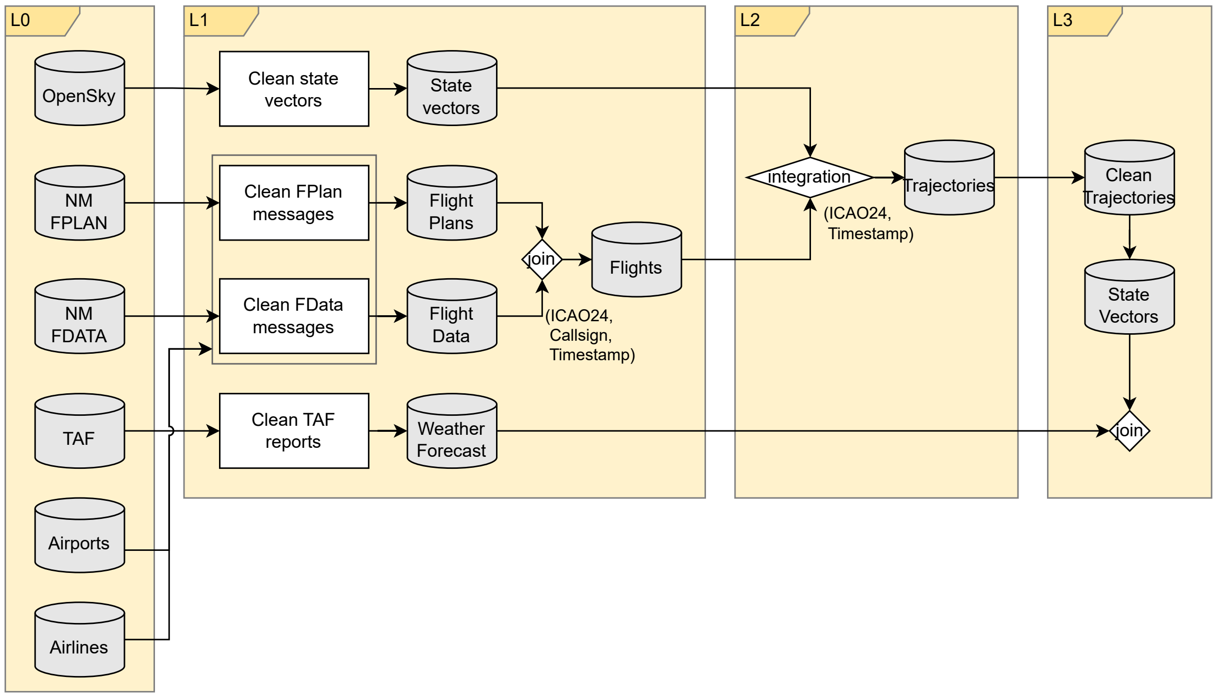

Figure 5.1: Description of the data pipeline.

5.1.1 Data partitioning

In addition to the heterogeneity, the volume of the data can also present a challenge. Both the OpenSky Network and Eurocontrol's Network Manager maintain large and continuously updated datasets, incorporating millions of new records daily. Furthermore, some of the operations required to construct 4D trajectories based on this data involve integrating data from these sources, and the complexity of the process rapidly becomes unmanageable in terms of memory and computation power for commodity hardware. Therefore, we need to partition the data to improve the scalability of the process and enable simple update mechanisms for our data repositories. We have decided to generate daily partitions for large datasets, such as surveillance data and flight plan data, and monthly partitions for datasets with a smaller volume, like data about the weather at the airports. This enables us to add new data daily by creating and populating the new partitions. Complementary datasets of small sizes with a low frequency of update are not partitioned, such as those related to the airports or the airlines.

In practice, the data partitioning for each data source is defined as follows:

- Surveillance data is partitioned according to the generation timestamp of the state vectors. Each daily partition contains all vectors generated between 00:00:00 and 23:59:59 on that date. Consequently, flights spanning two calendar days will have their state vectors split across two partitions.

- Flight plan data is partitioned by the estimated off-block time of each flight. All flights scheduled to depart within a given day are included in the corresponding partition, along with all related flight plan messages, regardless of the timestamp of message publication.

- Flight data is also partitioned by the estimated off-block time of the flight. As a result, flight plan and flight data messages for a given flight are always stored within the same partition.

- Weather forecasts at the terminal are partitioned according to their issue date, with each partition containing all forecasts issued during a given month. Since weather reports covers up to 30 hours into the future, the validity of the last report in the partition can span into the next partition.

The misalignment between the different data collections must be addressed during the integration process. On the one hand, although all flight plan and flight data messages for a given flight are stored in the same partition, its trajectory may be split across two daily surveillance data partitions. On the other hand, the forecast in effect at the timestamp of a given state vector may have originated from a weather report issued the previous day, if its validity period extends into the following month. These factors will be taken into account later in the data processing workflow to ensure a successful integration between the different datasets.

5.1.2 Case study: European airspace

The European airspace has been chosen to evaluate our data workflow. We made this decision for multiple reasons. First, we have access to data sources providing exhaustive data from flights taking place in that airspace, since there is where both EUROCONTROL and the OpenSky Network are located. Second, the European airspace is a complex scenario that allows us to study a large volume and variety of flights, including international and domestic (between airports in the same country) flights.

Our main surveillance data provider has excellent coverage on European territory thanks to its large network of receivers, which contributes to achieve a good availability and density of data to conduct a relevant study. However, the European territory is considerably rugged, with significant mountain systems and water masses. Combined with the high density of flights, we can expect a sufficient frequency of occurrence for the main problems that can be found in aviation trajectory data.

In order to facilitate the analysis of the experiments, we have limited our study to a subset of the flights taking place within European airspace, between European airports. This subset is representative of the considered airspace, since it covers a significant portion of the air traffic using the the main airways defined over European territory, and result in a large coverage of the European airspace. As a consequence, all surveillance data generated outside of this geographical scope is ignored since they can not belong to any flight within the defined parameters. In particular, we have chosen to study all flights between eight airports, which are listed below:

- EDDF - Frankfurt (GE)
- EGLL - Londres-Heathrow (UK)
- EHAM - Amsterdam-Schipol (NE)
- EKCH - Copenhague-Kastrup (DK)
- LEMD - Madrid-Barajas (SP)
- LFPG - París-Charles de Gaulle (FR)
- LGAV - Atenas-Elefterios Venizelos (GR)
- LIRF - Roma-Fiumicino (IT)

We have also limited our study to flights completed between July 3-16, 2023 (that is, two complete weeks). A threshold of a minimum of 200 vectors per trajectory is set to remove all trajectories that are too short due to faulty data.

However, surveillance data cannot be filtered in such ways until we construct the trajectories and identify the origin and destination for each of them, since ADS-B data does not provide with information about flight scheduling. Hence, we will process and analyze all the data available in OpenSky for the considered period until the trajectories are adequately identified. The rest of the analyzed sources (flight plans data and weather data) effectively have information about airports, which allows us to select the relevant data beforehand.

5.2 Data extraction

The first step for processing the data consists on retrieving the data from their original sources. These sources provide storage and filtering capabilities so we can query for the data we are interested on, but the transformations that can be applied on data in this way are very limited. Furthermore, an excessive retrieval of data from these sources may overload the data sources systems or quickly exhaust our usage quotas. We decided to store a copy of the raw data in our own storage to be able to work with the data without these restrictions. In this section, we describe how the data is extracted from the data sources and stored in our systems.

5.2.1 State vectors

State vectors data is served by OpenSky as JSON objects grouped by their assigned timestamp, as shown below. State vectors are represented as JSON objects, and grouped by their assigned timestamp. This fact leads into a structure with two levels of granularity: (i) a first level where the time information is given for sequences of individual state vectors; and (ii) a second level where the data contained by each of these state vectors is provided.

```
[ timestamp1 : [ { "icao24": icao241,
                   "callsign": callsign1,
                   "latitude": latitude1, ... },
                 { "icao24": icao242,
                   "callsign": callsign2,
                   "latitude": latitude2, ... },
                 ... ],
  timestamp2 : [ {...}, ],
...]
```

In this Thesis we require the data of individual state vectors to construct the corresponding 4D trajectories. Therefore, it is necessary to transform these data into a tabular form to work at the state vector level. Each timestamp and JSON object representing a state vector create a new row in a table with the corresponding columns. The attributes of each state vector object are aligned according to their key names. Then, a new column *timestamp* is added, whose value is the timestamp assigned to the original JSON object.

5.2.2 Flight plan data

A similar problem arises when processing flight plan data. Network Manager delivers these data in a deeply nested form, grouping in the structure the attributes that are related to each other. Each record describes a single flight plan or flight data version for a flight, which updates or extends the previous version for that particular flight. Its processing poses two main difficulties. On the one hand, the varying schema of the data across messages, even with messages associated to the same flight, since only a few attributes are mandatory and, depending on the state of the flight, new attributes are added over time. On the other hand, the existing attributes can be updated over time, whether by adding new values or changing the existing ones. In the second case, later messages contain data that was already included in previous messages, so there is a degree of data redundancy that may need to be treated.

5.2.3 Weather data

Weather data require different approaches depending on whether they are forecasts (TAF) or actual readings (ASOS). ASOS reports are generated hourly by each weather station based on actual observations, in METAR format. Each report can be identified by the station that generated it and the timestamp of its generation, since only one report is emmited each time. TAF works a bit differently: with each update, one to many METAR reports are generated based on the conditions forecast at each station. Then, for each station and update timestamp, multiple reports with different probabilities of occurrence can be generated. Moreover, in the event of sudden and significant changes, new TAF reports can be generated out of schedule to inform of the new forecast weather conditions. In these cases, the new reports can amend or overwrite the previous ones. Thus, keeping track of the current forecast at each moment requires additional care. Weather data is structured in tabular form based on the decoded METAR provided by the pytaf Python module [124].

5.3 Problem discussion

All of these data are originated by different systems and sensors that can eventually malfunction and provide incorrect readings, or even fail completely and stop providing any data for some time. Therefore, the integrated representation of these data may present different errors or missing data for some of its attributes. In this section, some of the most common errors for the different categories of data are reviewed succinctly.

5.3.1 State vectors

OpenSky aims to provide with surveillance data that is useful for research. Hence, they do not perform a heavy post-processing of the data to ensure that it is as raw and realistic as possible. This means that OpenSky data may contain several data flaws inherited from ADS-B messages, such as errors in positioning or missing attributes, already covered in Section 4.3.1. Some of these problems are alleviated by the fact that each state vector is constructed by aggregating the data of several messages received during lapses of 5 seconds. This time is higher than the theoretical update rate of ADS-B, and multiple messages of the same type should be available to generate each vector. Therefore, state vectors are more robust to data loss caused by missing payload, message loss or message corruption.

However, prolonged losses of data still constitute a significant problem in areas with no coverage, or in high density areas due to congestion of the medium or the receivers. While OpenSky Network has reached a good coverage over the years in Europe and North America, the availability of data can become very limited outside of these areas, or over large masses of water (such as the Mediterranean Sea). OpenSky tackles these situations "reusing" old data, if new messages have not been received, to reduce interruptions in the state vector generation, which introduces additional redundancy in the data. The most notable cases occur when position is reused: during several seconds, the aircraft seems to have stopped in the air, since the reported position remains constant for some time. If the position has stopped updating, but velocity data keeps being captured, the position is left empty after 15 seconds, or three state vectors. We will remove these vectors since position data is key to construct the aircraft trajectory.

The collaborative nature of the network also introduces several challenges derived from the heterogeneity in the devices deployed to capture ADS-B data. Different software and hardware configurations, unsynchronized clocks, buffer mechanisms or network limitations are some of the factors affecting the quality of the resulting data [93]. These problems are tackled during the processing of the data as it arrives to the OpenSky servers, but it is difficult to fully account for each of these situations and their effects.

For instance, the timestamps assigned to ADS-B messages are relative to the reception time by the sensor that captured it, and then rounded to the closest second in the OpenSky servers; moreover, the timestamps calculated for state vectors correspond with the timestamp of the last message included in each state vector. If an ADS-B message is delayed at any stage of the processing pipeline (e.g. if a receiver takes too long to forward its captured messages), it may be assigned to an incorrect state vector. As a result, this vector would include information that actually corresponds to an earlier time than the timestamp it carries, leading to a misrepresentation of the aircraft's state, as shown in Figure 5.2. The diagram shows the altitude of the plane across the sequence of state vectors, sorted by their timestamp. It is clear that several vectors might have been constructed with messages including altitude values from a few minutes before, and not with the actual altitude of the aircraft at that moment. However, these problems only become apparent once the trajectories are reconstructed, and are difficult to approach in this phase.

Further, vectors can inherit some of the problems of ADS-B covered in Section 4.3.1 beyond the lack of data. It is possible that some values used to build state vectors are incorrect due to sensor or GPS malfunctioning (particularly in position and altitude values). Again, some of these problems can remain unnoticed until the trajectories are identified, although others can be identified as such very quickly. For instance, it is not uncommon to find vectors with altitudes well beyond 42,000 feet, which is the usual maximum altitude for commercial flights.

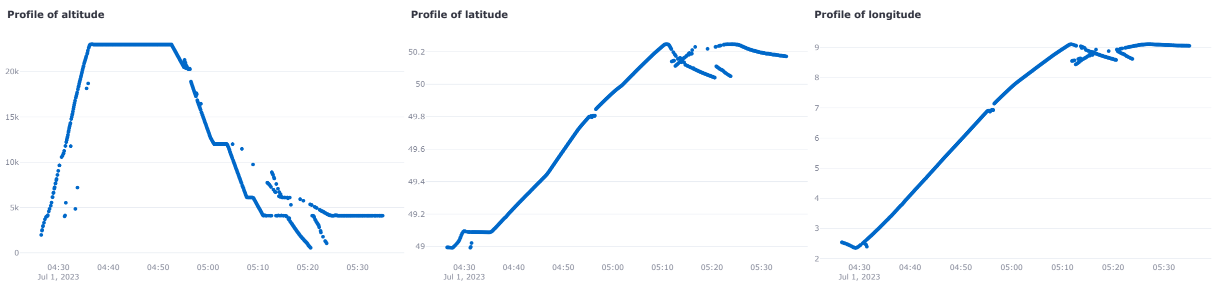

Figure 5.2: Example of delayed vectors (flight AT02523950 in July 1, 2023). Several vectors at the end of the trajectory show altitude, latitude and longitude values that are inconsistent with the progression of the trajectory.

5.3.2 Flight plans

We have found several problems that can interfere with the processing of flight and flight plan data from Network Manager. We discuss these problems below:

- The ICAO code (ICAO24) of the aircraft changes once the flight has already started. ICAO24 can change if the flight is assigned to a different aircraft in the pre-tactical phase of the flight or before, but obviously should not change after the flight has already started. In these cases, we prioritize the last informed value.

- Some attributes, such as the ICAO code or the flight rules applicable to the flight, are not included in the last message or messages, which hinders the integration between the flight plans and other sources. Specially in the Flight Plan message sent after the flight ends. A sensible solution would be propagating the previously observed value to these last records.

- The source repository where the data was stored used the local time zone (Madrid, Spain) to define the time information, instead of UTC. It is required to shift the time data to account for winter or summer time before integrating flight plans with other sources that use UTC, such as OpenSky.

- There exists some inconsistencies between flight plan and flight data records. Some data is present in one but not in the other, even if the attributes exist on both sources (e.g. callsign).

- Reused flight identifier: The flight identifier (ifplId) has been found to not be unique in the repository, which can cause an incorrect integration between flight plan and flight data records. However, these IDs only reappeared after a long time (months or years after). We have defined a more strict time frame in which this integration can take place, so flight plan records can only be associated with flight data records generated within that time frame.

5.3.3 Weather forecasts

Although the TAF records were already decoded with the pyTAF library, we have observed several data issues that must be addressed to enable their usage for our purposes. First, and similar to flight plan data, the structure of the decoded TAF reports is nested and variable. In particular, all the attributes that describe weather phenomena at different altitudes (cloud characteristics, icing or turbulence caused by wind) can be empty or contain a variable number of elements, each containing additional attributes. This structure can be translated into a wide tabular form, with columns for each of the possible sub-attributes. However, the resulting table would be very sparse, and the high number of columns could hinder the use of the data in later stages. Therefore, we opted to extract only a subset of these elements (i.e. the records closest to the airport, since they are of most importance for the studied use cases), and keep the nested attributes in their current form. The attribute wx_string is a special case: it is constructed as a string containing the concatenation of a variable number of codes describing different weather phenomena. While not explicitly an enumeration, the challenge it poses is similar to the other variable length attributes.

Second, TAF reports correspond with evolutionary forecasts, and thus they are subject to change with time. Therefore, to associate a particular state vector with the relevant forecast weather conditions, it is necessary to keep track of the latest available report according to their validity period and the publication of new reports with updated forecasts or amendments to the existing ones. This requirement is tackled after the cleaning step. Third, a significant portion of the forecasts have associated a particular probability, in contrast with other reports that describe expected changes (with a high probability) in the weather conditions. Depending on the application of TAF data, the more confident forecasts could be prioritized in detriment of probable changes, which also influences how the second consideration can be approached. Finally, in rare cases there are errors in the report type like PROB04 or PROB30TEMPO, which can be easily replaced with the correct values.

5.4 Data cleaning

In this first phase, we apply several common operations in data cleaning to remove problems that have incidence on individual fields from each dataset. We follow the methodology indicated in Figure 5.3, with few additional tweaks when required. For example, in some cases the values that are out of their defined domain will be removed, while in others it is possible to set them to a correct value. These operations are conditioned by the particularities of each data source: design decisions, data format and structure, the storage system, etc. All these factors introduce different characteristics and transformations that must be standardized before working with the data. We start by adapting the extracted data to our data model by removing irrelevant fields and changing the names of fields that are used later. The data types are also changed to ensure

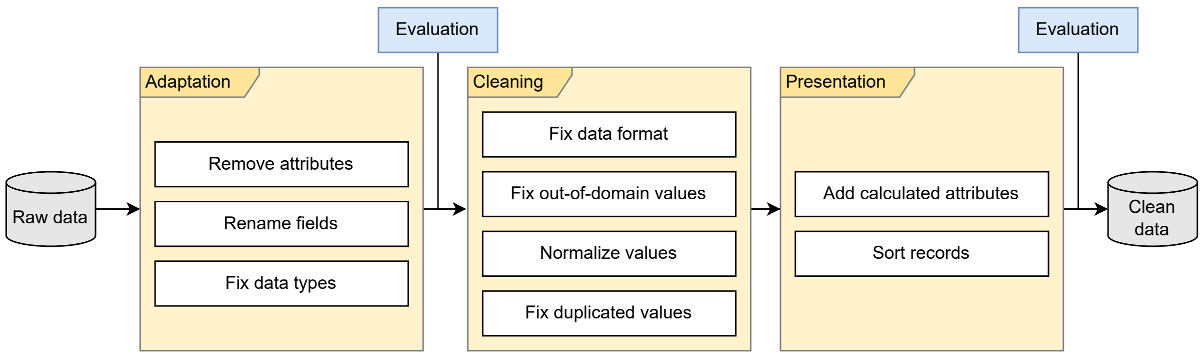

Figure 5.3: Data cleaning methodology. After each evaluation, the data quality metrics are measured on the processed dataset.

an adequate interpretation of the data (e.g. surveillance data is stored as JSON files, where all values are strings).

Once the data have refined, several cleaning operations are applied to correct different defects in the data: replace default values, normalize the existing values (e.g. changing the case of letters), and remove vectors with incorrect or empty values. Finally, we add the calculated attributes that could only be generated when the raw data is clean (or are more convenient to generate at the end of the process so they do not have to be carried throughout the cleaning process), and sort the resulting data before the storage. During the data processing, we measure the data quality according to a set of metrics right before the cleaning (once the structure of the data has been normalized) and at the end of the process, to monitor the changes in the metrics before and after the cleaning of the data.

5.4.1 OpenSky state vectors

The summary of transformations on the individual attributes is shown in Table 5.1. First, the *sensors*, *spi* and *position_source* fields are discarded, since they are mostly empty, and they serve no purpose for our data model. The attributes related to flight times are provided as strings comprising the date and the time, so we convert them to Unix epoch to work with them as integer values. To this end, the date and time are interpreted, and translated into seconds. The *callsign* and *icao24* fields are switched to upper case and trailing spaces are removed to comply with their expected formats (callsign strings are padded with spaces up to a length of 7).

Then, we search for values that are out of the scope, or are null, in a subset of the fields where missing or incorrect data would turn the vector irrelevant. Hence, we discard all vectors with null values in any timestamp or position fields. If latitude and longitude have a value, but it is out of their domains (recall that they are expressed in degrees), the corresponding vector is removed as well. After these filters, we look for any vectors having duplicated values for the combination of fields (*ICAO24*, *time_position*, *latitude*, *longitude*), to remove the state vectors constructed with outdated positions. Also, *true_track* values are applied the module 360 operation. The state vectors with missing altitude are not removed, since it can be estimated later once the trajectories have been identified. This also holds true for altitude values that are unrealistic.

Finally, two auxiliary columns are added: timestamp and altitude, and they are assigned the values of *time_position* and *baro_altitude*. These fields will act as the main time and altitude values, which will be overwritten during the trajectory cleaning stage of the workflow, if their value is determined as incorrect. The remainder time and altitude fields are kept intact for traceability reasons. The result is sorted by ICAO24 and timestamp, so the vectors of the same aircraft are stored adjacently.

Raw attribute	Process	Final attribute
hexid	Convert to upper case, align to format: [A-F0-9]{6}, remove vectors with identical position	icao24
callsign	Convert to upper case, align to format: [A-Z0-9]{3,9}	callsign
time_stamp	Convert to epoch	time_position
time_stamp_velocity	Convert to epoch	last_contact
longitude	Remove if null or out of the valid range	longitude
latitude	Remove if null or out of the valid range	latitude
altitude		geo_altitude
baro_altitude		baro_altitude
on_ground		on_ground
ground_speed		velocity
vertical_rate		vertical_rate
track	Apply module(360)	true_track
squawk		squawk
origin_country	Remove attribute	-
spi	Remove attribute	-
sensors	Remove attribute	-
position_source	Remove attribute	-
	New attribute, copy of *time_position*	timestamp
	New attribute, copy of *baro_altitude*	altitude

Table 5.1: Logical map for OpenSky state vectors (L0 to L1)

5.4.2 Network Manager Flight Plans and Flight Data

To start cleaning Flight Plan data (FPLAN), we project only the fields described in Appendix A to remove all unnecessary attributes. Tables 5.2 and 5.2 show the operations applied on the remaining attributes. First, the names of these attributes are adapted to match our data model. All timestamps are converted into epoch in a similar fashion to state vectors, and then modified into UTC from the corresponding local time zone in Madrid, Spain, where the flight plan message were generated. The local time zone can be GMT+1 or GMT+2 depending on the period of the year, so the process is parametrized to apply the adequate factor to the conversion. Then, all duplicate vectors, that is, vectors that are identical except for the *uuid* (message ID), are removed. All messages missing at least one of a set of key attributes are also discarded. These attributes are the identifier of the flight in NM systems (*ifplId*), the real identifier of the flight (*callsign*), and the departure and arrival airports. These attributes are necessary to identify trajectories, and can not be inferred through other methods. As pointed in previous sections, some flight plans “lose” some values in their last versions, once the flight has landed. In these cases, the last valid value is propagated into these messages for several attributes. All identifiers are converted to upper case to align them with the rest of data sources. Finally, the result is sorted by the *ifplId* and the *flightDataVersionNr*, which indicates the version of the flight plan, so the flight plan messages for each flight are stored together.

A similar approach is followed to process flight data messages (FDATA).

Raw attribute	Process	Final attribute
ifplId		ifplId
aircraftId	Convert to upper case, align to format: [A-Z0-9]{3,9}	callsign
registrationMark		registrationMark
aircraftAddress	Convert to upper case, align to format: [A-F0-9]{6}, propagate if missing	icao24
ssrInfo		ssr
aerodromeOfDeparture	Convert to upper case, remove if null	aerodromeOfDeparture
aerodromeOfDestination	Convert to upper case, remove if null	aerodromeOfDestination
alternate1	Convert to upper case	alternateAerodrome
aircraftType		aircraftType
totalEstimatedElapsedTime	Convert from HHMM format to minutes	totalEstimatedElapsedTime
wakeTurbulenceCategory		wakeTurbulenceCategory
flightType		flightType
flightRules		flightRules
estimatedOffBlockTime	Convert to epoch, shift from local time to UTC	estimatedOffBlockTime
icaoRoute		icaoRoute
aircraftOperator		operator
operatingAircraftOperator		operatingOperator
timestamp	Convert to epoch, shift from local time to UTC	timestamp
uuid		uuid

Table 5.2: Logical map for Flight Plan messages (L0 to L1)

5.4.3 Flights

Once the flight plans and the flight data have been cleaned, we can integrate these datasets to identify the individual flights for that day. Flights will be constructed using the most updated information about that flight, so only the last FPLAN and FDATA messages are used. We leverage the *timestamp* and *flightDataVersionNr* attributes in the datasets, respectively, to identify the last message for each flight. Only flights that have been reported to have finished (e.g. their *flightState* attribute has been set to *‘terminated’*) are considered.

The integration is easy using the unique flight ID, *ifplId*, although the problem of duplicated ID in this data source must be taken into account (although we have not detected repeated IDs that are close in time to each other, e.g. a few days). To this end, we integrate only the flight

Raw attribute	Process	Final attribute
id		ifplId
aircraftId	Convert to upper case, align to format: [A-Z0-9]{3,9}	callsign
aerodromeOfDeparture	Convert to upper case, remove if null	aerodromeOfDeparture
aerodromeOfDestination	Convert to upper case, remove if null	aerodromeOfDestination
estimatedOffBlockTime	Convert to epoch, shift from local time to UTC	estimatedOffBlockTime
aircraftAddress	Convert to upper case, align to format: [A-F0-9]{6}, propagate if missing	icao24
aircraftType		aircraftType
flightState		flightState
actualOffBlockTime	Convert to epoch, shift from local time to UTC	actualOffBlockTime
estimatedTakeOffTime	Convert to epoch, shift from local time to UTC	estimatedTakeOffTime
actualTakeOffTime	Convert to epoch, shift from local time to UTC, propagate if missing	actualTakeOffTime
estimatedTimeOfArrival	Convert to epoch, shift from local time to UTC	estimatedTimeOfArrival
actualTimeOfArrival	Convert to epoch, shift from local time to UTC, propagate if missing	actualTimeOfArrival
calculatedtedTakeOffTime	Convert to epoch, shift from local time to UTC	calculatedtedTakeOffTime
calculatedTimeOfArrival	Convert to epoch, shift from local time to UTC	calculatedTimeOfArrival
aircraftOperator		operator
operatingAircraftOperator		operatingOperator
icaoRoute		icaoRoute
routeLenght		routeLenght
estimatedElapsedTime	Convert from HHMM format to minutes	estimatedElapsedTime
flightDataVersionNr		flightDataVersionNr
timestamp		timestamp
uuid		uuid

Table 5.3: Logical map for Flight Data messages (L0 to L1)

data messages whose actual take-off time is close in time to the estimated off-block time defined in the flight plan. The defined thresholds are at most one day in advance, and two days later. In that way, messages from advanced or delayed flights would still be considered.

FPLAN and FDATA integration requires the consolidation of redundant information. Many attributes are present in both datasets, but some discrepancies have been detected in their values, such as different values of ICAO24, if the aircraft that was planned to be used changes at some point, or the estimated off-block time, if the flight is delayed short before the takeoff. In these cases, FDATA values are prioritised over FPLAN as they are considered more up to date.

5.4.4 Weather forecasts

TAF reports only required light processing to bring them into an appropriate form, as shown in Table 5.4. First, we extracted the relevant attributes located in nested structures. The minimum and the maximum temperatures at the airport, and their respective timestamps, were extracted from the *temperature* attribute. The main sky condition was also extracted from the corresponding attribute, and characterized by three new attributes: *sky_cover*, *cloud_base_ft_agl* and *cloud_type*. The fields describing the turbulence and the icing conditions were not extracted, since these attributes were found to be mostly empty. For all these nested attributes, we identified the empty list of elements as missing values, and treated them accordingly so the completeness of the data could be correctly assessed.

The attributes that define the validity period of the forecast can be empty if this period has not been defined. In these cases, they are implicitly relative to the issue time: routine forecasts are generated one hour in advance of the expected weather conditions, and all reports have a validity of 30 hours if it is not stated otherwise. Therefore, when these values are unknown the default validity of a TAF report can be assigned to this report based on the issue_time.

Raw attribute	Process	Final attribute
raw_text	Remove	
station_id	Convert to upper case, align to format: [A-Z]{4}	station_id
issue_time		issue_time
valid_time_from	If it has no value, assign issue_time + 1 hour	valid_time_from
valid_time_to	If it has no value, assign issue_time + 30 hours	valid_time_to
time_from		time_from
time_to		time_to
change_indicator		change_indicator
probability		probability
wind_dir_degrees	Apply module(360)	wind_dir_degrees
wind_speed_kt		wind_speed_kt
wind_gust_kt		wind_gust_kt
wind_shear_hgt_ft_agl		wind_shear_hgt_ft_agl
wind_shear_dir_degrees		wind_shear_dir_degrees
wind_shear_speed_kt		wind_shear_speed_kt
visibility_statute_mi		visibility_statute_mi
altim_in_hg		altim_in_hg
vert_vis_ft		vert_vis_ft
wx_string		wx_string
turbulence_condition	Identify empty values	turbulence_condition
icing_condition	Identify empty values	icing_condition
sky_condition	Extract fields, identify empty values	sky_condition
	New attribute	sky_cover
	New attribute	cloud_base_ft_agl
	New attribute	cloud_type
temperature	Extract fields, identify empty values	temperature
	New attribute	min_temp
	New attribute	min_temp_timestamp
	New attribute	max_temp
	New attribute	max_temp_timestamp

Table 5.4: Logical map for TAF reports (L0 to L1).

5.5 Data quality metrics

All of the considered data is subject to the usual data quality problems that affect the completeness, consistency, duplication, uniqueness, value (domain, range or structure) [100]. To this end, several common metrics are measured for each data source and each attribute, relative to the partition criteria defined for each data set. The list of calculated metrics is shown in Table 5.5.

Univariate analysis	
Completeness	Ratio of non-empty values for each attribute
Cardinality	Number of unique values for each categoric attribute
Distribution	Frequency of values for each categorical attribute with low cardinality
Duplicated values	Detect and process duplicated data in attributes defined as unique
Domain	Number of values outside of the defined range or domain
Structure	Number of values that do not conform to a defined pattern
Multivariate analysis	
Duplicated records	Duplicated values for combinations of attributes (such as 3D position and time) which may denote redundant or incorrect data
Descriptive metrics	
Number of records	Number of records in the dataset

Table 5.5: Summary of data quality metrics.

5.6 Experiments

The datasets described in Section 5.1.2 has been processed following the methodology explained above. In the next sections, we explore the characteristics of the resulting dataset, and analyze the values obtained for the defined metrics.

With the aim of demonstrating and evaluating the data workflow described in this chapter, we have defined a test dataset in line with the case study presented in Section 5.1.2. In particular, we have captured data to construct the trajectories of all flights between July 3-16, 2023. To this end, we have to process surveillance and flight plans data extracted from their respective data sources. In the following sections, we discuss the main results of the data cleaning step for this dataset, before we can integrate the data in the next chapter.

5.6.1 OpenSky dataset

A total of 447.5M state vectors were extracted from the OpenSky Network for that period, with an average of 32M state vectors per day. After applying the operations explained in previous sections, 5.4M state vectors (1.2065%) were removed due to having a repeated position, as indicated in Table 5.6. To this end, adjacent vectors with the same positions were removed, while keeping the first state vector, for each aircraft (identified by their ICAO24). Figure 5.4 shows the distribution of the state vectors across the studied days, before and after the cleaning. The amount of daily vectors is similar, with a slight increase before and during the weekend. In total, 174,761 aircraft were identified in the surveillance data, which performed 463,151 different flights (slightly reduced after the cleaning).

Table 5.7 shows the completeness of the fields of surveillance data. The most important attributes, such as the callsign, the ICAO24, the timestamps, the ground flag and the position, are almost always available. Interestingly, the completeness of the callsign and the origin country decreases after the cleaning, because default values are removed during this step since they do not provide any relevant information. Some examples of these default values are ‘TEST‘, ‘1234‘, ‘000‘ or an empty string, which are all valid values and counted towards the completeness metric. Such default values have not been found in other attributes. The geometric and barometric altitudes are missing more often, with 3.97% and 4.33% of the state vectors lacking a value for these fields. These results are slightly improved after the cleaning.

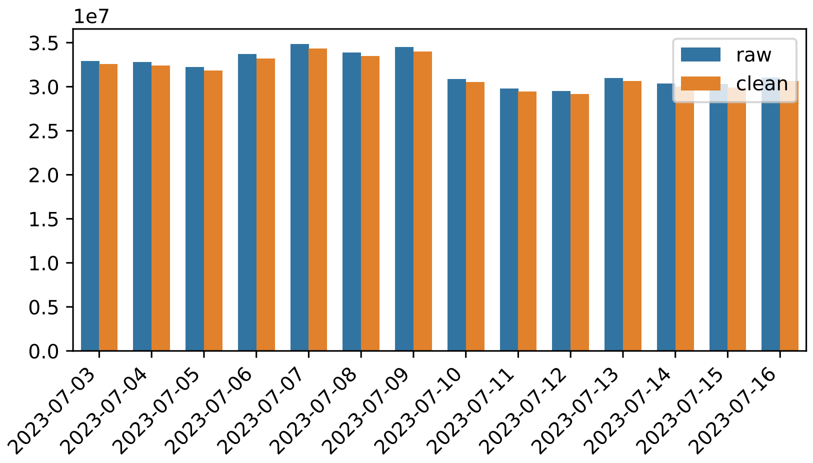

Figure 5.4: Number of vectors captured each day for the raw and clean data.

5.6.2 NM Flight Plans dataset

A total of 890,200 messages were extracted from the Network Manager relative to 348,388 different flights, as indicated in Table 5.8. This means that almost 25,000 flights were planned for each day. After cleaning, only 244 duplicate messages were found and removed, so the cardinality of the dataset remains almost unchanged. The main improvement can be observed in Table 5.9, where many of the attributes with low completeness (wakeTurbulenceCategory, registrationMark, flightType, totalEstimatedElapsedTime, and more importantly the ICAO24) were significantly improved thanks to the value propagation to the latest messages for each flight, which we have observed to have many fields blank. These flights were planned to be performed by 85,400 different aircraft. Note that there is a discrepancy between the number of flights and the number of identified callsigns: the difference can be attributed to repetitive flights, which reuse the same callsign in different flights.

Another prominent change is the size of the dataset, which is reduced from 3,786 MB to 51.5 MB. We decided to change how the data was stored for efficiency, and transformed the original JSON files into a tabular representation, and then stored them as (compressed) Parquet files. Finally, the distribution of flight plan messages across the studied days, which is similar to the

	Raw data		Clean data	
Value	Total	Mean	Total	Mean
File size (MB)	13,152	939	10,291	735
Records	447,426,023	31,959,001	442,027,582	31,573,398
Removed records	5,398,441	385,602	-	-
Reused position	5,398,441	385,602	-	-
Other concepts	0	0	-	-
Unique ICAO24	174,761	12,482	174,761	12,482
Unique callsign	463,151	33,082	463,104	33,078

Table 5.6: Total and mean values (per partition) of the state vectors dataset.

	Raw data		Clean data	
Attribute	Mean	StD	Mean	StD
callsign	100.00%	-	98.89%	0.28%
icao24	100.00%	-	100.00%	-
last_contact	100.00%	-	100.00%	-
latitude	100.00%	-	100.00%	-
longitude	100.00%	-	100.00%	-
on_ground	100.00%	-	100.00%	-
origin_country	100.00%	-	99.64%	0.13%
time_position	100.00%	-	100.00%	-
true_track	100.00%	-	100.00%	-
velocity	99.95%	0.02%	99.95%	0.02%
vertical_rate	96.50%	0.17%	96.55%	0.16%
geo_altitude	96.03%	0.18%	96.07%	0.19%
baro_altitude	95.67%	0.13%	95.79%	0.12%
squawk	81.75%	1.49%	81.76%	1.50%

Table 5.7: Mean completeness per day for numeric attributes in the state vectors dataset.

distribution of state vectors, is shown in Figure 5.5.

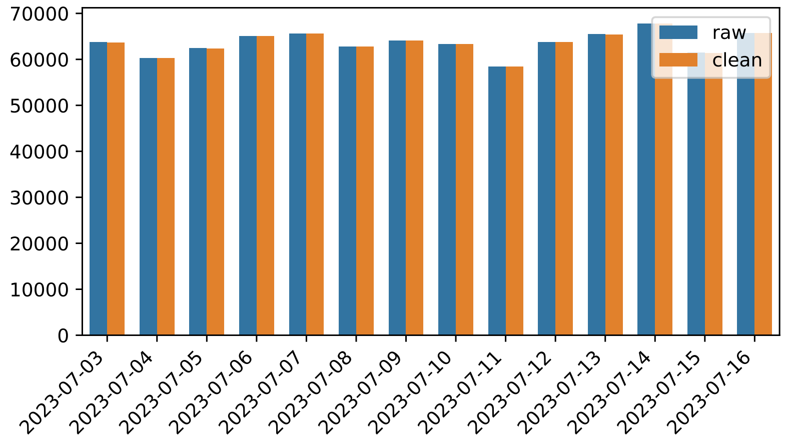

Figure 5.5: Number of messages captured each day for the raw and clean data.

5.6.3 NM Flight Data dataset

Similar conclusions can be extracted from the flight data dataset. We captured 13,04M flight data messages, corresponding with 248,402 planned flights, as indicated in Table 5.10. This number is lower than the observed in the flight plans dataset: this is because many flights are canceled well before the pre-tactical phase, that is, before they are transferred to the IFPS system and their data becomes available as flight data. In this case, a total of 76,248 aircraft performed flights with 237,706 different callsigns. The distribution of flight data messages differs from the flight plan messages: a significant decrease can be observed the days 7 and 11, but we have not been able to identify the cause. Finally, small improvements were achieved after the cleaning with respect to completeness: while most of the attributes are always present, some of the remaining fields could be partially recovered (in particular, the actual takeoff and arrival times). Once again, the size of the data was reduced by changing the storage format to improve the performance of reading files in later stages of the data workflow.

	Raw data		Clean data	
Value	Total	Mean	Total	Mean
File size (MB)	3,786	270	51.5	3.7
Records	890,200	63,585	889,953	63,568
Removed records	244	17	-	-
Duplicated	244	17	-	-
Unique flights	348,388	24,884	348,388	24,884
Unique ICAO24	85,397	6,099	85,397	6,099
Unique callsign	333,567	23,826	333,567	23,826

Table 5.8: Total and mean values (per partition) of the flight plans dataset.

	Raw data		Clean data	
Attribute	Mean	StD	Mean	StD
aerodromeOfDeparture	100.00%	-	100.00%	-
aerodromeOfDestination	100.00%	-	100.00%	-
callsign	100.00%	-	100.00%	-
estimatedOffBlockTime	100.00%	-	100.00%	-
timestamp	100.00%	-	100.00%	-
uuid	100.00%	-	100.00%	-
ifplId	99.99%	0.02%	99.99%	0.02%
operator	98.34%	0.10%	98.34%	0.10%
operatingOperator	92.24%	0.33%	92.24%	0.33%
aircraftType	90.75%	0.26%	90.75%	0.26%
wakeTurbulenceCategory	90.75%	0.26%	99.99%	0.02%
registrationMark	90.53%	0.28%	99.87%	0.07%
flightRules	89.56%	0.28%	89.56%	0.28%
flightType	89.56%	0.28%	99.99%	0.02%
totalEstimatedElapsedTime	89.56%	0.28%	99.99%	0.02%
icao24	86.29%	0.48%	95.23%	0.36%
ssr	7.14%	0.17%	9.14%	0.27%

Table 5.9: Mean completeness per day for numeric attributes in the flight plans dataset.

	Raw data		Clean data	
Value	Total	Mean	Total	Mean
File size (MB)	12,158	868	532	38
Records	13,038,699	93,133	1,3038,628	93,128
Removed records	71	5	-	-
Duplicated	71	5	-	-
Unique flights	248,402	17,743	248,402	17,743
Unique ICAO24	76,248	5,446	76,248	5,446
Unique callsign	237,706	16,979	237,706	16,979

Table 5.10: Total and mean values (per partition) of the flight data dataset.

	Raw data		Clean data	
Attribute	Mean	StD	Mean	StD
aerodromeOfDeparture	100.00%	-	100.00%	-
aerodromeOfDestination	100.00%	-	100.00%	-
aircraftType	100.00%	-	100.00%	-
callsign	100.00%	-	100.00%	-
estimatedOffBlockTime	100.00%	-	100.00%	-
estimatedTakeOffTime	100.00%	-	100.00%	-
estimatedTimeOfArrival	100.00%	-	100.00%	-
flightState	100.00%	-	100.00%	-
flightDataVersionNr	100.00%	-	100.00%	-
ifplId	100.00%	-	100.00%	-
routeLength	100.00%	-	100.00%	-
timestamp	100.00%	-	100.00%	-
uuid	100.00%	-	100.00%	-
icao24	98.28%	0.27%	98.32%	0.26%
operator	98.07%	0.10%	98.07%	0.10%
operatingOperator	91.05%	0.40%	91.05%	0.40%
actualOffBlockTime	75.55%	3.71%	75.55%	3.71%
actualTakeOffTime	75.55%	3.71%	77.08%	3.59%
actualTimeOfArrival	75.55%	3.71%	77.08%	3.59%
calculatedTakeOffTime	45.19%	5.83%	45.19%	5.83%
calculatedTimeOfArrival	45.19%	5.83%	45.19%	5.83%

Table 5.11: Mean completeness per day for numeric attributes in the flight data dataset.

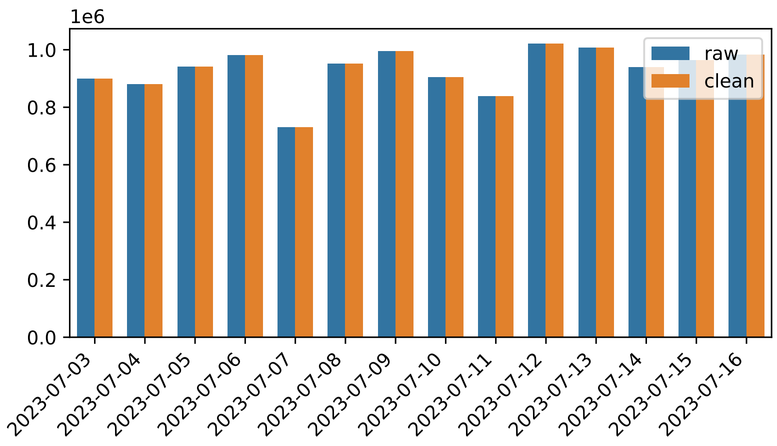

Figure 5.6: Number of messages captured each day for the raw and clean data.

5.6.4 TAF dataset

Due to the volume of the data, the TAF dataset was partitioned monthly instead of daily. Therefore, Tables 5.12 and 5.13 shows the statistics of the complete partition of July, 2023. A total of 1,578,281 reports were generated by 3,158 weather stations located at an airport. No duplicated records were found in this partition. The FM reports, which indicate certain changes in the weather conditions, were the most common, with more than 440k reports. Reports informing of temporary weather phenomena (TEMPO) were second in frequency, closely followed by reports informing of gradual changes towards new weather conditions (BECMG), with 270k and 260k reports, respectively. The corrections on previous forecasts (AMD and COR) were comparatively rare (2.41%). The remaining reports corresponded with probable forecasts.

With respect to the completeness of the data, the nested attributes showed a false high value of the metric since they were assigned a value, even if it means that there were no readings. We interpreted such situation as no data, thus providing a more accurate assessment of this aspect on the clean data. It becomes evident that these weather conditions are rarely informed, in contrast with the sky_condition, which has at least one record in 82.71% of the reports. The attribute valid_time_from has increased its completeness thanks to the attribution of the default validity period during the cleaning. The rest of the attributes remain unchanged, since no action was taken on them.

Value	Raw data	Clean data
File size (MB)	146	25.8
Records	1,578,281	1,578,281
Removed records	0	-
Reports per type		
FM	441,334	441,334
BECMG	261,649	261,649
TEMPO	272,650	272,650
PROBXX	119,163	119,163
AMD	34,200	34,200
COR	3,759	3,759
Unique stations	3,158	3,158

Table 5.12: Total and mean values (per partition) of the TAF reports dataset.

	Raw data	Clean data
Attribute	Mean	Mean
issue_time	100.00%	100.00%
part_date_utc	100.00%	100.00%
station_id	100.00%	100.00%
valid_time_to	100.00%	100.00%
icing_condition	100.00%	0.07%
sky_condition	100.00%	82.71%
temperature	100.00%	4.80%
turbulence_condition	100.00%	0.29%
valid_time_from	98.96%	100.00%
sky_cover	82.71%	82.71%
wind_speed_kt	80.57%	80.57%
visibility_statute_mi	73.18%	73.18%
change_indicator	71.77%	71.77%
cloud_base_ft_agl	71.02%	71.02%
time_from	69.37%	69.37%
wind_dir_degrees	66.96%	66.96%
time_to	41.40%	41.40%
wx_string	34.82%	34.82%
wing_gust_kt	12.70%	12.70%
cloud_type	11.15%	11.15%
probability	7.55%	7.55%
max_temp	4.68%	4.68%
max_temp_timestamp	4.68%	4.68%
min_temp	4.64%	4.64%
min_temp_timestamp	4.64%	4.64%
altim_in_hg	3.50%	3.50%
vert_vis_ft	0.64%	0.64%
wind_shear_hgt_ft_agl	0.08%	0.08%
wind_shear_dir_degrees	0.08%	0.08%
wind_shear_speed_kt	0.08%	0.08%

Table 5.13: Mean completeness per day for numeric attributes in the flight data dataset.

Chapter 6

Trajectory processing

At this stage, the individual data sources have been processed to characterize different aspects of flights, such as the surveillance data, the flight plans and the weather conditions. However, these data are disjoint and barely usable to support air traffic management in their current form. In Chapter 2 we introduced the concept of 4D trajectory as the cornerstone to conciliate all these flight-related concepts and provide an effective support to air traffic operations. This chapter discusses how 4D trajectories can be created based on the aforementioned data, and how to tackle some of the challenges that arise when structuring this data following this abstraction.

6.1 Trajectory definition

4D trajectories comprise static and dynamic information, providing a complete representation of all aspects related to the state of a flight. For the purposes of this Thesis, we have defined a enriched 4D trajectory representation containing the information necessary for implementing the identified use cases and that is available in the identified data sources.

A trajectory is uniquely identified (we use the attribute *ifplId* from Network Manager for this purpose), although several additional attributes should be incorporated to fully characterize it. These attributes are defined in the flight plan and have a single value for that trajectory. In particular, the following attributes are assigned to the trajectory: the callsign of the flight, the ICAO24 code of the aircraft, the aerodromes of departure and destination, the flight date, and the operating airline. We also include the schedule information, such as the estimated and actual times of arrival and departure. Since these trajectories are constructed a posteriori, we use the last value available for these attributes. Schedule attributes are subject to change as the flight progresses, so any online processing would require to keep track of their latest value during the flight. We also store the flight status reported in the last flight plan message to keep track of whether the flight has ended or the available data for that trajectory is incomplete.

The dynamic data in the trajectory corresponds with the surveillance data available for that trajectory. This information is represented as a time series with all the updates of the state of the flight, described by the following subset of attributes of those provided by OpenSky: the 2D position of the aircraft (latitude and longitude), the altitude (both geometric and barometric), the track of the aircraft, and the flag indicating whether the aircraft is on the ground or in the air. Based on this sequence of data, the metadata of the trajectory is enriched with the indication of the used sources of the surveillance and flight plan data, the number of vectors contained in the sequence and the time period that it describes; i.e. the timestamps of the initial and final state vectors. Finally, we add an attribute indicating the maturity of the trajectory in our data workflow; i.e. *raw* for the newly created trajectories, and *clean* for the refined trajectories.

6.2 Data integration

In previous chapters we have discussed some of the main data resources available to support ATM operations. Surveillance data allows to monitor the state of every aircraft equipped with the adequate systems in a near-real time manner. Flight plans provide schedule information that enables organizing the airspace in advance, in order to increase the predictability and efficiency of the operations. It is apparent that these resources bring complementary views of the same reality, although separately they are partial and limited. Therefore, it is necessary to integrate them in a single resource to unlock their full potential.

Trajectories, and in particular 4D trajectories, constitute a great abstraction to reconcile the concepts behind the flight plan data and the surveillance data. Trajectories provide a context (the state vectors describing the same flight) to surveillance data, increasing the interpretability of each data point in isolation. Trajectories also provide a deeper description of the progression of a flight beyond the defined time and route schedules or the summary submitted after the flight ends. After the integration of these data, trajectories can be enriched with different metadata to increase the characterization and value of the individual trajectories.

In the following sections, we delve into the common problems we can find when constructing trajectories based on flight plan and surveillance data, and how this problem is tackled in the context of our investigation.

6.2.1 Discussion of the main quality problems

The constructed trajectories can show several anomalies derived, in most of the cases, from problems present in the surveillance and flight plan data, although they only become apparent when we interpret these data as trajectories.

a. Missing vectors

- At the beginning of the trajectory
- At the end of the trajectory
- In the middle (measure % of missing data in terms of distance, number of gaps, size of the gaps in time and distance, estimate number of lost vectors (one each 5s))

Theoretically, the surveillance data provides a continuous and detailed description of the state of the aircraft, updated every few seconds. However, as discussed in the previous chapter, several factors can lead to ADS-B messages being lost during transmission and never reaching any receiver. Depending on the severity of these losses, we can distinguish between several scenarios.

In the case of punctual losses of ADS-B messages, OpenSky can account for these lost vectors by using messages emitted close in time. The theoretical update rate of ADS-B for position is 1Hz, so five or more ADS-B messages should be lost to break the continuity of the sequence of state vectors. Moreover, losing the first messages would not result in any consequences. Take the example of two messages of the same type received within the same vector generation cycle.

Only the latest message is used to construct the vector, so the loss of the first message is not relevant since the resulting vector is identical. If the second message was lost, the first one can provide slightly obsolete data from the aircraft state, but close enough to remain relevant.

When more ADS-B messages are lost, OpenSky reuses the latest message available during three more vector generation cycles to ensure continuity. This becomes more apparent when the position data is not adequately refreshed; during up to three vectors, the aircraft is situated at the exact same point. While new vectors are being generated, albeit only partially updated, they could be considered as anomalous depending on the use case, and thus they should be taken into account during the data process. In our case, such vectors describe an impossible movement pattern that introduces noise in the sequence. Hence, we identify those vectors based on their position data, and define them as "vectors with reused location" to be excluded during the trajectory reconstruction. In more severe cases, such as zones where high traffic density leads to massive message losses, the density of state vectors (amount of state vectors per time unit) can decrease, or even show gaps where no state vectors exist for a prolonged period of time and the state of the flight is unknown.

It is necessary to study such events, since they can be relevant depending on the destination of the data. In this project, we assess the continuity of the constructed trajectories and annotate them accordingly. First, we compute the average update rate, and determine those parts of the trajectory where this rate is not consistent. Second, we determine the existence and impact of gaps in the sequence, and whether they appear at the beginning, the middle or the end of the trajectory. In that way, we can evaluate the quality of each trajectory for the intended use cases. These metrics are described in more detail later in this chapter.

b. Coverage problems and duplication

While state vector losses are occasional and affect to individual trajectories, there exist zones where these losses occur more frequently. The most evident example are zones with no coverage, that is, those where there are no receivers to capture ADS-B messages from aircraft flying over. This situation causes gaps on all trajectories that fly over these zones, causing a complete absence of data on those zones. Such is the case of large water masses, where no receivers can be deployed. Orography is another common cause for persistent message losses: since ADS-B receivers are deployed on the ground, the terrain can block aircraft transmissions when flying over particular areas. In all cases, the lack of ADS-B data leads to being able to construct state vectors, and thus to gaps in the data sequence.

On the contrary, it is possible that ADS-B messages reach multiple receivers, leading into data duplication; for example, due to overlapping reception areas or bounces on geographical accidents. This problem is lessened by the state vector construction process: duplicate messages received during a single generation cycle would have no effect onto the produced state vector. However, if the delay between the two copies of the same ADS-B message is large enough, it could result into vectors with duplicated data (albeit timestamp would be different). These cases must be acknowledged and processed.

c. Incomplete payload

The different ADS-B messages serve specific purposes by carrying specific data. Therefore, all state vectors require at least one message of the necessary types to be complete. When some messages are not received in time, the resulting vector can have a part of their data missing, leading to an incomplete representation of the flight state. The problem of reused positions can be also be considered as an incomplete payload problem, since its caused by position data not being updated with enough frequency.

In some cases, this problem can be overcome by data imputation or even interpolation:

- Static data (which generally does not change across the trajectory) can be determined based on other vectors in the trajectory. Callsign is an example of this.
- Time-dependent attributes, such as timestamps, position, velocity or direction can be inferred or interpolated based on the surrounding vectors that have a valid value. However, these techniques should be applied with care, since introducing artificial data can lead to significant distortions in the reconstructed trajectory.
- Some attributes may be deemed to not be relevant, and left empty without consequences if data is not available and cannot be recovered in any other way. This decision depends on many factors, being one of the main ones the use case for the data.

d. Characterization of trajectories

An adequate description of trajectories via metadata can be a good option to help the value assessment and effective management of these data. For the purposes of the project, several measures are proposed to characterize trajectory data, which results in the definition of several metrics that are covered later in this chapter:

- Geographical range (latitude, longitude and altitude).
- Rate of update/Vector density per distance.
- Number of duplicated positions.
- The presence and impact of gaps and segments of low density.
- The percentage of vectors emitted when the aircraft is on ground.
- The presence of holdings in the trajectory, specially at the end of it.

6.2.2 OpenSky Flight API - OpenSky

OpenSky's flight data is inferred directly from surveillance data, following a set of heuristics to identify the individual flights in the stream of state vectors [94] (this process is outlined in Section 4.3.2.1). Therefore, flight data is subject to errors if surveillance data shows incorrect information or incomplete data; e.g. missing vectors at the start or the end of the trajectory, incorrect callsign values, etc. When the data is complete and correct, other information can be inferred based on the trajectory data, like actual time of takeoff or landing, the departure and destination airports, or the airline operating the flight (based on callsign). However, other data can not be determined indirectly, and require to rely on other data sources to fully describe the flight, such as schedule data. Hence, OpenSky flights API does not satisfy our data model, and will not be used hereafter.

Surveillance data and flight data can be joined using the *icao24* attribute. Then, *firstSeen* and *lastSeen* attributes are used to assign each trajectory the state vectors created during its duration.

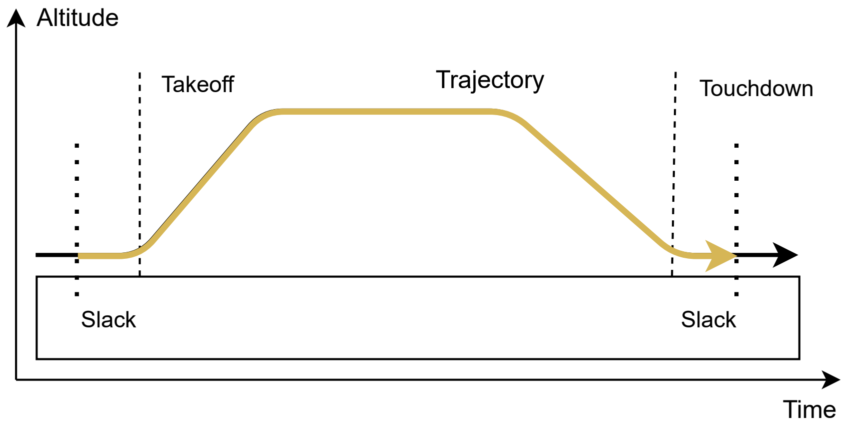

Figure 6.1: Trajectory identification mechanism based on the timestamps from the flight plan.

6.2.3 Network Manager - OpenSky

NM's flight plan data and flight data can be merged using the flight plan ID, as explained in the previous chapter. However, joining these data with surveillance data requires a more elaborated integration strategy:

- Flight plan data and flight data do not always include the ICAO24 of the aircraft. However, the callsign of the flight is always present.
- Surveillance data can have the callsign attribute missing, but the ICAO24 attribute is always present.
- The partitions defined for these datasets can not be integrated directly: a flight that started at the end of a day will continue at the beginning of the next day, so the corresponding surveillance data for that flight belong to the partitions of both days. Therefore, each partition of flights should be integrated with two partitions of surveillance data.

The missing ICAO24 values in flights datasets are those lost in the last messages for each flight. These values were filled during cleaning, so ICAO24 can be used reliably to join the two datasets. However, it is not sufficient: a single aircraft can make multiple flights during a day, so the surveillance data in a day for that aircraft comprises data from different trajectories. We can leverage the information about the flight plan to identify the individual trajectories in the sequence of surveillance data for that day. In particular, timestamps included in the flight data messages once the flight has ended can be used for this purpose. These timestamps describe the time the flight actually took off (*actualTakeOffTime*) or landed (*actualTimeOfArrival*). If the taxiing trajectory on ground is needed, *actualOffBlockTime* can be used to determine the moment the aircraft started moving at the terminal.

During the integration, we use the *actualTakeOffTime* and *actualTimeOfArrival* because the two use cases we are investigating utilize surveillance data of aircraft in flight, and the taxiing phases of the flight are not relevant. We also need to account for clock desynchronization and other factors that may cause the times included in the flight plan to be inaccurate with respect to the surveillance data. To this end, the time window defined by the above timestamps is extended by 10 minutes before and after to ensure that the complete trajectory is effectively identified as such, as shown in Figure 6.1.

6.2.4 OpenSky

The process of identifying individual trajectories can also be tackled using surveillance data only, and heuristics to determine when the flight starts and ends based on the flight state. For instance, the start and end of a flight can be determined using the *callsign*, *on_ground*, *speed* or *altitude* can be leveraged to determine the moments the aircraft lands or takes off [80], or when the aircraft actually stops at the gate of the destination airport (or starts moving from the gate during the departure). The GPS position of the aircraft can also be used with similar purpose, if the precision is good enough. The time patterns can also be helpful, since once the aircraft stops, it does not emit ADS-B messages.

However, all these heuristics are vulnerable to incorrect or missing data, which can be alleviated to a degree by using a combination of heuristics that help to overcome the errors or the lack of a value for an attribute if the problem is sporadic. Nevertheless, if no surveillance data is available, the heuristics become ineffective. For example, in Madrid Barajas-Adolfo Suárez airport it is frequent that almost no state vectors are generated under 2k feet once the aircraft is heading to the runway, which means that the end of the trajectory and the touchdown (the first contact of the aircraft with the ground in the runway) is not described in the surveillance data sequence. Thus, none of the heuristics explained above would be able to determine the end of the trajectory.

6.2.5 TAF reports

There are multiple possibilities for integrating surveillance and forecast weather data. The described weather conditions in these reports are relative to a particular position; e.g. at the terminal, or in intermediate areas during the flight. Influenced by the use case explained in Chapter 7, in this Thesis we associate each surveillance data point (each state vector) with the current weather forecast at the destination airport at the moment of generation of that state vector. Recall that several TAF reports of different types can be valid at the same time, as explained in Chapter 4. Then, it is required to reconcile the different forecasts in all these reports to characterize the weather conditions that can be expected at the destination airport.

To this end, the different types of reports are prioritized: routine reports define the base weather conditions, which are valid unless stated otherwise. The changes in these base conditions can be originated in three ways:

- New AMD or COR reports: AMD reports partially overwrite the contents in routine forecasts, while COR reports replace the base forecast entirely.
- Changes announced in the routine report, which can be permanent (FM and BECMG) or temporary (TEMPO).
- New routine TAF reports replace the last routine report and invalidate any other forecast change.

Taking these considerations into account, FM and BECMG clauses take precedence over the underlying base forecast, and are themselves overridden by TEMPO clauses, which describe more significant variations. When an AMD (Amendment) or COR (Correction) report is issued, it updates or corrects the base forecast accordingly, although the precedence order is not altered. Furthermore, the publication of a new routine TAF report invalidates all previous forecasts for the same validity period, establishing a new base forecast and new associated change periods, if any.

Figure 6.2 illustrates a feasible scenario where the forecast at a given time results from the overlap of several valid forecast elements. When a BECMG clause becomes valid (shown in blue), it supersedes the conditions defined by the base forecast. This period is temporarily overridden by a TEMPO clause (shown in red), which describes short-lived weather variations. Once the temporary event ends, the forecast reverts to the conditions defined by the BECMG clause, lasting until the end of the validity of the base forecast. Later, a new TAF report (shown in green) establishes a different base forecast, which may include new FM, BECMG, or TEMPO clauses describing further expected changes. Once the current forecast is reconstructed by resolving overlapping TAF reports and their associated amendments and changes, we can accurately identify the prevailing weather conditions at any point in time. This, in turn, allows us to associate each state vector with the relevant forecast according to its timestamp.

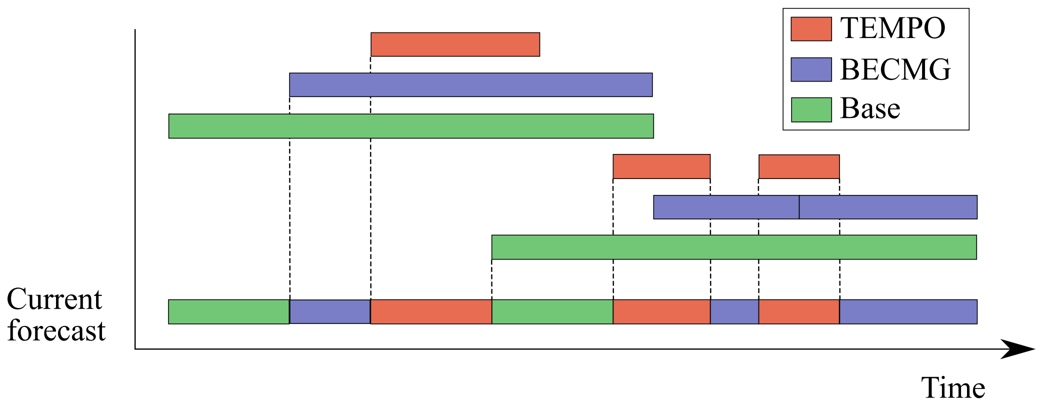

Figure 6.2: Prioritization of TAF reports. Routine reports (green) set the expected weather conditions. The gradual changes defined by BECMG reports overwrite the base conditions until the end of its validity period. Temporary events (TEMPO) have a short life span and overwrite the other forecasts.

6.3 Data quality problems of trajectories

Once reconstructed, the trajectory can show several problems that could not be identified before in the data workflow. These problems arise when each data point is put into a context, and are related to the availability of data (e.g. the density or frequency of the data, or the presence of gaps), the consistency of the data (e.g. incorrect or inconsistent values of individual data points, or an incorrect order in the data sequence) or the expectations on the characteristics of the trajectories (e.g. some trajectories might be too short or be incompatible with their pursued use). In this section, we discuss these problems and how they can be tackled.

6.3.1 Data density

In previous chapters, it was indicated that ADS-B has a theoretical update rate of 1Hz, and OpenSky Network generates a new vector every 5 seconds if the necessary ADS-B data has been captured. However, these are ideal values – we have seen that many factors can influence the actual availability of these data and reduce the volume and frequency of generated records. This fact becomes more prominent after trajectory identification: trajectories can have segments where the density of state vectors (that is, the number of state vectors per unit time) is strongly diminished below the ideal value of approximately 20 vectors per minute. As a result, the density of the data sequence is not homogeneous, which in some cases can hinder the application of these data. For instance, many deep learning methods to process time series data (sequences of data where each record is associated to a timestamp) require the data to be consistent in terms of the time separation between the elements. In these cases, the differences in the frequency of the records may harm the success of the training processes and the effectiveness of the developed models.

In other situations, even if the density is consistent along the sequence, it may not be the most convenient for a particular use case. Going back to the example above, some models do not benefit from having high density data, either because of the extra computations needed to train on a large amount of data, or because it could overfit on data that have too much detail, and fail to generalize the task it aims to perform. On the contrary, it is possible that a higher density of data is required, like if it is needed to fill the gaps in the trajectory or to make up for several state vectors that were lost. In these cases, it might be necessary to tweak the density of the data through different methods. Depending on the situation, one may want to reduce or to increase the density of a trajectory or a segment of it.

Reducing the data density requires that some vectors are removed from the sequence, providing a kind of summarized version of the trajectory. This process is called downsampling. While there exist different strategies to implement downsampling, in this project we opted for implementing a simple mechanism called bucketization. According to this method, the state vectors in the trajectory are divided into buckets of ST seconds (sampling time) according to their timestamp. Then, one vector is chosen for each bucket, and the rest of vectors in the bucket are discarded. The selection of the vector can be implemented through different strategies, like choosing the first or the last in the bucket, the nearest vector to the centre of the bucket, or the centroid of the bucket (the vector that is most similar to all other vectors in the bucket). These strategies are illustrated in Figure 6.3. Another strategy could be to construct a synthetic state vector that summarizes the vectors in the bucket, but creating synthetic data should be considered with care since it could lead to different problems and anomalies in the data.

The complementary operation is oversampling. In this case, new vectors are generated to fill in the spaces between the existing vectors. Once again, there exists different methods to tackle this synthetic data generation problem, although one of the simplest is interpolation. Trajectory data is a time series, so each element in the sequence is related to the adjacent elements. This dependence can be leveraged to apply interpolation techniques, such as linear, polynomial or spline interpolation, to find feasible intermediate values between the existing elements. However, synthetic data generation is inherently complex, and its use requires a very careful consideration. In the context of this project, we use interpolation on specific attributes to recover anomalous values in existing vectors, such as the altitude.

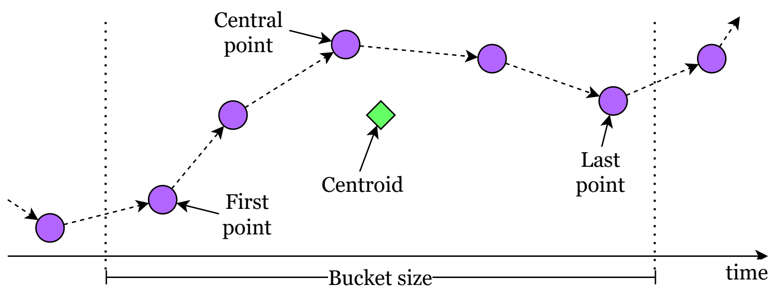

Figure 6.3: Different strategies to choose a vector in a bucket.

6.3.2 Gaps

During a flight, it is possible for more than just a few state vectors to be lost, resulting in significant gaps in the trajectory data. As explained in Chapter 4, ADS-B data availability depends on the presence of receivers within an effective range. Adverse weather, geographic obstacles, or the distance to the receiver may prevent data reception, causing entire segments of the trajectory to go unrecorded and leaving gaps in the data sequence. Gaps can be either accidental or systematic. Accidental gaps are caused by temporary factors, such as weather or equipment malfunction. Systematic gaps, however, result from persistent issues like the absence of coverage in specific areas, affecting all flights passing through them. The nature of the gap influences the severity of the problem: trajectories with accidental gaps can be replaced by other trajectories once the problem has passed. However, systematic gaps lack the possibility of replacement. Additionally, the size and frequency of the gaps are relevant factors to assess the impact of gaps in a trajectory. Larger gaps significantly reduce the certainty about the true state of the aircraft during the affected periods.

Like missing vectors, gaps can be addressed using data generation techniques such as interpolation or regression methods, as the state of the flight before and after the gaps are known and can be used to estimate the intermediate states. The trajectory prediction methods can be applied with this purpose (these methods are reviewed in Chapter 8). However, estimating entire flight segments is even more challenging than individual state vectors, and the results may be far from satisfactory. A better alternative could be incorporating data from complementary sources of surveillance data to fill the gaps, since OpenSky is limited by the coverage of its receiver network. Alternative surveillance data providers may offer coverage in regions where OpenSky does not.

6.3.3 Anomalous values

In the previous chapter, we said that some attributes can show incorrect values, which in some cases correspond with unfeasible values that could be immediately identified as such. An example is the altitude of 126,400 feet that has been observed in some state vectors. Since we know that commercial flights rarely exceed 42,000 feet of altitude, these errors can be easily found and removed. However, the error can be much more subtle, although some of them become more noticeable after the trajectory identification.

As mentioned earlier, trajectory data can be regarded as time series. Provided that the data are dense enough (i.e., there are no large gaps between elements), each data point should be consistent with its neighboring points. The changes between elements should be gradual, resulting in every element being either quite similar to their neighbors, or showing a similar trend than previous elements. For example, an aircraft flying at 800 kilometers per hour would travel about a kilometer in 5 seconds; this means that the position difference between two consecutive state vectors should not exceed this amount (provided that there exist vectors every 5 seconds). We can leverage this fact to define different methods to identify strange or atypical values, which we also call outliers (values that are outside of the expected distribution of the data). There exist more advanced methods, such as median filters or Kalman filters, that can be experimented with to process these trajectories.

In the context of this project, we search for outliers in the altitude and the position, since 3D position plays a relevant role in the studied use cases that are presented in Chapters 7 and 8. In particular, we apply a combination of median filters to identify anomalous values and treat them accordingly. In the case of position, the state vector is removed, since the error could be caused by an error in the onboard GPS, and could compromise the validity of other attributes of the vector (such as the speed). If the affected attribute is the barometric altitude, it could have been caused by an occasional error on the altimeter (e.g. it was being set up or adjusted by the pilot), and its value could be estimated based on the geometric altitude.

6.3.4 Delayed vectors

In Chapter 4 we introduced the problem of delayed vectors. The heterogeneity of the receivers that submit ADS-B data to the OpenSky Network can cause different problems, specially when it comes to the timestamp of the ADS-B messages and the delays that can occur before OpenSky consolidates all ADS-B data, and delayed vectors are one of the most prominent problems. These vectors are constructed using outdated ADS-B messages, which were assigned timestamps that are much later than they should be. Consequently, they represent true positions and states of the flight, but are placed out of their position in the chronological sequence of data, as illustrated in Figure 6.4. Figure 6.5 shows a real example of this situation: the altitude profile of a chronologically ordered trajectory where several vectors generated during the descent phase were delayed. Seemingly, the aircraft is jumping up and down with differences of more than 2,000 feet, a description that clearly does not match the reality of the flight. A similar observation can be made when we analyze the latitude and longitude profiles: the aircraft is suddenly displaced several hundreds of meters back and forth. After sorting the state vectors according to their position, the trajectory makes much more sense, as shown in Figure 6.6. Identifying these situations is sometimes difficult, and only becomes more apparent when one of these magnitudes (altitude, latitude or longitude) are represented in relation with time. Other representations, like the visualization on the map (see Figure 6.7), effectively hide the problems in the timestamp.

As a result, the trajectory describes changes in the descriptive attributes of the aircraft state that are unrealistic. Therefore, they could be detected by outlier detection methods and classified as anomalous vectors. However, timestamp errors have been identified as systematic, potentially affecting a significant number of vectors and trajectories, so their removal or overwriting through value imputation can have a large impact on the quality or density of these trajectories. Hence, this problem should be tackled on its own.

Detecting and recovering delayed vectors poses multiple challenges:

- Distinguishing delayed vectors from vectors with anomalous attribute values.

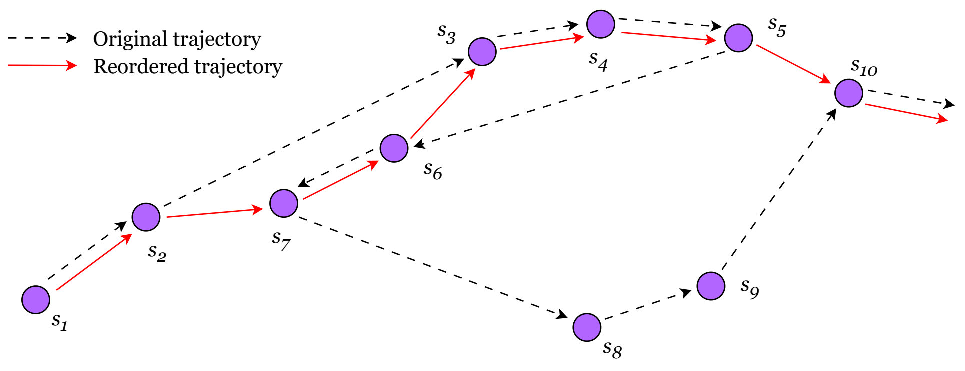

Figure 6.4: A schematic example of delayed vectors and outliers. The path in black indicates the original 4D trajectory ordered by the timestamp of the vector, with two delayed vectors s_6 and s_7. The reordered trajectory (in red) resituates those vectors, and skips the s_8 and s_9 positions since they are inconsistent with the rest of the trajectory (outliers).

- Determining the correct position in the sequence for these delayed vectors.
- Identifying which vectors have been resorted, as moving a vector back in the sequence shifts the position of subsequent vectors, but those vectors themselves have not been resorted.
- Assigning new timestamps to the reordered vectors.

6.3.5 Trajectory selection

Some trajectories might be discarded if their quality is insufficient and can not be fixed, or they just do not met the requirements for a particular use case. An exhaustive list of the potential problems that could affect the trajectory data is beyond the scope of this chapter, but below we list some additional problems we have found when working with the trajectory data and are taken into account in later operations with these data:

1. Trajectories that are too short (have too few state vectors) are excluded since the description they provide of the flight path has a severe lack of details, or they are incompatible with some of the techniques applied later, which require long sequences of data.
2. Trajectories whose origin and destination are the same airport (failed flights that had to return to the departure airport).
3. Trajectories whose maximum altitude is too low, which can correspond with test or training flights, unmanned flights (drones) or helicopters, since they are out of the scope of this study.
4. Trajectories with insufficient coverage of the flight path, e.g. trajectories with severe losses of data in the form of gaps.
5. Trajectories that are missing too many vectors at the start or at the end, which causes the trajectory to not describe the most critical phases of the flight: the ascent and the descent and landing of the aircraft.

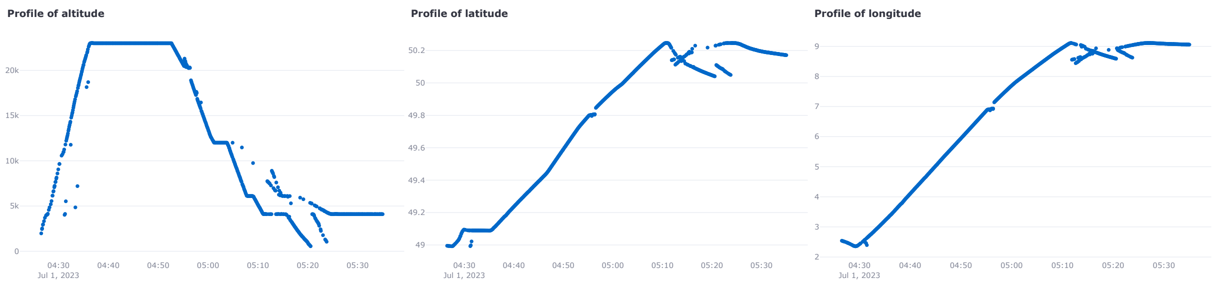

Figure 6.5: Flight profiles of flight AT02523950 on July 1, 2023. Several vectors at the end of the trajectory show altitude, latitude and longitude values that are inconsistent with the progression of the trajectory due to vectors being assigned incorrect timestamps.

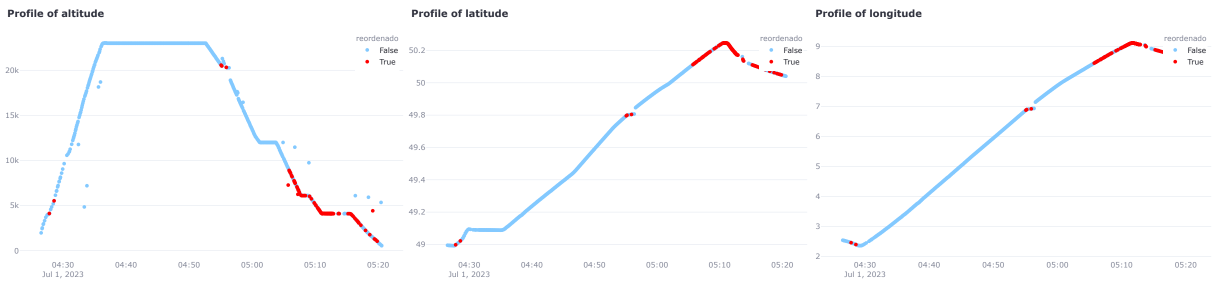

Figure 6.6: Example of delayed vectors (flight AT02523950 on July 1, 2023) after sorting the state vectors of the trajectory. The delayed vectors have been identified and assigned a consistent timestamp. Some vectors show an incorrect altitude value due to errors in the surveillance data.

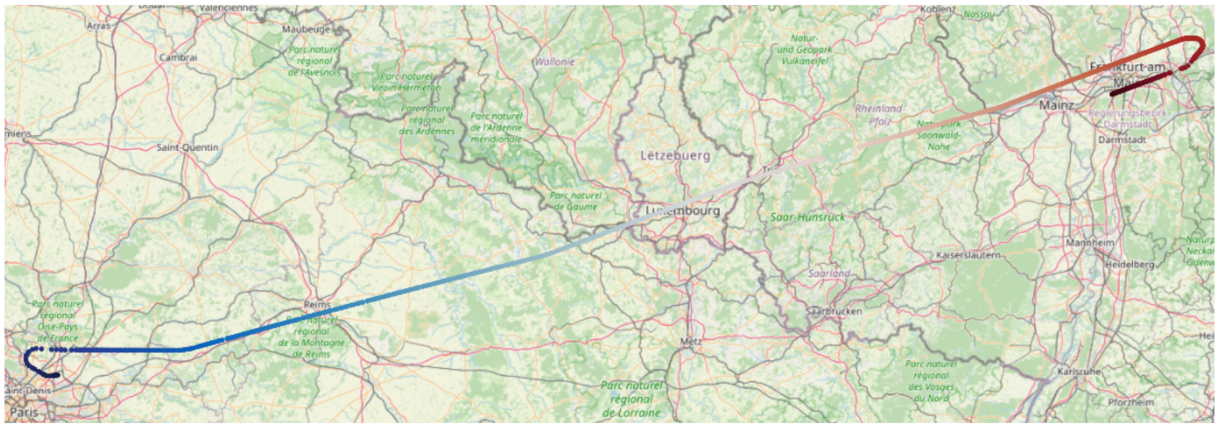

Figure 6.7: Representation in the map of the flight AT02523950. The state vectors are color-coded according to their timestamp; the smooth change in color reflects the correct progression of the timestamps.

Trajectories should be annotated with different metadata and quality metrics to facilitate the identification of such common problems in the trajectories, enabling the quick identification of which trajectories can be used for a particular use case. In the following sections we describe the set of metadata we have developed to assess the validity of the trajectories data for our purposes.

6.4 Trajectory quality metrics

In this section we define a set of data quality metrics that help to describe a trajectory following different criteria. Table 6.1 shows the list of defined metrics for trajectory data.

Generic	
Number of records	Number of state vectors associated to the trajectory
Completeness	Ratio of non-null values for each attribute in the time series
Cardinality	Number of unique values for each categoric attribute
Flight path characterization	
Geodesic/direct distance	Distance in kilometers between the origin and destination airports in a straight line; e.g. the minimum distance traveled by the aircraft.
Actual flight time	Effective flight time in seconds defined as the difference between the actual arrival time and the actual takeoff time reported in the flight plan. The trajectory duration for that flight should be at least this value.
Coverage	
Total duration	Difference in seconds between the timestamps of the first and last state vectors
Total distance	Accumulated distance in kilometers defined by the sequence of reported positions
Distance to origin	Distance in kilometers between the first vector and the departure airport
Distance to destination	Distance in kilometers between the last vector and the destination airport
Initial segment	Flag indicating if the start of the trajectory is missing
Final segment	Flag indicating if the end of the trajectory is missing
Last altitude	The last reported altitude in feet before touchdown, to help detecting missing data in the last phase of the descent.
Density	
Density	Number of vectors per traveled distance unit
Time separation	Mean and StD of the time differences in seconds between all adjacent vectors
Distance separation	Mean and StD of the distances between all adjacent vectors
Continuity	
Gaps	Characterize all gaps in the sequence: start and end vectors, time elapsed in the gap, ratio between gap time and total flight time
Continuous segments	Characterize all the segments in the sequence under continuity conditions: start and end vectors, time elapsed, ratio between continuous time and total flight time
Low density segments	Characterize all the segments in the sequence under low density conditions: start and end vectors, time elapsed, ratio between low density time and total flight time
Thresholds	
Thresholds	Values used to determine continuity and low density conditions, and the minimum threshold for gaps.

Table 6.1: Summary of trajectory quality metrics.

6.5 Improving trajectory quality

Some errors only become apparent in the context of a trajectory. In previous sections we have tackled individual datum errors, such as missing values or out-of-domain values. However, even in cases where the measurements included in a state vector are present and look feasible, they may be inconsistent with the situation described by other state vectors in the trajectory. For example, it is possible to find that a state vector reports an altitude of 42,000 feet (corresponding with a feasible cruise altitude) a few minutes after it started descending to approach to the destination airport. Obviously, an aircraft flying at 16,000 feet to approach to an airport can not be at 42,000 feet five seconds later. The same situation comes when a state vector locates the aircraft at a coordinates it crossed several minutes ago, like doing a quick jump back in the trajectory, and then returning to its advanced position. These and other facts are strong indicatives of a delayed vector.

The disorder in trajectories from a time perspective is caused fundamentally by the assignment of incorrect timestamps to one or more vectors in the trajectory. Thus, to carry any data analysis that depends on time, such as working with 4D trajectories, it is a necessity that these incorrect timestamps are identified and corrected. Since using the timestamps of the state vectors is not a reliable sorting criteria, we draw upon position data to find the correct order of the state vectors, identify which state vectors are out of their natural place, and try to fix their time information based on their new position within the trajectory.

However, there are cases in which different errors in the data may cause these inconsistencies in the data, even if the time information is correct. These errors are more common in position data (for example, due to positioning errors of the GPS), and specially in altitude data. For instance, it is common that an incorrect configuration of the aircraft's barometer or sudden changes of the atmospheric pressure can cause inconsistent variations in the reported barometric altitude. These problems also should be detected and solved to enable a reliable usage of the position data.

6.5.1 State vector sorting as an instance of the travelling salesman problem.

The Travelling Salesman Problem [49], or TSP, is a classic combinatorial optimization problem where the shortest path that includes all the nodes in a graph has to be found, under the condition that each node can only be included once, except for the first one. The TSP uses the analogy of a salesman that needs to visit a set of cities, and he needs to find the shortest tour (that is, a path that visits them all and goes back to his initial position): "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?". In this analogy, cities are the graph's nodes, and the distances between cities correspond with the weights of the edges between the nodes. If we consider that all cities can be visited from any other city, the underlying graph is complete (that is, all nodes are linked to all of the remaining nodes); and if these edges can be traveled in either way, then the graph is also undirected.

Vector sorting based on the reported positions in aircraft trajectories can be interpreted as an instance of the Travelling Salesman Problem (TSP), since the trajectory can be defined as a complete, undirected, weighted graph in which there are no cycles. Let us examine these affirmations. First, the graph is complete: there are no restrictions to travel from any point in the airspace to any other point, since aircraft are not bound to travel through a defined pathway like other vehicles. Second, the edge between a particular pair of points can be weighed though different means (being the euclidean distance in three dimensions a common choice), and the cost of travelling in either direction is the same. Third, each state vector is unique in the sequence of states (the same state vector is not emitted twice), so the same state can not be "visited" more than once. These characteristics allows us to find the shortest path (that is, the correct order of the trajectory vectors) using techniques developed to solve this problem. In the following sections, we discuss some of these techniques.

6.5.1.1 Approaches to TSP

Since TSP is a NP-hard problem, exact methods are not feasible for long sequences and an approximation method is required. In particular, we discuss two well known algorithms: Nearest Neighbour, and 2-opt (an instance of the more general k-opt).

Nearest Neighbour (NN) is a greedy algorithm where an initial node is chosen, and the closest unvisited node is iteratively visited until all nodes have been chosen once. This method has a time complexity of $\mathbf{O}(N^2)$, and can lead to non-optimal solutions depending on the chosen initial node. However, it is simple to implement, and can be effective in some scenarios.

k-opt is a local search optimization heuristic that starts with an initial tour (a path travelling across all nodes), and iteratively tries to identify a shorter tour until a local minimum is reached, although the probability of finding the global minimum is higher than using NN. The process starts with a random tour (albeit different heuristics can be applied to improve this initial our), and iteratively changes k non-contiguous edges. If the length of the tour decreases as a result, the change is applied and the iteration continues with the new tour. Once all changes have been evaluated and the length does not decrease with any of them, then the optimal solution has been reached. However, it is important to note that the resulting tour may not correspond with the global optimum. 2-opt and 3-opt are the most common variants of the k-opt heuristic. The greater the k, the higher the number of replacements that have to be calculated on each iteration, so depending on the size of the graph higher values of k can make the task unfeasible (starting with $k = 5$, the running time becomes exponential [51]). We can retake the example in Figure 6.4, and apply 2-opt to the initial trajectory (see Figure 6.8). In this case, we do not remove any outliers. First, we define the trajectory as a graph, where the positions are the nodes of the graph, and define the edges between them according to their initial order by their timestamps. Then, we link the first and last nodes to create a tour, a closed path. Recall that this graph is complete and undirected, but the edges have different weights depending on the selected cost function. Let us use the Euclidean distance for this example.

On each iteration, we perform a 2-change: we have to choose two non-consecutive edges, remove them, and then reconnect their nodes in a different way. If the length of the resulting path is smaller than the previous one, the 2-change was successful. If not, the iteration continues. Obviously, the number of combinations to evaluate on each iteration grow with the number of nodes in the graph. In particular, for a graph with n nodes, at each iteration we can try up to $n(n-2)$ changes [33]. For each candidate change, we also need to calculate the length of the resulting path, which depends linearly of the number of nodes. Moreover, for a particular pair of edges, there is no guarantee that any of the changes will result in any improvement. In fact, to the best of our knowledge, the theoretical bounds of applying 2-opt is still under discussion.

In our example, we choose to remove the edges $\overline{s_5 s_6}$ and $\overline{s_8 s_9}$, and define two new edges $\overline{s_5 s_8}$ and $\overline{s_6 s_9}$. In this case, the resulting path is not shorter, so this candidate change is not applied.

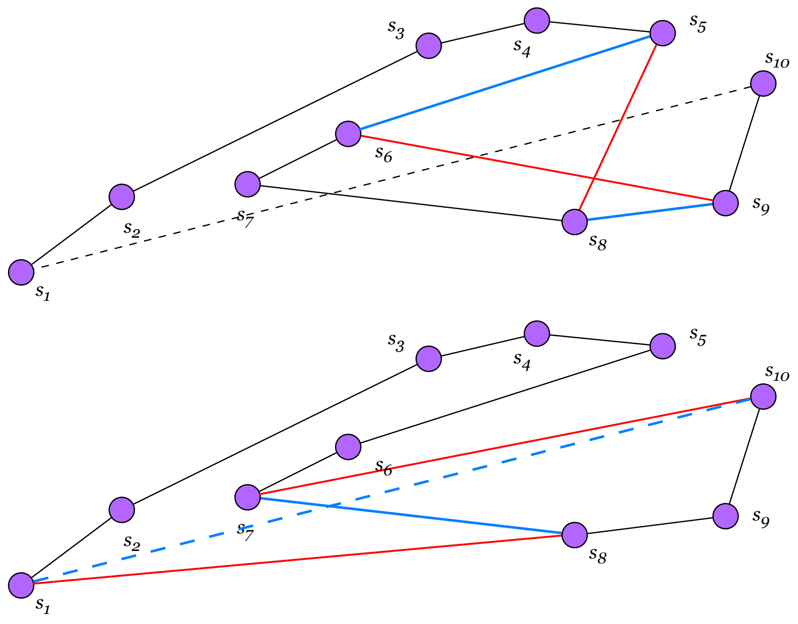

Figure 6.8: Two candidate changes of the 2-opt heuristic. The two edges in blue are replaced with the two edges in red, and the total cost of the new path is calculated. The top change does not improve the path's length, while the bottom change does (since it clears a crossing of edges).

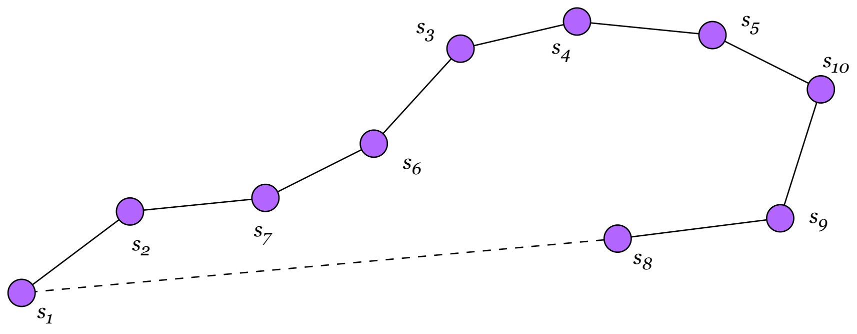

Figure 6.9: The result after a full run of the 2-opt algorithm using only the distance between points to weigh the edges.

After several iterations, we can reach the result in Figure 6.9. However, although in this case it is the optimal solution for this graph, this result is not guaranteed since 2-opt can fall into a local minima, where none of the 2-changes improves the length of the tour and no further progress can be made.

At this point, it is easy to see that applying 2-opt to aircraft trajectories can be a burden, since each trajectory usually comprises several hundreds or thousands of positions (nodes). Thus, we have made several adaptations to alleviate the computational cost of 2-opt for the particular case of aircraft 4D trajectories, which are discussed in the next section.

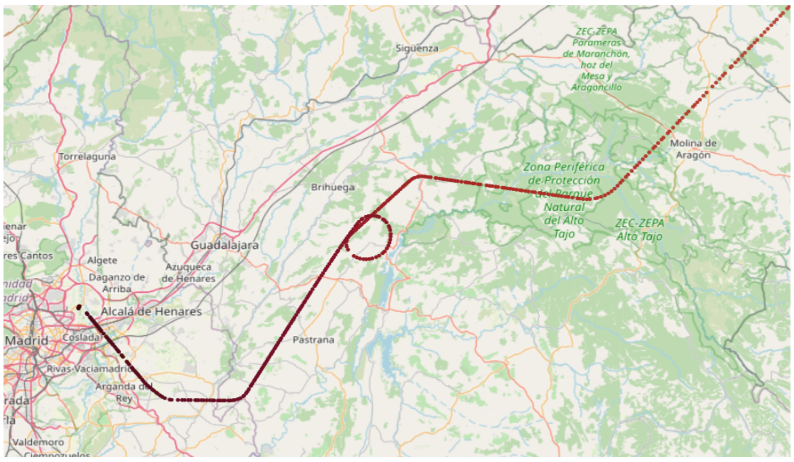

Figure 6.10: Flight AT02845701 on July 10, 2023. The aircraft approaches from the east and enters the designated area for holding procedures located to the north-east of LEMD airport. The aircraft performs one loop, and then it is cleared to attempt the landing.

6.5.1.2 The problem of loops and holding patterns

In Chapter 2 we introduced holding procedures, in which an aircraft performs an stationary pattern near to the destination airport in wait to the controller to allow the landing operation. These manoeuvres are always performed in designated areas near the airport. As shown in Figure 6.10, during a holding procedure the trajectory of the aircraft crosses with itself, and it can overlap if the manoeuvre is composed of multiple loops. However, loops can appear even in common manoeuvres during the ascent or descent phases, for example. 2-opt is not able to solve these situations: this heuristic naturally undoes this kind of crossings between the edges, so it is not possible to apply the 2-opt heuristic based on distance directly on trajectories with such patterns. In these cases, the 2-opt converts the loop into a “sac”, in which the subsequence is reversed between the points with crossing edges, as shown in Figure 6.11 with the arrows in red. Other approaches, like Nearest Neighbours (designated in the figure as NV, Nearest Vectors), can find some trouble at finding the next vector. While the next vector is often the closest one, this is not always true, specially in a loop. In the figure, the state vector s_7 is closer to s_2 than s_3 (which is the correct state vector) or even s_6 (which is the state vector indicated by 2-opt). In fact, if s_8 was closer than s_3 to s_7, the state vectors in the loop would be completely ignored until the rest of the trajectory has been processed (since NN looks for the closest unvisited node).

Several approaches can be applied here. The first one is taking no action – if a loop or a holding pattern is detected, the segment of the trajectory containing the loop is kept in the initial order. Another option would be applying 2-opt on segments of the trajectory so the cross is never included in any of these segments. In this way, 2-opt can process the positions in the loop without having to resolve the crossing edges. However, the number of positions and the characteristics of the loops vary from trajectory to trajectory, and it is difficult to automatize the construction of suitable segments for every trajectory, since multiple factors can affect it (coverage losses, delayed vectors, etc). Using small segment sizes could avoid such problems, but working with such segments can make the locality of the process excessive, and make the process ineffective. Another approach is including the altitude in the distance calculation, so

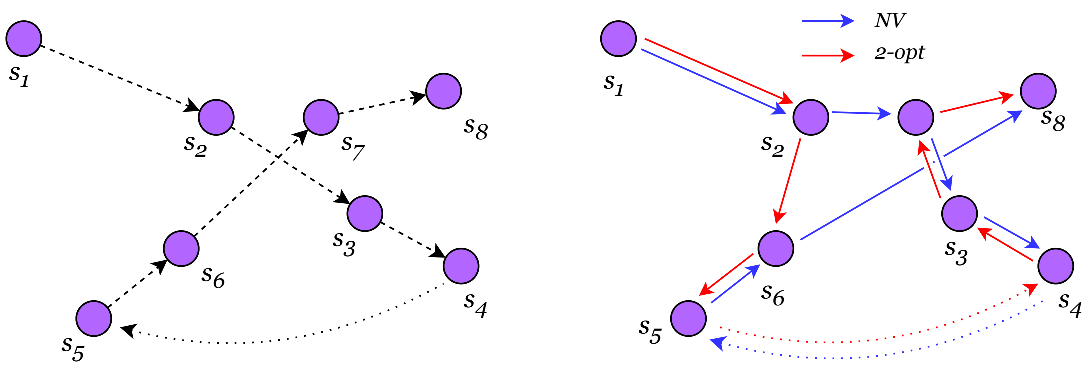

Figure 6.11: Example of a simple loop. The methods based on distance (Nearest vector and 2-opt) fail to follow the correct path.

the apparent low distance in 2D between the points with crossing edges become more prominent thanks to the vertical distance between these points. However, in our experience this approach has not been effective: we have found that the altitude values in the state vectors is subject to artifacts and missing values, particularly near the airport, which hinders the calculation of the distances, and therefore of the length of the path. Moreover, the differences in altitude are very small (in the order of tens or hundreds of feet), and have little impact in comparison with the vertical distances between those positions.

Based on our observations, we have developed several heuristics to tackle these situations, albeit there are still some challenging scenarios (such as holding procedures with multiple loops), which we detect and left without sorting. In the following, we discuss these heuristics.

1. Identify the presence of a loop (whether simple or multiple) by evaluating the accumulated rotation of the plane in areas where such manoeuvres are more probable; i.e. the manoeuvreing segment near the airport.

2. Implement a cost function, instead of a distance function, that accounts for the inertia of the movement so any candidate "next position" other than the correct position has a larger weight.

3. Parametrize our efficient 2-opt implementation to adapt to these patterns.

6.5.1.3 Cost functions

To assign weights to the edges between nodes, a cost function is required. An intuitive option are distance functions, such as the Euclidean distance or a great circle distance such as the haversine distance. However, these functions may behave badly in some situations and alternative methods could be needed. Let us analyse the application of these techniques to flight trajectories.

Distance functions. The distance functions give a measure of the distance between two points based on their coordinates in a particular space. The nature of aircraft 4D trajectories suggest using either the Euclidean or the great cicle distances. The Euclidean distance between two points is the length of the segment between the two points. The great circle distance, on the contrary, indicates the length of the arc between two points on the surface of a sphere. Both can be applied to measure the distances between two positions of an aircraft, although the use of

Euclidean distance is discouraged if the points are very far apart. The aircraft's movement takes place in a spherical shell with a depth of, at most, 42,000 feet (around 14 kilometers) above the surface of the Earth, with an inner radius of 6,378 kilometers. This means that the movement between two positions is not linear, but follows the spherical surface of the earth at a particular distance (the flight altitude). Hence, the haversine formula is more accurate at measuring the distances traveled by an aircraft. However, if the distance between the two points is small, the Earth radius is large enough to approximate the Earth surface (the portion that is traveled above during the movement) to a plane, and these distances are rather interchangeable (at least for our purposes). The difference become more noticeable as the two points are farther, in which case the haversine formula would be preferred. This problem is exacerbated by the fact that the Earth is more like an ellipsoid, and the movement is not equal in all directions. In fact, the error of the Euclidean distance is bigger in the latitude than in longitude. Given the geographical scope of our study, which implies the study of long trajectories in long distances, we opted for using the haversine formula to determine the distances between positions.

Another aspect to discuss is whether to use two (latitude and longitude) or three dimensions (latitude, longitude and altitude) to describe the points. If the altitude is not considered, the measured distance would not be between two points, but between a projection of those two points in 2D, which would be always lower or equal. Again, for points that are close the difference would be negligible, and grow with the altitude difference between the points. However, the altitude difference will always be much lower than the difference in the other two dimensions (since the ascending and descent should be smooth); thus, the impact of the third dimension in the distance is reduced in the Euclidean distance. In the haversine distance, the altitude would have to be added to the radius of the sphere, and its value is negligible with respect to the Earth radius. Moreover, we have found that altitude is frequently not available in all vectors in the trajectory, which would difficult the calculation of distances. Therefore, we decided to use only two dimensions, as it is precise enough for our purpose.

In conclusion, the formula we use to calculate distances is shown in Equation 6.1, where (ϕ_1, λ_1), (ϕ_2, λ_2) are the coordinates (in radians) of the two points, and R the average radius of the Earth (6378 kilometers or 3959 miles).

$$hav((\phi_1, \lambda_1), (\phi_2, \lambda_2)) = 2R \arcsin \sqrt{\sin\left(\frac{\phi_2 - \phi_1}{2}\right)^2 + \cos(\phi_1)\cos(\phi_2)\sin\left(\frac{\lambda_2 - \lambda_1}{2}\right)^2} \tag{6.1}$$

Since the 2D position coordinates in our data is expressed in degrees, these can be easily converted into coordinates in radians as:

$$(\phi, \lambda) = (\pi x/180, \pi y/180) \tag{6.2}$$

Cost functions. However, the previous solutions only take into account the initial and final positions, without accounting the role of the "inertia" of the aircraft movement. Let us suppose that an aircraft is moving over a plane and is at the point s_1(0,0); let s_2 and s_3 two points, (5,0) and (0,5), and the movement vector of the aircraft is $\boldsymbol{v} = (1, 0)$ (as shown in Figure 6.12). Would the cost of arriving to s_2 be equal to the cost of reaching s_3, even if they are at the same distance from s_1? The answer is no: the aircraft can not perform sudden 90 degrees turns, and the traveled distance to arrive to s_3 would be larger (as it would need to make an arc) instead of travel straight to s_2. In fact, the distance difference would depend on the speed of the aircraft: the higher this speed, the larger the radius of the circle needed to turn. Therefore, the path

$\overline{s_1 s_2}$ is more likely than the path $\overline{s_1 s_3}$ according to its initial direction, since it requires a lower traveled distance.

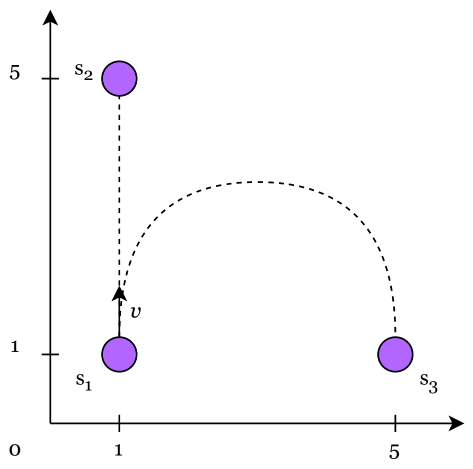

Figure 6.12: The path of an aircraft to a point depends on the speed and the direction. To reach the point C, the aircraft should perform a turn since it can not change its initial direction instantly. On the contrary, it can reach the point B by following a straight line.

This kind of movement can be modelled mathematically, but it is far from trivial (an example of these methods are Dubins paths[1]) and would require to know the turn rate of the aircraft (to calculate the maximum curvature of the trajectory), which depends on the aircraft model, among other factors. Hence, we have developed a simple method to penalize excessive turns in favour of more straight movements, where the cost is defined as the distance between the two positions, penalized by the multiplication of an exponential factor dependant on the angle between their respective directions. In this way, equal directions receive no penalties, and the penalty increases

$$Cost(\boldsymbol{v}_1, \boldsymbol{v}_2) = hav(\boldsymbol{v}_1, \boldsymbol{v}_2) \cdot \exp\left(\alpha/2\right) \tag{6.3}$$

Where α is the angle in degrees between the vectors v_1 and v_2.

We have explored two ways of introducing the direction in the cost calculation. The first one requires that we define the two vectors that join the current vector with the previous and the next vectors, with the same direction as the aircraft movement, and calculate the angle between them. The second one uses the difference between the *true_track* (that is, the bearing of the aircraft with respect to the north) values of two vectors. In both cases, larger angles would be penalized since aircraft tend to make slight to no turns during most of the flight, and sudden and significative changes in heading are not likely to occur. We can see an example of these techniques in Figure 6.13. Let us take a segment of a trajectory described by four state vectors. The vectors s_1 and s_2 are correctly sorted, but we need to attend to the direction of the aircraft to determine whether the next state vector is the s_3 or the s_4. To this end, we evaluate the coherence of the current direction with the direction required to reach each of the two candidate vectors.

[1] A Dubins path is the shortest curve between two points with a constraint on the maximum curvature, given an initial direction.

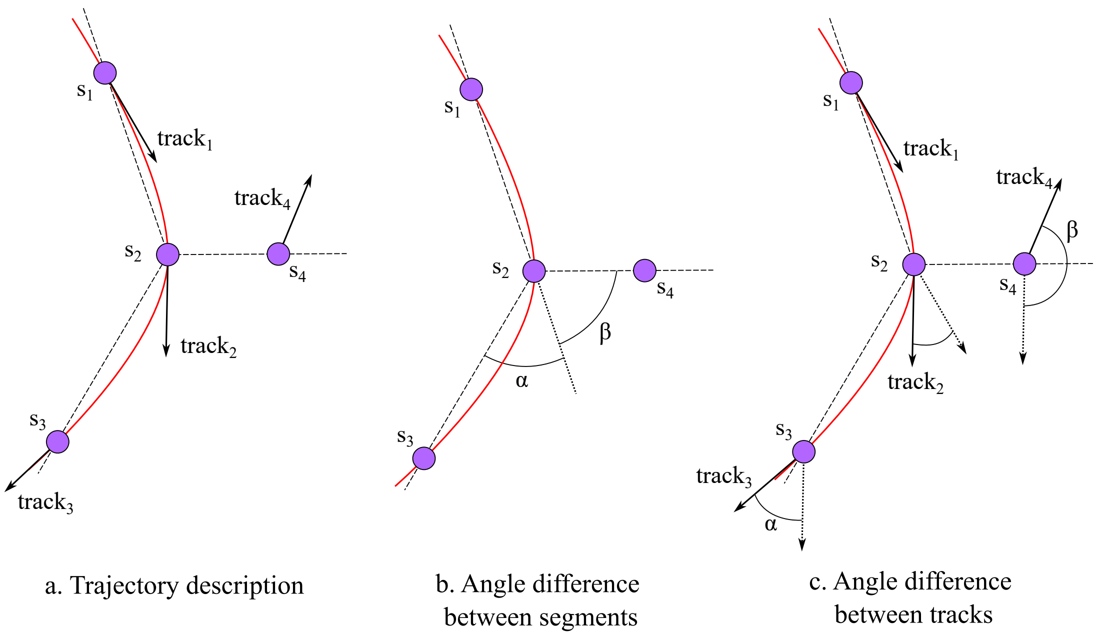
s1
track1
track4
s2
s4
track2
s3
track3
a. Trajectory description
s1
s2
s4
β
α
s3
b. Angle difference between segments
s1
track1
track4
β
s2
s4
track2
s3
track3
α
c. Angle difference between tracks

Figure 6.13: Example of failure of distance functions. The state s_4 is closer to s_2 than s_3, although their directions are inconsistent (left diagram). We can calculate the difference between the directions using the angles between the vectors connecting the positions (center), or the bearing in the state vectors for each position (right). The difference between α and β is larger in the second case.

With the first method, we define the vectors $\overrightarrow{s_2s_3}$ and $\overrightarrow{s_2s_4}$, and calculate their angles with the vector $\overrightarrow{s_1s_2}$. We can calculate the angle between two vectors v_1, v_2 as:

$$\alpha = \arccos\left(\frac{v_1 \cdot v_2}{|v_1||v_2|}\right) \tag{6.4}$$

For the second method, we leverage the feature that describe the bearing of the aircraft in each state vector (*true_ track*). Thus, the angle between the movement direction of two positions is:

$$\alpha = bearing_{s_2} - bearing_{s_1} \tag{6.5}$$

The two methods allows us to characterize the magnitude of the change in the direction of the aircraft between two positions. On the one hand, the method based on the bearing relies on the availability of its value, although it is mostly guaranteed to be defined in all state vectors. On the other hand, the calculation of the angle between vectors is computationally more expensive, and does not describe the actual direction of the aircraft at that position, as seen in the Figure 6.13. In consequence, we choose to use the bearing-based method.

6.5.2 Sorting algorithm

While ordering vectors in a trajectory is similar to TSP, these data have some characteristics that can be used in our favour to improve the probability of finding a better result, or reduce the computational cost of 2-opt by tweaking the initial tour. In the following, we discuss these characteristics, and how they can be leveraged to order our trajectories.

6.5.2.1 Flight segments

The trajectories can be divided in segments to improve the performance and computational cost of the sorting using our efficient 2-opt implementation. This division is related to the flight phases and airspace division described in Chapter 2, since these aspects determine the characteristics of the flight and its statuses along the trajectory. We can leverage these characteristics to apply several heuristics and tune the parameters t and q that regulate the 2-opt algorithm.

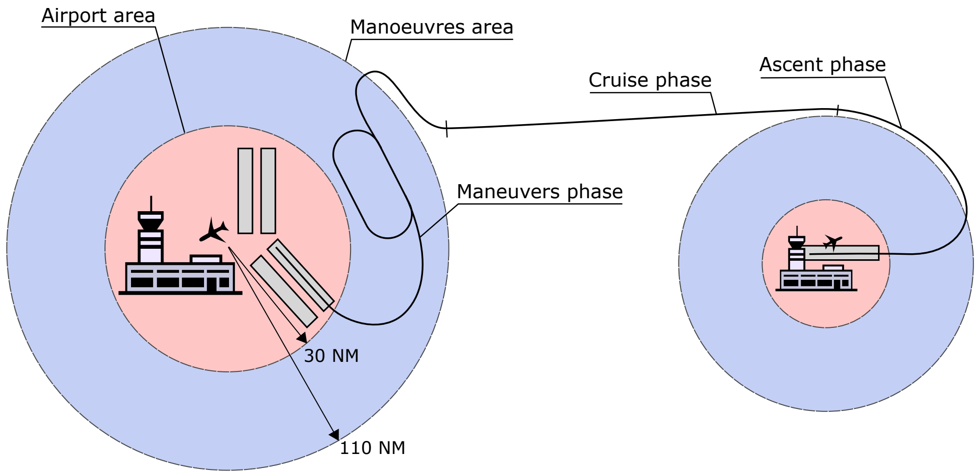

Figure 6.14: Segments identified in the flight.

In particular, we have defined four segments, defined by the following characteristics:

Ascent phase. During this phase, the aircraft moves steadily away from the departure airport, while gradually ascending until it reaches the cruise altitude. The direction in which the aircraft moves is determined by the runway used for take-off and the SID indicated by the control tower. This phase includes the manoeuvres to direct the aircraft towards destination airport, such as travelling around the airport TMA to avoid interference with other departing or arriving air traffic.

Cruise phase. After reaching the cruise altitude and speed, the aircraft moves towards the destination airport until it arrives at its TMA. In general, this movement has constant speed and direction, unless required by the planned route (for instance, moving around high obstacles or keeping close to the coast line). The aircraft gets closer to its destination with time, so the distance to destination is a reliable measure to leverage during the vector sorting.

Manoeuvreing phase. The manoeuvreing phase is the most complex for vector sorting. It takes place in the TMA of the airport, and may include approach manoeuvres, holding procedures, or even go-arounds (missed approaches). During this segment, the aircraft adopts different directions, altitudes and speeds. It also becomes more difficult to monitor the aircraft state, since more ADS-B messages are lost by network saturation due to the high traffic density in the area.

Approach phase. Once the pilot has been cleared to approach and attempt the landing operation, the aircraft leaves the manoeuvres area and heads towards the designated runway. This phase extends until the aircraft land at the runway and the flight ends.

Taxiing phase. A fifth segment can be defined with the parts of the trajectory that take place on the ground; i.e., the movement of the aircraft on the ground, from the terminal to the runway at the departure airport prior to takeoff, and from the runway to the terminal at the destination airport after landing. In these parts of the trajectory, the speed of the aircraft is low, and the state vectors are very densely distributed. However, these segments are not present in all trajectories, since it is common that there is no ADS-B data at the airport or close to it due to saturation or other factors.

These segments are aligned with the flight phases, and are determined by the different areas defined for organizing the airspace (TMA, en-route, etc.). In previous works the state of the flight has been used to determine these phases, either by the application of rules [107] or machine learning [108]. A different approach is using spatial operations to determine the area in which the flight is based on the specifications of the airspace. However, in this project the flight segment identification is used as a pre-processing operation and thus a more coarse approach is enough to fulfil our purposes. In particular, we divide the trajectories into the following segments based on the distance to the origin and destination airports of each position, according to the heuristics below:

- The taxiing segments are constructed with all vectors that are close to any of the two airports (a distance of <5 NM with respect to a point of reference in the airport), and were emitted on the ground. These segments remain unchanged, as they are not relevant to our objectives; however, they are preserved in the final trajectory for completeness.

- The approach or airport segment is defined at the destination airport, before the aircraft lands and after exiting the manoeuvreing area, as all vectors in the air that are close enough to the airport. We set this threshold at 30 NM, since within this area there are no manoeuvres beyond the required by taking-off and landing operations, by aircraft that are already committed to these operations.

- The ascent and manoeuvreing segments are identified based on the approximate size of their respective TMA. Since a TMA is defined by a bounded volume of the airspace, the distance to the airport at which the area starts is variable with the direction in which it is left or approached, and the SID or STAR used to leave or arrive to or from that area. However, for our application of 2-opt we define a generic distance to an airport to approximately characterize the true TMA of the airport. We have set this threshold distance at 110 NM for all airports, since we have observed it to be adequate for LEMD airport.

- The cruise segment corresponds to the intermediate segment between the identified ascent and manoeuvreing segments.

In the following steps of the sorting algorithm, we can leverage these segments to apply specific operations and configurations to each in order to improve the performance of the process.

6.5.2.2 Efficient implementation of 2-opt

Since the application of 2-opt on complete trajectories can become a burden computationally, we can approach it by dividing the process into smaller tasks. To this end, we can decompose the trajectory into subsequences, and perform the 2-opt heuristic on each of them. Hence, the number of nodes is reduced in each iteration, and the number of combinations is reduced drastically.

The trajectory is decomposed into smaller sequences using a sliding window of size t, which defines the number of positions (nodes) included in each sequence, and an overlapping factor q, which indicates the number of nodes in common between two adjacent windows. Thus, a trajectory with n nodes is effectively decomposed into s sequences, where $s = \left\lfloor \frac{n-t}{t-q} \right\rfloor + 1$. Then, 2-opt is successively applied to these sequences to find the shortest tour in each. Note that nodes belonging to two or more windows (depending on the values of t and q) can change position multiple times as part of the corresponding windows, with the order defined by the last window to which they belong prevailing over the previous orders.

The parameters t and q can be set attending to different factors. On the one hand, a high value of t implies a larger number of nodes to sort in each subsequence; however, a low value can lead into an excessive locality of the process and lead into incorrect results. On the other hand, a high value of q introduces more redundancy (since more nodes are taken into consideration in more iterations), but a low value reduces the dependency between adjacent windows. In practice, the order of the overlapping vectors remain unmodified (since they were already sorted in the previous window), and the rest of the elements in the window are sorted accordingly. The configuration of parameters can be changed for each trajectory, or even for different segments within the same trajectory, as explained in the next section.

There are some further considerations to make when setting their values, which also are attended in the following sections. Firstly, the initial order of the elements in the trajectory is not trivial, particularly when they are sorted by timestamp. In this case, delayed vectors may fall outside their corresponding window if their assigned timestamp is sufficiently later than their emission timestamp. Since the windows are processed sequentially, they may never be able to "go back" in the sequence, resulting in an incorrect order. This situation can be resolved by either setting a sufficiently large value of t (which increases the computational cost); randomising the order in which the windows are processed and performing multiple passes; or using a different sorting criterion to define the initial order of the sequence. In the next section we propose a set of heuristics for defining a good initial order at a low computational cost.

Secondly, the 2-opt heuristic can not handle the presence of (correct) loops in trajectories. 2-opt replaces the edges forming a cross in order to reduce the total length of the path, and is unaware of whether the crossing makes sense in the trajectory or not. It is not frequent that a trajectory crosses with itself, but not impossible – specially when the aircraft have to make a holding procedure, or have to adjust their approach to an airport. This topic is discussed further in a later section.

6.5.2.3 Heuristics based on the distance to the airport

We can leverage the characteristics of the flight during each of the identified segments to apply several heuristics to help defining a better starting point for the sorting algorithm. These heuristics use the position reported in the state vectors to sort them independently of their timestamp. We have stated in previous chapters that the assigned timestamp is not always reliable due to delays in the reception process, among other causes. Thus, the ordering by time may not coincide with the correct order, and an alternative approach needs to be applied. The position is also subject to errors, such as an incorrect GPS positioning, but errors in are much less likely to happen. Moreover, errors in position are more likely to be identified and processed accordingly (either by solving them, or by excluding the anomalous points from the data).

These heuristics help our 2-opt implementation in two ways. First, these heuristics allows to reallocate many vectors with an incorrect order in a more lightweight manner than 2-opt, thus reducing the total number of changes that have to be evaluated during 2-opt. Second, out 2-opt implementation depends on the chosen window size. To reallocate a state vector into their correct position, both the vector and the correct position has to be “present” in the same window. Thus, if the state vector falls outside all of the windows containing the position, it may not be reallocated successfully (in general, it would be moved to the first positions of the first window containing the state vector). This situation would require to use larger window sizes, which implies significantly longer processing times. By using these heuristics, many of these state vectors are moved in the sequence closer to their appropriate spot.

The pre-sorting heuristics are applied in sequence as follows:

- First, the vectors in the taxiing phase are excluded from the sorting process.
- The manoeuvring segment at the origin airport can be sorted either by timestamp, or using the Nearest Vector heuristic. The state vectors in this phase are less likely to be out of their order from a time perspective, since there can not be delayed vectors from a previous segment of the trajectory. Also, the NV method is effective for simple paths like the ascent manoeuvres in case they are disordered.
- The cruise phase often consists on a simple, straight-line path, so the Nearest Vector can also be applied on this segment. A more lightweight approach is sorting the vectors ascending or descending by their distance to the departure or destination airport, respectively.
- Within the TMA, including the manoeuvring and approach segments, Nearest Vectors can be used if there are no loops in the trajectory. As discussed before, these patterns potentially break both Nearest Vectors and 2-opt approaches. Otherwise, the Nearest Vector method is robust enough to pre-sort these segments.

Applying these heuristics reduces the workload of the application of 2-opt, which is responsible of dealing with the most difficult positions and solving any error caused or not tackled by the previous methods. They also allows 2-opt to work with more performant parameters, such as smaller windows and a lower degree of overlapping between consecutive windows.

6.5.2.4 Nearest vector

In Section 6.5.1.1 we have introduced the Nearest Neighbour as an approach to solve the TSP. This method leverages the similarity between elements with different purposes; e.g. classification of elements, sorting, etc. The similarity of two elements is given by a distance function in a particular vector space, under the assumption that the more similar are two elements, the lower is the distance that separates them. In TSP, the Nearest Neighbours approach tries to determine the shortest tour by iteratively visiting the closest unvisited node in the graph, starting with one node (either random or chosen by a different initialization strategy). In the case of 4D trajectories, the “next vector” for each state vector would be the closest position that has not been visited yet. In this project, we call this method as the Nearest Vector method.

The Nearest Vector approach (like any Nearest Neighbour method) relies on a distance function. For calculating distances between 2D/3D positions, the Euclidean distance is a common choice. However, as discussed in Section 6.5.1.3, this distance function may not be suitable for

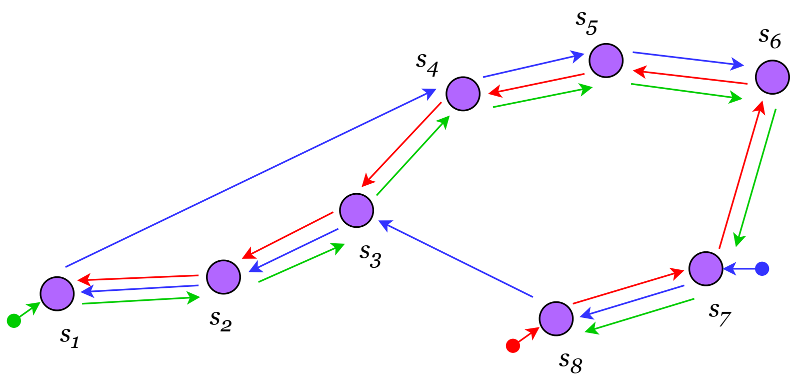

Figure 6.15: Different initial points chosen for initializing the Nearest Vector algorithm. Each of them generates different paths, with the path in green being the only correct one.

4D trajectories in some scenarios. In this case, the cost function explained in that section can be used when necessary as the distance function in NV.

The Nearest Vector approach still has three shortcomings. Firstly, it is vulnerable to the presence of loops in the trajectory, as explained in Section 6.5.1.2. Even trajectories that do not intersect with themselves can be problematic, such as those involving very closed sacs. These problems are more frequent in trajectories with low continuity, with gaps in parts of the trajectory describing turning manoeuvres. Secondly, its complexity is $O(n^2)$, since each iteration requires the calculation of the distance to all unvisited elements. Therefore, processing long sequences could also be problematic. Thirdly, Nearest Neighbours methods are dependant of their initialization; that is, the initial conditions (e.g. the state vector chosen as the first vector) determine the output of the algorithm, which may not correspond with the optimal solution. NV is not different: depending on the first vector of the trajectory, the result can change drastically. Specially since the distribution of the positions in the vector space is linear-like, in contrast with more homogeneous distributions in other problems such as element classification. For instance, choosing an intermediate position during the cruising phase could result in the trajectory being traveled backwards from that point.

Several strategies can be adopted to optimize the initialization of the algorithm, although we have explored two. First, choose as the starting point the state vector with the earliest timestamp (whether in the trajectory, or in the considered segment). This strategy is safe at the start of the trajectory, since there can not be delayed vectors prior to the starting of the trajectory, and any delayed vector in that segment would have a later timestamp. However, it may become problematic when processing intermediate vectors in the trajectory, which could result in a similar situation to that described above. NV can also benefit from using overlapping windows like 2-opt, particularly for setting up a good starting point for each window with several vectors already sorted in the previous window. For intermediate segments (specially during the cruising phase), we evaluated using the position instead and choose the closest or farthest position to the departure and destination airports, respectively. Either method (lowest timestamp or lowest distance) have been found to be effective, since we have not found them to introduce errors in the processed trajectories.

6.5.2.5 Loop and holdings detection

Assuming continuity of the trajectory within the manoeuvres area, the total rotation performed by the aircraft with respect to its entry direction is a reliable indicator of the presence of a loop. Generally, aircraft enter this area via an appropriate route to land on a designated runway, in accordance with the current runway configuration. Thus, the path they follow is straightforward, requiring only slight turns to align with the runway entry point. However, loops (that is, paths that intersect with themselves) and holdings require larger turns, exceeding the 360 degrees. In some scenarios, an aircraft may turn left and then right, or vice versa, thus reducing the accumulated turn in the manoeuvre area. To account for these cases, the maximum accumulated turn is calculated instead of the accumulated turn at the exit of the manoeuvre area.

As with the directional factor explained in Section 6.5, either the track variation or the angles between subsequent positions can be used for this purpose. However, since trajectories are more likely to be affected by delayed vectors in the areas surrounding the airport, the latter method is less reliable. Therefore, we calculate the changes in the bearing of the aircraft to determine the maximum accumulated turn in either direction. If the total turn exceeds 360º, there is a loop in the trajectory. While we can determine the number of loops, we can not currently identify whether this loop is due to a holding procedure, or a simple manoeuvre, because these trajectories are excluded from our study in later chapters. However, a more precise analysis could be conducted based on pattern recognition and where the loops are located (since holding manoeuvres are made in predetermined places within the TMA).

6.5.2.6 Further considerations

During the data cleaning applied in previous steps, we removed all state vectors with duplicate positions and position update timestamp for each ICAO24 in the dataset. However, we have found that some vectors with repeated positions are still present at this point of the data processing, possibly due to message duplication: the same ADS-B message that reports the position is captured by different sensors and receives different timestamps, and can arrive to OpenSky servers at different times. Then, these copies are then used by OpenSky to construct multiple state vectors with different time information, but the same position. Since we consider duplicated positions as redundant data to reconstruct the 4D trajectories, we apply a further clean up of vectors with repeated positions independently of their timestamp. These state vectors should not interact with the sorting algorithms, since they are at a distance 0 between them and do not influence the total traveled distance, but they could be problematic for timestamp and altitude inferences.

6.5.3 Reordered vector identification

Detecting whether or not an element has been moved is not trivial. Let there be a sequence $X = \{x_i^j\}, i, j \in [1, 6]$ an ordered sequence of five elements, where i is the initial index of the element, and j is the current index of that element. The moved elements will be underlined. Let us suppose that the element x_4 is moved back into the second position: $X' = \{x_1^1, \underline{x_4^2}, x_2^3, x_3^4, x_5^5, x_6^6\}$. Note that the index of the elements between the original position and the new position of x_4 has changed, albeit those elements have not been moved. If, on the contrary, the element x_2 is moved forward up to the index 4, the sequence results as follows: $X' = \{x_1^1, x_3^2, x_4^3, \underline{x_2^4}, x_5^5, x_6^6\}$. Again, not only the index of the moved element has changed, but also the indices of all elements between the initial and final positions. Moreover, if the vector is moved to an adjacent position, like x_4 to position 3, the result is as follows: $X' = \{x_1^1, x_2^2, \underline{x_4^3}, x_3^4, x_5^5, x_6^6\}$. Looking at the sequence, it is not obvious to tell whether x_3 or x_4 has been moved. This situation is common when a single vector is delayed and appears later than its next state vector. After each of these operations, the position of several elements in the sequence changed, although only one element was moved each time. If we move backwards two elements, s_4 and s_6 to positions 3 and 2, respectively: $X' = \{x_1^1, \underline{x_6^2}, \underline{x_4^3}, x_2^4, x_3^5, x_5^6\}$, all intermediate elements have their position modified, and the discrepancy between their original position and their new index is two. We can move one element forward and other element backwards; e.g. swap the position of elements x_2 and x_4.

Based on these examples, we can identify two situations when a single vector is moved:

- The vector is moved forward: when a vector is moved forward, the next vectors in the sequence decreases their indices by 1 until the moved element is found in the sequence.
- The vector is moved backwards: when a vector is moved backwards, the vectors between the new index and the original index increases their indices by 1.

Based on these observations, we have to check the elements in the sequence and keep track of the state vectors that are found outside of their initial position. In this way, we can distinguish between elements that have been effectively moved, and elements whose index has changed due to another element being moved. We define the following rules:

- Each element has two indices, p and q that indicate the initial and final positions of the element in the sequence, respectively. The index p is used to identify the element, and the index q is used to check whether the element has been moved or not.
- We follow the sequence after being reordered.
- Let F and M be the sets of *found* and *missing* elements, respectively. Found elements are those moved backwards in the sequence; that is, they are found before arriving at their initial position. Missing elements are those moved forward in the sequence; thus, they are not at their initial position when we reach there. These sets are initially empty, and should be empty after the iteration ends.

Let x_i^j be the j-th element in the sequence after being reordered, and the i-th element in the initial sequence. Let f and m be the cardinality of F and M sets (the number of elements found or missing before processing the element x_i^j in the position i, respectively). Let us evaluate the possible scenarios:

- If $f = m = 0$, there are no state vectors out of position that have been detected or missing up to the position j, and the indices i and j coincide.
- If $f = 0$ and $m > 0$, there are m state vectors that have not been found at their position. In those positions, the index i is increased in m, while j is not. If a state vector has not been moved, then its indices should verify that $j = i - m$.
- If $f > 0$ and $m = 0$, then f elements have been found whose initial index i is higher than it should be. If x_i^j has not been moved, then $j - f = i$.

- Finally, if $f > 0$ and $m > 0$, there are missing state vectors pending of being found, and state vectors found before their initial position. Any other state vector that has not been moved should verify that $j - f = i - m$.

- If a state vector does not validate these expressions, then the state vector in the initial position i is a reordered vector.

At the start of each iteration, we update the F and M sets. If $i \in M$, we have found a missing element and it is removed from that set. For found vectors it is a bit more complex: to remove a position from F, we must find that $j + m - f \in F$ to indicate that we have reached the position of an element that we found before. If we have already found f elements before in the sequence, the position of that element is displaced f positions ahead; on the contrary, if m elements were missing, the position of that element is advanced in m positions. We can also remove all subsequent positions $(j + 1, j + 2, ..., i - 1)$ that are present in F until $i - 1$. When one of these positions is not in F, we stop removing positions from that set. In this way, we can reduce the total iterations of the algorithm. Then, the expression $j - f = i - m$ is evaluated to determine the type of the element under analysis:

- If the equality holds, then the element is correct: it is not either a moved vector, a found vector, or a missing vector.

- If $i - m < j - f \equiv i < j - f + m$, the element i is a missing element, and the element j has been moved. The element i is added to M.

- If $i - m > j - f \equiv i > j - f + m$, the element i is a found element, and the element j has been moved. The element i is added to F.

Once the sequence has been exhausted, the sets F and M are empty, and the state vectors that were moved have been flagged accordingly to enable the posterior operations on reordered vectors; i.e. find a valid timestamp to replace the original timestamp that was found incorrect.

6.5.4 Timestamp calculation

Once the vectors with incorrect timestamps have been identified and corrected, we need to determine new time marks for them to be used as time series. Since the recovery of the original timestamp is not possible, we can calculate an estimation of that timestamp based on the timestamps of the adjacent state vectors that have not been moved. The new timestamp is calculated as follows:

$$t'_i = \frac{d_{i,i-1}}{d_{i+1,i-1}}(t_{i+1} - t_{i-1}) \tag{6.6}$$

where $d_{p,q}$ is the Euclidean distance between the state vectors s_p and s_q. Thus, the time difference with the previous state vector is proportional to the traveled distance with respect to the previous state vector.

A different approach would be using the speed indicated in the previous state vector to calculate the time it would take the aircraft to travel that distance. However, this approach would require additional validation since we have found this attribute to suffer from some noise, and it introduced some anomalies in our tests.

6.5.5 Outlier detection and attribution

Many of the dimensions described in the state vectors progress gradually, and their values are similar in vectors that are close to each other. Thus, it can be assumed that, provided the absence of gaps in the trajectory, the values that introduce significative changes in the sequence are incorrect.

Different methods can be applied to identify and solve such cases. Median filters are common for these kind of sequences given their simplicity and lightweight computation. Median filters consist on calculating the median of a window of elements around a given element to determine if its value is consistent with those found in the window; that is, if the central value is close enough to the median value of the window. The threshold of the likelihood can be set either statically (fixed or proportional threshold values) or dynamically (training the filters on example sequences). Median filters are more robust to extreme values than mean filters (or moving averages), which makes them more appropriate for finding anomalies in a sequence. We have evaluated this approach with the altitude, which we have found to be more prone to errors such as extreme values than other attributes.

Generally, these methods require to know several elements in advance in the sequence, although they can be implemented so the considered element is located at the end of the window instead of its centre if we consider an implementation where future elements are unknown. Then, the element is compared with the mean/median of the previous elements, and not the elements surrounding that element in both directions. More complex approaches implement statistic autoregressive methods or machine learning to predict appropriate values to compare with the real element. Some examples of these techniques are ARIMA, DeepAR or LSTM, which learn the patterns of the sequence to predict one or more elements based on a window of past elements.

6.5.5.1 Altitude calculation

Once the state vectors in the trajectory have been reordered, we face two problems that can be present in the reported values for the altitude. Firstly, the altitude can be null, specially when the aircraft is on the ground at the start or at the end of the flight. However, we have found state vectors without an altitude value at any point in the trajectory. Secondly, the altitude can be erroneously reported and provide a value that is inconsistent with the surrounding state vectors. This situation is frequent in delayed vectors, specially during the descent of the aircraft, since the reported altitude corresponds with the altitude of several seconds or minutes before. However, some incorrect values are still present even after sorting the state vectors according to their reported position.

We tackle these problems in two phases. In the first place, state vectors without an altitude value and indicating that the aircraft is on the ground are assigned the altitude in feet of the airport in which the aircraft is located. The state vectors with no altitude values during the flight are flagged to be corrected later on. In the second place, inconsistent values are detected using the aforementioned median filter. To ensure the correct calculation of the median, null values are replaced by a linearly interpolated value just for this purpose. Each altitude value is compared with the value provided by the median filter, and if the difference exceeds a particular threshold (arbitrarily set at 100 feet), the state vector is flagged as incorrect. Finally, the altitude of all flagged state vectors is replaced by a linearly interpolated value, in a similar fashion that was done for the timestamps. We measure the altitude variation between the previous and posterior state vectors, and assign the intermediate incorrect state vector a value proportional to the time elapsed since the previous vector, according to the expression below::

$$h'_i = \frac{t_i - t_{i-1}}{t_{i+1} - t_{i-1}}(h_{i+1} - h_{i-1}) \tag{6.7}$$

Again, the vertical rate attribute included in the state vectors could be helpful to estimate the altitude. However, the vertical rate is applied a rounding to 25 feet that influences the quality of this estimation. Thus, we opted for this more simple approach instead.

6.5.5.2 Position outliers

A similar approach can be applied for other attributes that are characterized by smooth changes in their values. However, in some cases we may prefer to remove the state vector altogether if we deem that it has incorrect data in critical attributes. In our case, the 2D position is vulnerable to GPS errors that locate the aircraft a large distance away from the rest of the trajectory. Such cases are easy to identify, but solving them by assigning an artificial value is not trivial. We could interpolate a feasible value, but the dependence between position and time makes this calculation a bit tricky. The timestamp can be incorrect, as shown in previous sections, and may be of no help to estimate the position. The middle point between the previous and the posterior state vectors could be used as the new , but this could bring some inconsistency into the evolution of the timestamp and other derived attributes (such as speed). Therefore, it is safer to flag the state vector as an outlier, and remove it later depending on the use case.

The detection of position outliers is performed using thresholds. Once the trajectory has been reordered, we calculate the average speed required to travel from each position to the next position. The speed is calculated as the traveled distance divided by the elapsed time between those two points:

$$speed_{avg} = \frac{distance(s_1, s_2)}{t_2 - t_1} \tag{6.8}$$

If $speed_{avg}$ exceeds an arbitrary threshold of 0.5 kilometers per hour (which is high for a commercial flight), the state vector is flagged as an outlier, since it is impossible that it could have traveled that distance in the elapsed time. However, this is only applied if the elapsed time is less than 60 seconds to avoid problems derived of gaps associated with delayed vectors.

6.6 Experiments

In this section, we will evaluate the methodology using the dataset defined in the Chapter 5. This dataset is described quantitatively in Table 6.2, which shows the daily aggregated values. Appendix B provide a more in-depth analysis with data aggregated by route A sample of 6,456 flights has been selected to fulfill the requirements of our case study, i.e. all flights between a set of eight major European airports between 3rd and 16th July 2023. All 442 million state vectors extracted from OpenSky after the data cleaning are potential candidates to belong to one of the identified flights. While we can easily filter the complete dataset to find the relevant flights based on their origin and destination airports, the state vectors can only be selected based on their date. Therefore, we must consider all state vectors generated during the duration of each trajectory as potential candidates for that particular trajectory. The results of the process are discussed in the next section.

Date	Flights	Vectors
2023-07-03	471	32,536,121
2023-07-04	466	32,395,989
2023-07-05	427	31,820,693
2023-07-06	503	33,194,943
2023-07-07	369	34,333,216
2023-07-08	434	33,486,486
2023-07-09	477	33,989,692
2023-07-10	499	30,517,151
2023-07-11	458	29,412,851
2023-07-12	492	29,157,569
2023-07-13	503	30,651,364
2023-07-14	487	29,988,333
2023-07-15	394	29,906,144
2023-07-16	476	30,637,029
Total	6,456	442,027,581

Table 6.2: Number of records available in our dataset. The number of flights corresponds with the flights described in Network Manager for the considered airports and period (July 3-16, 2023). The number of vectors corresponds with the all the data captured from OpenSky.

6.6.1 Integration results

The aggregated results of the integration between flights extracted from Network Manager and surveillance data generated by OpenSky are shown in Table 6.3. The columns reflect two phases in the process: the column named Joined indicates how many records were successfully matched based on the criteria defined in Section 6.2.3: the ICAO-24 code of a state vector coincides with the ICAO-24 code of a flight, and its timestamp is within the time period defined by the flight plan in which the flight actually took place. Thus, the number of joined flights indicates how many flights matched at least one state vector, and the number of joined vectors shows how many vectors were assigned to a flight. However, some of the flights populated in this way are invalid; e.g. the amount of vectors that were assigned to it was too low, so those flights are removed later. We have set this threshold at 200 state vectors to ensure dense enough trajectories. The vectors assigned to those flights are consequently removed. The columns called Final indicate the number of records after this cleaning step.

First, we review the effectiveness of the process regarding flights. The success rate is between 97.5% and 99.4% for most of the considered days, which demonstrate that OpenSky provides a notable coverage for all flights in European airspace. Recall that a relevant amount of state vectors were already removed due to repeated positions derived from the processing by OpenSky, so the (>200) vectors assigned to these flights should all be relevant and informative. However, the degree in which trajectories are complete can not be observed in these results; we rely on the trajectory quality metrics for that purpose. The percentages of the state vectors may seem low in comparison, ranging between 1.05% and 1.5%. However, we are considering all the state vectors extracted from OpenSky, since state vectors can not be segregated like the flight plans. Moreover, we are only working with a selection of routes, which is why so few vectors actually match a flight in our dataset.

6.6.2 Reordering results

In this section, we delve into the reordering of state vectors within the defined trajectories. Table 6.4 shows several values to assess the operations applied on the data: the number of sorted

	Flights		Vectors	
Date	Joined	Final	Joined	Final
2023-07-03	465	464 (98.5%)	466,948	466,795 (1.43%)
2023-07-04	456	455 (97.6%)	453,821	453,801 (1.40%)
2023-07-05	419	419 (98.1%)	425,487	425,487 (1.34%)
2023-07-06	485	485 (96.4%)	470,100	470,100 (1.42%)
2023-07-07	362	362 (98.1%)	361,491	361,491 (1.05%)
2023-07-08	427	427 (98.4%)	449,854	449,854 (1.34%)
2023-07-09	470	470 (98.5%)	483,470	483,470 (1.42%)
2023-07-10	496	496 (99.4%)	458,467	458,467 (1.50%)
2023-07-11	453	453 (98.9%)	408,497	408,497 (1.39%)
2023-07-12	486	486 (98.8%)	426,309	426,309 (1.46%)
2023-07-13	500	500 (99.4%)	452,550	452,550 (1.48%)
2023-07-14	481	481 (98.8%)	433,302	433,302 (1.44%)
2023-07-15	388	388 (98.5%)	385,312	385,312 (1.29%)
2023-07-16	473	473 (99.4%)	439,274	439,274 (1.43%)
Total	6361	6359	6,114,882	6,114,709

Table 6.3: Results of the integration between state vectors and flight plans, aggregated daily. A majority of the flights in our dataset have at least one state vector. However, the sample of flights only accounts for a small portion of the OpenSky data.

vectors, the incidence of loops in the trajectories of the dataset, and the variation introduced by the timestamp recalculation. Similar to the previous section, in Table 6.5 we provide aggregated values to characterize the impact of the reordering process on the constructed trajectories according to different metrics. We delegate into Appendix B a more thorough analysis of individual routes.

A total of 4,515 trajectories, out of 6,163, had some of their vectors changed their position in the sequence, and their timestamps adjusted. That is, the 73.26% of the trajectories presented this kind of error in the data, and required to be corrected. In average, 21.3 vectors were moved within the trajectory, or 30 if we only take into account the trajectories in which any state vectors were moved. This amount corresponds with an average of 3.12% of the vectors in the trajectories. The variation of the timestamp in the reordered vectors (the calculated timestamp minus the original timestamp assigned by OpenSky) can be used to characterize the amount of time in which these vectors were delayed with respect to their correct position in the sequence. To this end, we calculated the mean difference for all moved vectors within each trajectory, and then calculated the mean of this value across all trajectories. The vectors were assigned a timestamp 257.62 seconds earlier than their original timestamp. That is, those vectors had a delay of 4.3 minutes in average, although the delay is highly variable across the vectors: the

Metric	Mean	StD.
Reordered vectors	21.30	48.58
Reordered vectors (resorted trajectories)	29.99	55.34
Timestamp variation	-175.73	278.04
Timestamp variation (resorted trajectories)	-257.62	303.71
Resorted trajectories	4,515 (73.26%)	
Loops	303 (4.77%)	

Table 6.4: Description of the incidence of sorting operations on the trajectories in the dataset.

	Reduced distance	Removed vectors	Process time
Mean	137.07	36.87	6.32
St.Dev.	285.49	44.33	4.45
Min	-339.61	0	0.41
25%	0.00	9	4.10
50%	15.47	22	5.61
75%	149.59	46	7.15
90%	411.93	92	9.22
Max	4,957.04	434	72.09

Table 6.5: Description of the results in terms of reduced distance (kilometers), number of removed vectors, and total processing time (seconds).

standard deviation of the timestamp difference is high at 303.7 seconds, more than 5 minutes. Finally, a loop (or a holding) was found in at least 303 trajectories, so the presence of these patterns in the trajectories is relatively frequent (4.77%).

In the following we discuss the results shown in Table 6.5. The most relevant measure is the variation of the traveled distance. As stated before, an incorrect order of the state vectors in a trajectory leads into an increased total traveled distance. Trajectories with a high level of disorder show a traveled distance that is much higher than reasonable to travel between two airports. In these cases, sorting the trajectory should lead into a notable decrease in the traveled distance. For trajectories with low to no disorder, the variation of this distance will not be as apparent, though. In our benchmark, we observed that the reordering algorithm did not change the traveled distance in 29% (1843) of the trajectories, and 12 trajectories had worse values for this measurement. After closer inspection, we found that these trajectories either had loops and holdings in their paths (11) or were incorrectly generated (1). The remaining 75% of the trajectories had reduced their traveled distance to some degree, with a mean improvement of 137.07 kilometers across all trajectories. The distribution of the differences in traveled distance are shown in Figure 6.16 in logarithmic scale.

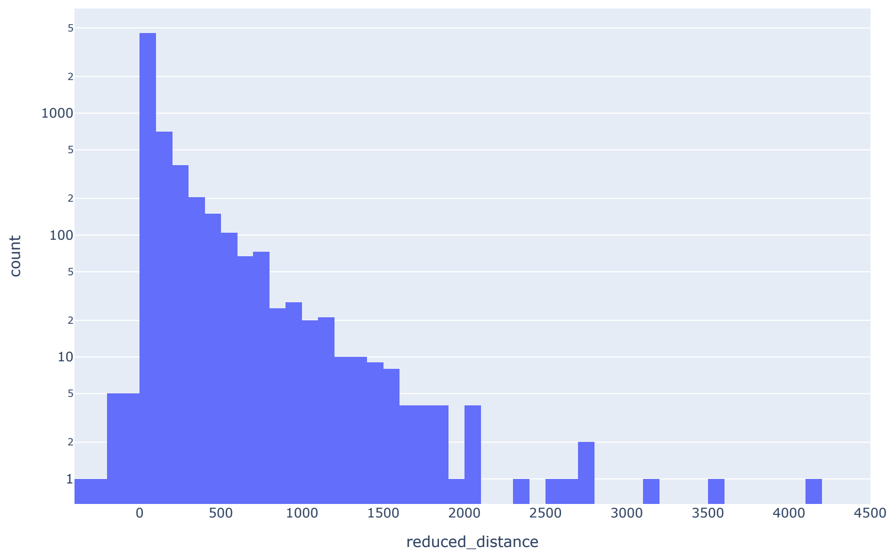

Figure 6.16: Distribution of the variation in the traveled distance after reordering the trajectories. The vertical axis has been applied a log scale to improve the visibility of less frequent values.

With respect to processing time, in previous sections we stated that the defined heuristics helped to alleviate the computational cost of the process by applying lighter operations than 2-opt only. As shown in Table 6.5, we reported a running time to 6.32 seconds per trajectory in average. Furthermore, for 90% of the trajectories the processing time is under 9.22 seconds, with only few trajectories that require more time. The distribution of running times is shown in Figure 6.17, with a clear high frequency around 6 seconds. We have investigated whether the number of vectors in the trajectory influences the running time, as per intuition, but Figure 6.18 shows that there is not a clear correlation between these two factors. The processing time remains constant to a degree, independently of the number of vectors with a notable density in the range (0,10), albeit a slight increase can be observed starting at 1000 state vectors.

Finally, additional vectors were removed during this step for having their positions repeated. This removal is applied trajectory-wise, and the first state vector (chronologically) is kept, and the rest are removed from the trajectory. Figure 6.19 shows that many trajectories still had a significant amount of such state vectors, with a mean of 37 vectors removed per trajectory, as indicated in Table 6.5.

6.6.3 Trajectory quality metrics

Once the vectors in the trajectories have been processed, we can use the set of quality metrics defined in Section 5.5 to characterize the trajectories before and after the sorting operation. In the following we present the aggregated results across all trajectories, with a special focus on some interesting aspects.

Table 6.6 shows the mean results for two categories of quality metrics. Under Flight path characterization, the actual flight time (as defined in the flight plan) and the distance between

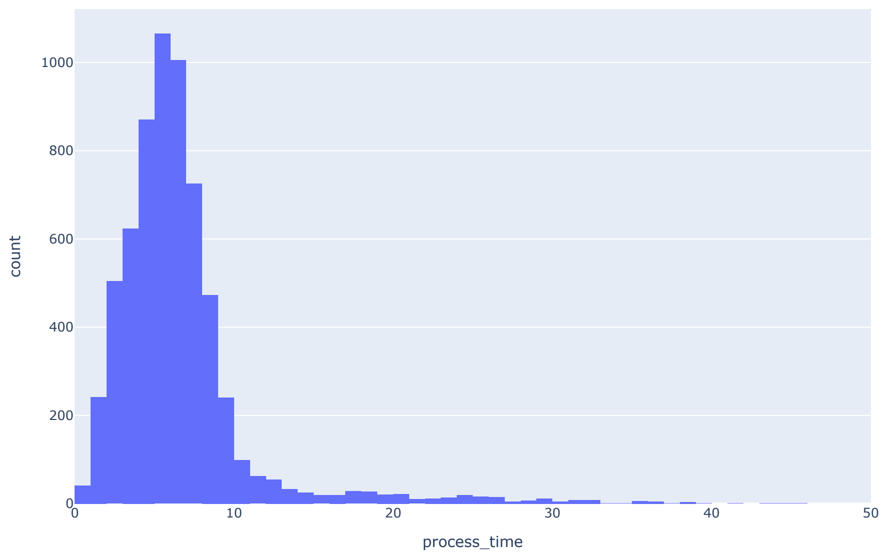

Figure 6.17: Histogram showing the distribution of processing time across all trajectories. Most of the trajectories are processed in 10 seconds or less, with some difficult cases extending up to 45 seconds. The longest time (72.61 seconds) has been excluded from the graph.

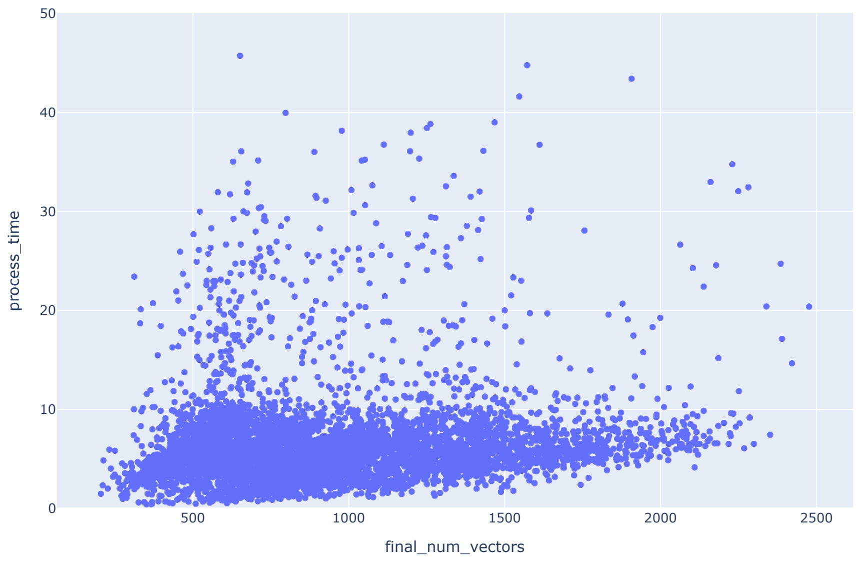

Figure 6.18: Correlation between processing time and the number of vectors in the trajectory. The longest time (72.61 seconds) has been excluded from the graph.

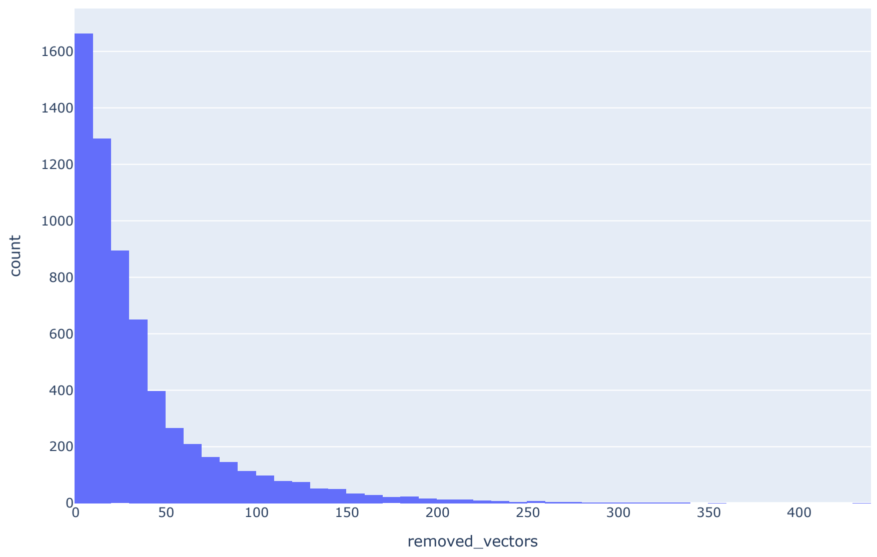

Figure 6.19: Number of vectors removed per trajectory.

	Raw trajectories		Clean trajectories	
Metric	Mean	StD	Mean	StD
Flight path characterization				
Actual flight time	5,810.30	2,686.82	5,810.30	2,686.82
Geodesic/direct distance	671.35	374.79	671.35	374.79
Coverage				
Total duration	6,056.95	2,595.73	6,018.41	2,599.93
Total distance	867.04	507.25	729.98	387.27
Distance ratio	1.37	0.61	1.12	0.18
Distance to origin	3.65	20.87	3.60	20.26
Distance to destination	5.93	17.54	4.45	15.09
Initial segment	1.0022%		1.0022%	
Final segment	1.1952%		0.6762%	
Last altitude	2,467.74	5,098.46	1,830.43	4,247.01

Table 6.6: Summary of the mean values of the trajectory quality metrics results for integrated and clean trajectories. Flight path characterization and Coverage metrics.

the departure and destination airports are calculated to set the reference values that can be used to evaluate the actual characteristics of the trajectory. Recall that these factors define the minimum values for the distance and duration of the trajectory, respectively These values become more valuable when the routes are analyzed separately, since they set a common ground for all trajectories of that route. The Coverage category shows the actual characteristics of the trajectory, including its duration, distance, and other aspects that help to assess its completeness.

The characterization of the flights does not vary from raw to clean data, since they are independent of the flown trajectory. In average, the considered routes require flights of 5,810 seconds (1.61 hours) that travel 671 kilometers in a straight line from airport to airport. The variance in these values is high, exceeding the 50% in both cases, though, so they do not provide an accurate representation of the flights. Nevertheless, they can be used to assess the mean values for the related measures in the next category of metrics. In fact, the average values observed in the trajectories are quite similar. The cleaned trajectories have a mean duration of 6,019 seconds (1.68 hours, a 4.34% more time), including the taxiing time at the airports (which is not contemplated in the values of the actual departure and arrival timestamps indicated in the flight plan). Further analysis could be done to identify the takeoff and landing in the trajectory and compare the real time in which the aircraft is flying with the flight time, but the lack of data during the landing operations at the destination airport prevented us from carrying out this analysis. The traveled distance is also higher at 730.07 kilometers (1.09%), but this fact was predictable since aircraft do not fly in a straight line from airport to airport. In many cases, even during the cruise phase the aircraft must maneuver to follow coastlines, avoid obstacles or follow airways. During the ascent and descent phases, the path is anything but a straight line, specially if there are holdings or other manoeuvres around the airports. These factors add up and set the minimum distance higher than the direct distance defined above, which is also reflected in the distance ratio metric that relates the traveled distance and the direct distance for each trajectory. Airlines and air traffic control organizations make use of theoretic reference trajectories to provide a more realistic value of the *required distance* for a trajectory in a route, but currently we do not have access to such information.

Starting with the comparison between the raw and cleaned trajectories, it can be observed that there is little difference in terms of the duration of the trajectories. With the exception of the last vectors in a trajectory having duplicated positions and being removed during the cleaning, the first and last vectors of the trajectory are not modified, so the duration is the same. The differences in traveled distance are more prominent: the distance reduces from 867 kilometers to 730, a reduction of 15.8% (137 kilometers). The standard deviation is also reduced in 23.7%, since the trajectories with delayed vectors become more similar to the *normal* trajectories of their route. According to the value of the distance ratio metric, raw trajectories are 34% longer in average than the distance between the airports, and this value decreases to 12% (and shows a lower standard deviation, from 0.54 to 0.18) in clean trajectories. The distance of the last vector in the trajectory to the destination airport also decreases from 5.93 to 4.45 kilometres in average, which is normal since the presence of delayed vectors in the last segment of the flight could make that the (chronologically) last state vector is farther than the closest state vector to the airport (which in general should be the last vector in the trajectory). A similar conclusion can be made for the last observed altitude value. Therefore, correctly reordering the last vectors in the trajectory contributes to improve the quality of the final segments of the trajectory, which become available more often and reduces the amount of trajectories without this segment from 1.2% to 0.68%. The availability of the initial segment does not vary since previous state vectors

	Raw trajectories		Clean trajectories	
Metric	Mean	StD	Mean	StD
Density				
Density	1.27	0.42	1.39	0.38
Density (no gaps)	1.30	0.40	1.44	0.34
Time separation	6.40	1.27	6.66	1.53
Time separation (no gaps)	5.98	0.89	6.26	1.09
Distance separation	0.90	0.42	0.78	0.24
Distance separation (no gaps)	0.87	0.39	0.73	0.18
Continuity				
Gaps	0.38	0.68	0.49	0.79
Continuous segments	1.38	0.68	1.49	0.79
Gap ratio	3.66%	7.35%	4.80%	8.45%
Continuity ratio	89.52%	10.65%	86.84%	12.64%
Discontinuity ratio	6.82%	5.97%	8.36%	7.69%
Gap time	279.39	604.79	344.61	657.52
Continuity time	5,335.57	2,228.50	5,155.98	2,252.32
Discontinuity time	441.99	457.40	517.82	539.86

Table 6.7: Summary of the mean values of the trajectory quality metrics results for integrated and clean trajectories. Density and Continuity metrics.

are not added at the start of the trajectory.

These values indicate that the constructed trajectories provide a good coverage during most of the flight, since they describe the initial and final segments of the flight. However, we require further analysis on the level of detail and completeness of this description to assess the quality of the trajectory. To this end, we proposed several Density and Continuity metrics that helps us to carry out such analysis. Let us review the observed results of these metrics, described in Table 6.7. Density metrics analyze the rate at which the flight state is updated with respect to the time and the traveled distance. The density indicates the mean number of state vectors per traveled kilometer. Two factors influence this value when going from raw to clean trajectories: firstly, the density increases due to the decrease in the traveled distance; secondly, several redundant state vectors are removed from most trajectories, thus lowering the density while keeping intact the relevant information. In the defined dataset, we observed a net increase in the density from 1.27 to 1.39 state vectors per kilometer. The time separation and distance separation metrics are also influenced by these operations. The time separation analysis indicates the average update rate between subsequent state vectors. Since some vectors are discarded, the average time between state vectors increases from 6.4 seconds to 6.66 seconds between reported positions, but there is not information loss in terms of the reported valid positions. Recall that OpenSky offers a theoretical refresh rate of 5 seconds; a mean time separation of 6.66 seconds indicates that the state vector density of these trajectories is close to the capacity of OpenSky even after removing the vectors with reused positions. On the contrary, the mean distance between subsequent reported positions decreases from 0.90 to 0.78 kilometers thanks to the state vector sorting, and the decrease in the standard deviation indicates that the separation between vectors became more homogeneous.

Nonetheless, the presence of gaps in the trajectories due to a lack of coverage raises this value up. The impact of gaps (that is, trajectory segments with no updates for more than 300 seconds) has been measured at an average of 4.8% of the total travel time across all trajectories. The increase of this value with respect to the raw trajectories was expected: when OpenSky stops capturing ADS-B messages, it keeps generating a few more vectors with redundant data. Since we are removing such vectors, in practice we are "extending" the existing gaps. If the gaps are excluded from the analysis, the density metrics vary significantly. The distance-related metrics improve slightly on raw trajectories, but the improvement is more notable in the average time separation, which reduces to 5.98 seconds. On clean trajectories, the values are better in value and more consistent. The density increases to 1.44 vectors per kilometer, the average time separation lowers to 6.26 seconds between state vectors, and the average distance separation also decreases to 0.73 kilometers between state vectors. Thus, the sorting process manages to increase the quality of the segments of the trajectories that are described in the data.

Part III

Use cases

Chapter 7

Estimation of the time of arrival

The Estimation of the Time of Arrival ETA is a key factor for ATM operations. Determining when a flight will arrive at its destination airport improves the predictability of the operations at the airport, and allows for a better resource allocation in the transit airspace and on the ground. As a consequence, accurately predicting the time of arrival is crucial to enhance the efficiency, capacity and safety of ATM operations, and therefore to reduce operational costs and the environmental impact of flights [9]. Conversely, inaccurate ETA predictions can trigger cascading delays at the airports, affecting both the departures of aircraft from the airport and the arrivals of subsequent flights, which may need to wait until the necessary resources are available for landing. However, the inherent uncertainty of flight operations, which are influenced by many dynamic and non-deterministic factors, makes accurate ETA prediction a challenging task.

Most current research focuses at predicting ETAs within the Terminal Manoeuvring Area (TMA) [46, 79, 117, 25, 18, 83], where some of the most critical ATM operations take place. Nevertheless, ETA prediction are valuable throughout the en-route phase, as it allows coverage of a wider portion of the airspace and enables earlier and more robust planning. On the other hand, some studies [9] adopt a more limited scope, targeting individual routes or segmenting flights by specific criteria. However, we found that a broader scope might benefit the performance of the predicting models. As flights approach the airport, the traffic coming from the different routes becomes more homogeneous when performing the approach manoeuvres defined for that airport. The behaviour of the aircraft may also be similar during the flight, especially during the cruise phase. In other words, learning from multiple routes should help to identify and model increasingly rich flight patterns, rather than focusing on single routes that require, on the other hand, training and maintenance of specific models, with the additional costs that this entails. In addition, the model can be trained with comparatively more data, which benefits the training of deep learning models in particular.

In this chapter, we describe our approach to ETA prediction for inbound aircraft at a destination airport. Our solution leverages data that describe factors influencing the progression of a flight, such as the realized trajectory, the origin and the planned schedule of the flight and the weather conditions at the destination airport, to provide accurate predictions accounting for all of these factors. To this end, we train and evaluate several models implementing different architectures of LSTM networks to determine the most suitable model for this task.

7.1 Related work

ETA prediction for commercial flights was traditionally addressed using deterministic methods [85], based on aircraft performance and physics simulations. The idea behind these methods is to determine the planned trajectory and then calculate the time needed to fly that trajectory under specific conditions. While effective, developing these methods is complex and expensive, and their predictions are very dependent on the simulation conditions for all considered factors: weather conditions, traffic congestion, the planned flight path... Any variation of these factors during the actual flight can make the prediction to be inaccurate. In recent years, data-driven approaches have gained relevance due to the increased availability of air-traffic related data and hardware resources capable of running computationally intensive machine learning algorithms. These methods leverage historical data to learn flight patterns and are better suited to adapt to unseen or rare circumstances, providing more accurate predictions in the uncertain conditions that govern ATM operations. Thus, data-driven approaches generalize better than fine-tuned deterministic ones, and have become a recurrent choice to address the problem at hand.

Most of the existing data-driven approaches [46, 69, 83, 18, 25, 117, 79] build a prediction model for a particular arrival airport, enabling ETA predictions for all incoming flights, regardless of their departure airport. All of them report figures for short-term ETA predictions, when the aircraft is close to the destination airport in terms of distance (mostly between 25 and 100 nautical miles, or NM, away from the airport) or time (between 5 and 60 minutes before landing), but do no report numbers at the early stages of the flight. In [105], a single model is proposed to predict all flights within a given area, but it is suggested that individual models for each destination airport (or groups of related airports) would be more effective. Ayhan et al. [9] follow this approach and build optimized models for each particular route (i.e. a pair of departure and arrival airports), enabling long-term accurate ETA predictions, at the price of a complex deployment that involves maintaining multiple models (one per route) at the destination airport.

A second aspect to consider is the data used to construct the prediction models. This decision must take into account the many factors that can affect the operation of a flight. *Surveillance information* (mainly ADS-B) is present in most of the proposals, allowing to describe enriched 4D trajectories, where time and situation (latitude, longitude, and altitude) are enhanced with other valuable features, such as speed or heading, to characterize the aircraft movement over time. *Weather data* (wind direction and speed, visibility, etc.) is also present in most of the solutions, but they differ in whether they consider this information only at the arrival airport [46, 105, 69, 18, 25], also at the departure airport [1] or throughout the flight [9]. *Flight plan data*, reporting origin and destination airports, the (scheduled and actual) off-block and takeoff times (if the flight has already started), the scheduled arrival time or the expected total duration of the flight, among other features, is also commonly used [105, 69, 1, 9, 83, 25, 117]. Finally, *seasonality information* [1, 18, 46, 69] (e.g. the day of the week, the month or the time of the day in which the flight will occur), *congestion information* [105, 1, 9, 18] (traffic density metrics at the airports or airspace levels) or *information about resource management* [46, 25] (e.g. configuration of runways) have proven to be useful for ETA prediction.

The main difference between the existing state-of-the-art solutions lies in the method they used to predict ETAs. Various machine learning methods have been evaluated for such purpose, highlighting bagging ensemble models, such as random forests (RF) [15] or Extra-Trees (ET) [45], and boosting methods, such as Gradient Boosting Machines (GBM) [44] or Adaptive Boosting

(AB) [43]. More recently, FeedForward Neural Networks (FFNN) [12] and other deep learning models, such as Long Short-Term Memory (LSTM) networks [53], which where already covered in Chapter 3, have been used with varying degrees of success.

Glina et al. [46] propose a RF-based model (called Quantile Regression Forest), which provide short-term predictions (between 3 and 60 NM) with RMSE values between 0.33 and 1.25 minutes, for flights arriving at Dallas/Fort Worth International Airport (ICAO code, *KDFW*). Kern et al. [105] also use random forests to predict ETAs for domestic routes in the United States and report a MAE reduction of 42.7%, compared to the ETA prediction provided by the Federal Aviation Administration (FAA) system. Kim [69] uses linear and median regression, and a nonparametric additive model to predict ETAs for incoming domestic flights at the Denver International Airport *(KDEN)*. In this case, the models were trained on 2010 data and then predictions were made for flights in 2011, reporting a mean absolute deviation of 8.63 minutes and concluding that departure delays are the most important factor for improving predictions. Dhief et al. [25] compare RF, ET, and GBM models, at the Changi Airport *(WSSS)*, concluding that ET performs better than the other methods, and reporting a RMSE of 1.92 minutes at 100 NM from the destination airport.

Subsequent publications demonstrate the superiority of GBM over bagging methods, at different time horizons. In [83], a GBM-based model is used to predict ETAs for incoming flights to the Malpensa-Milan Airport *(LIMC)*, reporting RMSE values of 175 seconds and 304 seconds, at 20 and 60 minutes from the arrival airport, respectively. Chen et al. [18] compare GBM, RF and FFNN predictions at different distances from the Zurich airport *(LSZH)*, concluding that GBM performed slightly better than RF and reporting RMSE values of 3.16 and 4.75 minutes, at 45 and 250 NM of distance, respectively. An ensemble "stacked" model is proposed in [117], reporting slightly better figures than GBM (and other methods) for ETA prediction at the entry point of Terminal Manoeuvring Area (TMA) of the Beijing Capital International Airport *(ZBAA)*. Achenbach et al. [1] also propose an ensemble model, which combines GBM and linear regression. This ensemble performs better than its constituent models, reporting effective predictions at long-term (RMSE of 5.9 minutes at departure), for flights flown by the A320 european fleet arriving at two airports. Ayhan et al. [9] make an exhaustive comparison of machine learning methods for ETA predictions on 10 major flight routes in Spain, including LSTM for the first time. In this case, GBM and AB report the best numbers, within 4 minutes of RMSE on average, regardless of the flight length. It is worth noting that LSTM reports unstable numbers in this evaluation, providing the most accurate prediction for a given route and reporting twice as much error as AB on another. Recently, Ma et al. [79] proposed a spatio-temporal neural network model that reports comparable numbers to a LSTM-based approach, for incoming flights to the ZBAA airport.

Table 7.1 summarizes the main features of the reviewed approaches: the *scope* of their models and the machine learning *method* they used (*RF*: random forests, *LR*: linear regression, *GBM*: gradient-boosting machines, *AB*: adaptive boosting, *ET*: extra trees, and *LSTM*: long short-term memory neural networks); the *data* used to build these models are then displayed (*Su*: surveillance, *W*: weather, *FP*; flight plans, *Se*: seasonality, *C*: congestion, and *R*: resources); and the flight points where the ETA is predicted. It is worth noting that the last row also describes our approach in the same terms, for comparison purposes.

Approach	Scope	Method	Data					
			Su	W	FP	Se	C	R
[46] (2012)	Arrivals at KDFW	RF	✓	✓		✓		✓
[105] (2015)	US territory	RF	✓	✓	✓		✓	
[69] (2016)	Arrivals at KDEN	LR		✓	✓	✓		✓
[1] (2018)	Arrivals at two airports	Ensemble	✓	✓	✓	✓	✓	
[9] (2018)	Individual route	AB / GBM	✓	✓	✓		✓	
[83] (2019)	Arrivals at LIMC	GBM	✓	✓	✓			
[18] (2020)	Arrivals at LSZH	GBM	✓	✓		✓	✓	
[25] (2020)	Arrivals at WSSS	ET	✓	✓	✓			✓
[117] (2020)	Arrivals at ZBAA	Ensemble	✓			✓		
[79] (2022)	Arrivals at ZBAA	LSTM	✓					
Our proposal	**Arrivals at LEMD**	**LSTM**	✓	✓	✓	✓		

Legend:
Su: Surveillance, W: Weather, FP: Flight plan, Se: Seasonality, C: Congestion, R: Resources.

Approach	Prediction
[46] (2012)	3-60 NM before landing
[105] (2015)	During the flight
[69] (2016)	Before departure
[1] (2018)	Before departure
[9] (2018)	Before departure
[83] (2019)	20-60 minutes before landing
[18] (2020)	45-250 NM before landing
[25] (2020)	100 NM before landing
[117] (2020)	Entry point of the TMA (20-25 NM before landing)
[79] (2022)	0-4 minutes after passing the entry point of the TMA
Our proposal	**15-150 minutes / 25-250 NM before landing**

Table 7.1: Summary of data-driven approaches for ETA prediction

7.2 Architecture

Estimating time of arrival based on 4D-trajectory data can be intuitively approached as a sequence modelling problem. 4D trajectories consist on long sequences (several hundreds or even thousands) of flight states, which in turn comprise multiple values describing different aspects of the flight. These data points are interdependent in a way that each of its values depends on the previous values in the sequence. Moreover, each data point has an associated timestamp, enabling the interpretation of these sequences as time series. As such, the analysis of these data using deep learning can be tackled using different types of Recurrent Neural Networks (RNN). However, the length of these sequences makes it difficult for traditional RNN to encode all their data; in practice, these networks fail to keep memory from distant values in the sequence, that is, they forget past elements as they advance in the sequence. Therefore, LSTM (Long Short-Term Memory) networks [53] should be better suited to tackle this task. As explained in Chapter 3, LSTM-based networks are specialized on learning long sequences of data, such as 4D trajectories, in order to capture both short-term and long-term dependencies between the elements in the sequence. This fact motivates our decision to build a LSTM-based neural network to predict the estimated time of arrival of a flight. This architecture consists of a single LSTM layer, with a hidden state of dimension n, and a fully connected (FC) layer with one cell and linear activation, to transform the output of the LSTM layer (a vector of length n) into a scalar value, which is the predicted RTA value for the input sequence.

7.2.1 Feature selection

In the following, we describe the set of features extracted from the enriched 4D trajectories used for training the ETA prediction models. For clarity, features are grouped according to their origin, although all of them are processed jointly by the models. Table 7.2 summarizes the selected features and provides examples for each of them.

Surveillance data. The surveillance data included in 4D trajectories comes from OpenSky network, as explained in Chapter 4. Surveillance data is used to describe the flight state over time. Each data point has an associated *timestamp*, and contains information about the 3D-position of the aircraft (*longitude*, *latitude* and *altitude*), the *instant speed* (both horizontal and vertical) and the direction the aircraft is heading (*track*). Additionally, we calculate a *distance* feature, which is the Haversine distance of the aircraft with respect to LEMD airport.

Flight Plan Data The flight plan data in 4D trajectories, originally extracted from Network Manager, provide us with schedule data such as the *expected times of departure and arrival*, the *actual times of departure and arrival* (which are calculated after the end of the flight), the *departure airport* and the airline operating the flight. We use this information to calculate the *delay of departure*, which is obtained as the difference between the scheduled time and the actual time of the takeoff, and two seasonality features, to leverage daily and weekly time patterns: (i) the *day of week*, in which the flight is scheduled to end; and (ii) the *time of day* [1], that describes the hour range the aircraft is scheduled to arrive; based in our observations, we consider three periods: morning (7-13h), evening (13-20h) and night (20-7h). The actual time is used to validate the remaining time to arrival, which is explained later in this section.

Weather data Weather forecast data is used to characterize the expected weather conditions at the destination airport for the flight's estimated time of arrival. The most relevant features are those related to *wind condition* (direction and speed), because these factors determine the direction of approach the aircraft must take, and influence its speed and maneuvers. Other selected features describe *temperatures*, *visibility* conditions and *sky conditions*. As described in Chapter 4, forecast reports are extracted from TAF reports generated by the weather station located at the LEMD airport. These reports include data about wind (direction and speed), temperatures, precipitation, icing probability and visibility, many of which may influence landing and takeoff operations.

Feature	Description	Sample value
	Surveillance	
Latitude	Latitude of a position update	51.477402
Longitude	Longitude of a position update	-0.4745
Altitude	Altitude over the ground at a position update (feet)	225.0
Distance	Haversine distance to LEMD airport (miles)	773.208995
Speed	Horizontal speed with respect to the ground (knots)	141.0
Vertical rate	Speed of climb or descend (feet per second)	2689.0
Track	Angle between aircraft heading and north	267
	Flight plans	
Departure airport	ICAO code of origin airport	*EGLL*
Day of week	Arrival day within a week	5
Time of day	Scheduled arrival period of the day	*evening*
Departure delay	Difference in minutes between planned and actual off-block time	0
Operator	IATA code of the operating airline	*BAW*
	Weather	
Wind direction	Direction from where the wind blows	35.0
Wind speed	Wind speed (knots)	4
Max. temperature	Maximum expected temperature	15.0
Min. temperature	Minimum expected temperature	1.0
Cloud altitude	Altitude of lowest clouds, if any (miles)	30
Visibility	Horizontal visibility in case of fog (miles)	4
Sky status	Qualitative description of sky status	*CAVOK*
	Target variable	
RTA	Remaining time to arrival (seconds)	7048

Table 7.2: Description of the features extracted from data to train the proposed models.

Target variable We define a Remaining Time to Arrival (RTA) value, which is used as the target variable for our model. RTA is the difference (in seconds) between the timestamp of each state vector and the actual landing time of the flight to which it belongs. We use surveillance information to obtain the landing time, even though flight plans provide an end time value, since this end time usually corresponds to the time at which the pilot was cleared to initiate the landing procedure, several minutes before actually landing.

7.2.2 Data preparation

To construct the datasets used for training the models, we extract the sequence data from the enriched 4D trajectories included in the considered dataset, which include both surveillance and weather information. Static attributes from the flight plan, such as the operating airline or scheduled times, are incorporated into the sequences by repeating their values at each timestep.

Since LSTM networks process sequences one element at a time, and each element must have the same structure, static features must be included in every timestep, even if their value does not change.

Trajectory selection

Before the data is Before adapting the data to the model requirements, trajectories containing holding procedures are excluded from the dataset. As discussed in Chapter 6, loops and holding procedures add additional complexity and unpredictability to the trajectory [24]. In the context of ETA prediction, they also distort the computation of the RTA, as these procedures introduce delays that cannot be foreseen before the aircraft enters the TMA. Since they depend on real-time landing slot availability, they result in unpredictable waiting times that shift the RTA across the entire trajectory. Moreover, in this Thesis, such manoeuvres may not have been correctly ordered during the trajectory sorting process, as their looping nature interferes with the 2-opt heuristic. For further details, see Chapter 6.

In addition, to ensure a balanced representation of all routes in the dataset, a maximum number of trajectories per month was defined for each origin airport. Since some routes to LEMD are significantly more frequent than others, the model could otherwise overfit to these dominant patterns. If the number of trajectories for a given route exceeds the monthly threshold, a random selection is applied to discard the excess and maintain the balance.

Data normalization

We further enhance the suitability of the data by downsampling the time series using the bucketization techniques introduced in Chapter 6. This operation serves two main purposes. First, it homogenizes the density of state vectors across the trajectory, distributing them more evenly in time in regions where the original update rate is higher than required. Since LSTM networks perform best when the time steps are uniformly spaced, reducing this variability facilitates more effective pattern learning. Second, downsampling contributes to noise reduction, simplifying the underlying patterns while preserving the relevant information. It also increases the total flight time covered by each sequence for a given sequence length, which has a significant impact in the performance of the developed models as discussed in Section 7.4. The update rate is treated as a configurable hyperparameter to evaluate which temporal resolution yields the best performance for the models. In this study, we consider update intervals of 30 and 60 seconds, compared to the nominal 5-second rate of OpenSky data.

Next, the data is transformed into numerical values suitable for model training. As discussed in Chapter 3, categorical features are converted to numerical format using label encoding, which maps each category to a unique integer. All features are then normalized to the [0, 1] range according to Equation 7.1, where v is the original value of feature f, and v^f_{min} and v^f_{max} are the minimum and maximum values of its distribution.

$$v' = \frac{v - v^f_{min}}{v^f_{max} - v^f_{min}} \tag{7.1}$$

The sequences must then be transformed into fixed-length segments to be used as input for LSTM networks. This is achieved through the windowing operation described in Chapter 3, which divides each trajectory into overlapping sub-sequences using a sliding window of length

l (lookback). Thus, for a trajectory $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}p)$ with p state vectors, we obtain $p - l$ sub-sequences $\omega_1, \ldots, \omega p - l$ of length l (as shown in Figure 7.1). All sub-sequences that contain a gap of more than 180 seconds between adjacent vectors are discarded to ensure the continuity of the time series in each sequence. Finally, each fixed-length sequence is labeled with the RTA value corresponding to the last state vector in the window, which will be the desired output associated to each example.

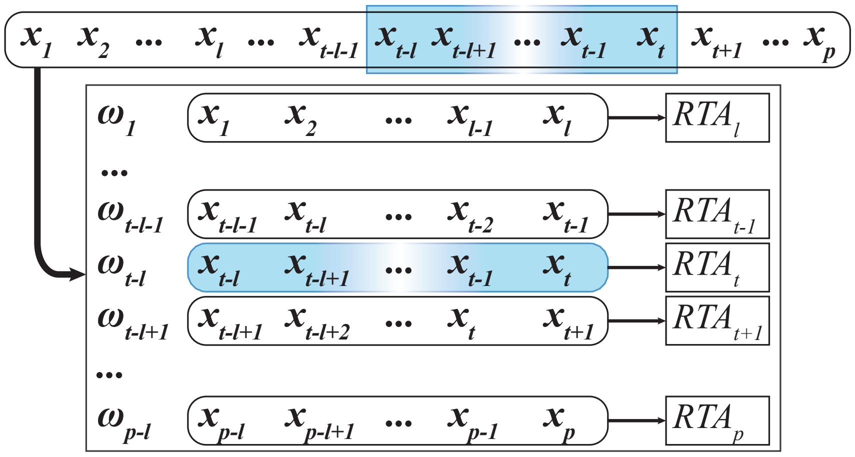
x_1 x_2 ... x_l ... x_{t-l-1} x_{t-l} x_{t-l+1} ... x_{t-1} x_t x_{t+1} ... x_p
ω_1 x_1 x_2 ... x_{l-1} x_l RTA_l
...
ω_{t-l-1} x_{t-l-1} x_{t-l} ... x_{t-2} x_{t-1} RTA_{t-1}
ω_{t-l} x_{t-l} x_{t-l+1} ... x_{t-1} x_t RTA_t
ω_{t-l+1} x_{t-l+1} x_{t-l+2} ... x_t x_{t+1} RTA_{t+1}
...
ω_{p-l} x_{p-l} x_{p-l+1} ... x_{p-1} x_p RTA_p

Figure 7.1: Extraction of sub-sequences using a sliding window.

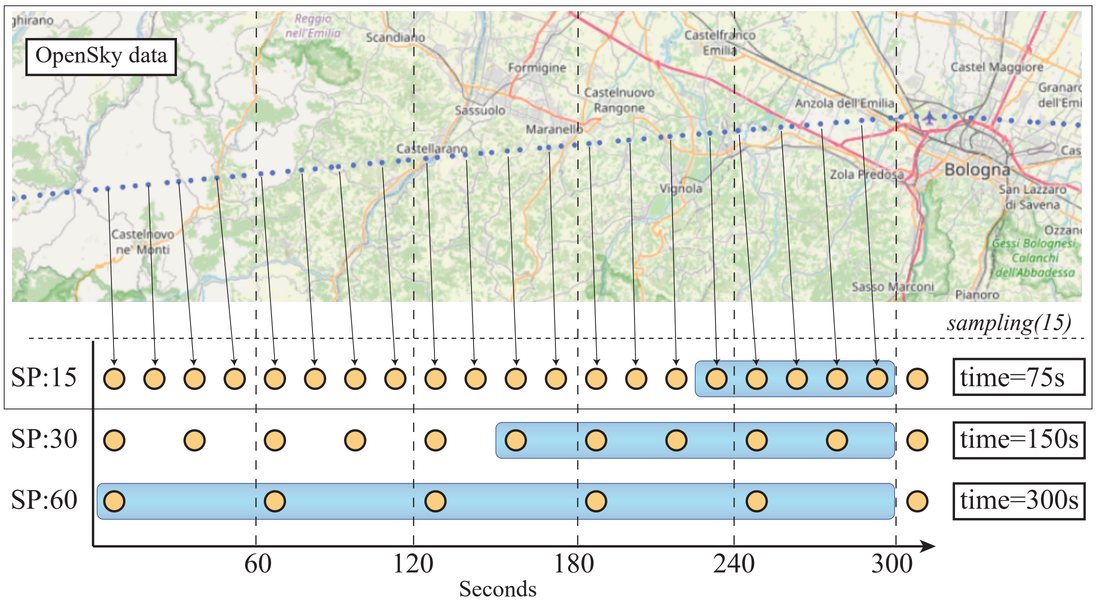

Figure 7.2: Example of a downsampled trajectory where the periodicity is increased from ~ 5 s. in the original Opensky data to 15, 30 and 60 seconds.

The lookback l, or the window size, is a determinant parameter to our model: longer input sequences provide more contextual information to the model, potentially improving predictions, but also increase computational cost and model complexity.

Importantly, the effective length of the "flight history"seen by the model is a function of both the sampling rate and the lookback. For example, with a sampling rate of 60 seconds and a lookback of 32 vectors, the model have access to the last 32 minutes of flight time to make a prediction. If the lookback is increased to 64 (keeping 60 seconds of sampling rate), then the "flight history" taken into account extends to the last 64 minutes of flight time. This is also true if we increase sampling period to 120 seconds, while keeping 32 vectors of lookback. However, these choices are not equivalent as the configuration with a lookback of 32 and a sampling rate of 120 seconds provides less detailed data, because it describes the same time period with half as many state vectors. Figure 7.2 shows an example with *lookback*=5. If we sample every 15 seconds (SP:15), a single window represents the previous 75 seconds of flight time. However, if the sampling period is 30 seconds, then each window amounts for 150 seconds of flight time. While the length of the window is the same in all cases (5 state vectors), the flight time and the level of detail in which the trajectory is described are different for each configuration.

7.3 Experiments

This section outlines the experimental design adopted to validate the proposed methods for ETA prediction. We first describe the experimental setup, including the models under evaluation, the case study dataset used for training and validation, and the metrics employed to measure the predictive accuracy and overall performance of each model.

7.3.1 Models

We briefly describe below the model configurations under evaluation. These descriptions are intended as a complement to the theoretical background provided in Chapter 3, where the underlying architectures and mechanisms are explained in greater detail.

LSTM

When we described LSTM networks in Chapter 3, we stated that LSTM models are defined by two main hyperparameters: the *lookback* and the number of *units*. On the one hand, lookback determines the number of individual elements (state vectors) the model expects to process in order to make a prediction. As discussed in the previous section, the longer the sequence, the more information the model has to characterize the evolution of the flight. However, longer sequences require more computing power or model complexity to learn long-term, complex patterns from the data. The number of units determines the dimensions of the internal representation (the hidden state) that the model constructs from the input data. Larger values of these hyperparameter allows to store more information in the hidden state, but also imply a significant increase in the computation cost and an additional risk of overfitting.

These two hyperparameters should be balanced. Low numbers of units combined with long sequences can lead into loss of information, since the hidden state does not have enough capacity. On the contrary, a large hidden state for processing short sequences could overfit to the data and being unable to generalize correctly, while increasing significantly the training times. In our study, we evaluated lookback values of 32 and 64, since our preliminary experiments reported poor performance for $l = 4, 8, 16$, and values larger than 64 were discarded as it was not possible to generate windows for many of the shorter routes considered. On the other hand, The higher this value, the more complex the model and the greater the risk of overfitting. After some preliminary testing, values of 10, 20, and 30 units were chosen.

We assigned fixed values to the other hyperparameters of the model: the hyperbolic tangent was set as the activation function; the batch size to 128; the loss function to mean absolute error (MAE), and Adam [70] was used as optimizer. We also experimented with the ReLU activation function, but it caused unstable training processes due to exploding gradient problems. During training, early stopping was used as a regularization measure to avoid overfitting. Models were trained for 30 epochs and the version with the lowest validation loss was selected for evaluation. As mentioned before, these configurations were applied to data with a sampling period of 30 and 60 seconds.

Baseline

We consider a baseline that includes the most prominent approaches to ETA prediction in the state of the art: Gradient Boosting Machines (GBM), Random Forest (RF) and Adaptive Boosting (AB). Similar to [9], all models were configured by using the default hyperparameters from their implementation in the Scikit-learn package (version 1.1.3), but reducing the number of estimators from 100 to 50 in all models due to memory constraints. Preliminary tests showed that AB performed poorly compared to other models because the decision trees used as the default base estimator were too shallow. We decided to replace the base estimator of AB with the estimator implemented in RF, reporting better results. None of these models are designed to work with time series, so we provide individual state vectors as inputs instead of the constructed windows.

We also explored the implementation of GRU networks instead of LSTM units, but we found no significant differences in model performance for comparable hyperparameter settings, so we excluded them from our study.

7.3.2 Case study: LEMD

In this work, we focus our case study on the *Adolfo Suárez-Madrid Barajas airport* (ICAO code, *LEMD*), the leading Spanish airport in terms of passenger traffic and the fifth in Europe in 2022. LEMD has four physical runways, arranged as two pairs of parallel runways, that can be used for either takeoff or landing operations, depending on the current runway configuration. LEMD uses two different configurations: *north* (north-facing runways are used for takeoffs and south-east-facing runways are used for landings) and *south* (vice-versa), which are chosen on the basis of weather conditions and available resources. A sample of 200 flights landing at LEMD for January 2022 are illustrated in Figure 7.3, which shows the prevailing runway configuration at that moment.

The *north* configuration was the most frequently used configuration in the first two months of 2022, but from March onwards, the distribution between configurations became more even due to the change in prevailing weather conditions. Due to this fact, the estimated time of arrival at Madrid-Barajas is even more uncertain. This is because incoming flights may need to execute different approximation maneuvers depending on the airport's current configuration in order to land on the assigned runway. These maneuvers may take several minutes and can not be anticipated, since the landing runway is only assigned and communicated to the pilot when the aircraft is already close to the airport and therefore cannot be used as input to the proposed model.

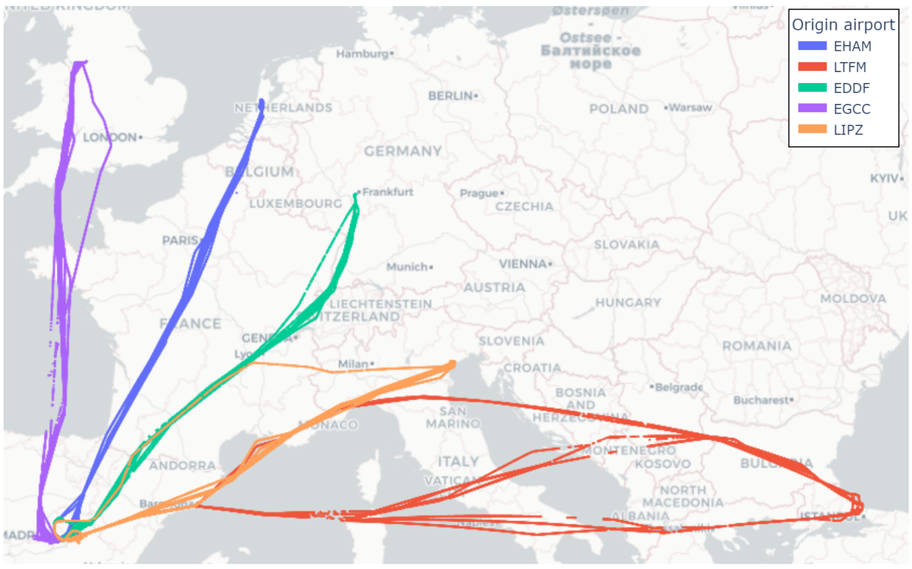

Figure 7.3: A sample of the trajectories from the five airports considered in the study of the individual models.

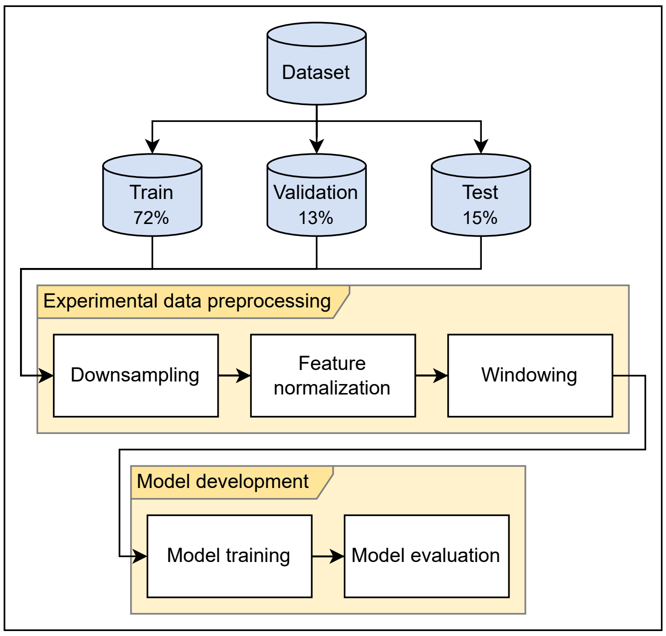

Figure 7.4: Model training workflow.

7.3.3 Datasets

Our study covers inbound flights at LEMD in the first nine months of 2022, that is, the collected data corresponds to the period between January 1 to September 30, 2022. As introduced in Section 7.2.2, we remove several trajectories due to the presence of holding manoeuvres. In total, 338 trajectories (1.61%) with holding patterns were removed. Since there are significant imbalance in the number of flights from each departure airport, we restrict our study to the 40 most frequent routes to ensure a minimum contribution from each origin. Then, we set the aforementioned limit of 70 trajectories per month from each origin airport. After these filtering operations, the resulting dataset consists of 19,633,275 state vectors from 20,560 trajectories. The monthly distribution of trajectories among the different routes is shown in Appendix C.

The dataset is divided into the *train*, *validation* and *test* subsets (containing 72.25%, 12.75%, and 15% of the trajectories respectively). A randomized, stratified approach is applied by distributing trajectories in direct proportion to their monthly and route frequency. In this way, the trajectories are evenly distributed across the three subsets according to the distribution of the original data. Finally, data in each subset is adapted to the particular model configuration, according to the process described in Section 7.2.2, as illustrated in Figure 7.4.

7.3.4 Metrics

The models are evaluated using MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) metrics. MAE is the mean of the absolute values of the differences between each objective RTA value, y_i, and the predicted value, $\hat{y}_i$, for every input i (Equation 7.2). RMSE is the square root of the mean value of the squares of the differences between each objective value and the predicted value across all examples (Equation 7.3). Due to its linear nature, MAE weights equally each example regardless of its error value. In RMSE errors are squared, so larger errors are weighted more than smaller errors, and can be used as a metric of the variance of the error values. The values are provided in seconds to enable direct comparisons.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{7.2}$$

$$\text{RMSE} = \sqrt{\frac{1}{\sum_{i=1}^{n}} (y_i - \hat{y}_i)^2} \tag{7.3}$$

In this study, we use two types of metrics: (i) *global metrics* to indicate the mean error value across all sequences (examples) in the dataset, regardless of the point of the trajectory in which they are placed; and (ii) *at-time* and *at-distance* metrics, which are used to characterize the prediction error at particular points of the trajectory. These particular points can be selected by time (e.g. the error 60 minutes before landing) or by distance (e.g. the error at 100 NM from the destination airport).

Note that longer trajectories are subject to greater uncertainty, but "cutting" all trajectories at the same remaining time or at the same distance to the arrival airport allows for fair comparison, regardless of the route they describe. To calculate these metrics, we evaluate the model on the last available sequence at the selected point in the trajectory, provided that it is close enough to the cutting point. We define maximum thresholds of 300 seconds and 10 NM for the difference between the cutting point and the RTA value and distance of the last state vector in the sequence. Otherwise, the sequence is not taken into account in the evaluation, as it is not representative of the designated point in the trajectory.

Model	SP	LB	Time	Units	MAE (s)	RMSE (s)
LSTM	30s	32	16m	10	224.27	338.52
				20	220.06	333.95
				30	233.31	341.55
	30s	64	32m	10	162.90	260.85
				20	**159.24**	**257.82**
				30	161.04	261.52
	60s	16	16m	10	218.77	321.59
				20	218.79	332.58
				30	211.56	320.56
	60s	32	32m	10	164.18	262.22
				20	163.60	263.56
				30	161.17	259.91
GBM	30s	-	-	-	220.45	323.73
AB	30s	-	-	-	222.41	458.28
RF	30s	-	-	-	256.92	552.01

Table 7.3: Global metrics.

7.4 Results

Table 7.3 shows global metric results for different model configurations. To make it easier to refer to individual configurations, we will use the notation (SP:X,LB:Y,U:Z) to indicate the model with sampling period of X seconds, lookback of Y vectors, and Z units, respectively. Note that the third column (*Time*) refers to the length (in minutes) of the input sequences, and it is obtained by multiplying the value of SP and LB. The results for the baseline methods are also reported at the bottom of the table using data sampled at 30 seconds, as in our most prominent configuration.

We first analyze the impact of downsampling by comparing same time values and different samplings. In this case, they report similar numbers for sequences describing either 16 or 32 minutes of flight time, but SP:30 outperforms SP:60 for longer sequences, reporting a reduction in MAE between 0.13 and 4.36 seconds. It is also shown that the number of units has little effect on the reported error values, but it is worth noting that more units imply higher training costs. Finally, the lookback value has the greatest impact on the result. By doubling the number of state vectors of the input sequences, the model has access to twice the flight time, which helps to better characterize its current state and provide a more accurate prediction. (SP:60,LB:32,U:30) reduces MAE in 50.39 seconds compared to (SP:60,LB:16,U:30), and this difference is consistent for 10 and 20 unit models. The gap is even larger for SP:30, where (SP:30,LB:64,U:20) outperforms (SP:30,LB:32,U:20) in 60.82 seconds in terms of MAE. The smallest MAE (159.24 seconds) is reported by the configuration (SP:30,LB:64,U:20), although (SP:60,LB:32,U:30) also reports competitive error values, increasing MAE by ≈ 2 seconds (161.17 seconds). In all cases, the

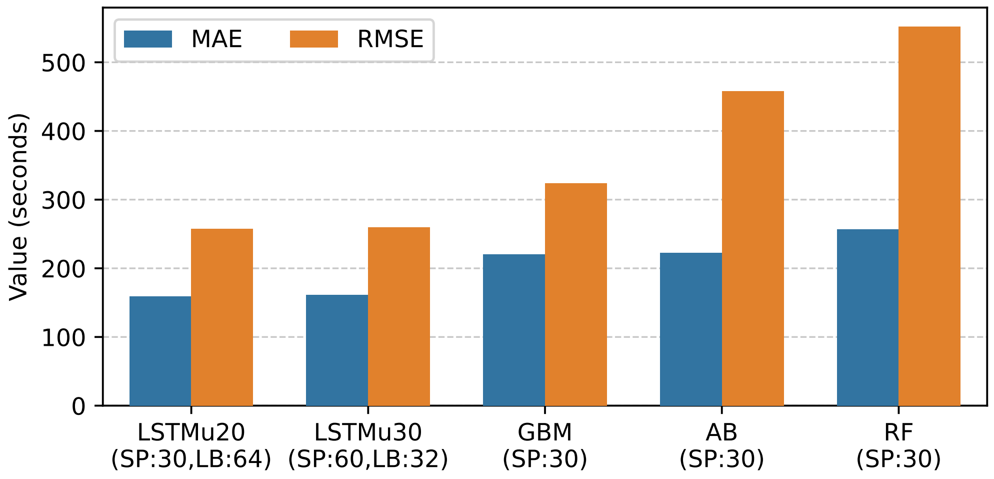

Figure 7.5: MAE and RMSE values of baseline models and best LSTM configuration.

RMSE values for each configuration confirm all of these observations.

We promote the configuration (SP:30,LB:64,U:20) for comparison purposes with the baseline models, which are trained on data with the same sample period. As shown in Figure 7.5, GBM and AB provide similar results, with a MAE of 220.45 and 222.41 seconds, respectively. However, GBM achieves a significantly lower RMSE value than AB, indicating that GBM error values have a lower variance and thus its predictions are more consistent. RF performs poorly in both metrics, confirming that boosting approaches are superior to ensemble approaches, as noted in the related work. The LSTM model is superior to all the baselines, improving their best result in terms of MAE by 61.23 seconds. The study of the RMSE yields the same conclusions, with LSTM providing a result 65.09 seconds better than the best baseline method, GBM, which highlights the improved stability of LSTM over GBM.

To complete this analysis, Tables 7.4 and 7.5 show the values for the at-time and at-distance metrics for our two selected configurations. Accordingly with the global metrics, (SP:30,LB:64,U:20) outperforms (SP:60,LB:32,U:30) on all at-time metrics up to 90 minutes before landing, although the differences are small, between 2 and 4 seconds. However, SP:60 model yields better results at 120 and 150 minutes, with a MAE 8.71 and 18.20 seconds lower, respectively. These evaluation metrics measure the performance when most of the remaining flights are still at cruise-level, although they exclude most short-range routes (e.g. those corresponding to national flights in Spain). SP:30 has a bigger density of data around the destination airport than SP:60, which may be biasing the model towards the area surrounding the airport in spite of farther parts of the routes. Also, sampling reduces the amount of noise in the data, and thus may help the model to generalize better at higher rates when there are not sudden changes in the trajectory. In consequence, SP:30 models might be best fitted to sequences near the airport, where the more detailed representation of the trajectory may benefit the model, while SP:60 performs better on sequences that are farther away from the airport, where there is less variability in the trajectory. This situation holds true for the at-distance metrics, although the differences are generally negligible, in concordance with the global results shown in Table 7.3.

In all cases, our models significantly improve the results reported by GBM, AB, and RF,

MAE at-time	15min	30min	60min	90min	120min	150min
SP:30, LB:64, U:20	**93.80**	**143.50**	**173.92**	**205.12**	264.36	345.03
SP:60, LB:32, U:30	95.84	145.15	175.83	209.12	**255.65**	**326.83**
GBM (SP:30)	153.33	189.21	209.46	253.62	298.01	348.24
AB (SP:30)	113.67	198.19	207.74	251.24	358.89	385.19
RF (SP:30)	191.30	225.64	243.60	320.15	385.22	395.16

RMSE at-time	15min	30min	60min	90min	120min	150min
SP:30, LB:64, U:20	**173.65**	**224.92**	**258.35**	299.46	378.32	477.81
SP:60, LB:32, U:30	175.76	229.11	259.39	**296.10**	**356.93**	473.09
GBM (SP:30)	255.73	299.60	304.55	342.98	398.16	**472.45**
AB (SP:30)	379.49	429.63	308.95	489.61	533.39	521.00
RF (SP:30)	693.32	467.57	543.27	638.97	534.18	525.14

Table 7.4: Metrics at-time for LSTM models (time=32 min). Lowest values (in seconds) for each metric are bolded.

MAE at-distance	25NM	45NM	60NM	100NM	125NM	250NM
SP:30, LB:64, U:20	**55.76**	**81.77**	**102.85**	**132.61**	137.54	**149.19**
SP:60, LB:32, U:30	58.69	83.72	106.45	133.16	**136.46**	149.62
GBM (SP:30)	103.94	149.69	183.82	204.01	211.04	205.24
AB (SP:30)	69.78	105.11	141.97	187.92	211.32	211.66
RF (SP:30)	127.34	173.51	215.69	248.06	248.70	218.16

RMSE at-distance	25NM	45NM	60NM	100NM	125NM	250NM
SP:30, LB:64, U:20	**98.34**	**138.25**	**167.18**	**215.28**	218.77	**233.83**
SP:60, LB:32, U:30	99.72	140.45	172.21	217.29	**215.09**	234.52
GBM (SP:30)	231.68	282.72	326.47	344.97	358.16	361.29
AB (SP:30)	372.43	396.10	424.19	451.72	474.65	469.91
RF (SP:30)	568.44	780.65	765.67	544.76	515.27	431.92

Table 7.5: Metrics at-distance for LSTM models (time=32 min). Lowest values (in seconds) for each metric are bolded.

both in terms of at-time and at-distance metrics, with the exception of RMSE at 150 min, where GBM performs slightly better than LSTM models.

Model	MAE	15min	30min	60min	90min	120min
SP:30, LB:64, U:20	**200.13**	**101.48**	169.57	215.46	286.24	**337.46**
SP:60,LB:32,U:30	201.62	106.02	**167.43**	**208.54**	**282.08**	344.30
GBM (SP:30)	252.89	142.71	191.18	226.06	289.77	383.67
AB (SP:30)	267.37	121.84	221.11	264.02	299.01	419.57
RF (SP:30)	313.49	272.07	275.76	248.12	336.18	473.83

Table 7.6: Evaluation with future data. Global MAE and at-time metrics for the best LSTM configuration and the baseline models are included. Lowest values (in seconds) for each metric are bolded.

7.4.1 Generalization assessment

This section focuses on further evaluating the performance, quality and reusability of our approach, but on a more realistic scenario. For this purpose, we use trajectories that were flown at a later date than the trajectories used to train the models. These trajectories describe incoming flights to Madrid-Barajas (from the 40 selected departure airports) from 1 to 31 October 2022. The resulting dataset (using the generation process described in Chapters 5 and 6) consists of 2,224 trajectories and 2,512,429 state vectors.

Table 7.6 reports global MAE and at-time metrics for this scenario, including our best model configurations and baseline methods. The performance deteriorates for all models, as expected, but ours still lead the comparison. The LSTM SP:30 model reporting a 25.7% higher global MAE, with an increment of 41 seconds in absolute terms, compared to the results shown in Table 7.3. The at-time metrics also increase, but each at a different rate: at 15, 30, 60, 90 and 120 minutes, the MAE increases by 8%, 18%, 24%, 39.5% and 27.5%, respectively. This analysis is also valid for the LSTM SP:60 model, since its figures are comparable. These results indicate that the model is more robust the closer the aircraft gets to the destination airport. The most likely reason is twofold. On the one hand, the maneuvers around the airport are standardized, and there is less variability in how the flight should progress. On the other hand, there is much more data on how the aircraft will behave in the surroundings of the destination airport than in any other area of the airspace, so the model may have learned better about this section of the flights. Having more data for each route should help the model to reduce the gap in the generalizability at different time horizons.

LSTM remains the leading model by a wide margin, improving the MAE by 52 seconds over GBM, which is the most effective state-of-the-art model in this experiment. This comparison also applies for every at-time metric considered, demonstrating LSTM's superiority.

7.4.2 Individual airport models comparison

We conducted several experiments to assess whether a global approach would be better than training specific models for each individual route, as stated in [105]. In particular, we trained several (SP:30,LB:64,U:20) models using data from each of these individual routes. Each model was trained and evaluated on the subset of the global dataset corresponding to the trajectories that belong to its particular route. Therefore, the global model and each individual model share

Table 7.7: Individual route model vs. global model approach. All the trained models are (SP:30, LB:64, U:20). Units in seconds.

Airport	Model	MAE	15min	30min	60min	90min
EDDF	Global	136.29	79.90	129.24	158.21	187.53
	Individual	222.97	176.10	199.61	239.29	229.86
EGCC	Global	170.68	93.88	176.37	200.84	189.73
	Individual	254.59	259.82	267.79	275.49	236.30
EHAM	Global	155.87	93.98	136.58	179.45	244.70
	Individual	214.70	146.67	157.59	237.75	298.91
LIPZ	Global	147.73	98.33	128.35	179.14	184.72
	Individual	203.36	129.06	195.23	222.24	223.89
LTFM	Global	221.89	149.14	143.61	128.44	161.11
	Individual	269.67	230.73	163.20	167.40	202.82
Mean MAE reduction		61.76	68.57	38.98	57.56	21.65

exactly the same data about the corresponding route. All models were trained for 40 epochs in the same conditions that were used to train the (SP:30,LB:64,U:20) global model, including the partition of the data in train, test and validation: the training data of each of these models were the data from the same route that were used to train the global model. The same holds true for test and validation datasets.

The results of these experiments are shown in the Appendix C.2 for all departure airports, but we focus on five of them: Frankfurt Main International (EDDF), Germany; Manchester Airport (EGCC), UK; Amsterdam Airport Schiphol (EHAM), Netherlands; Aeroporto Internazionale Marco Polo di Venezia (LIPZ), Italy; and Istanbul Airport (LTFM), which are chosen to cover the main European routes arriving at LEMD airport. A sample of the corresponding trajectories are depicted in Figure 7.3. Nevertheless, it is worth noting that similar conclusions can be drawn for routes that follow the same airways to fly to Madrid.

Table 7.7 shows the results of evaluating global and individual models on the test datasets corresponding to each of the selected routes. The global model performed consistently better on every metric for each of the routes. The bottom of the table reports the mean improvements observed on these five routes. The largest reductions in MAE can be observed closer to the airport. EDDF and EGCC benefit the most from using the global model: they achieve improvements of 86.7 and 83.9 seconds in global MAE. In particular, EDDF shows a great improvement at 15 minutes, with a reduction of MAE of 96 seconds. These routes have in common that the origin airport is located in the hearth of the European airspace, and thus the synergy with trajectories from other routes, which is observed in the surroundings of the airport, is extended along most of the trajectory. Other routes, such as LTFM, do not have as much path in common with the other trajectories used to train the global model, so they do not benefit as much as the rest of the considered individual routes, with a MAE improvement of 47.8 seconds. In a middle ground we find EHAM, which still achieves a MAE reduction of almost a minute with the global model.

Table 7.8: Results of the ablation test. Units in seconds.

Feature set	MAE	RMSE
Surveillance	208.77	326.09
Surveillance + Flight Plans	206.43	319.23
Surveillance + Weather	215.70	324.33
Full dataset	159.24	257.82

Provided that global and individual models were trained with the same data from each route, it becomes clear that the global model takes advantage of the availability of data from other routes. The European airspace is structured as a route network, where flights converge on airways, which roughly determine the route that an aircraft must follow to fly to the destination airport. Once in an airway, most flights will behave similarly under similar conditions (aircraft model, weather conditions, etc). This synergy becomes more apparent closer to the airport, given that flights follow standard procedures in the surroundings of the airport to approach to the runway and perform a landing procedure, so the model benefits from having data from more flights, even if they belong to different routes. Therefore, global models can learn a wider variety of patterns using data from different routes, thereby improving their performance on each route.

These results show that using a global model is better than maintaining multiple models, one per route (i.e. between an origin airport and a destination airport), for three main reasons: (i) the global model can predict the ETA at any point in the flight more accurately than any of the individual models that only model flights for a specific route; (ii) maintaining a single model is less costly than maintaining a large number of smaller models; and (iii) the global approach can improve predictions even on routes where there are fewer flights due to a lower flight frequency or less data availability.

7.4.3 Ablation test

Our approach combines data from different sources to account for different factors that influence the course of a flight (surveillance, flight plans and weather conditions). We have conducted an ablation test to determine how each factor affects the performance of our approach. In particular, we trained a LSTM model (SP:30,LB:64,U:20) with three different datasets: (i) a dataset containing only surveillance data, which is the core of 4D trajectories; (ii) a dataset enhancing surveillance with flight plan data; and (iii) a dataset combining surveillance and weather data.We replicated the training configuration used for LSTM models in previous experiments, changing only the features used by the model from the original dataset.

The results are shown in Table 7.8. Surveillance data are the backbone of the predictive power of the model, given that they provide detailed information about the trajectory itself, rather than factors influencing on it. On the one hand, adding *flight plan* features (time of day, departure airport, day of week and departure delay) slightly lowers the MAE results of the model. On the other hand, the *weather* features causes an increase in the error. However, flight plan and weather data in combination help the model to refine the results of using surveillance data by 49.5 seconds.

Ap.	Scope	Method	Metric	Main results	Our proposal
[46]	KDFW	RF	MAE$_{20NM}$	58 s	55.76 s
			MAE$_{60NM}$	75.4 s	102.85 s
[69]	KDEN	LR	MAE$_{fut}$	8.63 min	3.33 min
			RMSE	12.2 min	5.35 min
[1]	Two airports	Ensemble	MAE	4.31 min	2.65 min
			RMSE	5.9 min	4.29 min
[9]	Indiv. route	AB/GBM	RMSE$_{LECO}$	3.12 min	2.61 min
[83]	LIMC	GBM	RSME$_{60\ min}$	304 s	258 s
[18]	LSZH	GBM	RMSE$_{45NM}$	3.16 min	2.45 min
			RMSE$_{250NM}$	4.75 min	4.03 min
[25]	WSSS	ET	MAE$_{100NM}$	85-101 s	132 s
			RMSE$_{100NM}$	104-125 s	215 s
[117]	ZBAA	Ensemble	MAE$_{25NM}$	48 s	55s
[79]	ZBAA	LSTM	MAE$_{20\ min}$	89.39 s	93.8 s
[105]	US territory	RF	No comparable results were provided		

Table 7.9: Overview of the main results of the works reviewed in Section 7.1. Each entry includes the used metric, the value reported in the corresponding paper and the value from (SP:30,LB:64,U:20) model.

7.5 Discussion

Our previous experiments confirm LSTMs models as a good choice for ETA prediction, outperforming other machine learning methods that have been successfully used for the same purpose. The comparison with [9] is particularly interesting, because the behaviour of the LSTMs in their evaluation was rather unstable and, in all cases, their accuracy was lower than that reported by methods such as AB or GBM, contrary to what happens with our proposal.

We will now take a closer look at our results in comparison with the main results of the state of the art, which are summarised in Table 7.9. It is worth noting that these results may not be directly comparable in quantitative terms, since each proposal analyzes different case studies, but it is valuable to consider these numerical results in order to elaborate the following qualitative analysis.

One of the strengths of our proposal is its ability to provide good predictions in both the long and short term (from few minutes to several hours), which is not common in the state of the art, where existing proposals focus on one or the other case. The approaches in [1] and [69] are the most similar to ours, given that they predict ETA during the whole trajectory and using one global model for all traffic incoming into a destination airport. [1] reports MAE/RMSE values (on the test set) of 4.31 and 5.9 minutes, respectively, using an ensemble model comprised of a linear regressor and several GBM models. In comparison, our LSTM model with the best configuration (SP:30,LB:64,U:20) achieves a MAE/RMSE of 2.65/4.29 minutes. In [69], the approach based on a non-parametric additive model reached a MAE and RMSE values of 8.63 and 12.2 minutes, respectively, far from those reported by LSTM. However, their model was trained on data from 2010, and tested on data from 2011, so a fairer comparison may be made with the results reported in Table 7.6; still, LSTM achieves 3.33/5.35 minutes when applied on data outside of the training dataset time frame.

Nevertheless, the state of the art is mainly focused on short-term predictions. Muñoz et al. [83] report an RMSE value of 304 seconds at 60 minutes before landing, while our best configuration achieves 258 seconds for the same metric. Wang et al. [117] report 48 seconds of MAE at 25NM from the destination airport, while we achieve 55 seconds for the same distance. It is worth noting that this approach trains specific models for each runway, which removes one of the main sources of uncertainty in the short-term, and provides only very short-term predictions in the TMA of the destination airport. Strottmann Kern and Medeiros [105] also used Ranfom Forest and reported an reduction of 42.7% in MAE with respect to the ETMF predictions (the ATM system used by the Federal Aviation Administration). However, they do not provide any result that enable direct comparison with the results of our study. Glina et al. [46] applied Random Forests to predict short-term ETA in the surroundings of the airport using 5 days of data. The authors indicate that, during those days, there were only one active configuration in the destination airport and good weather conditions, which may reduce the complexity of the problem. The model is evaluated in a short term scenario, at 20 and 60 NM away from the destination airport, with a MAE of 58 and 75 seconds, respectively. Our LSTM model reaches slightly better results at 20 NM, and worse at 60 NM, with 56 and 103 seconds. This difference may be due to the airport characteristics: in LEMD, the 60 NM radious falls inside the area where the aircraft usually maneuver, and therefore is the most difficult part of the flight to be predicted.

Dhief et al. [25] report competitive numbers at 100NM: MAE of 85-101 seconds and RMSE of 104-125 seconds, depending on the runway and the model, outperforming ours: 132 and 215 seconds of MAE and RMSE. The difference between both errors is particularly interesting in our case, because denotes the presence of outliers (which are more heavily penalized by RMSE than by MAE). This is due to the fact that, in our dataset, we kept trajectories with single-loop holding procedures in our dataset (while in [25] they are removed). However, the exact impact of holdings in our results is yet to be determined, and will be addressed as part of our future work. Finally, the GBM model presented in [18] reports 3.16 and 4.75 minutes of RMSE at 45NM and 250NM, while our model reports 2.45 and 4.03 minutes at the same distances.

Ma and Du [79] proposed a complex model that combines trajectory clustering, convolutional neural networks, LSTM and attention mechanisms to predict ETA in the TMA using surveillance, congestion and weather data. That is, their results are based on the elapsed time since the aircraft enters into the TMA of the airport. As indicated in this study, the average time between the entering in the TMA and the landing time is approximately 20 minutes, so their results (MAE of 89.39 seconds) are comparable to our 15 minutes at-time metric, which is a slightly higher (93.80 seconds).

Finally, Ayhan et al. [9] created different models for a sample of individual routes between Spanish airports, and the results supported the authors' claim that LSTM produced unstable predictions. The results of this work may not be directly comparable with our proposal since they make the prediction before the aircraft takes off; however, we include the following observation. In our study, there is only one route in common with [9]; i.e. the route between LECO (A Coruña Airport, Spain) and LEMD. They reported that AdaBoost performed best at predicting this route, with a RMSE value of 3.12 and 3.71 minutes for their AdaBoost and LSTM models, respectively. For reference, our global LSTM model yields a RMSE value of 2.61 minutes, which is the same than that of the individual model for this route. This route was not included in Section 7.4.2 for a deeper analysis because the short length of its trajectories did not allow the calculation of at-time metrics beyond 15 minutes.

Chapter 8

Trajectory prediction

Trajectory prediction plays a central role in ATM, particularly during the tactical phase of flight operations—that is, shortly before and during the flight itself [121]. Accurate trajectory forecasts improve situational awareness, support conflict detection and resolution, and enable more efficient planning of airspace resources. Traditionally, trajectory prediction has been approached using model-based methods grounded in aircraft kinematics. While these approaches offer a simplified and interpretable view of aircraft motion, they often fail to capture the full complexity of real-world flight operations. In particular, they typically neglect the influence of dynamic and context-dependent factors such as delays, adverse weather, airspace restrictions, or interactions with other traffic—factors that significantly affect the evolution of a flight.

In this context, data-driven methods offer a promising alternative. Some of these approaches leverage large volumes of historical data to model the relationship between a flight's observed progression and the operational context in which it takes place. By learning from actual flight trajectories and associated contextual data—such as weather conditions, airport congestion, and scheduling constraints, these models are able to capture complex dependencies and provide more accurate and adaptive predictions of future aircraft positions.

In this chapter, we present our approach to trajectory prediction using models based on Long Short-Term Memory and Temporal Fusion Transformers architectures. We focus on predicting future positions of en-route flights using the 4D trajectory data defined in previous chapters, particularly the position, the speed and the bearing of the aircraft along its flight path.

8.1 Related work

LSTM networks [53] have been extensively applied to approach sequence prediction tasks, whether alone or combined with different techniques to improve the natural capabilities of these neural networks. Shi et al.[99] used LSTM to predict all aircraft trajectories in an airspace sector using surveillance data obtained from ADS-B sources. Their approach was later improved using LSTM networks with constraints [98], which force the model to take into account different aspects of the kinematics and behavior of an aircraft depending on the flight stage. With this new proposal, they provided the best performance of all the studied LSTM-based approaches, with a mean absolute error (MAE) of 0.0050 and 0.0105 degrees for latitude and longitude predictions, and a MAE of 9.96 feet for altitude. However, these results correspond with a single route (that is, flights between two particular airports), in contrast with other works where multiple routes can

Ref.	Method	Strategy	Scope	Inputs					
				Pos	Spe	Tr	Org	Clu	Dst
Single route									
[98] (2021)	ConstLSTM	One-step	Complete traj.	✓	✓	✓			
[78] (2020)	CNN-LSTM	One-step	Complete traj.	✓	✓	✓			
[68] (2022)	LSTM+Att	One-step	Not indicated	✓	✓	✓			
[92] (2022)	BiLSTM	Direct	Complete traj.	✓	✓	✓			
Multiple routes									
[99] (2018)	LSTM	One-step	Airspace sector	✓	✓	✓			
[73] (2020)	RF	One-step	Airspace sector	✓	✓	✓		✓	
[122] (2020)	LSTM	One-step	TMA (<90s).	✓	✓	✓			
[21] (2021)	RM-IMM	One-step	TMA (<60NM)	✓	✓	✓		✓	
[96] (2022)	CNN-GRU	MIMO	TMA	✓	✓	✓			
Ours (2024)	LSTM/TFT	MIMO	Complete traj.	✓	✓	✓	✓		✓

Legend: Pos: Position, Spe: Speed, Tr: Track, Org: Origin, Clu: Clustering, Dst: Distance.

Table 8.1: Summary of data-driven approaches for trajectory prediction.

be predicted with the same model. Zeng et al. [122] also applied LSTM networks to model the aircraft dynamics in a complex scenario, such as the environment around an airport, by analyzing landing and takeoff operations at the Guangzhou airport. More recently, Sahadevan et al. [92] successfully applied bidirectional LSTM to leverage both backwards and forward dependencies in the trajectories time series. They evaluated their approach on a single direction from a single route (from Chhatrapati Shivaji Maharaj airport (VABB) to Kempegowda airport (VOBL)), and reported MAE values of 0.0206 and 0.0160 degrees, and 33.75 feet, for latitude, longitude and altitude, respectively.

Subsequent work has introduced various elements to enhance the LSTM capabilities. For instance, Convolutional Neural Networks (CNN) were combined with LSTM to exploit the spatial aspect inherent in trajectory data. Ma et al. [78] proposed a hybrid model where spatial and temporal inputs are processed sequentially using CNN and LSTM networks. Each component is responsible for extracting different patterns in order to characterize better the current and future state of the aircraft. The case study in [78] is similar to [92] in terms of characteristics (single direction on a single route) and results. Shafienya et al. [96] proposed CG3D, a more complex architecture that combines conventional CNN, GRU and 3D CNN to analyze the different types of information from surveillance data. GRU (Gated Recurrent Units) [20] are similar to LSTM, but with a simpler internal structure. As in other proposals, spatial and temporal features are processed separately by CNN and GRU networks, respectively; however, they also add a 3D-CNN network that combines both sets of features, which in combination with the CNN-GRU defines the proposed CG3D architecture. Their results report a global MAE of 0.1176 for all predicted features.

More recently, attention mechanisms have gained prominence in sequence prediction problems, and therefore have also been applied to trajectory prediction. Jia et al. [68] added self-attention layers after LSTM layers to refine their output by promoting their main features and improving their prediction accuracy. LSTM have also been combined with physical simulation models [21], where the LSTM network processes dynamics information about the aircraft (e.g. surveillance data) and its output is used to refine the result of the RM-IMM estimation algorithm. This approach yielded good results in terms of predicting altitude (MAE of 2.94), although it is outperformed by other works given that latitude and longitude are predicted with a MAE of 0.0373 and 0.0397 degrees, respectively.

With respect to the implemented prediction strategy, most of the analyzed works aim to predict only the next state update given the m last updates: $y_{m+1} = f(x_1, ..., x_m)$. However, this approach has limited use given that the prediction is too short-term and immediate, so many of them extend this approach to perform multiple timestep-ahead predictions in a recursive manner. Sahadevan et al. [92] studied the degradation of the prediction accuracy with the number of predicted timesteps when following a direct strategy. The results are best at 1 timestep for all features considered (latitude, longitude, altitude and time), after which the results rapidly deteriorate as the predicted sequence lengthens. Similar conclusions can be drawn when using a recursive strategy, as demonstrated in [21], due to the prediction error propagation. [96] is the only work that proposes a MIMO approach, similar to ours, where 5 timesteps are predicted based on the previous 100 timesteps.

With regard to the context in which each study is carried out, most of the analyzed proposals focus on the prediction of trajectories on individual routes [78, 99, 92]; i.e. flights between two particular airports, a specific airspace sector [73] or at the TMA of an airport [96, 21, 122], which may include both landing operations, take-off operations, or both. In terms of the inputs for the prediction models, all previous work uses surveillance data (mainly ADS-B data) such as latitude, longitude and altitude. Most of them also account for the speed of displacement (vertical or horizontal) and the heading of the aircraft, which have demonstrated to be determinant in trajectory prediction. Zeng et al. [122] also utilize the aircraft type, and the speed and acceleration of change of the considered features.

Table 8.1 summarizes the main features of the reviewed approaches: the *scope* of their models and the deep learning *method* they used (*RF*: random forests, *RM-IMM*: Residual-Mean Interacting Multiple Models, *CNN-GRU*: convolutional neural networks + gated recurrent unit); the *strategy* followed to make predictions (see Section 3.1); and the *data* used to build these models. It is worth noting that the last row also describes our approach in the same terms, for comparison purposes.

8.2 Architecture

8.2.1 Feature selection

In this section, we describe the features extracted from the processed 4D trajectories used to train the trajectory prediction models. These features are derived from surveillance and flight plan data, as described in Chapter 5. For consistency with the approach followed in Section 7.2.2, the features are grouped by their origin, although they are processed jointly by the models. Table 8.2 summarizes the selected features.

Surveillance data Surveillance data describe the flight state over time according to the considered dimensions. Each data point has an associated *timestamp* and contains information about the 3D-position of the aircraft (*longitude*, *latitude* and *altitude*), the *horizontal speed* and the direction the aircraft is heading (*track*).From these, we also compute the *distance* to the destination airport (using the Haversine formula) to provide a spatial reference with respect to the flight's endpoint..

The surveillance data used in our study are provided by the OpenSky Network [94], as introduced in Chapter 4. OpenSky collects ADS-B messages and post-processes them into state vectors sampled every 5 seconds. To improve the quality and continuity of the trajectories, additional data cleaning and preprocessing steps are applied, as detailed in Chapter 5.

Flight Plan Data Flight plans provide with scheduling data such as the *flight ID* data, and the *departure* and *destination airports*. We use this information to filter the relevant data for the selected case study, and as input feature in the case of the origin airport.

Target Variable The prediction task is formulated as a multivariate time series forecasting problem, where the objective is to predict the future position of the aircraft over the next n time steps. Therefore, for each input sequence, the target variables are the *longitude*, *latitude*, and *altitude* values corresponding to the next n state vectors in the trajectory.

Feature	Description	Sample value
Latitude	Latitude of a position update	51.477402
Longitude	Longitude of a position update	-0.4745
Altitude	Altitude over the ground at a position update (feet)	225.0
Distance	Haversine distance to LEMD airport (miles)	773.208995
Speed	Horizontal speed with respect to the ground (knots)	141.0
Track	Angle between aircraft heading and north	267
Sector	Approach direction to destination airport	1
Departure airport	ICAO code of origin airport	*EGLL*

Table 8.2: Model features extracted from the data sources.

8.2.2 Data preparation

To prepare the dataset used for training the trajectory prediction models, we need to apply similar operations to those applied in Section 7.2.2. In the following, we succinctly describe this process, also illustrated in Figure 7.4.

Trajectory selection and cleaning

The first step consists in selecting and filtering the trajectories relevant to our case study. Similar to Section 7.2.2, trajectories with less than 300 state vectors and trajectories with multiple loops during a holding procedure are discarded. In addition, we remove trajectories that include holding procedures or sustained looping behavior. As discussed in Chapter 5, these maneuvers are often imposed by external air traffic control constraints, such as runway congestion, and are thus exogenous to the trajectory itself. Their presence introduces significant variability and unpredictability in the flight path, which cannot be inferred from the preceding trajectory data alone.

Normalization

Similar to Section 7.2.2, trajectories are downsampled to ensure a regular distribution over time of their state vectors. We set the sampling rate at 15 seconds, which we have found to be a good compromise between reducing the computational cost of the model training and keeping the fidelity to the original trajectory data.

Data is further transformed to fit the model requirements. First, the track feature is transformed according to $f(x) = \sin\left(\frac{x}{2}\right)$ which captures the circular nature of angular measurements and preserves the proximity between 0º and 359º. The track is divided by 2 to fit the output in the $[0, 1]$ range. Since this transformation is not injective in the interval $[0, 360)$, we introduce an auxiliary binary feature (*sector*) to distinguish between headings in the ranges $[0, 180)$ and $[180, 360)$.

As explained in Chapter 3, the categorical features are transformed into real values using label encoding (i.e. replacing each categorical value with an integer), and all features are normalized into [0,1] range according to Equation 8.1, where v is the original value of the feature f, and v^f_{min} and v^f_{max} are the observed minimum and maximum values.

$$v' = \frac{v - v^f_{min}}{v^f_{max} - v^f_{min}} \tag{8.1}$$

The final step consists in transforming each trajectory into fixed-length sequences suitable for model training. As detailed in Chapter 3, we apply a windowing operation with a sliding window of length l, generating all possible sub-sequences of l consecutive state vectors within each trajectory. For each input window, the target output is defined as the values of the next p vectors, corresponding to the desired prediction horizon. To ensure temporal continuity within each sequence, we discard any sub-sequences that include a gap of more than 180 seconds between consecutive vectors.

8.3 Experiments

This section presents the experimental framework used to assess the proposed methods for trajectory prediction. It begins with a description of the setup, detailing the evaluated models, the case study dataset employed to develop the models, and the performance metrics used to evaluate predictive accuracy and overall effectiveness.

8.3.1 Models

In this section, we describe the implemented models based on the LSTM and Temporal Fusion Transformer architectures. Since these architectures were introduced in Chapter 3, we do not provide a detailed discussion of their relevance to this task.

8.3.1.1 LSTM

4D trajectories consist on long sequences (several hundreds or even thousands) of flight points, which hide long and short-term temporal dependencies within the multiple time series they comprise. As stated in Section 3, LSTM-based networks can learn from long sequences of data, such as 4D trajectories, to capture both short-term and long-term dependencies between the elements in the sequence. This fact motivates our decision to build a LSTM-based neural network to predict sequences describing the next states of the aircraft based on the last available states. This architecture consists of a single LSTM layer, with a hidden state of dimension n, and a fully connected (FC) layer to transform the output of the LSTM layer (a vector of length n) into an interpretable trajectory prediction. In this work, this prediction include multiple future states (i.e. timesteps) characterized by three features: latitude, longitude and altitude, in order to determine the short-term future positions of the aircraft. In particular, we aim to predict the next 10 timesteps (i.e. 150 seconds of flight time with the defined sample rate of 15 seconds). The predictions made by LSTM networks are sometimes unstable, and describe maneuvers that are unrealistic for an aircraft in the air (such as a jagged paths that oscillate around the true trajectory followed by the aircraft). We experimented with a LSTM-based architecture with two additional FC layers after the LSTM layer to apply a quadratic polynomial interpolation as a curve smoothing operation [19] on the output of the network. Thus, LSTM layers are responsible of extracting and interpreting time-dependent patterns, and FC layers should transform this interpretation into more feasible trajectory predictions.

We evaluated two main hyperparameters: the input window size (m) and the number of units of the LSTM network. On the one hand, input window size determines the number of individual elements (state vectors) the model expects to receive to make a prediction. The longer the sequence, the more information the model has to characterize the evolution of the flight. However, longer sequences require more computing power or model complexity to learn long-term, complex patterns from the data. On the other hand, the number of units determines the dimensions of the internal representation that the model constructs from the input data. The higher this value, the more complex the model and the greater the risk of overfitting. We tested window sizes between *25* and *60*, in increments of *5*, and a range of different numbers of units between *10* and *35*, based on previous experiences with this task.

We assigned fixed values to the other hyperparameters of the model: we set the *sigmoid function* as the activation function; the batch size to *128*; the *loss function* to mean absolute error (MAE), and Adam [70] is used as optimizer. We experimented with different activation functions (namely, linear, hyperbolic tangent and sigmoid), but the sigmoid function provided the best results in terms of training stability and convergence speed. During training, early stopping was used as a regularization measure to prevent overfitting. The models were trained for *40* epochs, and the version with the smallest validation loss was selected for evaluation. The best model was configured with 30 LSTM units and input sequences of size 55.

8.3.1.2 TFT

We also applied the original TFT architecture [75] on 4D trajectory data to explore its capabilities at the trajectory prediction task. While the architecture also involves LSTM, the multi-head attention mechanisms should extract the key information from the output of LSTM layers to further improve the trajectory predictions.

We tuned three hyperparameters in the TFT model: head attention size, number of elements in the hidden layer, and the input window size. The head attention size is the number of parallel attention layers applied to the input of the model. Each of these layers may learn to detect different patterns, effectively "paying attention" to different aspects in the time series. The number of hidden elements corresponds with the number of units in the LSTM network that is integrated in the TFT architecture. The evaluated head attention sizes were 1, 2 and 4, and the number of hidden elements ranged from 140 to 380, in increments of 60 units. Finally, for the window size we experimented with values of m between *50* and *65* elements. The rest of the hyperparameters were set to the default values defined in the used implementation. The best results were achieved using 4 attention heads, a hidden size of 320 and input sequences of 60 state vectors.

8.3.2 Datasets

Our dataset (referred to as *Complete*) includes data from incoming flights at LEMD airport in the first nine months of 2022 (January 1 to September 30, 2022). It is worth noting that there is a significant imbalance in the number of flights from each departure airport to LEMD, which could bias the model in favour of more frequent routes. Similar to Chapter 7, we choose the 40 most frequent routes to reduce this effect, and limit each route to have a maximum of 50 trajectories per month. As a result, the dataset comprises data from 7,146 trajectories (2,971,108 state vectors). The datasets used for ETA prediction and for trajectory prediction differ in two main aspects. First, the threshold of maximum trajectories per month and route was lowered in this chapter, since the trained models are more complex and required additional training time and computational power. Thus, we reduced the amount of data to alleviate the training process. Second, the feature set is more restrictive in this chapter; many of the features used for ETA prediction are not relevant to predict the trajectory throughout the flight, so including them would serve no purpose for this task.

A second dataset is defined (called *Airport* from now on), which is a subset of the dataset described above. This dataset contains all state vectors that were emitted closer than 80NM to the destination airport. The purpose of *Airport* dataset is two-fold: (i) to evaluate the performance of the developed models in the TMA of the airport; (ii) to produce results that are comparable to those reported in the state of the art, which mainly focus on this area around the airport, and not on the whole trajectories.

The dataset is divided into the usual *train*, *validation* and *test* subsets (containing 72.25%, 12.75%, and 15% of the trajectories respectively), as depicted in Figure 7.4. A randomized, stratified approach is applied by distributing trajectories in direct proportion to their monthly and route frequency. In this way, the trajectories are evenly distributed across the three subsets according to the distribution of the original data. Finally, data in each subset is adapted to the particular model configuration, according to the windowing process described in Section 7.2.

8.3.3 Metrics

The models are evaluated using MAE (Mean Absolute Error), RMSE (Root Mean Squared Error) and MSE (Mean Squared Error) metrics.

- MAE is the mean of the absolute values of the differences between each objective sequence, $(x_{t+1}, ..., x_{t+m})$, and the predicted sequence, $(y_{t+1}, ..., y_{t+m})$, for every input $(x_{t-p}, ..., x_t)$ (Equation 8.2).

- MSE is the mean value of the squared differences between each objective value and the predicted value (Equation 8.3).
- RMSE is the square root of the mean value of the squares of the differences between each objective value and the predicted value across all examples (Equation 8.4).

$$\mathrm{MAE} = \frac{1}{n}\sum\nolimits_{i=1}^{n} |\mathbf{y}_i - \mathbf{x}_i| \tag{8.2}$$

$$\mathrm{MSE} = \frac{1}{n}\sum\nolimits_{i=1}^{n} (\mathbf{y}_i - \mathbf{x}_i)^2 \tag{8.3}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n} (\mathbf{y}_i - \mathbf{x}_i)^2} \tag{8.4}$$

Where n is the number of examples in the evaluation dataset, $\mathbf{y}_i$ is the tensor comprising the predicted next p future states, and $\mathbf{x}_i$ is the tensor that contains the actual next p timesteps.

Due to its linear nature, MAE weights equally each example regardless of its error value. In RMSE and MSE errors are squared, so larger errors are weighted more than smaller errors, and can be used as a metric of the variance of the error values. The MAE and RMSE values are provided in degrees (latitude and longitude) and feet (altitude) to enable direct comparisons. MSE values are normalized in the range [0,1).

MSE was used as metric during the preliminary experiments. MAE and RMSE are used in the evaluation of the final models in order to enable direct comparisons with the state of the art.

8.4 Results

8.4.1 Preliminary studies

In this section, we report some findings in our preliminary study to guide the experimentation process.

Approach	MSE LSTM	MSE LSTM-FC	MSE TFT
Basic	0.000126	0.000124	**0.0000749**
Basic+sector	0.000125	0.000119	0.0000877
Expanded	0.000118	0.000124	0.0000942
Expanded+sector	**0.000117**	**0.000117**	0.0000768

Table 8.3: MSE (scaled) on the validation set for all feature sets

8.4.1.1 Iterative vs. Complete sequence

In our first tests, we evaluated the recursive and MIMO strategies to approach the trajectory prediction task. We implemented two simple LSTM networks, which predicted one timestep and ten timesteps ahead, respectively. The results confirmed the conclusions exposed in [13].

The recursive approach performed worse, with a MSE of 0.004400. This strategy led to an accumulation of errors that increased with successive predictions. In particular, the predicted trajectory often showed non-existent turns after a small deviation in the predicted values (mainly latitude or longitude). On the other hand, the MIMO strategy, while making the model more difficult to fit, had a much lower MSE value: 0.000129 (an order of magnitude lower). Therefore, we adopted MIMO strategy for the rest of our experiments.

8.4.1.2 Feature set

Two different sets of input features were put under evaluation: *Basic* (latitude, longitude, altitude and track), and *Expanded* (basic features plus speed, departure airport, and distance to destination airport). We also evaluated the effect of the sector feature in the prediction capabilities of the models.

The results are shown in Table 8.3. Both LSTM models benefited from having the additional features and improved their MSE values with the *Expanded+sector* feature set, while the TFT model performed best with the *Basic* feature set. However, in both cases the differences were rather small. The best sets of features for each architecture are used for the rest of the experimentation.

8.4.2 Main results

Table 8.4 shows the main results of our experiments on the *Complete* dataset, corresponding to the complete trajectories from flights arriving at LEMD airport, and the *Airport* dataset, where only the vectors within the destination airport TMA are considered. The MAE and RMSE values on each dataset are also presented in Figure 8.1 (top and bottom, respectively). We report per-feature values in their original scale and units (i.e. latitude and longitude in degrees, and altitude in feet) to facilitate their interpretation and analysis.

On the one hand, the LSTM and LSTM-FC models report very similar numbers for the Complete dataset on all the considered features, in terms of both MAE and RMSE. There are small differences depending on the predicted feature: LSTM performs better in longitude (0.0364 vs 0.0373 degrees, equivalent to 4.04 and 4.14 km, respectively) and altitude (242.8 vs 259.2 feet), while LSTM-FC has a slight advantage at predicting the latitude (0.0216 vs 0.0208 degrees, around 2.3 kilometers). The TFT model report different results. The MAE for the latitude is the same as the achieved by LSTM, although with half the variance according to the RMSE (0.0403 vs 0.0910 degrees). The MAE for longitude is worse, with 0.0574 degrees (6.38 km), but the RMSE halves those presented by the LSTM and LSTM-FC models. Finally, the results for altitude are better in terms of MAE and RMSE, with the lowest values of the three models (189.8 and 465.9 feet, respectively). The difference between MAE and RMSE leads to the conclusion that LSTM models perform better than TFT in the individual trajectories that are closer to the most frequent trajectory, and worse in less frequent scenarios (such as adverse weather conditions, diverted flights or special procedures like holdings or go-arounds), where they make larger errors than TFT and therefore get penalized by the RMSE metric.

We also trained and evaluated the models on the *Airport* dataset (with the same model configuration used for the Complete dataset). The bottom line in Table 8.4 shows the corresponding results. Here, the TFT model provides the best performance at predicting the 2D position (latitude and longitude), and improves significantly its altitude predictions with respect

Model	Dataset	MAE			RMSE		
		lat	lon	alt	lat	lon	alt
LSTM	Complete	0.0216	**0.0364**	242.8	0.0910	0.1470	820.2
LSTM-FC	Complete	**0.0208**	0.0373	259.2	0.0910	0.1490	820.2
TFT	Complete	0.0216	0.0574	**189.8**	**0.0403**	**0.0834**	**465.9**
LSTM	Airport	0.0238	0.0544	360.9	0.0350	0.0940	557.7
LSTM-FC	Airport	0.0255	0.0855	367.5	0.0360	0.1430	590.6
TFT	Airport	**0.0091**	**0.0104**	**225.3**	**0.0162**	**0.0238**	**381.9**

Table 8.4: Evaluation results for the Complete and Airport datasets. Values in original units (degrees for latitude and longitude, and feet for altitude). The best values for each result set and feature are marked in bold.

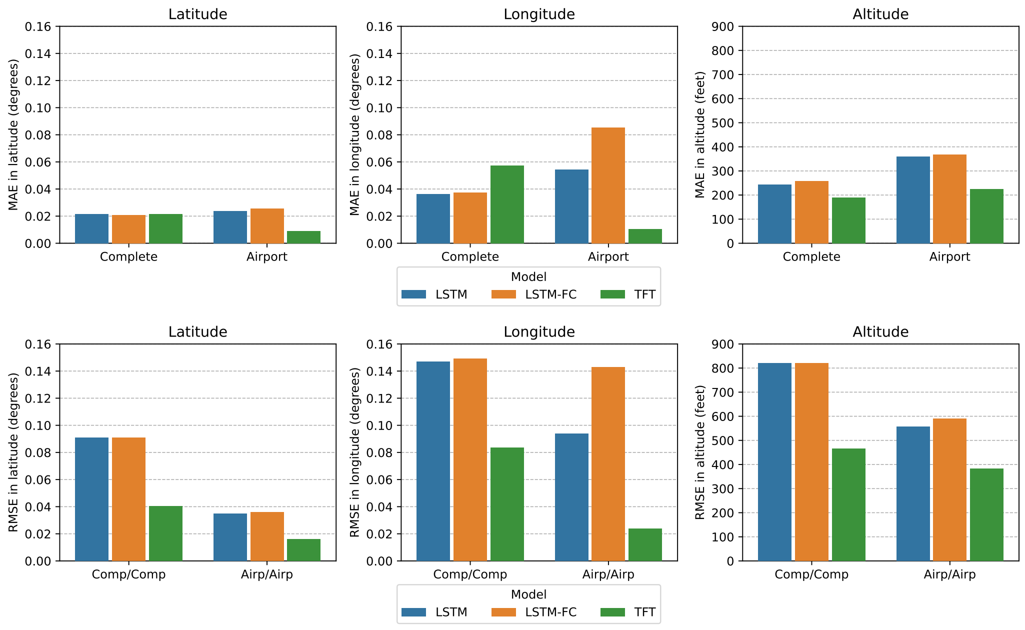

Figure 8.1: Comparison of the MAE (top) and RMSE (bottom) values of the models in the considered scenarios.

to the previous experiment. Our intuition is that data from outside the TMA is too disperse and scarce to train the TFT model correctly, so its performance should improve as we increase the size of the dataset. This hypothesis is further discussed in a later section. Inside the TMA, all the trajectories converge, which reduces the variability found in the data, and the size of the dataset is large enough for the model to capture the movement patterns around the destination airport. On the other hand, the LSTM models provide error values that are on-par than those observed on the other experiments in terms of latitude and latitude, while the longitude predictions are worse.

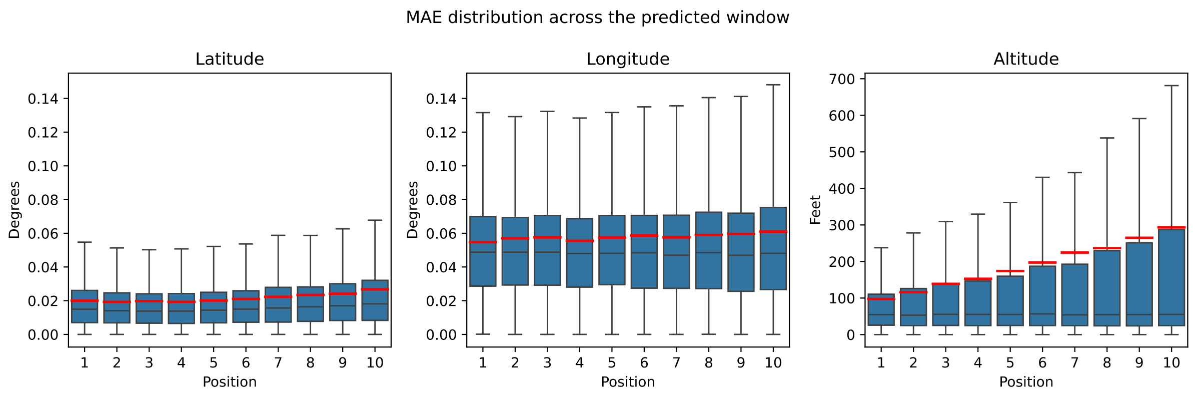

Figure 8.2: Distribution of the mean absolute error of the TFT model across the predicted window. The red lines indicate the mean values across the vectors in each position.

	Latitude		Longitude		Altitude	
Position	Mean	StD	Mean	StD	Mean	StD
1	0.0205	0.0251	0.0559	0.0506	99.70	195.86
2	0.0193	0.0255	0.0559	0.0507	116.93	246.48
3	0.0194	0.0276	0.0562	0.0537	137.73	294.76
4	0.0196	0.0299	0.0564	0.0562	157.96	340.22
5	0.0202	0.0322	0.0570	0.0590	179.30	386.18
6	0.0210	0.0342	0.0576	0.0615	200.48	429.71
7	0.0221	0.0363	0.0581	0.0637	221.22	470.83
8	0.0233	0.0385	0.0588	0.0660	241.56	510.90
9	0.0247	0.0405	0.0594	0.0686	261.88	551.68
10	0.0260	0.0439	0.0592	0.0711	281.24	594.38

Table 8.5: Mean and standard deviation values for the MAE values on each of the positions in the predicted window.

Ref	Year	Approach	Strategy	Scope	MAE			RMSE		
					lat	lon	alt	lat	lon	alt
Single route										
[98]	2021	ConstLSTM	One-step	Complete trajectory	**0.0050**	**0.0105**	9.96	**0.0203**	0.0482	45.34
[78]	2020	CNN-LSTM	One-step	Complete trajectory	0.0170	0.0710	33.83	0.0230	0.0860	40.14
[68]	2022	LSTM+Att	One-step	Not indicated	0.0373	0.0397	**2.94**	0.0464	0.0494	**3.9**2
[92]	2022	BiLSTM	Direct	Complete trajectory	0.0206	0.0160	33.75	0.0264	**0.0222**	52.34
Multiple routes										
[99]	2018	LSTM	One-step	Airspace sector	0.0725	0.0552	**77.95**	0.2295	0.1337	**134.51**
Ours		TFT	MIMO	TMA	**0.0091**	**0.0104**	225.3	**0.0162**	**0.0238**	381.9
		LSTM	MIMO	Complete trajectory	0.0216	0.0364	242.80	0.0910	0.1470	820.20

Table 8.6: MAE and RMSE values for latitude, longitude and altitude features from comparable proposals.

8.5 Discussion

Table 8.6 summarizes the main results of the state of the art described in Section 8.1, including those reported by the best LSTM and TFT-based approaches presented in this study. Although none of these approaches is directly comparable, as they all represent different case studies (different airports, number and type of routes, etc.), we think it is interesting to analyze these numbers to position our current contributions with respect to the existing literature. To do this, we have arranged the results in Table 8.1 according to the number of routes that each approach able to predict: the top part shows the results of approaches that work with data from a single route, while the bottom part shows the reported results of techniques (including ours) that are able to predict the trajectory for multiple routes. On the other hand, it is worth noting that most of the studied approaches implement an one-timestep prediction (with the exception of [92] and its Direct approach), while ours uses a MIMO approach. This means that the errors reported in those works correspond to the error in the prediction of the next timestep (which is the one with the least uncertainty), while our results describe the errors for the next 10 timesteps, whose uncertainty increases progressively. In consequence, the error made by the model would vary across the predicted window. This intuition is confirmed in Figure 8.2, where the mean absolute error distribution on each of the predicted window positions is shown for the Complete dataset. The outliers (values that exceed 1.5 times the interquartile range) have been removed from the graph for clarity. The mean and standard deviation values (without excluding the outliers) are shown in Table 8.5. The mean MAE for altitude increases monotonically from 100 to 281 feet. The dispersion of the values also increases progressively due to higher uncertainty the further in time we aim to predict. The error distribution in latitude and longitude is more uniform along the predicted window in terms of mean errors and dispersion. However, there is still a slight but steady increment of the error the further we advance within the window. It is worth noting that this fact is not evident in the Figure 8.2 due to outlier removal, although it becomes clear in Table 8.5. The only exception is that the error in latitude is slightly higher in the first predicted vector than in the second one.

Taking these aspects into account, our numbers are competitive with those reported in the state of the art in terms of latitude and longitude. TFT outperforms all the considered works in terms of the MAE in 2D positioning except [98], albeit this case study is performed on a single route, which significantly reduces the variance of the predicted trajectories, and only predicts the next timestep. Similar conclusions can be made for our LSTM model, which manages to compete with, or even outperform, most of the analyzed LSTM-based proposals in terms of MAE.

The situation is different when it comes to altitude, with our models performing worse than the state of the art. Figure 8.3 shows the distribution of the MAE values of the TFT model at different distances to the destination airport. The mean and median error values are mostly uniform up to 650 NM, although the dispersion of the error values fluctuates significantly. These variations in the metrics are aligned with the start of the routes: as shown in Figure 8.4, many trajectories manifest ascending patterns at several points within the considered geographical scope, introducing additional variability into the data of flights occurring in those areas. As the aircraft gets closer to the airport, the flight patterns become more clearly defined, which help the model to provide more consistent results. This can be observed in the range (200-650), where most of the air traffic organizes at different, but defined, flight levels. However, the error is still lower than acceptable from a domain point of view: MAE in altitude is 318 feet, which is much lower than the minimum vertical separation for aircraft crossing paths standardized by ICAO.

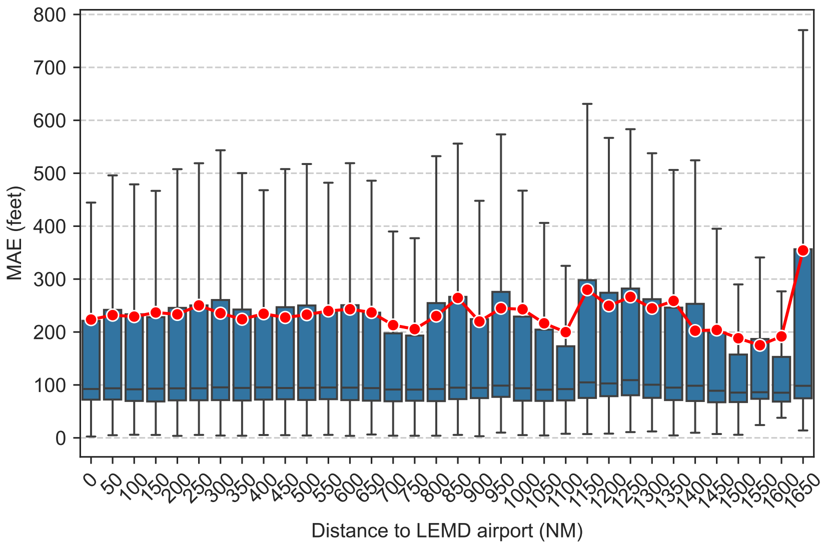

Figure 8.3: Distribution of the mean absolute error values of the TFT model on the altitude feature at different distances to the destination airport. The mean value at each distance is indicated in red.

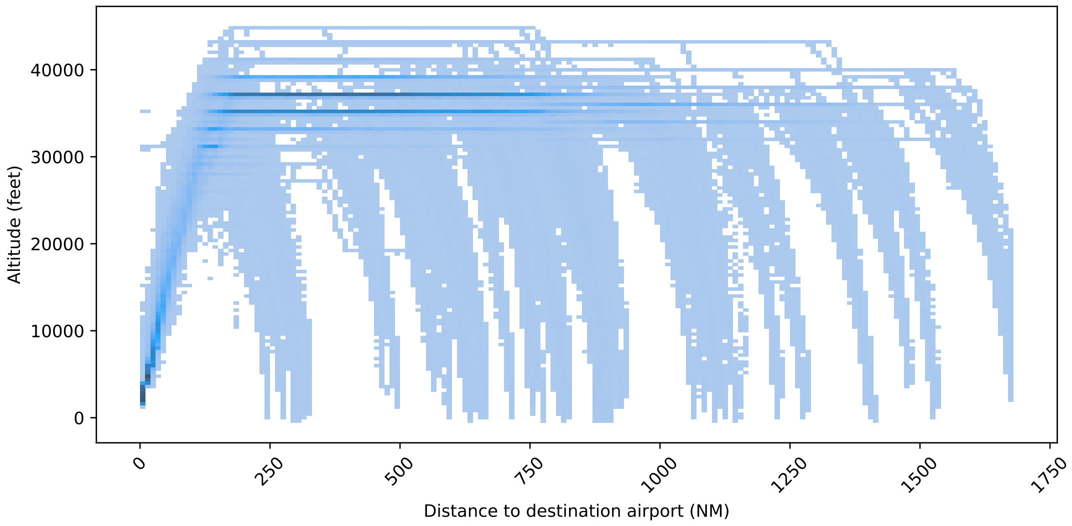

Figure 8.4: Heatmap describing the correlation between altitude and distance values.

ICAO specify a minimum separation of 1,000 feet for aircraft flying at 29,000 feet at most, and 2,000 feet if they fly above this value [62].

Chapter 9

Conclusions and future work

This chapter presents the main conclusions of the work carried out throughout this Thesis. After reviewing the initial objectives and hypothesis, we summarize the key contributions, discuss the results obtained, and outline potential directions for future research.

At the beginning of this Thesis, we proposed three main objectives for guiding our research and helping to validate the research hypothesis. These objectives presented strong dependencies among them, and this fact required us to tackle them in order to ensure some progress at each of them before being able to advance to the next objective. However, thanks to the methodology adopted in this Thesis, we were able to make concurrent progress on all objectives after consolidating the initial results. As a consequence, it has been possible to leverage our increasing knowledge on each area to reinforce advancement in the other areas.

Objective RO-01. Conceptual model. The first objective was the consolidation of a conceptual domain model that captured the key entities and relationships in the relevant areas of Air Traffic Management. Our initial efforts were dedicated to surveying existing resources, including ATM ontologies, reference data models and other specialized resources like official documentation published by authorities and organizations such as ICAO or EUROCONTROL. Although comprehensive, most of these resources were either too broad in scope or misaligned with the specific needs of this work. In particular, models such as AIRM, while highly detailed and standardized, introduced levels of abstraction or coverage that were not practical for our integration and analytical goals. Consequently, we developed a tailored domain model centered on the concept of *Flight* as the core entity. Around this central concept, additional entities were progressively incorporated based on their relevance to unify and accurately describe the temporal and operational aspects of flights. This pragmatic, flight-centric modeling allowed us to focus on the integration of heterogeneous data sources and the generation of enriched 4D trajectories, which would serve as the foundation for subsequent tasks. Finally, to ensure consistency with existing standards and promote interoperability, we established a mapping between our domain model and the AIRM concepts. This mapping was not intended to provide full compliance, but rather to align our model semantically with recognized references.

Objective RO-02. Data workflow. The second objective was the design and implementation of a data integration workflow capable of transforming the raw data obtained from heterogeneous data sources into enriched 4D flight trajectories. This task was built upon the domain model developed previously, which determined the information requirements that had to be satisfied by the integration of these data sources. First, we started by identifying and characterizing the data sources we could have access to, so the suitability of each of them for feeding our data workflow could be assessed. The selection process for data sources was primarily driven by two criteria: data availability and accessibility. To ensure coverage of the information requirements established in the domain model, we first examined the availability of data elements across candidate sources. In those cases where no single source could provide complete coverage, integration of heterogeneous datasets was considered a necessary step to be addressed later in the process. Accessibility, on the other hand, was greatly enhanced thanks to the support provided by BR&TE, which granted us access to a wide set of datasets and sufficient computational resources to explore and process them effectively.

Additionally, we performed a comparative assessment of data quality across sources for concepts where alternatives existed. While some entities, such as flight plan data, could only be obtained from a single reliable source, other concepts like ADS-B surveillance data were available from multiple providers. The selection was thus based on identifying those sources that best aligned with our requirements in terms of coverage, reliability, and usability within a research context. Some additional sources were consulted to enhance our understanding of the domain, though they were not incorporated into the final workflow due to scale, licensing, or usage constraints.

Once the relevant sources had been identified, we designed the data integration pipeline. This process addressed issues of data quality, solved inconsistencies and duplicated records, and produced an integrated representation of flight trajectories in the form of enriched 4D trajectories. Such 4D trajectories comprised sequences of aircraft states that combine spatial and temporal information with additional attributes describing the status of the flight. Building upon this integrated data, we defined a trajectory abstraction that combines the raw 4D trajectory with relevant metadata (e.g. information extracted from the flight data, or the quality assessment of the trajectory), facilitating semantically meaningful access to flight data. On this abstraction, additional processing was applied to enhance their data quality. The most significant among them included the temporal reordering of state vectors within each trajectory and the correction of anomalous values, as exemplified by the filtering and smoothing of altitude data. To monitor and validate the quality of the processed data, we also designed a set of data quality metrics. These metrics allowed us to evaluate the impact of the integration and cleaning steps, and to iteratively refine the pipeline.

Finally, the data integration process was evaluated through a case study involving a significant dataset. We selected over 6000 flights corresponding to all commercial movements between a group of eight European airports over a two-week period. This experiment provided a solid empirical basis to assess the effectiveness and scalability of our approach.

Objective RO-03. Use cases. The third objective aimed to leverage the constructed 4D trajectories for implementing different solutions to support ATM operations. To this end, we selected two tasks of critical importance, such as predicting the estimated time of arrival (ETA) and predicting the future trajectory of an en-route flight, to propose two novel solutions based on deep learning methods that adressed these tasks.

The **first use case** addressed the prediction of arrival times for en-route flights. This task is particularly challenging due to the wide range of factors that can influence the evolution of a flight, especially during its cruise phase, where uncertainty is high. However, accurately estimating the arrival time well in advance is critical for improving the management of airport resources, such as runway assignments and terminal airspace, and for increasing the efficiency of key operations like sequencing arrivals and minimizing holding times.

To address this, we applied deep learning techniques capable of capturing the complex dependencies between multiple features that describe the flight context at each moment. The selected model architectures were based on Long Short-Term Memory (LSTM) networks, given their effectiveness in modeling long temporal sequences. Although some studies had noted stability issues with LSTM models [9], our initial experiments showed promising results compared to alternative approaches. In addition to LSTM, we explored other architectures such as convolutional networks for sequences and encoder-decoder structures, but none of them surpassed the performance of LSTM in our preliminary experiments.

Unlike many previous studies that focus on specific origin-destination pairs or predefined routes, we opted for a broader approach that aimed to predict the arrival time for any inbound flight to a given airport, regardless of its origin. This required not only a careful selection of predictive features, but also the inclusion of data sources that were not initially covered by the domain model. To this end, we performed a comparative analysis of the features used in related works and evaluated the feasibility of integrating missing features (such as the airport weather forecasts) into our data workflow.

The prediction model was evaluated through an extensive experiment covering a nine-month period and including all flights arriving at Madrid Barajas Adolfo Suárez Airport (LEMD). Results showed that our LSTM-based models outperformed the all the ensemble methods that constitute the state-of-the-art for the task (e.g., AdaBoost, Gradient Boosting Machines, Random Forests), both in terms of accuracy and robustness across heterogeneous flight patterns. In particular, our models showed higher accuracy across all stages of the flight, with the mean error reduced in 10-48 seconds with respect to the best ensemble method at the different measured time milestones (between 15 and 150 minutes before landing). The error also showed greater consistency, with a quadratic error lower in 41-82 seconds in the last two hours of flight time. These results demonstrate that the implemented methods enhance the predictability of flight ETA for an improved awareness of the airspace state.

The **second use case** focused on short-term trajectory prediction. The goal in this case was to anticipate the aircraft's future path by predicting several upcoming positions, which would enable the early detection of potentially hazardous situations, such as loss of separation or entry into restricted areas. Accurate trajectory prediction contributes directly to the enhancement of airspace safety and efficiency.

We began by reviewing the state of the art to understand the different strategies for sequence prediction and to identify the most relevant features and modeling approaches. Building on the experience gained in arrival time prediction, we adopted a similar LSTM-based approach. Additionally, we explored the use of a novel architecture particularly suited to this problem: the Temporal Fusion Transformer (TFT). TFT is designed to combine sequential data present in our enriched 4D trajectories with static contextual information (e.g., flight plan metadata), leveraging attention mechanisms to capture relevant patterns over time. This makes it especially appropriate for our integrated data setting.

The feature set used in this case was more compact, focusing on spatial and dynamic attributes such as position, bearing, speed, and the origin airport. We again conducted a comprehensive experiment using nine months of data for flights arriving at Madrid, testing the performance of the models over a wide variety of trajectory shapes and entry points into the terminal area. The LSTM-based architectures yielded results comparable to those in the state of the art, while TFT models showed promise due to their ability to exploit both temporal dependencies and static contextual inputs. In particular, the TFT model predicted latitude and longitude for the next 2.5 minutes of flight-time within the TMA with mean errors of 1 and 1.15 kilometers, respectively, and both the LSTM and the TFT models performed well with errors of 2.3 and 3.4 kilometers at any moment of the flight. Notably, our models addressed the full set of inbound traffic rather than being limited to a single route, marking a significant step toward generalizability and operational applicability.

The results achieved in the three research objectives allow us to validate the different components of our research hypothesis:

> *The integration of data from heterogeneous sources related to Air Traffic Management through a data transformation workflow to produce enriched 4D trajectory data according to a unified data model, can improve the operational efficiency of Air Traffic Management operations by leveraging data-driven approaches based on machine learning.*

Therefore, we consider the research hypothesis validated. The overall goal of this Thesis, that is, to demonstrate the value of integrating heterogeneous ATM data into a unified, enriched representation that enables advanced data-driven applications, has been successfully achieved.

9.1 Future work

This Thesis lays the groundwork for several promising directions of future research, which can be grouped along the three main axes of the work: the domain model, the data transformation workflow, and the application of data-driven methods.

Domain model evolution

There are several relevant areas in the developed domain model that can be further developed to improve coverage and support additional use cases. In particular, a more comprehensive description of the airspace structure is needed, including the definition of controlled airspace divisions, airways, runways, and airport surroundings (such as SID/STAR procedures and relevant navigation points). Additionally, better integration between trajectory data and meteorological conditions is an important challenge. This would involve associating trajectories with weather observations not only at the airport, but also along the route.

Data workflow extension

Extending the domain model will require the incorporation of new data sources to satisfy the emerging information requirements. Some candidate sources are already known and were partially studied during this work. For instance, ASOS (Automated Surface Observing System) can provide accurate weather observations at the airport, while GFS (Global Forecast System) offers global meteorological forecasts along the route. The integration of additional surveillance data sources such as FlightRadar24 could also be beneficial to mitigate some limitations observed in OpenSky Network, particularly in areas with low reception coverage and problematic areas like the closest airspace to the airport, including the runways, where OpenSky lacks coverage, which restricts further analysis of landing operations.

Exploration of advanced models

The enriched 4D trajectory data enables further exploration of novel deep learning architectures beyond those evaluated in this Thesis. Preliminary experiments with architectures such as TSMixer [17] and Liquid Neural Networks [50] have shown encouraging results. These approaches offer promising alternatives for modeling complex temporal and spatio-temporal patterns and should be studied in greater depth. Additionally, emerging techniques in self-supervised learning and model interpretability could be explored to enhance the robustness and transparency of prediction models in ATM.

Implementation of additional use cases

The constructed 4D trajectories can support additional use cases beyond those explored in this Thesis. Their detailed spatio-temporal resolution, combined with integrated flight and contextual information, makes them well suited for the support of critical tasks. Some examples of these are airspace capacity assessment, by analyzing traffic density and factors affecting sector throughput; or early conflict detection, by anticipating potentially hazardous situations through the analysis of predicted aircraft positions, especially within terminal areas. They also provide a solid basis for flight efficiency analysis, enabling comparisons between actual and optimal trajectories and facilitating the study of contributing factors to inefficiencies. The use cases implemented in this Thesis confirm the feasibility of this enriched representation for operational applications.

Overall, these research directions offer a natural continuation of the work developed in this Thesis and open the door to new applications and improvements in Air Traffic Management efficiency and safety.

Part IV

Appendices

Appendix A

Data sources description

This appendix provides an overview of all data sources used in our research, according to their constituent data fields. All fields are described, along with their domains and illustrative examples. For certain categorical attributes, or attributes that contain codes that require interpretation, additional tables are provided with a description of their values with the purpose of clarifying their meanings.

A.1 OpenSky state vectors

Property	Description	Domain/range	Example
icao24	ICAO 24-bit ID of the transponder (hexadecimal)	[0-9A-F]{6}	34444b
callsign	Callsign of the flight (can be null)	[A-Z]{1,8}	IBE05ZB
origin_country	Country name inferred from the ICAO 24-bit address		Spain
time_position	Unix timestamp (seconds) for the last position update		1688335192
last_contact	Unix timestamp (seconds) for the last update in general		1688335191
longitude	WGS-84 longitude in decimal degrees	(-180,180]	-105.7453
latitude	WGS-84 latitude in decimal degrees	[-90,90]	42.5375
baro_altitude	Barometric altitude in meters	[0,inf)	10675.62
on_ground	Boolean value which indicates if the position was retrieved from a surface position report	{0,1}	false
velocity	Velocity over ground in m/s	[0,inf)	238.66
true_track	True track in decimal degrees clockwise from north (north=0°)	[0,360]	71.14
vertical_rate	Vertical rate in m/s		9.1
sensors	Receiver ID		
geo_altitude	Geometric altitude in meters		11163.3
squawk	The transponder code (squawk)	[0000,7777]	2604
spi	Whether flight status indicates special purpose indicator.		false
position_source	Origin of this state's position (ADS-B, ASTERIX, MLAT or FLARM)	0,1,2	0
category	Aircraft category		

Table A.1: OpenSky state vectors attributes. All fields can be null except *icao24* and *timestamp*.

A.2 OpenSky Flights API

Property	Description	Domain/range	Example
icao24	ICAO 24-bit ID of the transponder (hexadecimal)	[0-9A-F]{6}	4ca61d
firstSeen	Estimated time of departure (Unix time)		1657573398
estDepartureAirport	ICAO code of the estimated departure airport	[A-Z]{4}	LPPR
lastSeen	Estimated time of arrival (Unix time)		1657576440
estArrivalAirport	ICAO code of the estimated arrival airport	[A-Z]{4}	LEMD
callsign	Callsign of the flight (if any)	[A-Z]{1,8}	RYR1AC
estDepartureAirportHorizDistance	Horizontal distance of the last received airborne position to the estimated departure airport	[0,inf)	218
estDepartureAirportVertDistance	Vertical distance of the last received airborne position to the estimated departure airport	[0,inf)	14
estArrivalAirportHorizDistance	Horizontal distance of the last received airborne position to the estimated arrival airport	[0,inf)	1915
estArrivalAirportVertDistance	Vertical distance of the last received airborne position to the estimated arrival airport	[0,inf)	60
departureAirportCandidatesCount	Number of other possible departure airports	[0,inf)	1
arrivalAirportCandidatesCount	Number of other possible arrival airports	[0,inf)	3

Table A.2: OpenSky flight attributes. All fields can be null except *icao24*, *firstSeen* and *lastSeen*.

A.3 TAF reports

Property	Description	Domain/range	Example
raw_text	The raw TAF report in METAR format		See footnote(1)
station_id	Station identifier	[A-Z0-9]4	UTSS
issue_time	Issue time (date and time the forecast was prepared)	Datetime	2023-07-08 21:00:00
valid_time_from	The start time of when the report is valid	Datetime	2023-07-08 21:00:00
valid_time_to	The end time for when the report is valid	Datetime	2023-07-09 22:00:00
fcst_time_from	The start of the forecast time	Datetime	2023-07-09 02:00:00
fcst_time_to	The end time of the forecast	Datetime	2023-07-09 11:00:00
change_indicator	Forecast change indicator	See Table A.4	TEMPO
time_becoming	The time a forecast transition ends (in becoming reports)	Datetime	2023-07-09 11:00:00
probability	Percent probability of the change (if any)	[0,100]	30
wind_dir_degrees	Wind direction (the direction from where the wind is blowing)	[0,360]	60
wind_speed_kt	Wind speed (in knots)	[0,inf)	10
wind_gust_kt	Wind gust, corresponds with the maximum wind speed in the last 10 minutes (in knots)	[0,inf)	20
wind_shear_hgt_ft_agl	Wind shear height above ground level (in feet)	[0,inf)	2000
wind_shear_dir_degrees	Wind shear direction	[0,360]	180
wind_shear_speed_kt	Wind shear speed (in knots)	[0,inf)	25
visibility_statute_mi	Horizontal visibility (in miles). The value 6+ denotes anything higher than 6	[0,6+)	6
altim_in_hg	Altimeter (in mmHg)	[0,inf)	29.9500007629394
vert_vis_ft	Vertical visibility (in hundreds of feet)	[0,inf)	2
wx_string	String encoding of the weather conditions		BR VCSH

(1)TAF AMD CYSJ 082159Z 0821/0906 23010KT P6SM SCT012 TEMPO 0821/0824 6SM BR BKN005 OVC012 FM090000 18007KT 6SM BR BKN005 BECMG 0900/0902 1/4SM FG VV002 RMK NXT FCST BY 090000Z

sky_condition	Describe the sky conditions at several flight levels		
↪ sky_cover	Sky coverage.	See Table A.5	FEW
↪ cloud_base_ft_agl	Cloud base		50
↪ cloud_type	Cloud type	See Table A.6	CB
turbulence_condition			
↪ turbulence_intensity	Turbulence intensity values	[0,9]	1
↪ turbulence_min_alt_ft_agl	Minimum altitude for turbulence (in feet)		100
↪ turbulence_max_alt_ft_agl	Maximum altitude for turbulence (in feet)		3100
icing_condition			
↪ icing_intensity	Icing intensity values	[0,9]	2
↪ icing_min_alt_ft_agl	Minimum altitude for icing conditions (in feet)	2000	10000
↪ icing_max_alt_ft_agl	Maximum altitude for icing conditions (in feet)		18000
temperature	An array of temperature data		
↪ valid_time	Date and time at which temperature will become valid		2023-07-09 09:00:00
↪ sfc_temp_c	Surface temperature	(-273,inf)	23
↪ max_temp_c	Max temp	(-273,inf)	40
↪ min_temp_c	Min temp	(-273,inf)	10

Table A.3: TAF report description.

Code	Description
TEMPO	Temporary change: the weather will revert to the previous state after the indicated period
BECMG	Progressive and permanent change to new weather conditions
FM	Immediate and permanent change to new weather conditions
PROB	For weather changes that are not certain, probability of change (30 or 40%)
COR	Corrected report (replaces completely the previous one)
AMD	Amended report (partial report that overwrites the previous one)

Table A.4: Description of change indicator codes in TAF reports.

Code	Description
FEW	Few clouds
SCT	Scattered clouds
BKN	Broken clouds
OVC	Overcast clouds
NSC	No significant cover
NCD	No clouds measured
SKC	No cloud cover
CLR	No cloud cover below 12K feet

Table A.5: Description of sky condition codes in TAF reports.

Code	Description
CB	Cumulonimbus cloud
TCU	Towering cumulus of great vertical extent

Table A.6: Description of cloud type codes in TAF reports.

A.4 Network Manager Flight Plans

Property	Description	Domain/range	Example
flightPlanData.structured			
↪ flightPlan			
↪ ifplId	Unique identifier assigned to flight by the NM system. Mandatory.	[A-Z]{2}[0-9]{8}	AT03394183
↪ aircraftId	Information regarding the aircraft. Mandatory.		
↪ aircraftId	Assigned callsign to the flight	[A-Z0-9]{2,7}	THY7JQ
↪ registrationMark	Registration mark of the aircraft	[A-Z0-9]{1,50}	TCJST
↪ aircraftAddress	ICAO address of the aircraft (hexident or transponder code)	[0-9A-F]{6}	4BAA74
↪ ssrInfo	SSR code assigned to the aircraft by the ATS and its transmission mode (squawk)	[0-9]{4}	
↪ aerodromeOfDeparture	Airport from which the aircraft is set to depart. Mandatory.		
↪ icaoID	ICAO ID of the designated airport		LTFM
↪ aerodromesOfDestination	Airports where the aircraft is set to arrive. Mandatory.		
↪ aerodromeOfDestination	Airport where the aircraft have to arrive		
↪ icaoID	ICAO ID of the designated airport		LKPR
↪ alternateX	Airports where the aircraft can be rerouted into if necessary. Multiple airports can be defined (X=1,2...)		
↪ icaoID	ICAO ID of the designated airport		EDDC
↪ enrouteAlternateAerodromes	Aerodromes where the aircraft can land in case of emergency along the route		
↪ icaoID	ICAO ID of the designated airport		LOWW
↪ takeOffAlternateAerodromes	Aerodromes where the aircraft can land in case of emergency at take-off		
↪ icaoID	ICAO ID of the designated airport		LTFM
↪ whatIfRerouteReference	Indication of AO What-If rerouting reference in a flight plan	[1,9]	
↪ numberOfAircraft	Number of aircraft in a formation flight	[1,99]	2

↪ aircraftType	Aircraft type of the aircraft performing the flight. Mandatory.		
↪ ↪ icaoID	ICAO identifier of the aircraft type	[A-Z][A-Z0-9]{1,3}	A321
↪ totalEstimatedElapsedTime	Total estimated elapsed time. Mandatory.	HHMM	0226
↪ wakeTurbulenceCategory	Wake turbulence category. Mandatory.	See Table A.8	MEDIUM
↪ flightType	Type of the flight. Mandatory.	See Table A.9	SCHEDULED
↪ flightRules	Flight rules applicable to the flight (visual or instrumental). Mandatory.	See Table A.10	IFR
↪ estimatedOffBlockTime	The estimated time at which the aircraft will start the movement associated with departure. Mandatory.		2024-01-02 14:35:00
↪ icaoRoute	Encoded Flight Plan ICAO Route. Mandatory.		See footnote(2)
↪ stayInformation	Type of activity (training, photographic mission, etc) to be performed during the stay periods mentioned in the route of the flight		
↪ enrouteDelays	A list of delays or holdings planned at given points		
↪ ↪ delay	Delay		
↪ ↪ point	The point where the delay is planned to occur (ICAO)		
↪ equipmentCapabilityAndStatus	Represents the capability and status of the equipment of the aircraft of the flight. EQUIPPED/NOT_EQUIPPED. Mandatory.		
↪ surveillanceEquipment	Surveillance equipment of the aircraft of the flight EQUIPPED/NOT_EQUIPPED. Mandatory.		
↪ otherInformation	Any other flight data Items specified in the bilateral agreement. Refer to ICAO 4444 field type 18 (Other information). Mandatory.		
↪ supplementaryInformation	Supplementary flight data. Refer to ICAO 4444 field type 19 (Supplementary information). Mandatory.		
↪ iataFlightNumber	Public ID of the flight assigned by IATA.	[A-Z]{2}[0-9]{1,4}	
↪ arrivalAirportSlot	Designated slot to start the landing operation.		
↪ departureAirportSlot	Designated slot to start the take-off operation.		

(2) N0441F180 NETEX5T NETEX Z282 DIBIR L179 ROMIN/N0463F320 L179 SASKI L608 LOGAN LOGAN1H

↪ flightPlanOriginator	Entity that submitted the flight plan to NM (usually the airline)	
↪ airFiledData	Estimate data provided when the flight plan was filed airborne	
↪ atsUnitId	ICAO ID of the ATS unit from which supplementary flight plan data can be obtained	
↪ startingPoint	Starting point at which the air filed flight plan is submitted	
↪ clearedLevel	Level at which the aircraft has been cleared to join controlled airspace over the given point	
↪ estimatedTimeOver	Estimated date/time over the given point	
↪ eetsToLocations	Array of locations and the corresponding accumulated elapsed time to these locations (FIR, named point or coordinates).	
↪ elapsedTime	The elapsed time.	11
↪ fir	ICAO ID defined for the reference FIR	LBSR
↪ point	ICAO ID defined for the reference point	
↪ latitude	Latitude of the reference point	
↪ longitude	Longitude of the reference point	
↪ aircraftOperator	ICAO ID of the aircraft operator that owns the aircraft	THY
↪ operatingAircraftOperator	ICAO ID of the operating aircraft operator	THY
↪ originatorAddress	Originator address. Mandatory.	EGLLBAWD
↪ knownErrorIndicators	List of the known error indicators	MODE_S
timestamp	Timestamp of the message.Mandatory	
uuid	Unique identifier assigned to message by the NM system. Mandatory.	

Table A.7: Flight plan description.

ICAO code	Value	Description
H	HEAVY	Take-off mass of 136,000 kg or more
M	MEDIUM	Take-off mass between 7,000 kg and 136,000 kg
L	LIGHT	Take-off mass of 7,000 kg or more
J	SUPER	Special cases where the take-off is much higher than the limit set for HEAVY

Table A.8: Description of wake turbulence category codes in flight plans.

ICAO code	Value	Description
S	SCHEDULED	Scheduled commercial air transport services
N	NOT_SCHEDULED	Not scheduled commercial air transport services: charter, on demand...
G	GENERAL	Other flight services: instructional flying, aerial work, non-commercial business aviation...
M	MILITARY	Military flights
X	OTHER	Any flight type that is not of a type described above

Table A.9: Description of flight type codes in flight plans.

ICAO code	Value	Description
I	IFR	Flight guided by instrumental flight rules
Y	IFR_THEN_VFR	Flight guided by instrumental flight rules that is expected to transition to visual flight rules
V	VFR	Flight guided by visual flight rules
Z	VFR_THEN_IFR	Flight guided by visual flight rules that is expected to transition to instrumental flight rules

Table A.10: Description of flight rules type codes in flight plans.

A.5 Network Manager Flight Data

Property	Description	Domain/range	Example
flightData			
↪ flightId	IFPL id and flight keys associated to the flight		
↪ id	Unique IFPL id.Mandatory	[A-Z]{2}[0-9]{8}	AT04873391
↪ keys			
↪ aircraftId	Assigned callsign to the flight	[A-Z0-9]{2,7}	ITY1463
↪ aerodromeOfDeparture	ICAO ID of the aerodrome of departure.		LIRF
↪ nonICAOAerodromeOfDeparture	True if the aerodrome of departure is not an ICAO one.		false
↪ airFiled	True if the flight plan was filed airborne		false
↪ aerodromeOfDestination	ICAO ID of the filed aerodrome of destination		LIPZ
↪ nonICAOAerodromeOfDestination	True if the filed aerodrome of destination is not an ICAO one		false
↪ estimatedOffBlockTime	Estimated off-block date/time according to the latest processed flightplan message (by IFPS)	Datetime	2023-09-06 07:20:00
↪ aircraftAddress	The 24 bit aircraft address		4CA8D1
↪ aircraftType	ICAO identifier of an aircraft type.	[A-Z]{1}[A-Z0-9]{1,3}	A320
↪ flightState	Flight state.	See Table A.12	ATC_ACTIVATED
↪ estimatedTakeOffTime	Estimated take-off time: the take-off time corresponding to the FTFM flight profile. The corresponding estimated off-block time of the FTFM flight profile is the flight.estimatedTakeOffTime - flight.taxiTime		2023-09-06 07:33:00
↪ calculatedTakeOffTime	Calculated take-off time: the take-off time corresponding to the RTFM flight profile. The corresponding calculated off-block time is the flight.calculatedTakeOffTime - flight.taxiTime		2023-09-06 07:57:00

↪ actualTakeOffTime	Estimated Actual take-off time: the take-off time corresponding to the CTFM flight profile. The corresponding estimated actual off-block time is the flight.actualTakeOffTime - flight.currentlyUsedTaxiTime.	2023-09-06 07:57:00
↪ estimatedTimeOfArrival	Estimated time of arrival: time of arrival according to the FTFM flight profile.	2023-09-06 08:21:00
↪ calculatedTimeOfArrival	Calculated time of arrival: time of arrival acorting to the RTFM flight profile.	2023-09-06 08:45:00
↪ actualTimeOfArrival	Estimated Actual time of arrival: time of arrival according to the CTFM flight profile.	2023-09-06 08:44:00
↪ actualOffBlockTime	Estimated Actual Off-Block time: off-block time according to the CTFM flight profile.	2023-09-06 07:33:00
↪ aircraftOperator	Aircraft operator.	ITY
↪ operatingAircraftOperator	Operating aircraft operator.	ITY
↪ readyEstimatedOffBlockTime	Last flight plan related estimated off-block time, but amended by NM or READY message (filing time). Present if different from the latest flight plan related estimated off-block time in flightId.keys; null otherwise	
↪ cdmEstimatedOffBlockTime	The last ready estimated off-block time amended by E-DPI or T-DPI_t message (target takeoff time). Present if different from the latest flight plan related estimated off-block time in flightId.keys; null otherwise.	
↪ icaoRoute	Complete ICAO 4444 item 15 information comprising of initial requested speed and flight level and route	See footnote(3)
↪ routeLength	Length of the route.	277
↪ estimatedElapsedTime	Estimated elapsed time.	

(3) N0436F350 LATRA7P LATRA UM133 LERGA/N0430F350 UY30 LATAM UY22 NISAR NISAR6R

↪ flightDataVersionNr	The version number of the flight data: the version number increases as the flight data changes over time.		27
↪ arrivalInformation	Arrival information from API messages and exchange of times over the coordination fix point.		
↪ registrationMark	The last aircraft registration mark provided	[A-Z0-9]{1,50}	TCJST
↪ aircraftType	The last ICAO Aircraft type provided		
↪ aircraftIATAId	The last aircraft IATA identifier provided		
↪ arrivalTaxiTime	The last estimated or actual taxi time from landing to gate provided		
↪ nmArrivalProcedure	The identifier of the Standard Instrument Arrival Route currently selected by NMOC		
↪ arrivalRunway	The identifier of the last assigned arrival runway provided		
timestamp	Timestamp of the message.Mandatory		2023-09-06 08:39:34
uuid	Unique identifier assigned to message by the NM system. Mandatory.		See footnote[4]

Table A.11: Flight data description.

[4]3168861d-c4a5-4eb2-a9a3-a67e42cecc-74

A.6 Network Manager Flight Data

Code	Value	Description
FILED	Filed	A flight is in the state Filed after ETFMS received its FPL
FILED_SLOT _ALLOCATED	Filed, Slot Allocated	A Filed flight is in the state Filed, Slot Allocated after slot allocation was applied to it
FILED_SLOT_ISSUED	Filed, Slot Issued	A Filed flight is in the state Filed, Slot Issued after slot allocation was applied to it and the corresponding SAM was sent
PLANNED	Planned	A flight is in the state Planned after ETFMS received its RPL or if its data comes from PREDICT Flight Data (PFD)
PLANNED_REROUTED	Planned, Re-routed	A Planned flight is in the state Planned, Re-routed after it was re-routed
PLANNED_SLOT _ALLOCATED	Planned, Slot Allocated	A Planned flight is in the state Planned, Slot Allocated after slot allocation was applied to it
PLANNED_SLOT _ALLOCATED _REROUTED	Planned, Slot Allocated, Re-routed	A Planned flight is in the state Planned, Slot Allocated, Re-routed after slot allocation was applied to it and it was re-routed
CANCELLED	Cancelled	Flight data in the state Cancelled is never shown in counts, flight lists, etc... and is almost equivalent to removed flight data, with the exception that a new FPL can correlate with cancelled flight data (rerouteing and departure information)
TACT_ACTIVATED	TACT Activated	A flight is TACT Activated when the ATOT is in the future. If there is a T-DPI-s TTOT: within the CTOT tolerance window (regulated flights); any TTOT (non-regulated flights). Or if there is a A-DPI TTOT: within the CTOT tolerance window (regulated flights); any TTOT (non-regulated flights)
ATC_ACTIVATED	ATC Activated	The reception of a coordination message (FSA, APL, ACH), an Air-Filed Flight Plan (AFIL), an APR, a DEP, an FAA Departure Information (FDI) and a CPR will put a flight in the state ATC Activated
TERMINATED	Terminated	An ATC activated flight (receiving ATC updates such as CPRs) will be Terminated 20 minutes after the latest Actual Time of Arrival (ATA), which is derived from ATC data. A TACT activated flight (no ATC updates received) will be Terminated 180 minutes after the latest ATA, which is derived from the flight plan

Table A.12: Description of flight state codes in flight plans.

A.7 Other datasets

Airlines and airports

Property	Description	Domain/range	Example
name	Complete name of the aerodrome		A Coruña Airport
iata	IATA 3-letter code of the aerodrome	[A-Z]{3}	LCG
icao	ICAO 4-letter code of the aerodrome	[A-Z]{4}	LECO
lat	Latitude of the aerodrome	[-90,90]	43.302059
lon	Longitude of the aerodrome	(-180,180]	-8.37725
country	Name of the country the aerodrome is placed		Spain
alt	Altitude over the sea level of the aerodrome in feet		326

Table A.13: Airports attributes

Property	Description	Domain/range	Example
Name	Complete name of the airline		Iberia
Code	IATA 2-letter code of the airline	[A-Z]{2}	IB
ICAO	ICAO 3-letter code of the airline	[A-Z]{3}	IBE

Table A.14: Airlines attributes

Appendix B

Results of the data processing

This appendix presents the detail of the results derived from the experimentation conducted in Chapter 6. The results are presented aggregated by route (that is, a pair of origin and destination airports) to enable a better interpretation of the results, since the globally aggregated statistics, while insightful, include trajectories with very different characteristics and can hide some relevant observations.

Table B.1 reflects the results of the integration between the flights derived from the flight plan data and the surveillance data to construct individual 4D trajectories, before any trajectory-level processing is applied. For each route, we indicate the origin and destination airports, the number of identified flights, the number of vectors that were integrated with these flights, and the average number of vectors that comprises each constructed trajectory in that route. The dispersion of the distribution for each route is calculated as the standard deviation.

Table B.2 presents a comparison of the same set of routes before and after applying the trajectory processing pipeline described in Chapter 6. For each route, we provide the number of flights, and the average number of vectors per trajectory before and after the trajectory cleaning. We also analyze the variation in the traveled distance by indicating the raw and the processed distances, as well as the dispersion of these values. Finally, the mean processing time is also shown to characterize the time required to compute the trajectories from each route.

Origin	Destination	Flights	Vectors	Mean length (±StD)
EDDF	EGLL	192	153,349	798.6 ± 125.10
EDDF	EHAM	134	78,661	587.0 ± 83.68
EDDF	EKCH	80	55,624	695.3 ± 86.41
EDDF	LEMD	121	169,107	1397.5 ± 189.64
EDDF	LFPG	146	80,767	553.2 ± 94.14
EDDF	LGAV	48	61,439	1279.9 ± 170.88
EDDF	LIRF	81	63,633	785.5 ± 103.82
EGLL	EDDF	195	136,187	698.3 ± 97.72
EGLL	EHAM	203	105,426	519.3 ± 80.35
EGLL	EKCH	132	113,999	863.6 ± 116.06
EGLL	LEMD	145	154,230	1063.6 ± 197.86
EGLL	LFPG	147	70,245	477.8 ± 79.97
EGLL	LGAV	94	159,631	1698.2 ± 199.71
EGLL	LIRF	95	102,067	1074.3 ± 131.61
EHAM	EDDF	151	76,373	505.7 ± 73.80
EHAM	EGLL	200	111,368	556.8 ± 108.98
EHAM	EKCH	168	107,430	639.4 ± 103.69
EHAM	LEMD	130	180,097	1385.3 ± 169.90
EHAM	LFPG	147	77,097	524.4 ± 86.10
EHAM	LGAV	66	102,998	1560.5 ± 180.60
EHAM	LIRF	92	96,197	1045.6 ± 139.71
EKCH	EDDF	81	60,920	752.1 ± 117.43
EKCH	EGLL	121	125,173	1034.4 ± 162.08
EKCH	EHAM	166	124,017	747.0 ± 99.04
EKCH	LEMD	24	44,254	1843.9 ± 213.99
EKCH	LFPG	106	107,153	1010.8 ± 153.65
EKCH	LGAV	34	56,346	1657.2 ± 241.71
EKCH	LIRF	44	59,094	1343.0 ± 168.51

LEMD	EDDF	118	148,991	1262.6 ± 178.19
LEMD	EGLL	152	167,159	1099.7 ± 181.54
LEMD	EHAM	126	161,949	1285.3 ± 178.53
LEMD	EKCH	28	45,021	1607.8 ± 229.72
LEMD	LFPG	111	102,880	926.8 ± 155.00
LEMD	LGAV	80	101,261	1265.7 ± 172.64
LEMD	LIRF	175	138,740	792.8 ± 117.79
LFPG	EDDF	164	95,284	581.0 ± 95.49
LFPG	EGLL	142	72,738	512.2 ± 98.28
LFPG	EHAM	147	81,200	552.3 ± 86.00
LFPG	EKCH	126	114,725	910.5 ± 125.03
LFPG	LEMD	109	111,188	1020.0 ± 174.39
LFPG	LGAV	120	177,412	1478.4 ± 220.03
LFPG	LIRF	117	99,827	853.2 ± 134.70
LGAV	EDDF	64	99,794	1559.2 ± 224.62
LGAV	EGLL	100	200,860	2008.6 ± 247.79
LGAV	EHAM	64	125,682	1963.7 ± 245.16
LGAV	EKCH	44	79,978	1817.6 ± 197.50
LGAV	LEMD	76	105,579	1389.2 ± 267.20
LGAV	LFPG	128	226,420	1768.9 ± 256.55
LGAV	LIRF	109	100,598	922.9 ± 190.59
LIRF	EDDF	79	73,455	929.8 ± 136.47
LIRF	EGLL	95	123,643	1301.5 ± 174.32
LIRF	EHAM	82	106,739	1301.7 ± 147.87
LIRF	EKCH	44	61,398	1395.4 ± 132.89
LIRF	LEMD	172	148,389	862.7 ± 146.07
LIRF	LFPG	143	138,646	969.5 ± 146.75
LIRF	LGAV	100	71,877	718.7 ± 121.65

Table B.1: Summary of integrated trajectory data per origin–destination pair, detailing the number of flights, state vectors, and trajectory lengths (in number of vectors).

Origin	Destination	Flights	Vectors		Distance		Process time
			Initial	Final	Initial	Final	
EDDF	EGLL	192	799	757	642.1 ± 236.49	483.3 ± 65.74	12.4 ± 25.15
EDDF	EHAM	134	587	568	408.9 ± 207.18	308.0 ± 88.55	8.1 ± 2.95
EDDF	EKCH	80	695	670	530.9 ± 132.84	481.6 ± 71.67	6.2 ± 2.24
EDDF	LEMD	121	1398	1361	1122.1 ± 376.94	976.6 ± 115.04	6.1 ± 3.07
EDDF	LFPG	146	553	532	388.4 ± 268.06	307.1 ± 71.98	5.7 ± 2.90
EDDF	LGAV	48	1280	1204	1327.0 ± 281.36	1175.7 ± 37.61	4.2 ± 1.42
EDDF	LIRF	81	786	741	868.2 ± 387.80	645.4 ± 79.66	2.4 ± 1.11
EGLL	EDDF	195	698	664	663.6 ± 268.01	466.7 ± 75.05	7.4 ± 3.04
EGLL	EHAM	203	519	505	324.4 ± 130.09	269.2 ± 17.97	6.9 ± 2.02
EGLL	EKCH	132	864	840	735.2 ± 177.55	654.9 ± 23.89	5.6 ± 2.09
EGLL	LEMD	145	1064	1015	1063.9 ± 320.32	840.0 ± 62.59	5.9 ± 3.54
EGLL	LFPG	147	478	468	299.2 ± 69.71	273.5 ± 21.49	6.3 ± 2.07
EGLL	LGAV	94	1698	1622	1747.3 ± 297.26	1567.7 ± 53.44	5.1 ± 3.27
EGLL	LIRF	95	1074	1035	1053.6 ± 267.02	949.4 ± 110.96	2.2 ± 0.79
EHAM	EDDF	151	506	480	379.6 ± 168.44	272.8 ± 18.28	7.0 ± 2.74
EHAM	EGLL	200	557	531	382.7 ± 163.72	283.8 ± 60.48	11.3 ± 8.89
EHAM	EKCH	168	639	626	468.5 ± 106.29	437.8 ± 23.85	7.8 ± 24.57
EHAM	LEMD	130	1385	1346	1086.8 ± 314.84	957.7 ± 53.59	6.8 ± 3.46
EHAM	LFPG	147	524	506	312.5 ± 84.82	278.0 ± 22.81	6.3 ± 3.08
EHAM	LGAV	66	1561	1487	1513.3 ± 208.99	1405.7 ± 36.54	4.9 ± 1.50
EHAM	LIRF	92	1046	1000	1083.2 ± 562.68	852.6 ± 103.37	3.3 ± 1.54
EKCH	EDDF	81	752	727	563.0 ± 207.22	471.4 ± 136.90	7.0 ± 4.25
EKCH	EGLL	121	1034	1001	802.1 ± 242.97	669.1 ± 60.46	12.9 ± 31.17
EKCH	EHAM	166	747	734	485.3 ± 206.78	420.1 ± 91.80	7.6 ± 4.53
EKCH	LEMD	24	1844	1813	1421.8 ± 193.19	1336.0 ± 71.69	9.1 ± 8.35
EKCH	LFPG	106	1011	992	750.8 ± 323.95	653.7 ± 80.75	6.0 ± 2.51
EKCH	LGAV	34	1657	1609	1536.7 ± 288.74	1376.4 ± 30.72	5.0 ± 2.01
EKCH	LIRF	44	1343	1320	1076.9 ± 217.19	972.6 ± 19.24	11.5 ± 47.45

LEMD	EDDF	118	1263	1225	1298.2 ± 621.88	1034.3 ± 188.41	8.2 ± 4.69
LEMD	EGLL	152	1100	1054	1127.8 ± 338.72	867.7 ± 94.99	8.9 ± 5.54
LEMD	EHAM	126	1285	1252	1130.5 ± 291.84	968.9 ± 80.70	8.1 ± 4.03
LEMD	EKCH	28	1608	1584	1470.7 ± 238.38	1341.9 ± 12.04	7.0 ± 3.19
LEMD	LFPG	111	927	906	838.4 ± 291.60	725.0 ± 49.92	6.6 ± 3.41
LEMD	LGAV	80	1266	1217	1574.4 ± 111.02	1501.7 ± 32.05	4.1 ± 1.50
LEMD	LIRF	175	793	761	973.7 ± 185.55	882.7 ± 52.84	2.6 ± 1.19
LFPG	EDDF	164	581	556	494.9 ± 285.55	366.6 ± 133.12	8.1 ± 4.34
LFPG	EGLL	142	512	498	359.4 ± 131.84	287.5 ± 67.18	11.2 ± 7.80
LFPG	EHAM	147	552	537	398.7 ± 195.47	292.4 ± 29.00	7.9 ± 3.55
LFPG	EKCH	126	911	898	708.6 ± 93.69	675.5 ± 38.67	8.6 ± 28.25
LFPG	LEMD	109	1020	994	788.2 ± 202.00	701.1 ± 72.12	9.3 ± 30.59
LFPG	LGAV	120	1478	1394	1591.8 ± 316.58	1389.1 ± 71.62	4.5 ± 1.81
LFPG	LIRF	117	853	811	896.6 ± 321.97	731.4 ± 32.35	2.7 ± 1.74
LGAV	EDDF	64	1559	1470	1311.8 ± 219.89	1194.5 ± 29.08	7.4 ± 3.46
LGAV	EGLL	100	2009	1896	1822.1 ± 302.14	1603.6 ± 69.13	14.4 ± 32.67
LGAV	EHAM	64	1964	1893	1608.8 ± 303.79	1455.5 ± 126.91	7.8 ± 3.56
LGAV	EKCH	44	1818	1759	1521.3 ± 260.62	1393.8 ± 41.86	8.4 ± 14.39
LGAV	LEMD	76	1389	1326	1608.6 ± 137.99	1525.0 ± 41.15	5.6 ± 3.49
LGAV	LFPG	128	1769	1669	1517.2 ± 513.79	1367.4 ± 106.77	6.3 ± 3.39
LGAV	LIRF	109	923	863	745.8 ± 89.75	721.2 ± 69.61	5.7 ± 2.65
LIRF	EDDF	79	930	890	787.1 ± 284.25	670.5 ± 144.96	6.5 ± 2.36
LIRF	EGLL	95	1302	1248	1163.0 ± 290.41	980.9 ± 78.22	10.7 ± 8.06
LIRF	EHAM	82	1302	1266	1033.6 ± 258.29	895.9 ± 33.30	8.7 ± 5.36
LIRF	EKCH	44	1395	1376	1138.3 ± 409.96	1006.3 ± 41.25	6.5 ± 2.86
LIRF	LEMD	172	863	825	974.6 ± 184.25	891.3 ± 26.38	4.2 ± 2.67
LIRF	LFPG	143	970	929	804.7 ± 194.74	712.3 ± 33.96	5.0 ± 2.36
LIRF	LGAV	100	719	673	730.9 ± 62.88	713.5 ± 19.52	3.3 ± 1.40

Table B.2: Per-route statistics before and after applying the trajectory processing pipeline, including the traveled distance, the trajectory length (in number of vectors) and the average processing time.

Appendix C

Data description of ETAs use case

C.1 Data distribution amongst routes

This Appendix presents the distribution of trajectories amongst the airports considered in the study. The data is divided in months, since the data was sampled to ensure a homogeneous time distribution, and an equivalent representation of each route.

		Month								
Airport	Total	01-22	02-22	03-22	04-22	05-22	06-22	07-22	08-22	09-22
EBBR	628	70	70	71	76	72	69	65	65	70
EDDB	571	73	69	72	67	43	47	58	70	72
EDDF	628	69	65	65	67	66	66	74	77	79
EDDH	345	31	35	26	46	43	31	50	41	42
EDDL	563	73	68	59	68	70	42	51	62	70
EDDM	616	66	68	78	68	66	74	67	60	69
EDDP	499	71	66	56	53	41	54	66	44	48
EGCC	252	39	39	32	35	31	14	16	25	21
EGKK	615	67	69	71	60	64	70	73	71	70
EGLL	612	59	68	74	60	61	69	66	82	73
EHAM	657	68	72	81	77	81	69	75	69	65
EIDW	649	71	75	72	69	77	73	76	70	66
EKCH	284	45	41	28	32	28	18	24	35	33
EPWA	200	32	22	23	20	9	13	18	28	35
LBSF	186	23	12	18	28	18	15	22	22	28
LEBB	587	69	69	68	63	65	62	65	69	57
LEBL	647	74	69	74	71	64	75	77	66	77
LECO	595	58	66	70	62	59	67	70	76	67
LEVC	541	58	70	34	54	66	49	73	71	66

LEZL	596	71	64	51	74	69	56	70	72	69
LFLL	601	70	70	58	69	70	61	63	71	69
LFML	606	67	62	68	72	71	64	64	71	67
LFMN	515	49	44	34	71	70	55	65	72	55
LFPG	555	62	56	59	66	54	66	63	60	69
LFPO	592	59	64	76	76	66	66	63	67	55
LFRS	466	32	28	22	72	73	68	58	62	51
LGAV	401	34	20	29	23	21	64	74	66	70
LHBP	338	31	11	28	53	44	37	34	55	45
LIMC	595	70	40	62	75	79	67	81	56	65
LIPE	561	70	16	62	63	65	66	73	69	77
LIPZ	574	66	27	60	64	76	70	72	66	73
LIRF	585	71	70	64	56	57	70	62	69	66
LIRN	356	41	10	35	-	27	56	59	68	60
LKPR	197	18	20	16	28	16	18	20	34	27
LOWW	570	71	18	62	76	67	73	67	71	65
LPPR	611	70	65	62	74	63	69	68	65	75
LPPT	560	58	55	72	54	63	67	68	61	62
LROP	440	56	30	38	62	50	48	48	58	50
LSGG	615	75	68	62	61	68	70	72	66	73
LTFM	551	50	33	50	72	67	69	70	69	71
Total	20560	2307	1984	2142	2337	2260	2257	2400	2451	2422

Table C.1: Number of total trajectories for each airport and month.

C.2 Results of the individual models

This Appendix presents the results of the evaluation of each individual model, and the corresponding results of the global model, on their respective test datasets, in order to show the improvements explained in the Section 7.5. Where the test dataset had too few examples (less than 10), the results of the evaluation were not included in the table. This happens for some at-time metrics when the route is too short to produce long enough windows, or some windows were discarded due to gaps in the trajectories (windows with gaps longer than 180 seconds with no data are discarded).

		MAE 15	MAE 30	MAE 60	MAE 90	MAE all
EBBR	Individual	130.32	182.54	263.29	410.46	224.42
	Global	107.58	161.66	211.04	376.60	185.19
EDDB	Individual	200.19	206.30	221.80	283.89	254.63
	Global	158.10	177.43	193.63	249.94	215.65
EDDF	Individual	160.81	190.38	235.26	243.22	217.95
	Global	80.98	136.54	166.99	206.64	142.90
EDDH	Individual	156.38	211.76	196.31	219.33	201.22
	Global	73.39	155.37	136.17	157.75	134.54
EDDL	Individual	177.09	249.97	276.45	353.59	274.62
	Global	108.79	214.02	211.17	238.49	196.62
EDDM	Individual	150.57	186.80	235.65	299.33	232.37
	Global	74.07	120.54	143.25	187.94	129.68
EDDP	Individual	111.68	134.04	151.12	224.01	181.29
	Global	83.39	124.43	147.30	200.04	164.77
EGCC	Individual	259.82	267.79	275.49	236.30	254.59
	Global	93.88	176.37	200.84	189.73	170.68
EGKK	Individual	157.17	222.29	200.15	508.26	197.85
	Global	109.61	206.12	189.18	520.55	176.48
EGLL	Individual	163.23	274.88	206.18	467.70	212.00
	Global	99.46	249.22	176.81	303.23	172.41
EHAM	Individual	135.99	148.01	225.03	279.65	205.32
	Global	98.95	141.74	190.47	249.56	163.76
EIDW	Individual	236.97	181.19	216.53	222.39	216.39
	Global	104.65	176.44	174.77	184.62	160.77
EKCH	Individual	147.00	147.99	152.89	203.44	199.87
	Global	81.64	114.08	106.48	151.01	142.20

EPWA	Individual	151.12	177.98	243.64	269.82	293.41
	Global	88.88	112.20	166.09	210.62	204.38
LBSF	Individual	276.49	331.08	370.60	355.86	399.45
	Global	103.98	158.41	158.77	190.08	217.38
LEBB	Individual	-	-	-	-	80.49
	Global	-	-	-	-	54.08
LEBL	Individual	115.01	-	-	-	101.52
	Global	78.41	-	-	-	60.03
LECO	Individual	77.05	-	-	-	79.25
	Global	106.91	-	-	-	83.33
LEVC	Individual	-	-	-	-	339.24
	Global	-	-	-	-	33.11
LEZL	Individual	119.62	-	-	-	66.00
	Global	87.74	-	-	-	64.50
LFLL	Individual	83.09	119.58	396.09	-	110.74
	Global	68.43	106.84	276.51	-	89.39
LFML	Individual	118.67	145.67	-	-	143.32
	Global	101.13	143.22	-	-	123.06
LFMN	Individual	122.94	185.06	298.67	-	175.55
	Global	99.78	162.31	235.79	-	147.18
LFPG	Individual	84.57	140.32	203.30	-	152.01
	Global	73.03	133.18	172.34	-	126.55
LFPO	Individual	136.66	181.16	216.72	-	185.63
	Global	97.48	137.69	208.40	-	146.33
LFRS	Individual	103.34	120.50	-	-	118.34
	Global	84.64	109.34	-	-	89.53
LGAV	Individual	173.54	194.06	283.32	291.90	283.07
	Global	121.66	167.63	252.61	152.02	186.60
LHBP	Individual	207.05	176.42	216.33	255.85	243.84
	Global	63.92	112.86	120.76	184.79	154.41
LIMC	Individual	148.90	147.00	201.68	-	185.53
	Global	91.72	115.08	144.50	-	125.07

LIPE	Individual	133.63	176.53	221.08	493.67	209.38
	Global	82.96	158.42	186.99	362.37	165.45
LIPZ	Individual	129.06	195.23	222.24	223.89	203.36
	Global	98.33	128.35	179.14	184.72	147.73
LIRF	Individual	131.07	181.82	320.35	343.06	178.70
	Global	114.27	163.34	234.84	216.52	145.39
LIRN	Individual	170.17	180.10	267.35	180.26	201.18
	Global	81.68	115.53	193.88	131.63	121.94
LKPR	Individual	282.28	232.62	282.53	325.89	303.93
	Global	101.44	145.04	188.74	171.69	154.58
LOWW	Individual	136.06	188.05	246.55	326.52	287.22
	Global	92.00	169.82	184.31	211.42	202.01
LPPR	Individual	100.28	-	-	-	79.89
	Global	88.14	-	-	-	70.82
LPPT	Individual	125.42	-	-	-	118.41
	Global	81.08	-	-	-	95.34
LROP	Individual	271.66	201.49	217.56	272.13	292.23
	Global	97.87	101.55	126.45	173.66	176.08
LSGG	Individual	66.51	101.89	141.21	-	103.82
	Global	67.87	106.49	125.93	-	92.78
LTFM	Individual	224.68	148.71	166.42	213.30	277.56
	Global	141.92	137.14	140.70	160.26	214.47

Table C.2: Results of the evaluation of individual and global models. Units in seconds.

Bibliography

[1] Anna Achenbach and Stefan Spinler. "Prescriptive Analytics in Airline Operations: Arrival Time Prediction and Cost Index Optimization for Short-Haul Flights". In: *Operations Research Perspectives* 5 (2018), pp. 265–279. ISSN: 2214-7160. DOI: 10.1016/j.orp.2018.08.004.

[2] Saeif Alhazbi, Savio Sciancalepore, and Roberto Di Pietro. "Reliability of ADS-B Communications: Novel Insights Based on an Experimental Assessment". In: *Procs. 34th ACM/SIGAPP Symposium on Applied Computing.* 2019, pp. 2414–2421.

[3] Busyairah Syd Ali, Washington Yotto Ochieng, and Rozaimah Zainudin. "An Analysis and Model for Automatic Dependent Surveillance Broadcast (ADS-B) Continuity". In: *GPS Solutions* 21.4 (Oct. 2017), pp. 1841–1854. ISSN: 1080-5370, 1521-1886. DOI: 10.1007/s10291-017-0657-y.

[4] Busyairah Syd Ali, Wolfgang Schuster, Washington Ochien, Arnab Majumdar, and Chiew Thiam Kian. "A Study of ADS-B Data Evaluation and Related Problems". In: *San Diego* (2013).

[5] Busyairah Syd Ali, Wolfgang Schuster, Washington Ochieng, and Arnab Majumdar. "Analysis of Anomalies in ADS-B and Its GPS Data". In: *GPS Solutions* 20.3 (July 2016), pp. 429–438. ISSN: 1080-5370, 1521-1886. DOI: 10.1007/s10291-015-0453-5.

[6] Busyairah Syd Ali, Wolfgang Schuster, Washington Ochieng, Arnab Majumdar, and Thiam Kian Chiew. "Framework for ADS-B Performance Assessment: The London TMA Case Study: Automatic Dependent Surveillance Broadcast". In: *Navigation* 61.1 (Mar. 2014), pp. 39–52. ISSN: 00281522. DOI: 10.1002/navi.53.

[7] Busyairah Syd Ali and Nur Asheila Taib. "A Study on Geometric and Barometric Altitude Data in Automatic Dependent Surveillance Broadcast (ADS-B) Messages". In: *The Journal of Navigation* 72.5 (Sept. 2019), pp. 1140–1158. ISSN: 0373-4633, 1469-7785. DOI: 10.1017/S0373463319000201.

[8] Gennady Andrienko, Natalia Andrienko, and Georg Fuchs. "Understanding Movement Data Quality". In: *Journal of Location Based Services* 10.1 (Jan. 2016), pp. 31–46. ISSN: 1748-9725, 1748-9733. DOI: 10.1080/17489725.2016.1169322.

[9] Samet Ayhan, Pablo Costas, and Hanan Samet. "Predicting Estimated Time of Arrival for Commercial Flights". In: *Proc. ACM 24th International Conference on Knowledge Discovery and Data Mining (SIGKDD).* Vol. 18. 2018, pp. 33–42. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3219874.

[10] Ana Azevedo and Manuel Filipe Santos. "KDD, Semma and CRISP-DM: A Parallel Overview". In: *Proc. IADIS European Conference Data Mining.* 2008, pp. 182–185. ISBN: 978-972-8924-63-8.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate.* May 2016. arXiv: `1409.0473 [cs, stat]`. (Visited on 09/2023).

[12] George Bebis and Michael Georgiopoulos. "Feed-forward neural networks". In: *IEEE Potentials* 13.4 (1994), pp. 27–31.

[13] Souhaib Ben Taieb, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. "A Review and Comparison of Strategies for Multi-Step Ahead Time Series Forecasting Based on the NN5 Forecasting Competition". In: *Expert Systems with Applications* 39.8 (June 2012), pp. 7067–7083. ISSN: 0957-4174. DOI: `10.1016/j.eswa.2012.01.039`.

[14] Sarah Bolton, Richard Dill, Michael R Grimaila, and Douglas Hodson. "ADS-B classification using multivariate long short-term memory–fully convolutional networks and data reduction techniques". In: *The Journal of Supercomputing* 79.2 (2023), pp. 2281–2307.

[15] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: `10.1023/A:1010933404324`.

[16] Andrés Carrillo López. *GitHub - andresC98/TSF_ Transformers_ TFM: Repository containing my Master Thesis for the M.Sc. Big Data Analytics, titled "Time Series Forecasting with Transformers".* `https://github.com/andresC98/TSF_Transformers_TFM`. Accessed: 2025-07.

[17] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. *TSMixer: An All-MLP Architecture for Time Series Forecasting.* Sept. 2023. DOI: `10.48550/arXiv.2303.06053`.

[18] Gong Chen, Judith Rosenow, Michael Schultz, and Ostap Okhrin. "Using Open Source Data for Landing Time Prediction with Machine Learning Methods". In: *Proc. 8th OpenSky Symposium.* Vol. 59. Dec. 2020, p. 5. DOI: `10.3390/proceedings2020059005`.

[19] Xi Cheng, Bohdan Khomtchouk, Norman Matloff, and Pete Mohanty. *Polynomial Regression As an Alternative to Neural Nets.* Apr. 2019. DOI: `10.48550/arXiv.1806.06850`.

[20] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations Using RNN Encoder for Statistical Machine Translation". In: *Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Oct. 2014, pp. 1724–1734. DOI: `10.3115/v1/D14-1179`.

[21] Hong-Cheol Choi, Chuhao Deng, and Inseok Hwang. "Hybrid Machine Learning and Estimation-Based Flight Trajectory Prediction in Terminal Airspace". In: *IEEE Access* 9 (2021), pp. 151186–151197. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2021.3126117`.

[22] Andrew Cook. *European Air Traffic Management: Principles, Practice, and Research.* Ashgate Publishing, Ltd., 2007.

[23] Augustin Degas et al. “A Survey on Artificial Intelligence (AI) and eXplainable AI in Air Traffic Management: Current Trends and Development with Future Research Trajectory”. In: *Applied Sciences* Computing and Artificial Intelligence (Jan. 2022). DOI: 10.3390/app12031295.

[24] Imen Dhief, Sameer Alam, Nimrod Lilith, and Chan Chea Mean. “A Machine Learned Go-around Prediction Model Using Pilot-in-the-Loop Simulations”. In: *Transportation Research Part C: Emerging Technologies* 140 (July 2022), p. 103704. ISSN: 0968-090X. DOI: 10.1016/j.trc.2022.103704.

[25] Imen Dhief, Zhengyi Wang, Man Liang, Sameer Alam, Michael Schultz, and Daniel Delahaye. “Predicting Aircraft Landing Time in Extended-TMA Using Machine Learning Methods”. In: *Proc. 9th International Conference for Research in Air Transportation (ICRAT)*. Sept. 2020, p. 9.

[26] Guozhu Dong and Jian Pei. *Sequence Data Mining*. Springer Science & Business Media, Oct. 2007. ISBN: 978-0-387-69937-0.

[27] Mariana M. G. Duarte and Mahmoud Sakr. “Outlier Detection and Cleaning in Trajectories: A Benchmark of Existing Tools”. In: *CEUR Workshop Proceedings*. Vol. 3379. 2023.

[28] Mekkaoui Djamel Eddine and Yanming Shen. “A deep learning based approach for predicting the demand of electric vehicle charge”. In: *The Journal of Supercomputing* 78.12 (2022), pp. 14072–14095.

[29] ENAIRE. *AIP España. Air Spaces.* 2024. URL: https://aip.enaire.es/AIP/contenido_AIP/ENR/LE_ENR_2_1_en.html (visited on 07/2025).

[30] ENAIRE. *AIP España. STAR 4 description with North Configuration.* 2024. URL: https://aip.enaire.es/AIP/contenido_AIP/AD/AD2/LEMD/LE_AD_2_LEMD_STAR_4_en.pdf (visited on 07/2025).

[31] ENAIRE. *AIP España. TMA Madrid.* 2024. URL: https://aip.enaire.es/AIP/contenido_AIP/ENR/LE_ENR_6_5-13_en.pdf (visited on 07/2025).

[32] Gabriele Enea and Marco Porretta. “A Comparison of 4D-Trajectory Operations Envisioned for NextGen and SESAR, Some Preliminary Findings”. In: *Proc. 28th Congress of the International Council of the Aeronautical Sciences (ICAS)*. Vol. 5. 2012, pp. 4152–4165. ISBN: 978-1-62276-754-0.

[33] Matthias Englert, Heiko Röglin, and Berthold Vöcking. “Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP”. In: *Algorithmica* 68.1 (Jan. 2014), pp. 190–264. ISSN: 1432-0541. DOI: 10.1007/s00453-013-9801-4.

[34] EUROCONTROL. *2024 European Aviation Overview*. Jan. 2025. URL: https://www.eurocontrol.int/publication/eurocontrol-european-aviation-overview-archive-2024 (visited on 07/2025).

[35] EUROCONTROL. *AIRM website*. URL: https://airm.aero (visited on 07/2025).

[36] EUROCONTROL. *AIXM website*. URL: https://aixm.aero (visited on 07/2025).

[37] EUROCONTROL. *All-causes delays to Air Transport in Europe - Quarter 3*. 2022. URL: https://www.eurocontrol.int/publication/all-causes-delays-air-transport-europe-quarter-3-2022 (visited on 07/2025).

[38] EUROCONTROL. *ATFCM Operations Manual v.25*. 2021.

[39] EUROCONTROL. *Network 4D Trajectory CONOPS*. July 2023.

[40] EUROCONTROL. *SESAR Factsheet: Business Trajectory / '4D' Trajectory*. 2010. (Visited on 07/2025).

[41] FIXM. *FIXM website*. URL: `https://fixm.aero` (visited on 07/2025).

[42] FlightRadar24. *FlightRadar24 Website*. 2025. URL: `www.flightradar24.com` (visited on 07/2025).

[43] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (Aug. 1997), pp. 119–139. ISSN: 00220000. DOI: `10.1006/jcss.1997.1504`.

[44] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29.5 (2001), pp. 1189–1232. ISSN: 0090-5364. DOI: `10.1214/aos/1013203451`.

[45] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely Randomized Trees". In: *Machine Learning* 63.1 (Apr. 2006), pp. 3–42. ISSN: 1573-0565. DOI: `10.1007/s10994-006-6226-1`.

[46] Yan Glina, Richard Jordan, and Mariya Ishutkina. "A Tree-Based Ensemble Method for the Prediction and Uncertainty Quantification of Aircraft Landing Times". In: *Proc. 10th Conference on Artificial and Computational Intelligence*. Jan. 2012, p. 6.

[47] Karthik Gopalakrishnan and Hamsa Balakrishnan. "A Comparative Analysis of Models for Predicting Delays in Air Traffic Networks". In: *MIT Web Domain* (June 2017).

[48] Guan Gui, Fan Liu, Jinlong Sun, Jie Yang, Ziqi Zhou, and Dongxu Zhao. "Flight Delay Prediction Based on Aviation Big Data and Machine Learning". In: *IEEE Transactions on Vehicular Technology* 69.1 (Jan. 2020), pp. 140–150. ISSN: 1939-9359. DOI: `10.1109/TVT.2019.2954094`.

[49] G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and Its Variations*. Springer Science & Business Media, May 2006. ISBN: 978-0-306-48213-7.

[50] Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. "Liquid Time-constant Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.9 (May 2021), pp. 7657–7666. ISSN: 2374-3468. DOI: `10.1609/aaai.v35i9.16936`.

[51] Sophia Heimann, Hung P. Hoang, and Stefan Hougardy. "The k-Opt Algorithm for the Traveling Salesman Problem Has Exponential Running Time for k>=5". In: *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*. Vol. 297. 2024, 84:1–84:18. ISBN: 978-3-95977-322-5. DOI: `10.4230/LIPIcs.ICALP.2024.84`.

[52] Sepp Hochreiter. "The Vanishing Gradient Problem during Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[53] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: `10.1162/neco.1997.9.8.1735`.

[54] Steffen Huber, Hajo Wiemer, Dorothea Schneider, and Steffen Ihlenfeldt. "DMME: Data Mining Methodology for Engineering Applications - A Holistic Extension to the CRISP-DM Model". In: *Proc. 12th Conference on Intelligent Computation in Manufacturing Engineering (CIRP)*. Vol. 79. Elsevier B.V., Jan. 2019, pp. 403–408. DOI: `10.1016/j.procir.2019.02.106`.

[55] ICAO. *ADREP Taxonomy*. 2017. URL: `https://www.icao.int/safety/airnavigation/aig/pages/adrep-taxonomies.aspx` (visited on 07/2025).

[56] ICAO. *Global TBO Concept v0.11*. 2018. URL: `https://www.icao.int/airnavigation/tbo/Pages/Why-Global-TBO-Concept.aspx` (visited on 07/2025).

[57] ICAO. *ICAO Annex 11. Air Traffic Services. 15th Edition*. 2018.

[58] ICAO. *ICAO Annex 14, Volume I. Aerodrome Design and Operations. 8th Edition*. 2018.

[59] ICAO. *ICAO Annex 15. Aeronautical Information Services. 14th Edition*. 2013.

[60] ICAO. *ICAO Annex 2. Rules of the Air. 10th Edition*. 2005.

[61] ICAO. *ICAO Annex 3. Meteorological Service for International Air Navigation. 17th Edition*. 2010.

[62] ICAO. *ICAO Document 4444. Procedures for Air Navigation Services: Air Traffic Management. 16th Edition*. 2016.

[63] ICAO. *ICAO Document 8896. Meteorological Practice*. 2021.

[64] ICAO. *ICAO Document 9854. Global Air Traffic Management Operational Concept. 1st Edition*. 2005.

[65] ICAO. *IWXXM website*. URL: `https://community.wmo.int/en/activity-areas/wis/iwxxm` (visited on 07/2025).

[66] ICAO. *The World of Air Transport in 2023*. 2024. URL: `https://www.icao.int/sustainability/WorldofAirTransport/Pages/the-world-of-air-transport-in-2023%5C_es.aspx` (visited on 07/2025).

[67] David Ison, Linda Weiland, Ian McAndrew, and Kat Moran. "Identification of Air Traffic Management Principles Influential in the Development of an Airport Arrival Delay Prediction Model". In: *Journal of Aviation/Aerospace Education & Research* 24.2 (2015). ISSN: 2329-258X. DOI: `10.15394/jaaer.2015.1618`.

[68] Peiyan Jia, Huiping Chen, Lei Zhang, and Daojun Han. "Attention-LSTM Based Prediction Model for Aircraft 4-D Trajectory". In: *Scientific Reports* 12.1 (Sept. 2022), p. 15533. ISSN: 2045-2322. DOI: `10.1038/s41598-022-19794-1`.

[69] Myung Suk Kim. "Analysis of Short-Term Forecasting for Flight Arrival Time". In: *Journal of Air Transport Management* 52 (Apr. 2016), pp. 35–41. ISSN: 09696997. DOI: `10.1016/j.jairtraman.2015.12.002`.

[70] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 2017. DOI: `10.48550/arXiv.1412.6980`.

[71] Billy V. Koen. "Toward a Definition of the Engineering Method". In: *European Journal of Engineering Education* 13.3 (Jan. 1988), pp. 307–315. ISSN: 0304-3797. DOI: `10.1080/03043798808939429`.

[72] Banage T.G.S. Kumara, Incheon Paik, Jia Zhang, T.H.A.S. Siriweera, and Koswatte R.C. Koswatte. “Ontology-Based Workflow Generation for Intelligent Big Data Analytics”. In: *2015 IEEE International Conference on Web Services*. June 2015, pp. 495–502. DOI: 10.1109/ICWS.2015.72.

[73] Thanh-Ha Le, Phu Tran, Duc-Thinh Pham, Michael Schultz, and Sameer Alam. “Short-Term Trajectory Prediction Using Generative Machine Learning Methods”. In: *Proc. 9th International Conference on Research in Air Transportation (ICRAT)*. Vol. 15. Tampa, Florida, US, July 2020.

[74] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. *Temporal Fusion Transformers for interpretable multi-horizon time series forecasting*. 2020. arXiv: 1912.09363.

[75] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. “Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting”. In: *International Journal of Forecasting* 37.4 (Oct. 2021), pp. 1748–1764. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.03.012.

[76] Yangdong Liu, Yizhe Wang, Xiaoguang Yang, and Linan Zhang. “Short-Term Travel Time Prediction by Deep Learning: A Comparison of Different LSTM-DNN Models”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2017, pp. 1–8. DOI: 10.1109/ITSC.2017.8317886.

[77] Alexandra Lukáčová, František Babič, and Ján Paralič. “Building the Prediction Model from the Aviation Incident Data”. In: *2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. Jan. 2014, pp. 365–369. DOI: 10.1109/SAMI.2014.6822441.

[78] Lan Ma and Shan Tian. “A Hybrid CNN-LSTM Model for Aircraft 4D Trajectory Prediction”. In: *IEEE Access* 8 (2020), pp. 134668–134680. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3010963.

[79] Yan Ma, Wenbo Du, Jun Chen, Yu Zhang, Yisheng Lv, and Xianbin Cao. “A spatiotemporal neural network model for estimated-time-of-arrival prediction of flights in a terminal maneuvering area”. In: *IEEE Intelligent Transportation Systems Magazine* 15.1 (2022), pp. 285–299. DOI: https://doi.org/10.1109/MITS.2021.3132766.

[80] Miguel A. Martínez-Prieto, Anibal Bregon, Iván García-Miranda, Pedro C. Álvarez-Esteban, Fernando Díaz, and David Scarlatti. “Integrating Flight-Related Information into a (Big) Data Lake”. In: *Proc. IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. Sept. 2017, pp. 1–10. DOI: 10.1109/DASC.2017.8102023.

[81] Juan Miguel Moine, Silvia Gordillo, and Ana Silvia Haedo. “Análisis Comparativo de Metodologías Para La Gestión de Proyectos de Minería de Datos”. In: *Procs. XVII Congreso Argentino de Ciencias de La Computación*. Argentina, 2011, pp. 931–938. ISBN: 978-950-34-0756-1.

[82] Leonardo Moreira, Christofer Dantas, Leonardo Oliveira, Jorge Soares, and Eduardo Ogasawara. “On Evaluating Data Preprocessing Methods for Machine Learning Models for Flight Delays”. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro: IEEE, July 2018, pp. 1–8. ISBN: 978-1-5090-6014-6. DOI: 10.1109/IJCNN.2018.8489294.

[83] Andrés Muñoz, David Scarlatti, and Pablo Costas. “Real-Time Estimated Time of Arrival Prediction System Using Historical Surveillance Data”. In: *Proc. 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).* Aug. 2019, pp. 174–177. DOI: 10.1109/SEAA.2019.00035.

[84] Andrés Muñoz, David Scarlatti, and Pablo Costas. “Real-Time Prediction of Flight Arrival Times Using Surveillance Information”. In: *Proc. 12th European Conference on Software Architecture: Companion Proceedings.* Sept. 2018, pp. 1–4. ISBN: 978-1-4503-6483-6. DOI: 10.1145/3241403.3241434.

[85] James Murphy, Ronald Reisman, Jesse Clayton, and Richard Wright. “Physics-Based and Parametric Trajectory Prediction Performance Comparison for Traffic Flow Management”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit.* Aug. 2003, p. 11. ISBN: 978-1-62410-090-1. DOI: 10.2514/6.2003-5629.

[86] Juan Jose Rebollo and Hamsa Balakrishnan. “Characterization and Prediction of Air Traffic Delays”. In: *Transportation Research Part C: Emerging Technologies* 44 (2014), pp. 231–241. ISSN: 0968090X. DOI: 10.1016/j.trc.2014.04.007.

[87] Juan José Rebollo de la Bandera and Juan José. “Characterization and Prediction of Air Traffic Delays”. Thesis. Massachusetts Institute of Technology, 2012.

[88] Álvaro Rodríguez–Sanz, Cecilia Claramunt Puchol, Fernando Gómez Comendador, Javier Pérez-Castán, Rosa Arnaldo Valdés, Francisco Serrano Martínez, and Mar Najar Godoy. “Air traffic management based on 4D-trajectories: requirements and practical implementation”. In: *MATEC web of conferences.* Vol. 304. EDP Sciences. 2019, p. 05001.

[89] Kaushik Roy, Benjamin Levy, and Claire Tomlin. “Target Tracking and Estimated Time of Arrival (ETA) Prediction for Arrival Aircraft”. In: *Proc. AIAA Guidance, Navigation, and Control Conference and Exhibit.* Aug. 2006. ISBN: 978-1-62410-046-8. DOI: 10.2514/6.2006-6324.

[90] RTCA. *Minimum Aviation System Performance Standards for Automatic Dependent Surveillance Broadcast (ADS-B).* Report DO-242A. 2006.

[91] Sergio Ruiz, Javier Lopez Leones, and Andrea Ranieri. “A novel performance framework and methodology to analyze the impact of 4D trajectory based operations in the future air traffic management system”. In: *Journal of Advanced Transportation* 2018 (2018), pp. 1–17.

[92] Deepudev Sahadevan, Harikrishnan P. M, Palanisamy Ponnusamy, Varun P. Gopi, and Manjunath K. Nelli. “Ground-Based 4d Trajectory Prediction Using Bi-Directional LSTM Networks”. In: *Applied Intelligence* 52.14 (Nov. 2022), pp. 16417–16434. ISSN: 1573-7497. DOI: 10.1007/s10489-022-03309-6.

[93] Matthias Schäfer, Martin Strohmeicr, Matthew Smith, Markus Fuchs, Vincent Lenders, and Ivan Martinovic. “OpenSky Report 2018: Assessing the Integrity of Crowdsourced Mode S and ADS-B Data”. In: *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC).* Sept. 2018, pp. 1–9. DOI: 10.1109/DASC.2018.8569833.

[94] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. “Bringing up OpenSky: A Large-Scale ADS-B Sensor Network for Research”. In: *Proc. 13th International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE Computer Society, 2014, pp. 83–94. ISBN: 978-1-4799-3146-0. DOI: 10.1109/IPSN.2014.6846743.

[95] Ken Schwaber and Jeff Sutherland. *The Scrum Guide*. 2011.

[96] Hesam Shafienya and Amelia C. Regan. “4D Flight Trajectory Prediction Using a Hybrid Deep Learning Prediction Method Based on ADS-B Technology: A Case Study of Hartsfield–Jackson Atlanta International Airport (ATL)”. In: *Transportation Research Part C: Emerging Technologies* 144 (Nov. 2022), p. 103878. ISSN: 0968-090X. DOI: 10.1016/j.trc.2022.103878.

[97] Umair Shafique and Haseeb Qaiser. “A Comparative Study of Data Mining Process Models (KDD , CRISP-DM and SEMMA)”. In: *International Journal of Innovation and Scientific Research* 12.1 (2014), pp. 217–222. ISSN: 2028-9324.

[98] Zhiyuan Shi, Min Xu, and Quan Pan. “4-D Flight Trajectory Prediction With Constrained LSTM Network”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.11 (Nov. 2021), pp. 7242–7255. ISSN: 1558-0016. DOI: 10.1109/TITS.2020.3004807.

[99] Zhiyuan Shi, Min Xu, Quan Pan, Bing Yan, and Haimin Zhang. “LSTM-based Flight Trajectory Prediction”. In: *Proc. 2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. Rio de Janeiro, July 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489734.

[100] Fatimah Sidi, Payam Hassany Shariat Panahy, Lilly Suriani Affendey, Marzanah A. Jabar, Hamidah Ibrahim, and Aida Mustapha. “Data Quality: A Survey of Data Quality Dimensions”. In: *Proc. 2012 International Conference on Information Retrieval & Knowledge Management*. Mar. 2012, pp. 300–304. DOI: 10.1109/InfRKM.2012.6204995.

[101] Jorge Silvestre, Miguel de Santiago, Anibal Bregon, Miguel A. Martínez-Prieto, and Pedro C. Álvarez-Esteban. “On the Use of Deep Neural Networks to Improve Flights Estimated Time of Arrival Predictions”. In: *Engineering Proceedings* 13.1 (2021), p. 3. ISSN: 2673-4591. DOI: 10.3390/engproc2021013003.

[102] Jorge Silvestre, Miguel A. Martínez-Prieto, Anibal Bregon, and Pedro C. Álvarez-Esteban. “A Deep Learning-Based Approach for Predicting in-Flight Estimated Time of Arrival”. In: *The Journal of Supercomputing* 80.12 (Aug. 2024), pp. 17212–17246. ISSN: 1573-0484. DOI: 10.1007/s11227-024-06060-6.

[103] Jorge Silvestre, Paula Mielgo, Anibal Bregon, Miguel A. Martínez-Prieto, and Pedro C. Álvarez-Esteban. “Multi-Route Aircraft Trajectory Prediction Using Temporal Fusion Transformers”. In: *IEEE Access* (2024), pp. 1–1. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3415419.

[104] Jorge Silvestre, Paula Mielgo, Anibal Bregon, Miguel A. Martínez-Prieto, and Pedro C. Álvarez-Esteban. “Towards Aircraft Trajectory Prediction Using LSTM Networks”. In: *Procs. 39th ACM/SIGAPP Symposium on Applied Computing*. SAC ’24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1059–1060. ISBN: 9798400702433. DOI: 10.1145/3605098.3636195.

[105] Christian Strottmann Kern, Ivo Paixao de Medeiros, and Takashi Yoneyama. "Data-Driven Aircraft Estimated Time of Arrival Prediction". In: *Proc. 9th Annual IEEE Systems Conference (SysCon).* Apr. 2015, pp. 727–733. ISBN: 978-1-4799-5927-3. DOI: `10.1109/SYSCON.2015.7116837`.

[106] Junzi Sun. *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals.* 2nd ed. TU Delft OPEN Publishing, 2021. ISBN: 978-94-6366-402-8. DOI: `10.34641/mg.11`.

[107] Junzi Sun, Joost Ellerbroek, and Jacco Hoekstra. "Flight Extraction and Phase Identification for Large Automatic Dependent Surveillance–Broadcast Datasets". In: *Journal of Aerospace Information Systems* 14.10 (Oct. 2017), pp. 566–572. ISSN: 1940-3151, 2327-3097. DOI: `10.2514/1.I010520`.

[108] Junzi Sun, Joost Ellerbroek, and Jacco Hoekstra. "Large-Scale Flight Phase Identification from ADS-B Data Using Machine Learning Methods: 7th International Conference on Research in Air Transportation". In: *7th International Conference on Research in Air Transportation* (2016). Ed. by D. Lovell and H. Fricke.

[109] Junzi Sun, Joost Ellerbroek, and Jacco Hoekstra. "Reconstructing Aircraft Turn Manoeuvres for Trajectory Analyses Using ADS-B Data". In: *9th SESAR Innovation Days 2nd – 5th December 2019.* SESAR Innovation Days. 2019.

[110] Asma Tabassum, Nicholas Allen, and William Semke. "ADS-B Message Contents Evaluation and Breakdown of Anomalies". In: *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC).* St. Petersburg, FL: IEEE, Sept. 2017, pp. 1–8. ISBN: 978-1-5386-0365-9. DOI: `10.1109/DASC.2017.8102001`.

[111] Asma Tabassum and William Semke. "UAT ADS-B Data Anomalies and the Effect of Flight Parameters on Dropout Occurrences". In: *Data* 3.2 (June 2018), p. 19. ISSN: 2306-5729. DOI: `10.3390/data3020019`.

[112] Nur Asheila Taib and Busyairah Syd Ali. "An Analysis of Geometric Altitude Data in ADS-B Messages". In: *2016 International Technical Meeting of The Institute of Navigation.* Monterey, California, Feb. 2016, pp. 697–704. DOI: `10.33012/2016.13392`.

[113] Gabor Takacs. "Predicting Flight Arrival Times with a Multistage Model". In: *Proc. IEEE International Conference on Big Data (Big Data).* Oct. 2014, pp. 78–84. ISBN: 978-1-4799-5666-1. DOI: `10.1109/BigData.2014.7004435`.

[114] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: `1706.03762`. URL: `http://arxiv.org/abs/1706.03762`.

[115] T Verbraak, Joost Ellerbroek, Junzi Sun, and Jacco Hoekstra. "Large-Scale ADS-B Data and Signal Quality Analysis". In: *Procs. 12th USA/Europe Air Traffic Management Research and Development Seminar.* Seattle, June 2017.

[116] Matthew Veres and Medhat Moussa. "Deep Learning for Intelligent Transportation Systems: A Survey of Emerging Trends". In: *IEEE Transactions on Intelligent Transportation Systems* 21.8 (Aug. 2020), pp. 3152–3168. ISSN: 1558-0016. DOI: `10.1109/TITS.2019.2929020`.

[117] Zhengyi Wang, Man Liang, and Daniel Delahaye. "Automated Data-Driven Prediction on Aircraft Estimated Time of Arrival". In: *Journal of Air Transport Management* 88 (Sept. 2020), p. 101840. ISSN: 09696997. DOI: `10.1016/j.jairtraman.2020.101840`.

[118] Philip B. Weerakody, Kok Wai Wong, Guanjin Wang, and Wendell Ela. "A Review of Irregular Time Series Data Handling with Gated Recurrent Neural Networks". In: *Neurocomputing* 441 (June 2021), pp. 161–178. ISSN: 0925-2312. DOI: `10.1016/j.neucom.2021.02.046`.

[119] Rüdiger Wirth. "CRISP-DM : Towards a Standard Process Model for Data Mining". In: *Proc. 4th International Conference on the Practical Application of Knowledge Discovery and Data Mining*. Vol. 1. 2000, pp. 29–39.

[120] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. "Spatio-Temporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks". In: *Sensors* 17.7 (June 2017), p. 1501. DOI: `10.3390/S17071501`.

[121] Weili Zeng, Xiao Chu, Zhengfeng Xu, Yan Liu, and Zhibin Quan. "Aircraft 4D Trajectory Prediction in Civil Aviation: A Review". In: *Aerospace* 9.2 (Feb. 2022), p. 91. ISSN: 2226-4310. DOI: `10.3390/aerospace9020091`.

[122] Weili Zeng, Zhibin Quan, Ziyu Zhao, Chao Xie, and Xiaobo Lu. "A Deep Learning Approach for Aircraft Trajectory Prediction in Terminal Airspace". In: *IEEE Access* 8 (2020), pp. 151250–151266. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2020.3016289`.

[123] Kai Zhang, Yushan Jiang, Dahai Liu, and Houbing Song. "Spatio-Temporal Data Mining for Aviation Delay Prediction". In: *2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC)*. Nov. 2020, pp. 1–7. DOI: `10.1109/IPCCC50635.2020.9391561`.

[124] Guangyu Zhao, Muqun Yang, Yizhao Gao, Yizhe Zhan, H. Joe Lee, and Larry Di Girolamo. "PYTAF: A Python Tool for Spatially Resampling Earth Observation Data". In: *Earth Science Informatics* 15.3 (Sept. 2022), pp. 1443–1448. ISSN: 1865-0481. DOI: `10.1007/s12145-020-00461-w`.

[125] Kai Zheng and Han Su. "Go Beyond Raw Trajectory Data: Quality and Semantics". In: *IEEE Data Eng. Bull.* (2015).

[126] Yu Zheng. "Trajectory Data Mining: An Overview". In: *ACM Transactions on Intelligent Systems and Technology* 6.3 (May 2015), pp. 1–41. ISSN: 2157-6904, 2157-6912. DOI: `10.1145/2743025`.

[127] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms.* CRC press, 2012.

Acronyms

ACC Area Control Centre. 19

ADS-B Automatic Dependant Surveillance – Broadcast. 4, 64

AICM Aeronautical Information Conceptual Model. 55

AIRM ATM Information Reference Model. 56

AIRMET Airmen's Meteorological Information. 19

AIXM Aeronautical Information Exchange Model. 55

ASM Air Space Management. 17, 21

ASOS Automated Surface Observing System. 73

ATC Air Traffic Control. 17–21

ATFM Air Traffic Flow Management. 17, 20

ATM Air Traffic Management. 15, 165

ATS Air Traffic Services. 17, 20, 22

ATZ Aerodrome Traffic Zone. 23, 24

CTA Controlled Traffic Area. 23

CTR Controlled Traffic Region. 23

EOBT Estimated Off Block Time. 35, 71

ETA Estimated Time of Arrival. 5, 143

ETFMS Enhanced Tactical Flow Management System. 4, 35

FIC Flight Information Centre. 18, 19

FIR Flight Information Region. 18, 21, 22

FIS Flight Information Service. 17–19

FIXM Flight Information Exchange Model. 55

GNSS Global Navigation Satellite System. 66

GPS Global Positioning System. 66

ICAO International Civil Aviation Organization. 15

IFPS Initial Flight Plan Processing System. 4, 20, 35, 70

IFR Instrumental Flight Rules. 21

IWXXM ICAO Meteorological Information Exchange Model. 55

METAR Meteorological Aerodrome Report. 72

NMOC Network Manager Operations Centre. 20, 35, 70

RTA Remaining Time to Arrival. 148

SID Standard Instrument Departure. 25

SIGMET Significant Meteorological Information. 19

STAR Standard Terminal Arrival Routes. 25

SWIM System Wide Information Management. 35, 36

TAF Terminal Aerodrome Forecast. 73

TMA Terminal Manoeuvring Area. 18, 23, 143

TWR Tower of the aerodrome. 19, 23

UAR Upper Air Routes. 22

UIR Upper Information Region. 22

VFR Visual Flight Rules. 21, 23

WXXM Weather Information Exchange Model. 55