ABSTRACT

Title of dissertation proposal: AIRSPACE PLANNING FOR OPTIMAL CAPACITY, EFFICIENCY, AND SAFETY USING ANALYTICS

Samet Ayhan, Doctor of Philosophy, 2019

Dissertation proposal directed by: Professor Hanan Samet
Department of Computer Science

Air Navigation Service Providers (ANSP) worldwide have been making a considerable effort for the development of a better method for planning optimal airspace capacity, efficiency, and safety. These goals require separation and sequencing of aircraft before they depart. Prior approaches have tactically achieved these goals to some extent. However, dealing with increasingly congested airspace and new environmental factors with high levels of uncertainty still remains the challenge when deterministic approach is used. Hence due to the nature of uncertainties, we take a stochastic approach and propose a suite of analytics models for (1) Flight Time Prediction (2) Aircraft Trajectory Clustering (3) Aircraft Trajectory Prediction, and (4) Aircraft Conflict Detection and Resolution long before aircraft depart. The suite of data-driven models runs on a scalable data management system that continuously processes streaming massive flight data to achieve the strategic airspace planning for optimal capacity, efficiency, and safety.

(1) Unlike other systems that collect and use features only for the arrival airport to build a data-driven model for predicting flight times, we use a richer set of features along the potential route, such as weather parameters and air traffic data in addition to those that are particular to the arrival airport. Our feature engineering process generates an extensive set of multidimensional time series data which goes through Time Series Clustering with Dynamic Time Warping (DTW)

to generate a single set of representative features at each time instance. The features are fed into various regression and deep learning models and the best performing models with most accurate ETA predictions are selected. Evaluations on extensive set of real trajectory, weather, and airport data in Europe verify our prediction system generates more accurate ETAs with far less standard deviation then those of European ANSP, EUROCONTROLs. This translates to more predictable flight arrival times, enabling airlines to make more cost-effective ground resource allocation and ANSPs to make more efficient flight scheduling.

(2) The novel divide-cluster-merge; DICLERGE system clusters aircraft trajectories by dividing them into three major flight phases: climb, enroute, and descent. Trajectory segments in each phase are clustered in isolation, then merged together. Our unique approach also discovers a representative trajectory, the model for the entire trajectory set.

(3) Our approach considers airspace as a 3D grid network, where each grid point is a location of a weather observation. We hypothetically build cubes around these grid points, so the entire airspace can be considered as a set of cubes. Each cube is defined by its centroid, the original grid point, and associated weather parameters that remain homogeneous within the cube during a period of time. Then, we align raw trajectories to a set of cube centroids which are basically fixed 3D positions independent of trajectory data. This creates a new form of trajectories which are 4D joint cubes, where each cube is a segment that is associated with not only spatio-temporal attributes but also with weather parameters. Next, we exploit machine learning techniques to train inference models from historical data and apply a stochastic model, a Hidden Markov Model (HMM), to predict trajectories taking environmental uncertainties into account. During the process, we apply time series clustering to generate input observations from an excessive set of weather parameters to feed into the Viterbi algorithm. The experiments use a real trajectory dataset with pertaining weather observations and demonstrate the effectiveness of our approach to the trajectory predic-

tion process for Air Traffic Management.

(4) We propose a novel data-driven system to address a long-range aircraft conflict detection and resolution (CDR) problem. Given a set of predicted trajectories, the system declares a conflict when a protected zone of an aircraft on its trajectory is infringed upon by another aircraft. The system resolves the conflict by prescribing an alternative solution that is optimized by perturbing at least one of the trajectories involved in the conflict. To achieve this, the system learns from descriptive patterns of historical trajectories and pertinent weather observations and builds a Hidden Markov Model (HMM). Using a variant of the Viterbi algorithm, the system avoids the airspace volume in which the conflict is detected and generates a new optimal trajectory that is conflict-free. The key concept upon which the system is built is the assumption that the airspace is nothing more than a horizontally and vertically concatenated set of spatio-temporal data cubes where each cube is considered as an atomic unit. We evaluate the system using real trajectory datasets with pertinent weather observations from two continents and demonstrate its effectiveness for strategic CDR.

Overall, in this thesis, we develop a suite of analytics models and algorithms to accurately identify current patterns in the massive flight data and use these patterns to predict future behaviors in the airspace. Upon prediction of a non-ideal outcome, we prescribe a solution to plan airspace for optimal capacity, efficiency, and safety.

AIRSPACE PLANNING FOR OPTIMAL CAPACITY, EFFICIENCY, AND SAFETY USING ANALYTICS

by

Samet Ayhan

Dissertation Proposal submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Advisory Committee:
Professor Hanan Samet, Chair/Advisor
Professor Michael O. Ball
Professor Dinesh Manocha
Professor David Mount
Professor Udaya Shankar

Dedication

Dedicated to my family — I love you all.

Acknowledgements

I owe my gratitude to all the people who have made this thesis possible.

...

Work in progress...

...

Finally, I want to thank my family for their ever-lasting love, support, encouragement, and unwavering faith in me. They are the greatest source of my energy and have made me what I am today. I dedicate this thesis to them.

Thank you everyone!

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation

Civil aviation underwent a considerable growth over the past years and it is agreed that it will have a continuously increasing trend of growth for the next years. It is estimated that by 2040 the U.S. alone can expect an increase of more than 68% in commercial air traffic [1]. Inclusion of massive number of unmanned aircraft in the U.S. airspace system, commonly referred to as National Airspace System (NAS) within the next decade is also unsurprising. One of the key activities of the Federal Aviation Administration (FAA), the Air Navigation Service Provider (ANSP) in the U.S. is to measure and monitor resources in the NAS. The NAS including airports and associated terminal areas is a complex system and require a number of metrics to properly characterize performance. Although the prior research offered different metrics to quantify and measure performance of the NAS, they all agreed that flow efficiency, runway utilization, and rate of cancelled or diverted flights are major factors of measuring the performance of the NAS [2, 3]. Hence, delays, diversions, and cancellations are solid indicators of efficiency and capacity issues in the NAS [4]. The FAA's Office of Performance Analysis estimates show in 2017, the

cost of delayed flights increased by 11.3%, from $23.9 to $26.6 billion. Most of this increase was due to inefficiency in the NAS, resulting in the sum of costs to airlines, passengers, lost demand, and direct costs. Between 2012 and 2017, the cost of flight delays increased from $19.2 to $26.6 billion, an increase of $7.2 billion. Increases in the number of diversions can also indicate capacity issues at the airports due to weather, construction, or volume. In 2017, flight departure cancellations at core 30 airports in the U.S. increased 16.9%. Cancellations may be due to weather, system delays, equipment issues, or other reasons. All these factors are likely to cause both safety and performance degradation [5]. The increasing demand for commercial aviation coupled with factors degrading the performance of the NAS make the overarching goals of ANSPs, which are increasing capacity and improving efficiency of airspace while maintaining safety, hard to achieve.

To address these challenges, new technologies and procedures for next generation ATM are being developed in the context of the Next Generation Air Transportation System (NextGen) [6] in the USA and the Single European Sky ATM Research [7] in Europe. The NextGen and SESAR require the expansion of Air Traffic Control (ATC) services together with the implementation of advanced Air Traffic Management (ATM) concepts such as dynamic resectorization, free flight, enhanced ground delay program, and airspace redesign, all with the goal of improving the efficiency of the NAS [8–11]. Moreover, they require a paradigm shift from a highly structured and fragmented system that is heavily reliant on tactical decision making and with few strategic planning functions based on uncertain information, to an integrated one based on collaborative strategic management of trajectories and information sharing [12].

In the future ATM systems to be built under NextGen and SESAR, the trajectory becomes the fundamental element of new sets of operating procedures collectively referred to as Trajectory-Based Operations (TBO). The underlying idea behind TBO is the concept of business trajectory which is the trajectory that will best meet airline business interests. The TBO concept of oper-

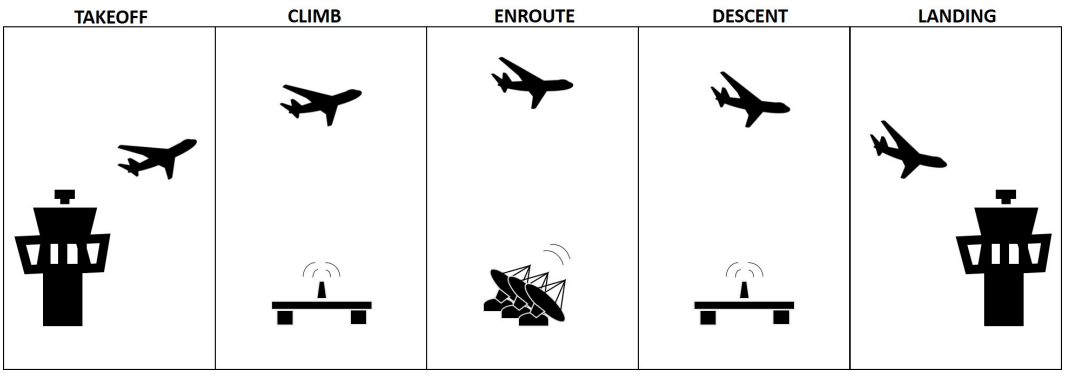


Fig. 1.1: Phases during a flight.

ations and the notion of a business trajectory will result in increased overall predictability of air traffic, reduced costs and emissions due to lowered fuel consumption and/or time, and increased capacities. Thus, the future ATM system should offer flexibility to accommodate business trajectories, while bearing the primary goal of safety and efficiency in mind. Overall, the NextGen and SESAR are expected to enable higher degrees of automation and predictability yielding a reduction in the air traffic controllers' workload.

1.2 Background

A major component of the NAS is the ATC system which is made up of a vast network of radars and air traffic control facilities. It is through the ATC system that ANSPs provide services such as takeoffs and landings, and managing the flow of air traffic between airports. Other components of the NAS include airports or landing areas, rules, regulations, procedures, and technical information such as aeronautical charts. A flight goes through five major phases during its lifetime; takeoff, climb, enroute, descent, and landing. Three of these are wheels-up phases, climb, enroute, and descent which are considered to contribute more significantly to airspace efficiency. Figure 1.1 shows phases of an entire flight.

ANSPs usually use three types of facilities to control air traffic. Airport towers direct aircraft on the ground, before landing, and after takeoff when they are about 5 nautical miles from the airport and up to about 3,000 feet above the airport. Terminal radar approach control (TRACON) facilities sequence and separate aircraft as they approach and leave busy airports, beginning about 5 nautical miles and ending about 50 nautical miles from the airport and generally up to 10,000 feet above the ground. Air route traffic control centers (ARTCC), called enroute centers, control aircraft in transit and during approaches to some airports. Most of the enroute centers' controlled airspace extends above 18,000 feet for commercial aircraft. Enroute centers also handle lower altitudes when dealing directly with a tower, or when agreed upon with a terminal facility.

The term "aircraft movements" is used to describe services, such as arrivals and departures, that the FAA provides to aircraft in the tower, terminal, and enroute airspace. According to FAA, enroute services are considerably more expensive to provide than other air traffic services. The NAS is composed of 517 airport towers, 155 Terminal Radar Control (TRACON) facilities, and 25 Control Centers. TRACONs handle descending flights received from a Center or ascending flights received from an ATC tower. Centers receive flights from or hand off flights to other Centers throughout the flights enroute phase of operation. They also receive flights or hand off flights to TRACONs when flights enter or exit the enroute phase of operation.

While the goals of ATC are the safe and efficient management of aircraft, the overarching goals of Air Navigation Service Providers (ANSP) are to provide optimal airspace capacity and efficiency. These goals require the separation and sequencing of airborne aircraft, which are achieved through air traffic clearances. Clearances are issued by ATC for aircraft to proceed under specified traffic conditions within a controlled airspace for the purpose of preventing collision. ATC also regularly conformance monitors aircraft to determine if they adhere to their assigned clearances so that safety, security, and efficiency can be maintained. Although these goals are

achieved through clearances today, NextGen and SESAR require a shift from clearance-based to trajectory-based ATC operations.

Trajectory Based Operations (TBO) is supposed to provide the capabilities, decision-support tools, and automation to manage aircraft movement by trajectory. It will enable aircraft to fly negotiated flight paths necessary for full Performance Based Navigation (PBN), taking both operator preferences and optimal airspace system performance into consideration. TBO is a cornerstone of NextGen and SESAR; it is a major operational transformation for aviation, basing safe separation on higher levels of automation that assesses the current aircraft positions, with respect to their future positions in time. TBO is a 2025 air traffic management system concept that manages aircraft through their Four-Dimensional Trajectory (4DT), gate to gate, both strategically and tactically to control surface and airborne operations. A 4DT includes a series of points from departure to arrival representing the aircraft's path in four dimensions: lateral (latitude and longitude), vertical (altitude), and time. 4DTs will be used for planning, sequencing, spacing, and separation based on the aircrafts' current and future positions. Hence, separation duties will be performed by a combination of airborne and Ground Based Automation (GBA). These functions integrate the traditional functions of navigation which is to define a path and create a path guidance, with flight control, which is to steer the aircraft to that path. With these functions, NextGen and SESAR will be enabled to perform tactical and strategic tasks such as conformance monitoring, trajectory negotiation, and trajectory planning.

Safety is typically quantified by the number of conflicts, i.e., situations where two aircraft come closer than a certain distance to one another. Accurate conflict detection depends upon accurate trajectory prediction that is a vital process for ATC, as the more accurate and reliable the predicted trajectories the more accurate the conflict detection, thereby the more safe and efficient is the deconfliction of these flights. Deconfliction can be achieved by periodically checking pre-

dicted trajectories against each other to detect any conflicts between them and then taking conflict resolution action, or, by comparing newly predicted trajectories against previously deconflicted trajectories. An aircraft following an agreed trajectory can then be assured a conflict-free and unperturbed flight. In order for ATC to attain this goal, the trajectory prediction tools should be able to provide high levels of accuracy for an adequate horizon of time. Air traffic density, i.e., a count of aircraft in a particular airspace sector and air traffic complexity, i.e., a count of aircraft predicted to be in conflict with another are two major factors defining the metric of air traffic controller workload [13]. Hence, the most optimal airspace density and complexity, the more automation and thereby the less air traffic controller workload.

1.3 Overview and Contributions

The goal of this dissertation is to enable ANSPs, airlines and other stakeholders involved in ATM to strategically improve capacity, efficiency, and safety of airspace by

1. detecting descriptive patterns in the behavior of aircraft through their massive amounts of historical trajectories, and

2. leveraging these learned insights to develop efficient computational models for predicting flight times, 4D trajectories, and their potential conflicts. These insights can also be used toward prescribing optimal resolutions when conflicts occur.

Hence, we study the characteristics of aircraft trajectories and develop several data-driven models in support of strategic airspace planning. Now, we briefly describe each problem and provide a list of our contributions in each problem space.

1.3.1 Aviation Big Data Management (Chapter 2)

In 1992 the Federal Aviation Administration (FAA) started a program to provide flight plan and track information from the NAS to airlines and other organizations in real-time or near real-time depending on the need. The feed, called ASDI, is a product of the Enhanced Traffic Management System (ETMS). The feed originates from the Traffic Flow Management (TFM) Production Center located at the William J. Hughes Technical Center (WJHTC) in Atlantic City, New Jersey. The FAA distributes the ASDI data stream to a tier of vendors who in turn distribute it to their clients [14]. As a subscriber to the ASDI streaming data feed, we have been processing 7.5 million main and 26.5 million supporting messages daily. If one assumes that this is typical for a given day, only a single year of data amounts to over 10 billion messages. The data set is big and stored as compressed XML messages. Clearly, for any analytics to be useful over this dataset a data management solution is required. Hence, we implemented an efficient data management solution, which is the underlying infrastructure for our descriptive, predictive, and prescriptive analytics solutions presented in this thesis.

1.3.2 Flight Time Prediction (Chapter 3)

Accurate prediction of Estimated Time of Arrival (ETA) for a flight is a key component of collaborative decision making (CDM), a process that attempts to keep the costs under control along with improvement in capacity, efficiency and safety of airspace. ETA is formulated as departure time plus flight time and is a crucial input to the aircraft trajectory prediction. Given a set of historical flights with their attributes such as flight number, trajectory, departure and arrival airport, actual departure and arrival date and time, airline name, aircraft type, and associated weather and air traffic parameters, we want to predict the ETA for the current flight before it departs.

Our approach to the ETA prediction is unique as 1) our system predicts ETA for runway

arrival times. 2) Our system predicts ETA before departure, when the flight trajectory is still unknown. In fact, our prediction system doesn't make use of costly flight plans, offering a more cost-effective solution. Hence, our approach addresses the ETA prediction problem strategically over a time horizon of several hours. 3) For our model, we collect and use a richer set of features not only for the arrival airport but also for the airspace along the potential route.

Our experiments on real trajectory, weather, air traffic and airport data verify that our system outperforms European ANSP, EUROCONTROL's ETA prediction by offering not only a higher accuracy but also a far smaller standard deviation, resulting in smaller prediction windows of flight arrival times.

1.3.3 Trajectory Clustering (Chapter 4)

Whether descriptive, predictive or prescriptive, most analytics approaches pertaining to aircraft trajectory data require clustering to group similar trajectories and discovering a representative trajectory for all as a single entity. During the process, considering a trajectory as a whole may mislead, resulting in overfitting and a failure to discover a representative trajectory. Hence, given a set of aircraft trajectories, we want to compute clustering to come up with a single trajectory that represents the massive trajectory dataset.

Our unique approach enables accurately clustering of three or more dimensional aircraft trajectories by employing a divide-cluster-merge framework. The framework divides trajectories based on a well-known aviation domain fact: each flight is made up of three major phases: climb, enroute, and descent. Hence, it clusters each phase in isolation, then merges them together. We also present a representative trajectory algorithm that reveals the trajectory model based on lateral and vertical smoothing.

Our experiments use a real aircraft trajectory dataset and show that our framework effec-

tively performs clustering and discovers the representative trajectory.

1.3.4 Time Series Clustering (Chapter 5)

To predict aircraft trajectories Hidden Markov Model (HMM) requires a set of input observations, which are unknown, although the weather parameters overall are known for the pertinent airspace. Hence, given an extensive set of weather parameters for the overall airspace volume of interest, we want to compute a time series of weather observations representing only the relevant sections of the airspace. To achieve this, we perform time series clustering on the current weather observations and generate one cluster centroid for each time instance along the potential path.

Previous research used standard clustering techniques to generate clusters that are largely independent of the time series from which they originate. Our unique approach generates an optimal sequence of weather observations used for accurate trajectory prediction in the climb phase of the flight. Our algorithm computes a cost value for each weather observation based on its frequency for each time instance, ranks, and stores them in a matrix. Using Dynamic Programming (DP), the algorithm computes the optimal sequence of weather observations, a set with the minimum cumulative cost that satisfies the continuity constraint.

Our experiments use a real trajectory dataset with pertinent weather observations and demonstrate the effectiveness of our algorithm over time series clustering with a k-Nearest Neighbors (k-NN) algorithm that uses Dynamic Time Warping (DTW) Euclidean distance.

1.3.5 Aircraft Trajectory Prediction (Chapter 6)

At the heart of ATM lies the DSTs that rely upon accurate trajectory prediction to determine how the airspace will look like in the future to make better decisions and advisories. Dealing with airspace that is prone to congestion due to environmental factors still remains the challenge

especially when a deterministic approach is used in the trajectory prediction process. Given a set of historical trajectories, aircraft types, aircraft performance models, and weather parameters along with the current flight's aircraft type, aircraft performance model, and weather parameters, we want to predict its trajectory.

Our approach considers airspace as a 3D grid network, where each grid point is a location of a weather observation. We hypothetically build cubes around these grid points, so the entire airspace can be considered as a set of cubes. Each cube is defined by its centroid, the original grid point, and associated weather parameters that remain homogeneous within the cube during a period of time. Then, we align raw trajectories to a set of cube centroids which are basically fixed 3D positions independent of trajectory data. This creates a new form of trajectories which are 4D joint cubes, where each cube is a segment that is associated with not only spatio-temporal attributes but also with weather parameters. Next, we exploit machine learning techniques to train inference models from historical data and apply a stochastic model, a Hidden Markov Model (HMM), to predict trajectories taking environmental uncertainties into account. During the process, we leverage our time series clustering algorithm to generate input observations from an excessive set of weather parameters to feed into the Viterbi algorithm. Our novel stochastic trajectory prediction approach can be used for more efficient and realistic flight planning and to assist airspace flow management, potentially resulting in higher safety, capacity, and efficiency commensurate with fuel savings thereby reducing emissions for a better environment.

1.3.6 Long-Range Conflict Detection and Resolution (Chapter 7)

At the present time, there is no mechanism for ANSPs to probe new flight plans filed by the Airlines Operation Centers (AOCs) against the existing approved flight plans to see if they are likely to cause conflicts or bring sector traffic densities beyond control. In the current Air Traffic

Control (ATC) operations, aircraft conflicts and sector traffic densities are resolved tactically, increasing workload and leading to potential safety risks and loss of capacity and efficiency. Given a set of predicted trajectories along with their pertinent weather observations, we want to detect and resolve conflicts among them in a scalable manner, i.e. perform conflict detection and resolution not only between two aircraft but also among two or more aircraft before they depart.

Our data-driven approach proposes two major capabilities; i) conflict detection, and ii) conflict resolution before the flight depart. In the conflict detection stage, our system declares a conflict when a Protected Zone (PZ) of an aircraft on its predicted trajectory is infringed upon by another aircraft at the same time interval in the future. In the conflict resolution stage, upon a conflict, the framework executes our conflict resolution algorithm that is derived from the Viterbi algorithm and prescribes a solution that resolves the conflict by perturbing at least one of the 4D trajectories.

Our experiments on real trajectory and weather datasets from two continents verify that our framework achieves lateral and vertical accuracies that are within the boundaries of conventionally accepted minimum separation values, set by the ANSPs. This translates to the fact that, ANSPs can now detect and resolve potential conflicts long before the aircraft depart, resulting in safer and greener skies with higher efficiency and capacity, and thereby reducing the air traffic controller workload.

1.4 Overarching Thesis Statement

The following statement summarizes the thesis aptly: Dealing with airspace prone to uncertainties still remains a challenge especially when a deterministic approach is used to predict its future behavior. Hence, we stochastically approach the problem and propose a suite of efficient computational models to detect descriptive patterns in historical air traffic data and use them to

predict the future behavior within the airspace. Overall this suite of capabilities can be delivered in the planning phase before the aircraft depart, resulting in improvement in the four ATM key performance areas; safety, capacity, efficiency, and environmental impact, and thereby improving ATM automation and reducing the air traffic controller workload.

1.5 Thesis Organization

The thesis is split into five major parts. Part 1 of the thesis introduces the problem space, motivates it, and provides the background information. Overview and our contributions are also included in this part. Part 2 of the thesis lays out the aviation big data management system that is the underlying infrastructure for the descriptive, predictive, and prescriptive analytics systems presented in the following chapters. Part 3 is divided into three chapters: Chapter 3 presents the analytics system that predicts flight time which is used as an input to the aircraft trajectory prediction system. Chapter 4 presents a novel clustering algorithm to group similar trajectories and discover a representative trajectory for all as a single entity. Chapter 5 discusses an efficient time series clustering algorithm which improves the efficacy of the trajectory prediction system. Part 4 is divided into two chapters: Chapter 6 discusses probabilistic aircraft trajectory prediction system and Chapter 7 delves into long-range aircraft conflict detection and resolution system. Finally, Part 5 concludes the thesis. Each chapter lists the existing literature that is most relevant.

Part II

Aviation Big Data Management

Chapter 2

Aviation Big Data Management

In this chapter, we describe a novel data management system that enables query processing and predictive analytics over streams of aviation big data. Our system makes predictions based upon descriptive patterns of archived aviation data. We have been receiving live Aircraft Situation Display to Industry (ASDI) data and archiving it for a few years. At the present time, there is not an easy mechanism to perform analytics on the data. The incoming ASDI data is large, compressed, and requires correlation with other flight data before it can be analyzed.

The service exposes this data once it has been uncompressed, correlated, and stored in a data warehouse for further analysis using a variety of descriptive, predictive, and possibly prescriptive analytics tools. The service utilizes a custom tool for correlating the raw ASDI feed, IBM Warehouse with DB2 for data management, WebSphere Message Broker for real-time message brokering, SPSS Modeler for statistical analysis, and Cognos BI for front-end business intelligence (BI) visualization. This chapter describes a scalable service architecture, implementation and the value it adds to the aviation domain.

2.1 Introduction

There is an interest in large scale data analytics in the aviation domain for analysis and prediction of capacity and flow in the US National Airspace System (NAS). This is facilitated by data analytics systems in addition to availability of massive amounts of surveillance data. To better make sense of current and historical aviation data, and make predictions using descriptive behavior, a scalable analytics service is needed.

In 1992 the Federal Aviation Administration (FAA) started a program to provide flight plan and track information from the NAS to airlines and other organizations in real-time or near real-time depending on the need. The feed, called ASDI, is a product of the Enhanced Traffic Management System (ETMS). The feed originates from the Traffic Flow Management (TFM) Production Center located at the William J. Hughes Technical Center (WJHTC) in Atlantic City, New Jersey. The FAA distributes the ASDI data stream to a tier of vendors who in turn distribute it to their clients [15].

We have been archiving ASDI data for over two years. At the present time, there is not an easy mechanism to perform analytics on the data. The data set is large and stored as compressed XML messages. Table 1 shows the number of ASDI messages for August 8, 2009. Table 2 shows the number of supporting records for the main message set. If one assumes that the information in the tables above is typical for a given day, two years of data amounts to approximately 5.5 billion messages and 19 billion supporting records. Clearly, for any analytics to be useful over this dataset a data management solution is required.

In addition to processing previously stored historical ASDI data in flat text files, we also needed to handle processing of continuous streams of ASDI data in near real-time which would then enable users to perform analytics operations once the data has been pushed into the data

Tab. 2.1: ASDI Message Counts for August 8, 2009

Message Type	**Message Count**
Arrival Information	46,000
Boundary Crossing Update	84,000
Departure Information	54,000
Flight Management Information	522,000
Flight Plan Amendment	303,000
Flight Plan Cancellation	121,000
Flight Plan	120,000
Oceanic Report	9,000
Track Information	6,259,000
Total Message Count	**7,518,000**

warehouse.

In response to these needs, we decided to build a data warehouse that contains summary, historical, and detail data to support tactical and strategic decision making. Data would be extracted from operational data sources, transformed, cleaned, reconciled, aggregated, and summarized in preparation for warehouse processing [16].

Our chosen data management system was IBM's InfoSphere Data Warehouse, as it manages large data sets by offering extended data mining and analytics support. The data warehouse is implemented by a backend DB2 database [17]. For near real-time ASDI feed consumption, message routing, transformation, and publish/subscribe capabilities, we used WebSphere Message Broker [18] and WebSphere MQ [19]. SPSS Modeler by IBM was our choice for the predictive data analytics tool that provides data collection, statistics, predictive modeling, and deployment capabilities [20]. Front-end visualization for business intelligence support was provided by IBM's

Tab. 2.2: ASDI Supporting Record Counts for August 8, 2009

Supporting Record	**Record Count**
Aircraft Specification	558,000
Airways	1,732,000
Centers	1,692,000
Computer Identification	5,551,000
Fixes	6,200,000
Oceanic Planned Position	18,000
Oceanic Reported Position	9,000
Qualified Aircraft Identification	532,000
Route of Flight	721,000
Sectors	1,773,000
Waypoints	7,667,000
Total Record Count	**26,413,000**

Cognos BI tool [21].

The rest of this chapter is organized as follows: Section 2.2 presents related work, Section 2.3 explains the process of ASDI correlation, Section 2.4 presents the overall system architecture including the physical and conceptual modeling of the data warehouse as well as consumption of the ASDI feed. Section 2.5 discusses some of the data representation and loading issues we encountered and optimizations we realized. Section 2.6 tackles a basic use-case where we perform analytics predicting distinct traffic volume within a boundary defined by the user. Section 2.7 concludes the chapter.

2.2 Related Work

Big data means data that cannot be handled and processed in a traditional manner. It will be so large as to not fit on a single hard drive, as a result, it will be stored on several different disks, and will be processed on a number of cores [22].

There are a number of articles and books on big data, analytics, data warehousing and OLAP technology and related research issues. While some of these focus on physical and conceptual design, others target maintenance issues and stream processing. However, to the best of our knowledge, no work has been done similar to ours in the aviation domain where operational real-time or near real-time surveillance data is turned into a warehouse enabling critical decision making and predictive analytics in the literature.

Research matters pertaining to data warehousing and OLAP technology can be found in various resources [23–27]. A significant amount of research in the database community has been dedicated to the physical and logical design of data warehouses [28].

With their research, Luján-Mora et al. [29] present a physical model of data warehouses using component and deployment diagrams of UML.

In their article, Arumugam et al. [30] describe a purely-push based, research prototype database system called DataPath, which is a ”data-centric” system. In the DataPath system, all data production, including system I/O, is undertaken asynchronously, whether a receiving operation is ready for the data or not.

Li et al. [31] concluded that in-order processing (IOP) systems in high performance stream processing not only enforce order on input streams, but also require that query operators preserve order. This order preserving requirement constrains the implementation of stream systems and incurs significant performance penalties, particularly for memory consumption. They introduced a new architecture for stream systems, out-of-order processing (OOP), that avoids ordering constraints, freeing stream systems from the burden of order maintenance by using explicit stream progress indicators.

In traditional DBMSs, it was assumed that the DBMS is a passive repository storing a large collection of data elements and that humans initiate queries and transactions on this repository. Abadi et al. [32] called this a human-active, DBMS-passive (HADP) model. They called the opposite a DBMS-active, human-passive (DAHP) model due to fact that, the role of the DBMS in the case of monitoring applications is to alert humans when abnormal activity is detected. According to them, monitoring applications are very difficult to implement in traditional DBMSs. First, the basic computation model is wrong: DBMSs have a HADP model while monitoring applications often require a DAHP model.

Borkar et al. [33] discuss the industry trends and compares monolithic parallel SQL database systems with variations on the Hadoop stack, concluding that the latter is preferable. They also highlight the fact that open-source stack implementators have forgotten almost everything about single system performance, focusing solely on scale-out.

In their article, Rusu et al. [34] introduce GLADE, a scalable distributed framework for

large scale data analytics. The main abstraction at the core of GLADE is Generalized Linear Aggregates (GLA) that are based on the notion of user-defined aggregates (UDA) which allow users to extend the functionality of a database system with specialized aggregation operations.

2.3 ASDI Correlation

In this section, we introduce ASDI data and the correlation process.

The purpose of the Correlator is to tag flight plan and track messages from multiple Air Traffic Control (ATC) centers relating to the same flight with a unique identifier called the `FLIGHT_KEY`. The consumer is free to process updates for each flight without the overhead of deciding which messages are associated with each flight. This is a powerful benefit because each ASDI message does not always, in and of itself, provide enough information to identify the flight it is associated with.

The ASDI data feed is a continuous stream of messages delivered over a TCP/IP network socket from an upstream ASDI vendor. A data distribution server was created to receive one stream from Embry-Riddle Aeronautical University (ERAU), record the ASDI data, and make it available locally.

The ASDI data feed produced by Enhanced Traffic Management System (ETMS) is a stream of data packets containing Zlib compressed XML documents of ASDI messages with a binary header. ASDI messages can be flight plan related data, oceanic reports, or Host track reports.

Each NAS center identifies a flight plan by its own three character alphanumeric Computer Identification (`CID`) code. The ASDI stream contains messages for a flight from each center the flight will pass through, although, flight plan related messages in the ASDI stream do not include the `CID` once a flight becomes active. Often a center will create a new flight plan and `CID` for an

existing flight, only to cancel one of them shortly thereafter. It is not always clear if the second flight plan was meant to be an amendment or if it is an additional flight. Sometimes, a message for an inactive flight may be missing a CID, which adds to the confusion.

Some centers will even change the origin of an international flight to a local facility once it enters their control. Anchorage Center will often issue a departure message for an arriving active flight over Canada, even though it departed hours before from the mainland US.

2.3.1 Design

The overall design of the correlator is to identify each unique flight plan that is filed in the NAS and to examine each ASDI message to deduce which flight a message is associated with. The centers operate independently of each other to assign their own CID, so the CID alone is not sufficient to correlate messages from different centers. The ETMS will also assign a unique CID to regularly scheduled flights before they are filed by a center. This is compounded by fact that flight plan related messages do not include the CID once a flight becomes active. Excluding the CID for an active flight should not be a problem since one would not expect more than one flight in the air with the same flight number, but it happens often. The opposite can also be true where messages arrive without a CID, but the departure message or updated flight management record have not arrived for the flight, potentially producing multiple candidates.

New flight plans can be created midflight, hours or even a day before the flight departs.

As the correlator identifies a new unique flight plan it generates a new unique FLIGHT_KEY record and stores a set of key parameters such as the aircraft identifier, flight status, departure center, aircraft type, destination, etc. that together make the record unique. As new messages arrive for a flight the parameters are filled in and updated as the flight progresses. The CID does provide useful information and may directly correlate a message if the CID for a center is already

stored for a flight plan. In particular the CID can provide an indication of the status of a flight.

2.3.2 Correlation Process for Position-Related Messages

For Host track and oceanic reports, if there is a flight plan with a matching aircraft identifier and CID, then the message is correlated with that flight plan. Otherwise, a list of all active flights with a matching aircraft identifier, as well as departure point and destination for oceanic reports, is compiled. If there is only one matching flight then the message is correlated with that flight. If there are multiple matching active aircraft identifiers then the flight plan with the closest matching last known position is correlated.

2.3.3 Correlation Process for Flight Plan-Related Messages

The correlation process for all flight plan related messages is based on key parameters. For each message a list is generated of all flight plans with matching aircraft identification, departure point and destination. If the flight plan related message, other than an Flight Management Information (RT) message, contains a CID, it should be for a flight that is not active and an attempt is made to find a match by CID. RT messages are different because they typically always contain a CID and the flight's status is indicated by the flight status field in the RT message. All other messages that do not contain a CID should be for active flights. There should not be more than one flight active at the same time with the same aircraft identification, however there are some seen throughout each day. The process is complicated by messages that are partially complete or contain incorrect data.

Because each center has its own CID, the list of possibilities is further reduced by comparing the departure and arrival times for reasonable matches. Flight plan, amendment and boundary crossing messages contain a coordination point and time that marks where and when the aircraft

should be handed off to the facility producing the message. These messages are matched to the most likely flight in which the coordination time falls within the bounds of the departure and arrival times, if known. Therefore, each flight record contains one or more tailored flight plans from one or more centers. Flight plan cancelation messages are matched to each center's tailored flight plan. Flights are considered complete when all center flight plans are canceled or an arrival message is received.

Once a single flight plan record is correlated to the incoming message the `FLIGHT_KEY` is set. If more than one flight plan record still matches then the incoming message and possible matches are logged and the `FLIGHT_KEY` is set to `UNCORRELATED`. If the incoming message cannot be matched to an existing record then a new record is created for the flight plan and a new `FLIGHT_KEY` is assigned to it. This process generally works for most of the flight plan related messages.

2.4 Architecture

In this section, we present the overall system architecture including the physical and conceptual modeling of the data warehouse as well as processing of archived and near real-time ASDI feed.

Figure 1 illustrates the overall physical structure of the system. A TCP/IP live ASDI stream is consumed by a message flow deployed on WebSphere Message Broker where messages are sorted and pushed to related queues based on their message types in WebSphere MQ. Another message flow deployed on Message Broker pulls from the related queues and populates the tables in the `SURVDB` database. Each queue in WebSphere MQ maps to a table in the `SURVDB` database. As tables are populated with raw XML messages in the `SURVDB` database, the XML Shredder, which is comprised of a set of stored procedures parses out raw messages and puts their content

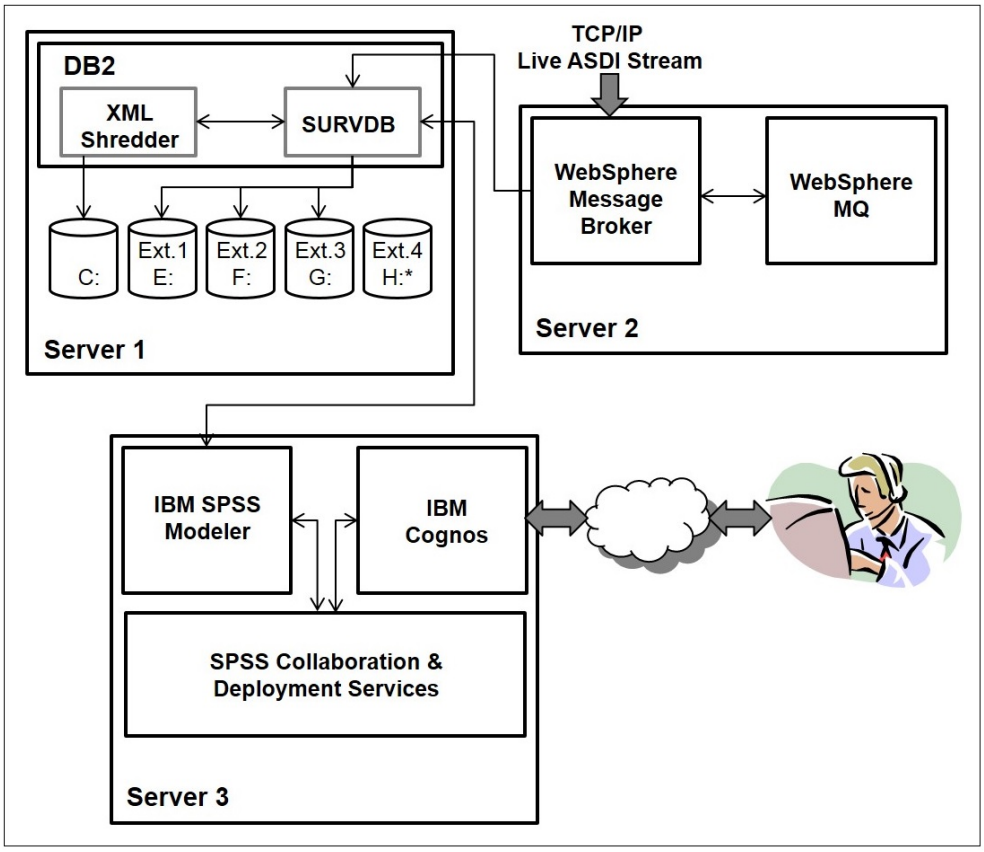


Fig. 2.1: Overall System Architecture.

per tag into separate tables in the `SURVDB` database. Tables are partitioned across the external hard drives, each 2TB of size. This summarizes the process of near real-time message flow into the database.

The end user interaction with the system is as follows: The end user is provided with a number of pre-set models, created and stored in SPSS Modeler, presented on Cognos, and available on a demilitarized zone (DMZ). The user passes parameters from Cognos to SPSS Modeler through SPSS Collaboration and Deployment Services. SPSS Modeler connects to the `SURVDB` database and pulls the data describing the past behavior. Upon receipt of new parameters, the SPSS Modeler invokes the scoring service to make predictions using descriptive behavior and presents the results back to the end user on Cognos.

2.4.1 Physical Modeling

Figure 2.1 illustrates the overall physical structure of the systems where we used total of three servers.

- Server-1 is a Dell T3400 computer with 4GB of RAM and 64-bit dual-core Intel Core 2 Duo processor where Infosphere Warehouse was installed along with DB2 on top of Windows Server 2008 R2. The server is connected to four external drives, each with 2TB of storage.

- Server-2 is a Dell M4500 computer with 4GB of RAM and 64-bit quad-core Intel Core i7 processor where WebSphere Message Broker is installed along with WebSphere MQ for near real-time correlated ASDI data consumption from a TCP/IP stream, and brokering. The server runs on Windows Server 2008 R2.

- Server-3 is a Dell T5500 computer with 16GB of RAM and 64-bit six-core Intel Xeon X5650 processor where SPSS Modeler and Cognos BI tools along with SPSS Collaboration and Deployment Services are installed. The server runs on Windows Server 2008 R2.

2.4.2 Conceptual Modeling of Database

Since the incoming ASDI messages are in XML, a standard format for the exchange of semi-structured data, we first translated them into an equivalent relational schema. Some approaches for translating XML documents into a relational database can be found in the literature, on DTD and XSD [35, 36] or other alternatives [37], but insufficient emphasis is given to the problem of determining cardinality of relationships, which instead has a primary role in multidimensional design [38].

We generated database tables based on classes created from the schema definitions for the correlated ASDI data. There are nine main database tables corresponding to an ASDI messages

as listed in Table 1.

There are eleven supporting database tables that hold portions of the data from the main ASDI messages as listed in Table 2.

Each table contains a UUID,SOURCE_DATE, and SOURCE_TIME element. The UUID element ties records in the supporting database tables to the corresponding ASDI message in the main database tables. The SOURCE_DATE element determines what DB2 partition table to place the data in. With DB2, range partition tables (RTP) provide the capability to have a virtual table that is actually composed several tables. This way query and data load performance is increased and large datasets are better managed.

Partitioning is important for various reasons. First, the query optimizer is aware of the partitioning structure, and can analyze predicates to perform partition exclusion: scanning only a subset of the partitions instead of the entire table. Second, each partition of a table can have a different storage format, to match the expected workload [39].

When a record is written to a database table, DB2 uses the SOURCE_DATE to write the record to the correct table. When a query is made with a specified range, DB2 will only check those tables within the range. If the data is ever deployed on another database provider, other means for partitioning may need to be deployed. In the worst case, one can use the original implementation where actual physical tables were created for each range.

Each main database table also contains a FLIGHT_KEY element. The FLIGHT_KEY enables a user to retrieve all messages that correspond to a particular flight.

2.4.3 Processing of ASDI Data

The system was built with two separate data flows in mind:

- Processing of archived ASDI data of two years.

- Processing of near real-time of ASDI data.

Both processes included correlation and database loading steps.

Processing of Archived ASDI Data First, we will go over the process of adding ASDI data to the database. The process consists of two steps. In the first step, the ASDI data must be correlated. In the correlation process, messages are correlated to flight plans and assigned a unique `FLIGHT_KEY`. For those messages that cannot be correlated, the `FLIGHT_KEY` is assigned the value `UNCORRELATED`. Given the `FLIGHT_KEY`, a user can obtain all messages corresponding to a given flight. Figure 2.2 shows the block diagram for converting ASDI data to correlated ASDI data. The ASDI Playback application is a server that accepts TCP/IP connections from a client. When a client connects, the Playback Server begins streaming ASDI data to the client. Currently there is no flow control between the server and the client. The Server publishes at a fixed message per second rate. In addition, no timing consideration is made based on the size of the message when publishing. The Playback Server reads a set of ASDI files where each file represents a day of messages. The output of the correlator is a set of four files for each input file. There is a file containing all flight related messages, a file containing all track information, a file containing all messages that could not be correlated, and finally a file containing all messages that contained errors.

It currently takes about thirty minutes for the correlator to process a day of ASDI data. Given that we have collected approximately two years of ASDI data, it would take approximately fifteen days to process the entire set. Recovery methods are in place in the event that one of the applications stops. In order to recover from a failure, the correlator periodically saves off a checkpoint of the correlation data. In the event of a failure, the Playback Server is restarted from the time of the last checkpoint.

In the second step, the correlated data must be processed and loaded into the database.

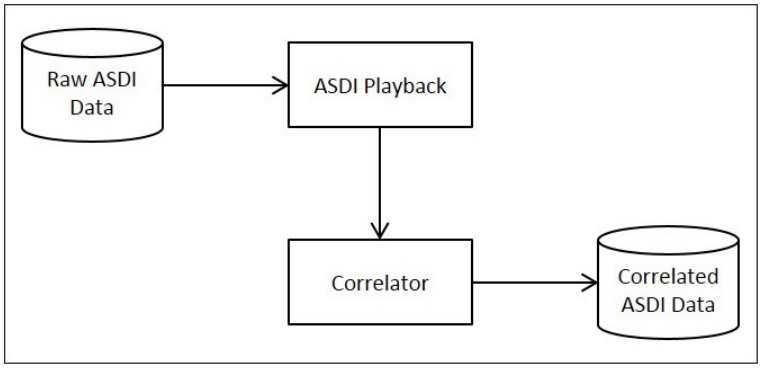


Fig. 2.2: Creating Correlated ASDI Dataset.

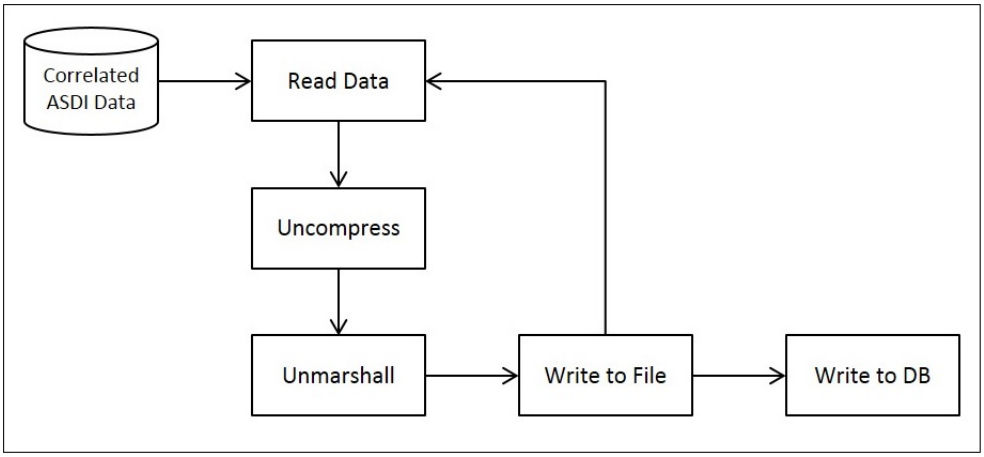


Fig. 2.3: Writing Correlated ASDI Messages to the Database.

Figure 2.3 shows the block diagram for processing the correlated ASDI data and loading it to the database. The application sequentially processes a list of files containing correlated ASDI messages. The files can be in one of two formats, binary or text. The binary format is the format used by the correlator. Content of the binary file consists of a repeating pattern of header data followed by compressed payload data. The text format is used when errors are found processing the binary data. If an error is encountered while processing a file, an error is logged. A user would then examine the message, make any required corrections and create a new file where the first line contains the text ”Modified Message” and the second line contains the modified message.

The process for adding archived ASDI data of two years to the database is as follows:

- The next file in the list is opened.
- The next message block is read.
- If the message block is a binary format, the file is uncompressed.
- The message block contains all ASDI messages received in the last 30 seconds or up to a maximum of 150 messages.
- The message block is unmarshalled using JaxB.
- Each message is processed and written to a Comma-Separated Value (CSV) file.
- Each message will have a main table and may have supporting tables.
- If there are no more records in the file all CSV files are written to the database using the SQL load command.
- The next file is opened (step 1).
- If there are more records in the file, the next message block is read (step 2).

Processing of Near Real-Time ASDI Data A number of approaches regarding stream processing can be found in the literature. Stonebraker et al. [40] laid out eight requirements of real-time stream processing. Some of the approaches in literature broadly deal with order of input in stream processing, classifying the techniques as IOP or OOP. Krishnamurthy et al. [41] described IOP techniques as most early streaming implementations which were intolerant of disorder and expected data to arrive in a strictly time-oriented fashion. They described OOP as next generation systems which implemented flexible and efficient query operators that can accept data out of order. Our approach is insensitive to the input order, in which the messages are received.

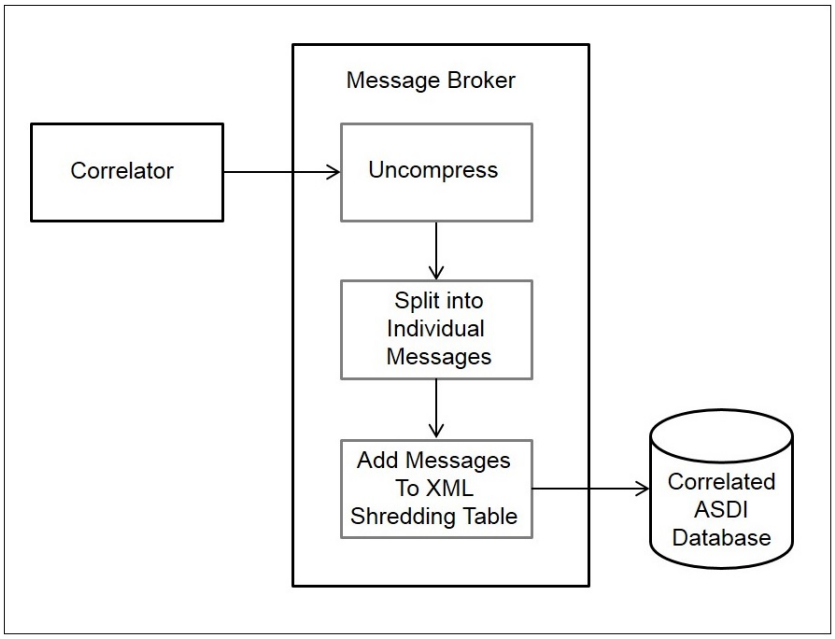


Fig. 2.4: Near Real-Time ASDI Processing.

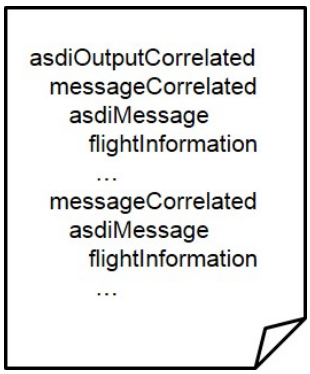


Fig. 2.5: Hierarchy of Correlated ASDI XML Document.

The approach for real time processing uses IBM MQ, IBM Message Broker, and XML Shredding. Figure 2.4 shows the block diagram for near real-time processing.

The Uncompress Compute Node connects to the correlator via a TCP/IP socket. The data is a continuous stream of correlated ASDI data consisting of a header and compressed ASDI data. The node reads the header file to determine the size of the compressed data block. The compressed data is then uncompressed resulting in an XML document consisting of correlated ASDI messages. The hierarchy of the document is shown in Figure 2.5. The XML document is then forwarded to the Message Splitting Node.

The Message Splitting Node extracts individual ”msgCorrelated” elements from the document. Each element is then wrapped to create a valid document and forwarded to the XML Shredding Node.

The XML Shredding node is responsible for adding each message to the SHRED_QUEUE table in the database. Each document maps to one of the nine message types discussed earlier. Each message maps to a main table with up to four supporting tables. There is a corresponding `XMLTABLE` statement for each table. The format for the `XMLTABLE` statement is:

```
XMLTABLE(xqueryExpression PASSING xmlSource COLUMNS columnName
columnDataType PATH pathXqueryExpression, ...)
```

Each message is placed in a Queue table in preparation for shredding where a stored procedure fetches the XML document and executes the `XMLTABLE` statements. Several columns require special processing to place the data into ”Host” format. The result of the shredding process is a record that is inserted into the appropriate table.

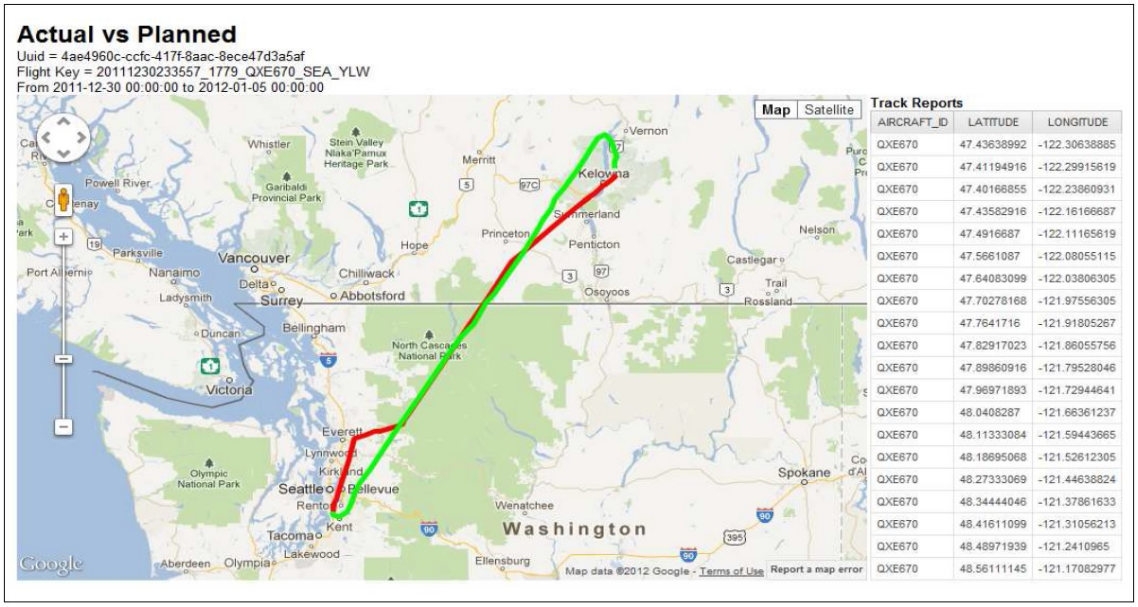


Fig. 2.6: Actual versus Planned Report (red: waypoints, green: track reports).

2.4.4 Client Interface

The client interface uses IBM Cognos Report Studio. Report Studio is a web application for creating reports. Users can tailor reports by specifying criteria such as date range, aircraft id, departure airport, or arrival airport. Report Studio supports several outputs including HTML, XML, PDF, Excel, and CSV. Reports are available for all nine messages and can contain links allowing users to obtain additional information. Figure 2.6 is a view of "Actual versus Planned", illustrating flight plan overlaid with trajectory. This report shows waypoints for the flight plan and the corresponding track reports.

IBM Framework Manager is used to create the data package used by Report Studio. Framework Manager is a metadata modeling tool that allows a user to define data sources, available data items, and relationships between data items. If the source is a database, Framework Manager uses the relationships to determine the necessary table joins when constructing the queries.

2.5 Discussion

In this section, we discuss some of the issues we encountered regarding data representation, and archived data loading. We also present the performance optimizations we realized.

2.5.1 Data Representation

According to the FAA's ASDI schema, several data elements can be represented in more than one form. Either a separate column in the database for each form is required or the data could be stored in the FAA's Host computer system format with a corresponding column containing the format code. In either case, some user requests may require multiple queries to obtain the information. For readability, the choice was to save the data in Host format. Here, two columns are required. One column contains the format code and the other column contains a string representation of the value. Table 3, 4, and 5 show the possible formats for altitude, fixes, and speed respectively.

2.5.2 Performance Optimizations

The first iteration of the application used SQL inserts to add messages to the database. To reduce overhead between the client and the database server, the messages were pushed in batches. At first, it took over seven hours to push an entire day of ASDI messages to the database.

Optimization-1

The original application created the SQL statement by appending elements to a `Java String`. Timing tests revealed that it was taking over five hours (of the seven hours) to create the `String`s. Instead of appending the elements to a `Java String`, the `Java StringBuilder` class was used to create the statements. This significantly reduced the amount of time spent. The

Tab. 2.3: Possible formats for altitude

Format	Representation	Example	Comment
a	(d)dd	340	Altitude of flight level in 100s of feet
b	OTP	OTP	Aircraft is flying VFR-ON-Top
c	OTP/(d)dd	OTP/250	Aircraft is flying VFR-ON-Top at 25,000 feet
d	(d)ddB(d)dd	250B260	Assigned altitude block between 25,000 and 26,000 feet
e	ABV/(d)dd	ABV/600	Aircraft is flying ABV above 60,000 feet
f	(d)dd/fix/(d)dd	250/DAL/260	Aircraft will fly at altitude flight level 250 to the DAL fix to flight level 260
g	VFR	VFR	Aircraft is flying VFR
h	VFR/(d)dd	VFR/75	Aircraft is flying VFR at 7,500 feet

Tab. 2.4: Possible formats for fixes

Format	Representation	Example	Comment
a	dddd(L)/(d)dddd(L)	3500/04000, 3500N/04000W	Latitude/longitude
b	aa(a)(a)(a)	DFW	Fix name
c	aa(a)(a)(a)dddddd	DFW300040	Fix radial distance. Fix DFW, radial 300, distance 040

Tab. 2.5: Possible formats for speed

Format	Representation	Example	Comment
a	(d)(d)dd	345	True air speed in knots
b	ddd	345	Ground speed in knots
c	Mddd	M085	Mach speed (0.85, for example)
d	SC	SC	Speed in classified

reason for the order of magnitude in increased performance is due to the fact that `Java Strings` are immutable. Every time `String` is appended, a new instance is created. Hence, continually appending to a large `String` will result in very poor performance.

Implementing the first optimization dropped the time to process a day of data to hundred and twenty minutes, resulting in time savings of 71.4%.

Optimization-2

The software was ported from IBM BladeCenter HS21 (with 4GB of RAM and 64-bit dual-core Xeon 5130 processor) to Dell M4500 computer (with 4GB of RAM and 64-bit quad-core Intel Core i7 processor). Both servers run on Windows Server 2008 R2.

Porting the software to a faster platform reduced the time to process a day of data to thirty minutes. This highlights the fact that database performance is highly correlated to the hardware it runs on.

Optimization-3

Nowadays, DBMS support standard bulk loading utilities for batch loading data. This approach is useful in traditional periodically updated data warehouses, in which the data load usually represents a large number of rows [42].

In our case, data was originally pushed into the database using the insert statement. By using the load command, the time to push a day's worth of data decreased from thirty minutes to fifteen minutes. To support the load command, the application was modified to write CSV files for each table. When the entire day of messages had been processed, the load command pushed each CSV file into the appropriate tables.

Optimization-4

To limit the size of the tables, the original code created multiple tables for each table type. For queries, this approach puts the burden on the application to query multiple tables when a range crosses several tables. DB2 supports the concept of RPTs. Here, when adding a record, one or more columns are used to determine the table. With RPT, the user is not required to make multiple queries when a range crosses a table boundary. With RPT, the time to process a day's worth of data actually increased to thirty minutes. However, this additional fifteen minute cost per day of partitioning enabled us saving significant amount of time and resources during database queries.

Optimization-5

Three changes were made to the database:

- Range periods were changed from a week to a month.
- The automatic table space resizing was changed from 32MB to 512KB.
- The buffer pool size was decreased.

The changes decreased the time to process a day's worth of data from thirty minutes to twenty minutes. The application of all optimizations reduced the total processing time of a day's data from over seven hours to twenty minutes, resulting in total time savings of 95.2%.

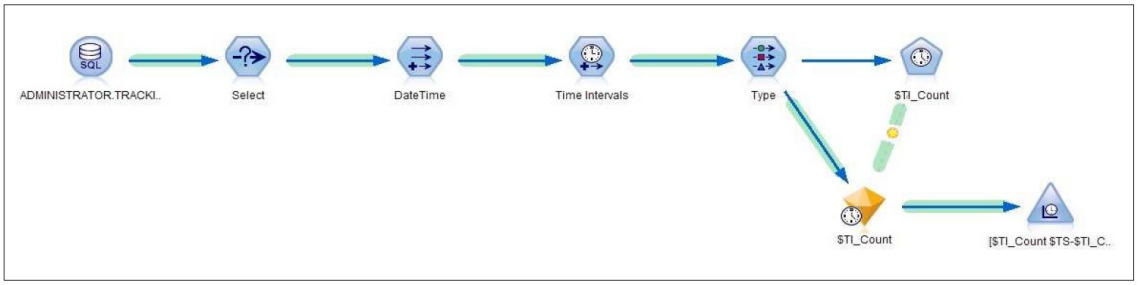


Fig. 2.7: The Use-Case in Stream Format in SPSS Modeler.

2.6 Basic Use-Case

In this section we present a basic use-case where we perform analytics predicting distinct traffic volume within an Airspace Volume of Interest (AVOI).

The use-case is as follows: For a given AVOI, compute the distinct traffic volume within that AVOI at any point in the future in 15 minute buckets.

Narrowing the time gap between data acquisition and acting on a business decision based on the data is referred to as near real-time business analytics [43]. With this use-case we aim to alert on congestion due to flow control areas or weather if certain thresholds are exceeded and propose alternate flight paths.

For this use-case we executed SPSS Modeler, which provides a way of building predictive models through a graphical user interface, utilizing many advanced modeling techniques. In addition, it also has the ability to perform various data processing operations to better prepare the data to be modeled. The model that the user designs is shown in a stream format in Figure 2.7, which allows for easy visualization of the model flow.

Illustrated in Figure 2.7, Node-1 pulls in the TRACKINFORMATION table which contains aircraft position data using SQL. Node-2 determines the boundary of the model stream using the

AVOI parameters (bounding box, minimum altitude, maximum altitude). Node-3 combines the SOURCE_DATE and SOURCE_TIME to a timestamp that can be understood by the modeling node. Node-4 computes the interval that the database entry will fall in. The time interval is 15 minutes. Node-5 defines the target and input fields needed for creating the model. Node-6 handles the creation of the model. Node-7 (the golden diamond) is the final result that is actually produced by the modeling node. It contains the predictive part of the model. Node-8 produces a graph based off of the model results.

Execution scenario is as follows: the user enters AVOI parameters along with a specific date for prediction. SPSS Modeler then performs data processing on the database, counting relevant records occurring in the past. SPSS Modeler will then return the predicted value along with likelihood estimations. Provided descriptive behavior of the past data, next year's flight count within the user-defined AVOI is predicted with certain likelihood.

2.7 Conclusion

Currently the raw ASDI data is compressed and uncorrelated to flight plans. There is no mechanism in place to extract meaningful insight from this data. With the ASDI data correlated to flight plans and stored in a structured fashion, meaningful data can be extracted. As shown in this work, the extracted data can then be used as input for analytics tools.

The nature of raw ASDI data is such that the sheer volume of raw surveillance data makes performing analytics very difficult. On average, over seven million ASDI messages are received per day. Total storage for one day of surveillance data is approximately 1GB. Based on the number of messages and the storage requirements, a scalable data warehouse approach was implemented. Once the database was populated clients could "slice and dice" the data in ways that were not possible when in the native format. Also, by using range partitioned tables, not only did query

performance increase, but table size restrictions were no longer an issue. This allows for years of data to be stored in a single database.

The raw ASDI data runs through a series of processes before it can be used for analytics. The raw data is sent through a correlation process, where messages are correlated to flight plans, and then saved in a compressed file. A Java application then runs through these files where they are uncompressed, unmarshalled, and then inserted into their designated database tables.

Once the stored historical ASDI data has been inserted into the database, a separate series of processes were put into place for inserting real time data. This strategy takes the compressed data blocks directly from the correlator output, and prepares them for insertion into the database. The individual messages are separated and placed in a raw-XML form into a queue table in the database. Stored procedures are then run against tuples in the queue table in a process called XML shredding.

Using XML Shredding to store real-time ASDI data allows for the database to be constantly updated with the most up to date information. This will allow for analysis in near real-time to discover current trends as they are happening. Additionally, aviation queries can be issued on the most current information.

The entire ASDI archival service has been developed, tested, and deployed entirely on commodity hardware. This hardware allows for easy maintenance and scalability as the database grows. It also allows for easy integration into existing laboratory infrastructure and networking. The current hardware architecture is made up of only three physical servers.

We plan on using aviation weather information in the analytics process to attain more accurate predictions. Among the potential data sources for this capability are Meteorological Aerodrome Reports (METAR) [44], and Rapid Refresh (RAP) [45].

Part III

Time of Arrival Prediction and Trajectory Clustering

Chapter 3

Flight Time Prediction

A major factor in increased airspace efficiency and capacity is accurate prediction of Estimated Time of Arrival (ETA) for commercial flights, which can be a challenging task due to a non-deterministic nature of environmental factors, and air traffic. Inaccurate prediction of ETA can cause potential safety risks and loss of resources for Air Navigation Service Providers (ANSP), airlines and passengers. This chapter presents a novel ETA Prediction System for commercial flights. The system learns from historical trajectories and uses their pertinent 3D grid points to collect key features such as weather parameters, air traffic, and airport data along the potential flight path. The features are fed into various regression models and a Recurrent Neural Network (RNN) and the best performing models with the most accurate ETA predictions are compared with the ETAs currently operational by the European ANSP, EUROCONTROL. Evaluations on an extensive set of real trajectory, weather, and airport data in Europe verify that our prediction system generates more accurate ETAs with a far smaller standard deviation than those of EUROCONTROL. This translates to smaller prediction windows of flight arrival times, thereby enabling airlines to make more cost-effective ground resource allocation and ANSPs to make more efficient flight schedules.

Fig. 3.1: Air traffic in Spanish airspace on July 12, 2016.

3.1 Introduction

Accurate prediction of Estimated Time of Arrival (ETA) for commercial flights is a key component of collaborative decision making (CDM), a process that attempts to keep the costs under control along with improvement in four key Air Traffic Management (ATM) areas; safety, capacity, efficiency and environmental impact. However, just a seemingly insignificant event such as a delay in obtaining a wheel chair can result in delays as slots are missed and reassigned which can be costly for airlines resulting in increased fuel-burn, emissions and expanded flight hours. This can also cause passenger dissatisfaction and loss of market share due to a hampered corporate image. The impact can grow exponentially due to ripple effect caused by delays in subsequent flights, missed connections, and even disruptions. Hence, due to the nature of unknowns, and complexity of airspace system, it is a challenging task to make an accurate ETA prediction. To illustrate the airspace complexity and volume of air traffic in a single day in Spain, we present Figure 3.1.

Traditional methods approach the ETA prediction problem deterministically, taking into

account aircraft performance models, along with either parametric or physics-based trajectory models. They usually begin by estimating the flight trajectory, which includes the lateral flight path together with altitude and speed profiles, and then proceed to calculating the time required to fulfill the predicted trajectory. These models by themselves fail to account for external operational circumstances such as weather phenomena, airspace sector densities, and airport congestion, which can directly affect the aircraft's actual flight profile. Note that decreasing the average delay per flight by one minute could save millions of dollars in annual crew costs and fuel savings for a mid-sized airline. Hence, unlike traditional methods, we propose a machine learning based systematic approach to address the ETA prediction problem. We make a considerable effort in feature engineering to use a richer set of features including general information about flights as well as weather, air traffic, and airport data for more accurate modeling and prediction. Our prediction system is built upon the concepts presented in the Aircraft Trajectory Prediction System [46] in which airspace is considered as a set of data cubes around grid points as part of the 3D reference grid network.

This chapter contains three main contributions:

- We propose an ETA Prediction System for commercial flights. Unlike other systems that collect and use features only for the arrival airport, we use a richer set of features along the potential route, such as weather parameters and air traffic data in addition to those that are particular to the arrival airport. Our feature construction process generates an extensive set of multidimensional time series data which goes through Time Series Clustering with Dynamic Time Warping (DTW) to generate a single set of representative features at each time instance.

- We present algorithms for two major features that highly contribute to the accurate ETA prediction: airspace sector congestion rate, and airport congestion rate.

- We perform a comparative analysis using a number of regression models and an RNN and present their performances. Using the best performing model, we compare our results with those generated by the EUROCONTROL. Our experiments on real trajectory, weather, air traffic and airport data verify that our system outperforms EUROCONTROL's ETA prediction by offering not only a higher accuracy but also a far smaller standard deviation, resulting in smaller prediction windows of flight arrival times.

The proposed system can be used by the ANSPs and airlines for more accurate scheduling and resource management resulting in improvement in the four ATM key performance areas; safety, capacity, efficiency, and environmental impact. The rest of the chapter is organized as follows. Section 3.2 reviews related work. Section 3.3 describes the data used in this study followed by Section 3.4 where we present a proper set of features selected for ETA prediction via feature engineering. Section 3.5 states the problem and reviews two top-ranking boosting methods, while Section 3.6 presents our experimental evaluation. Section 3.7 draws concluding remarks.

3.2 Related Work

There has been much work and an abundant literature on predicting estimated rate of arrival for humans [47] and ETA for various transportation vehicles on land (road and rail) [48, 49] and water [50], in addition to air. Traditional methods to ETA prediction for air transportation, particularly in the case of commercial flights use a deterministic approach by heavily relying on aircraft performance models along with either parametric or physics-based trajectory models. They usually begin by computing the flight trajectory, which includes the lateral flight path together with altitude and speed profiles, and then proceed to calculate the time required to fulfill the predicted trajectory [51–55]. However, the time of arrival at a fixed point on the ground is dependent not only on the airspeed the aircraft will fly, but also on the winds, temperatures, air traffic along the

route and congestion at the arrival airport. Therefore, if the atmospheric, airspace and airport traffic data is uncertain, the reference trajectory may not have been laid out correctly with respect to the real world situation, yielding inaccurate prediction of estimated time-of-arrival for the specified airport. Probabilistic methods usually develop stochastic linear models of aircraft motion to estimate future aircraft positions. They model aircraft trajectories as Discrete-Time Stochastic Hybrid Linear Systems in which aircraft path decisions form hybrid modes which then inform mode transitions between turning and straight flight modes [56–58].

Unlike traditional methods, we propose a machine learning based systematic approach in which general information about the flight as well as weather and air traffic data are all taken into consideration. Our objective is to learn from historical data, possibly stored in an aviation data warehouse [59], discover patterns and generate a set of features to build a model for accurate ETA prediction. There are only a few papers following the machine learning approach to the ETA prediction problem for air transportation in the literature [60–62].

Gopalakrishnan et al. [63] demonstrate the use of machine learning techniques to predict delays in air traffic networks a few hours ahead of time.

In their study, Kern et al. [60] aim to enhance ETA predictions generated by the Federal Aviation Administration (FAA)'s Enhanced Traffic Management System (ETMS). They select a set of baseline features and gradually increase complexity by adding new ones to check if they interfere with each other. The features are selected in 4 phases; 1) flight data, 2) flight and weather data, 3) flight and air traffic data, and 4) flight, weather, and air traffic data. Next, they apply Random Forest (RF) to generate the model. According to their results, their model predicts ETA 78.8% more accurately than the FAA's ETMS. Although their model's performance is applaudable, they use weather and air traffic data only for arrival airports during their feature collection, missing all the data along the routes between departure and arrival airports.

In his paper, Takacs [61] presents a solution to the General Electric's Flight Quest contest to make flights more efficient by improving the accuracy of arrival estimates. The contestant presents his prediction approach to runway and gate arrival times of en route U.S. domestic flights based on flight history, weather, air traffic control, and other data available at a given time. His approach follows 6 consecutive stages of ridge regression and gradient boosting machines, trained on a total of 56 features. The model structure is a result of a heuristic, hand-run optimization process. The workflow iteratively defines features and tries to incorporate the modeling to decrease the RMSE as much as possible. However, due to time constraints imposed by the contest, the author decided to keep the model simple by using only 56 features.

Glina et al. [62] use Quantile Regression Forests (QRF), which is an extension of RF as part of ensemble of Classification and Regression Tress (CART), to generate point predictions and pertinent conditional probability distributions for the ETA of individual flights. They validate their model on data from the Dallas/Fort Worth International Airport, obtaining mean absolute errors (MAE) of less than 60 seconds for their estimates of time-to-wheels-on for flights with a distance to the airport less than or equal to 20 nautical miles. Unfortunately, their data pertains to a single airport and it spans the period of only 5 days.

To summarize, our approach is distinguished from past efforts with the following respects: 1) Our system predicts ETA for runway arrival times. 2) Our system predicts ETA before departure, when the flight trajectory is still unknown. In fact, our prediction system doesn't make use of costly flight plans, offering a more cost-effective solution. Hence, our approach addresses the ETA prediction problem strategically over a time horizon of several hours. 3) For our model, we collect and use a richer set of features not only for the arrival airport but also for the airspace along the potential route. 4) We validate our model using an extensive set of real trajectory, weather, and airport data over the period of 11 months.

Tab. 3.1: A set of major air routes in Spain.

Departure Airport	**Arrival Airport**
Barcelona–El Prat Airport (LEBL)	A Coruña Airport (LECO)
	Málaga Airport (LEMG)
	Vigo–Peinador Airport (LEVX)
	Seville Airport (LEZL)
Adolfo Suárez Madrid-Barajas Airport (LEMD)	Almeria Airport (LEAM)
	A Coruña Airport (LECO)
	Jerez Airport (LEJR)
	Menorca Airport (LEMH)
	Palma de Mallorca Airport (LEPA)
	Vigo–Peinador Airport (LEVX)

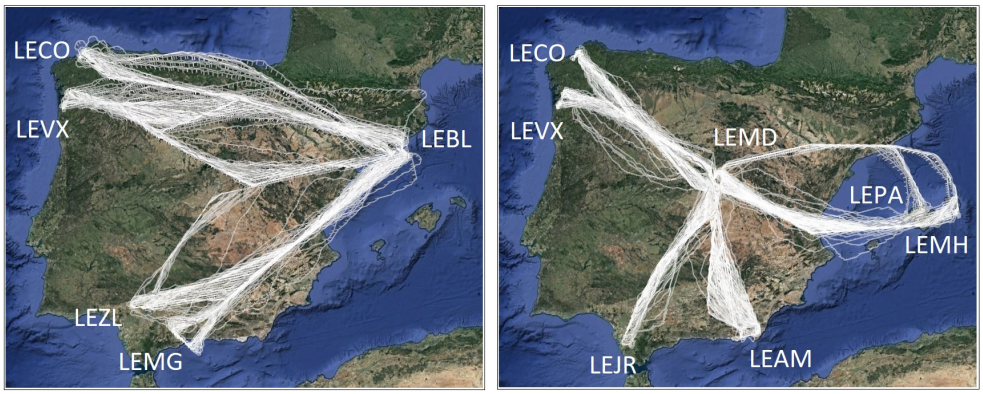
3.3 Data Description

This section introduces the datasets used for ETA prediction.

3.3.1 Trajectory Data

Trajectory data plays a large role in our study. Trajectories have been the subject of much work in the spatial domain with an emphasis on cars along roads [64]. The focus has been on their generation (e.g., [65]), queries (e.g., [66–70]), and matching (e.g., [71–73]). This data is collected continuously and is quite voluminous. Instead, our focus here is on the flight domain.

Due to the fact that there exists no system that continuously records and stores exact positions of an aircraft's original trajectory [74], only a discrete set of sample data are recorded and stored which presumably represent a close approximation of the original trajectory. We call



a) Flights departing from LEBL. b) Flights departing from LEMD.

Fig. 3.2: A set of trajectories for flights departing from Barcelona and Madrid.

this a raw trajectory. The raw trajectory data is provided by Spanish ANSP, ENAIRE, using radar surveillance feed with a 5 seconds update rate. The raw data is wrangled as part of the Data-driven AiRcraft Trajectory prediction research (DART) project under the SESAR Joint Undertaking Work Programme [75]. The data contains all commercial domestic flights for Spain, a total of 119,563 raw trajectories and 80,784,192 raw trajectory points for the period of January through November 2016. The fields of the raw trajectory data are as follows: *Flight No, Departure Airport, Arrival Airport, Date, Time, Aircraft Speed* in *X, Y, Z* directions, and position information *(Latitude, Longitude, Altitude)*. Note that, as a preprocessing step, we downsample raw trajectory data from the original resolution of 5 seconds to 60 seconds and align them to our 3D reference grid [1]. This is to build a set of features for each trajectory point, i.e. weather observations along the aligned trajectories. Table 3.1 lists a set of major flight routes departing from two major cities in Spain and Figure 3.2 provides their visual representation on a map.

3.3.2 Meteorology Data

The meteorology data is obtained from the National Oceanic and Atmospheric Administration's (NOAA) Global Forecasting System (GFS) [76]. The original data has 28-km spatial and

6-hour temporal resolution and it contains over 40 weather parameters including *Atmospheric, Cloud* and *Ground* attributes for each grid point as part of its 3D weather model. Hence, for this study's geographic volume of interest for the time period of January through November 2016, over 80TB of weather data is collected.

3.3.3 Airport Data

The airport traffic data is provided by EUROCONTROL as part of the Data-driven AiRcraft Trajectory prediction research (DART) project under the SESAR Joint Undertaking Work Programme [75]. The data contains airport traffic data for Spain, a total of 1,252,571 sets of records, where each record is composed of *Date, Flight No, Departure Airport, Arrival Airport, Aircraft type, Actual Departure Time, Actual Arrival Time, Scheduled Departure Time* and *Scheduled Arrival Time*.

3.3.4 Airspace Data

The airspace sector data is computed using raw trajectory data along with airspace sector volumes. The data contains aircraft counts within 15-minute bins for each sector in Spain for a period of 11 months. The records include *Sector Name, Date, Time of Sector Entry* and *Time of Sector Exit* with counts of aircraft.

3.4 Feature Engineering

In this section, we select and construct a proper set of features for ETA prediction via feature engineering. Feature engineering refers to the process of using domain knowledge of data to create features that best represent the underlying problem of the predictive models, resulting in improved model accuracy on unseen data. We adopt high-dimensional features to build models so

Tab. 3.2: The description of features.

Feature Type	**Feature**	**Description**
Flight	Airline	Name of airline
	Flight no	Flight number
	Aircraft type	Type of aircraft
Spatial	Latitude	Latitude of an occurrence
	Longitude	Longitude of an occurrence
	Altitude	Altitude of an occurrence
	Sector	Boundaries of a 3D airspace volume
Temporal	Date	Date of an occurrence
	Time	Time of an occurrence
	Time bin	15-minute time bin, an occurrence falls
Meteorological	Atmospheric temperature	Atmospheric temperature recording along the flight route
	Atmospheric wind speed	Atmospheric wind speed recording along the flight route
	Atmospheric wind direction	Atmospheric wind direction recording along the flight route
	Atmospheric humidity	Atmospheric humidity recording along the flight route
	Atmospheric pressure	Atmospheric pressure recording along the flight route
	Ground temperature	Ground temperature recording at the arrival airport

Tab. 3.3: The description of features continued.

Feature Type	**Feature**	**Description**
Meteorological	Ground wind speed	Ground wind speed recording at the arrival airport
	Ground wind direction	Ground wind direction recording at the arrival airport
	Ground humidity	Ground humidity recording at the arrival airport
	Ground pressure	Ground pressure recording at the arrival airport
	Ground wind gust	Ground wind gust recording at the arrival airport
Airport	Arrival airport	Airport the flight arrives at
	Departure airport	Airport the flight departs from
	Airport arrival count	Number of arrivals at an airport
	Airport departure count	Number of departures from an airport
	Airport congestion rate	Current vs. historical arrival+departure counts
Airspace	Sector aircraft count	Number of aircraft in a sector along the flight route
	Sector congestion rate	Current vs. historical aircraft count in a sector

Tab. 3.4: The description of features continued.

Feature Type	**Feature**	**Description**
Combinational	Flight-spatial	Traversed locations of particular flights
	Flight-spatial-temporal-airspace	Traversed locations of particular flights at particular time instances
	Meteorological-temporal	Observed weather parameters at particular time instances
	Airport-temporal	Arrival and departure counts of aircraft at a particular airport
	Spatial-temporal-airspace	Traversed locations by an aircraft at particular time instances
	Spatial-temporal-meteorological	Observed weather parameters at particular positions and time instances

they possess the ability to predict ETA more accurately. Features extracted from each domain are listed in Table 3.4.

3.4.1 Basic Features

Basic features are extracted from each individual domain. We exploit *Airline, Flight No* and *Aircraft Type* as the flight features. The first three letters of *Flight No* identify the *Airline*. Intuitively, each of these basic flight features or combination of them exhibit a distinct pattern, potentially yielding a different ETA. For instance, as *Aircraft Type*, Bombardier Canadair Regional Jet (CRJ) has different performance parameters than that of Boeing 737-800's (B738), likely to generate a different ETA.

We use *Latitude, Longitude, Altitude*, and *Sector* as the spatial features. Aircraft trajectories are nothing more than a set of joint 3D coordinates, where each coordinate is associated with a time stamp. Trajectories are usually of different length, formed by a different set of 3D coordinates. Hence, different spatial features may cause various patterns and biases in ETA prediction.

We adopt *Date, Time*, and *Time Bin* as the temporal features. Note that the temporal features listed in Table 3.4 are kept at a high level for the sake of saving space. Time bins are used to generate histograms. Note that, these features can refer to a flight departure or arrival or sector crossing as well as observation of a weather parameter. Essentially, ETA is an accumulated set of time intervals along the trajectory. Hence, a trajectory of 60 segments, where each segment is one-minute long, translates to an ETA of departure time plus 60 minutes.

Meteorological data is one of the major factors impacting the spatial-temporal patterns of trajectories. Among over 40 weather parameters, we exploit 5 atmospheric features for airspace *Temperature, Wind Speed, Wind Direction, Humidity*, and *Pressure*, and 6 ground features for airports *Temperature, Wind Speed, Wind Direction, Humidity, Pressure*, and *Wind Gust*. Convective

weather shaping up along the aircraft's planned route may cause a trajectory deviation yielding a different ETA.

The most recent work that aimed at addressing ETA prediction with a machine learning approach utilized particular features such as weather observations and traffic data for the arrival airport only, disregarding features along the potential trajectory [60]. This is partly due to fact that the exact trajectory is unknown before departure. However, excluding these key features along the potential trajectory greatly degrades the accuracy. Now, a critical question arises: *How can we obtain a set of features along the potential trajectory when the trajectory is still unknown?* Our approach to this problem is as follows: As a preprocessing step, we align raw historical trajectories to the 3D reference grid [46]. The process generates a time series of aligned historical trajectories. Figure 3.3 illustrates a set of 3D grid points historically traversed by a sample flight between a departure and arrival airport. Using spatially and temporally closest set of data points from the weather model, we perform interpolation and extract the pertinent parameters for each aligned trajectory point. Once weather observations have been extracted for each grid point at a particular time instance, we omit spatial information. Next, we perform Time Series Clustering with Dynamic Time Warping (DTW) [77, 78] on weather observations of variable length. The process aggregates the weather observations and generates a single set of representative features for each time instance. The result is a set of features in the form of time series. For instance, we use 300 meteorological features for a flight that is 60-minutes long. This is due to fact that each trajectory point is recorded once a minute and at each trajectory point 5 meteorological features are adopted along the potential route. The same applies to the sector congestion rates. For the same flight of 60 minutes, we use 4 time bins, resulting in 4 sector density features along the flight route.

Airport features consist of *Arrival Airport, Departure Airport, Airport Arrival Count* and

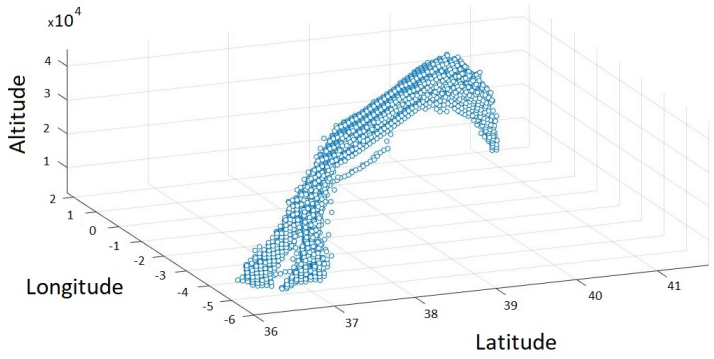


Fig. 3.3: Historically traversed 3D grid points between departure and arrival airports.

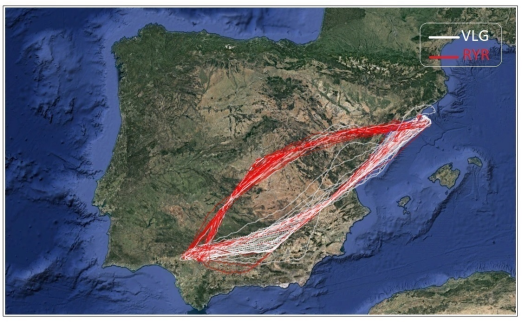
Arrival Departure Count, in addition to *Airport Congestion Rate*, which is computed using the actual and scheduled arrival and departure counts. Each of these features are likely to have an impact on flight's ETA, as the higher the airport congestion rate, the higher the likelihood of delay on arrival time.

We exploit *Sector Aircraft Count* and *Sector Congestion Rate* as airspace features. Similar to the impact by the airport congestion rate, the higher the airspace congestion rate along the planned route, the higher the likelihood of trajectory diversion causing a delay on arrival time.

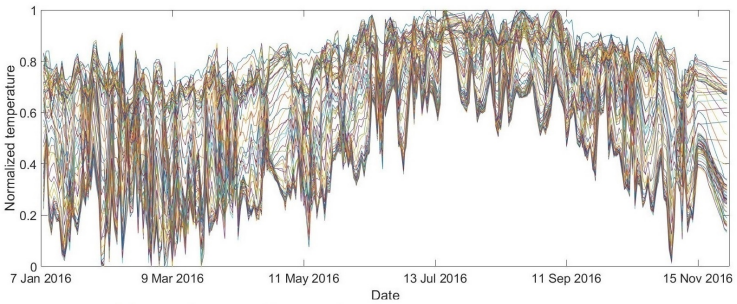
3.4.2 Combinational Features

We combine various features from two or more domains to build combinational features. Combining flight and spatial features produces *Flight-spatial* combinational feature. Intrinsically, each airline's flight exhibits a distinct trajectory pattern, yielding a different ETA. Figure 3.4a shows Ryan Air (RYR) versus Vueling Airline's (VLG) flights between the LEBL and LEZL airports. Red trajectories representing the RYR flight dominates the northern part, while white trajectories representing the VLG flight dominates the southern part of the routes.

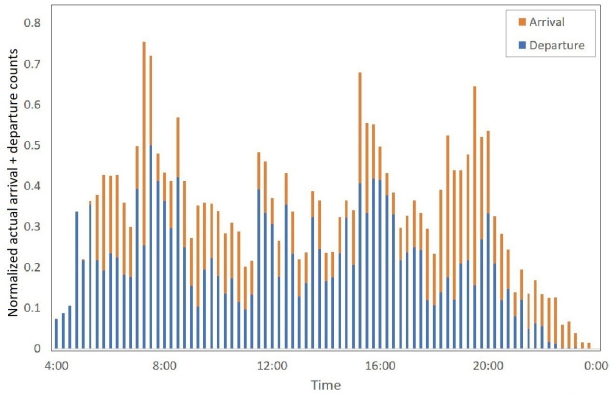
Combining meteorological and temporal features produces *Meteorological-temporal* combinational feature. Since weather observations vary over time and



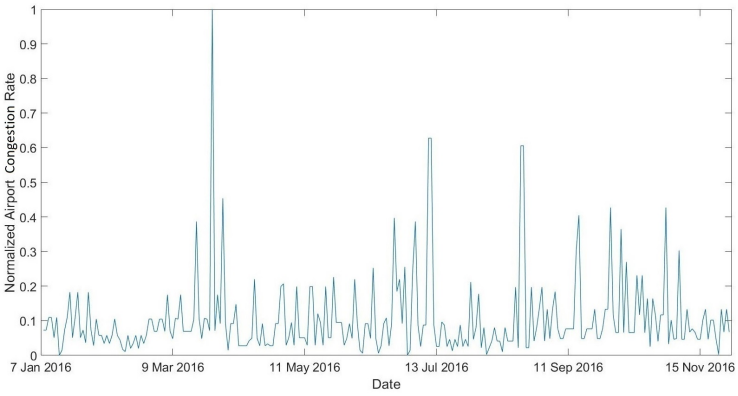
a) Two distinct trajectory patterns by two airlines.



b) Distribution of normalized temperature over time.



c) Distribution of normalized arrival and departure counts over 15-min. bins.



d) Congestion rates over 15-min. bins at LEZL airport for fight VLG22XV.

Fig. 3.4: Distribution of various features.

ETA is likely to be impacted by the weather patterns, it is intuitive to combine these features. Figure 3.4b shows distribution of normalized temperature over the period of January through November 2016 for the flight VLG22XV between the LEBL and LEZL airports. Due to fact that average flight time for the flight VLG22XV is 75 minutes and that each trajectory point is recorded once per minute, each time instance on Figure 3.4b has 75 temperature recordings. As shown in the figure, normalized temperature increases during the summer months.

We combine airport and temporal features to generate the *Airport-temporal* combinational feature. This enables us to gain insight on the distribution of aircraft counts on various airports at various time bins during the day, potentially impacting the ETA prediction. Figure 3.4c captures the distribution of normalized arrival and departure counts within 15-minute time bins for the LEZL airport. The airport appears to have more traffic at time bins around 8:00.

To examine combination of various features and derive insights on trends, find anomalies and perform strategic planning, a novel interactive visualization tool, NormSTAD [79] can be used.

3.4.3 Airport Congestion Rate

Given the actual (historical) arrivals and departures along with scheduled arrivals and departures for a particular airport, we want to compute the airport congestion rate for the 15-minute time bin, the flight of interest is scheduled to arrive. First, we create a set of 15-minute time bins throughout the day. Next, we compute the average counts for the actual arrival and departures for each of these 15-minute time bins. Given the scheduled arrival time of the flight of interest, we determine which time bin it falls into. For that particular time bin, we compute the airport congestion rate as presented in Algorithm 3.1. Figure 3.4d. illustrates congestion rates in each 15-minute bins at the LEZL airport for the flight VLG22XV, indicating a potential delay in late

March due to a high congestion rate at the LEZL airport.

Algorithm 3.1: Airport Congestion Rate

Result: Airport congestion rate

Input : Counts of scheduled arrivals $ctSA$, counts of scheduled departures $ctSD$, average count of actual arrivals $avgCtAA$, and average count of actual departures $avgCtAD$ of a flight with a flight number fno and timestamp ts for a particular airport

Output: Congestion rate cr of a particular airport, given the time stamp ts and a flight number fno of an arriving flight fl

$TB \leftarrow [tb_1, tb_2, ..., tb_k]$

$3dp \leftarrow lat, lon, alt$

$fl \leftarrow fno, 3dp, ts$

foreach $tb \in TB$ **do**

  if $fl.ts \subset tb$ **then**

    return $acr[tb] \leftarrow (ctSA[tb] + ctSD[tb])/(avgCtAA[tb] + avgCtAD[tb])$

  end

end

3.4.4 Sector Congestion Rate

Given the actual (historical) flights' sector crossings, along with scheduled flights' sector crossings for a particular sector, we want to compute the sector congestion rate for each sector, for each 15-minute time bin, the flight of interest is scheduled to cross. Due to fact that airspace sectors are nothing more than a set of extruded polygons, we use a Point-In-Polygon algorithm [80] to compute the counts of aircraft in each sector. To achieve this, we first create a set of 15-

minute time bins throughout the day. Next, we compute the average counts for the actual flights' sector crossings for each of these 15-minute time bins. Given the scheduled crossing time of the flight of interest, we determine which time bin it falls into, and which sector it likely crosses. Due to fact that the flight's trajectory is unknown at the time of ETA prediction, we consider all sectors the flight of interest is likely to cross based on actual sector crossings. Next, for that particular time bin, we compute the airport acceptance rate as presented in Algorithm 3.2.

Algorithm 3.2: Sector Congestion Rate

Result: Sector congestion rate

Input : Counts of scheduled flights' sector crossings $ctSF$, average counts of actual flights' sector crossings $avgCtAF$, for each time bin tb for each sector

Output: Congestion rate of a particular sector scr, given the time stamp ts and a flight number fno of a crossing flight fl

$TB \leftarrow [tb_1, tb_2, ..., tb_k]$

$3dp \leftarrow lat, lon, alt$

$fl \leftarrow fno, 3dp, ts$

foreach $tb \in TB$ **do**

  if $fl.ts \subset tb$ **then**

    return $scr[tb] \leftarrow (ctSF[tb]/avgCtAF[tb])$

  end

end

3.5 Problem Statement and Model Reviews

Given a set of historical flights with their attributes such as flight number, trajectory, departure and arrival airport, actual departure and arrival date and time, airline name, aircraft type, and associated weather and air traffic parameters, we aim at learning a model that predicts the current flight's ETA before it departs. Note that ETA can refer to both runway and gate arrival times. Our system predicts runway arrival times i.e. times between aircraft wheels-off and wheels-on, as we exclude airport surface data when we build our model. To address this problem, we build a number of models, and rank them based upon their performance.

3.5.1 AdaBoost

Adaptive Boosting is a meta-estimator to fit a sequence of weak learners on iteratively modified versions of the data. All the predictions are then combined through a weighted sum to produce the final prediction. The data modifications at each iteration consist of applying weights to each of the training samples. The first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence [81].

Formally, we assign an initial weight $w_i = 1 \quad i = 1, ..., N_1$ to each training pattern. We repeat the following procedure while the average loss $\bar{L}$ is less than 0.5.

(1) The probability that training sample i is in the training set is $p_i = w_i / \sum w_i$, where the summation is over all members of the training set. Select N_1 samples with replacement to form the training set. This may be implemented by marking a line of length $\sum w_i$ and subsection of length w_i. A uniform number picked from the range $[0, \sum w_i]$ and landing in section i corresponds to picking pattern i.

(2) Build a regression machine t from the training set. Each machine makes a hypothesis: $h_i : x \rightarrow y$.

(3) Pass every member of the training set through this machine to obtain a prediction $y_i^{(p)}(x_i) \quad i = 1, ..., N_1$.

(4) Compute a loss for each training sample $L_i = L\Big[|y_i^{(p)}(x_i) - y_i|\Big]$. The loss L may be of any functional form as long as $L \in [0, 1]$. If we let $D = sup|y_i^{(p)}(x_i) - y_i| \quad i = 1, ..., N_1$ then we have three candidate loss functions:

$$L_i = \frac{|y_i^{(p)}(x_i) - y_i|}{D} \quad (linear)$$

$$L_i = \frac{|y_i^{(p)}(x_i) - y_i|^2}{D^2} \quad (squarelaw)$$

$$L_i = 1 - \exp\left[\frac{-|y_i^{(p)}(x_i) - y_i|}{D}\right] \quad (exponential)$$

(5) Calculate an average loss: $\bar{L} = \sum_{i=1}^{N_1} L_i p_i$

(6) Form $\beta = \dfrac{\bar{L}}{1 - \bar{L}}$. β is a measure of confidence in the prediction.

(7) Update the weights: $w_i \rightarrow w_i \beta^{[1-L_i]}$. The smaller the loss, the more weight is reduced making the probability smaller that this pattern will be selected as a member of the training set for the next machine in ensemble.

(8) For a particular input x_i, each of the T machines makes a prediction. Obtain the cumulative prediction h_f using the T predictors: $h_f = \inf\left\{y \in Y : \sum_{t:h_t \leq y} \log(1/\beta_t) \geq \dfrac{1}{2} \sum_t \log(1/\beta_t)\right\}$ This is the weighted median. Equivalently, each machine h_t has a prediction $y_i^{(t)}$ on the ith pattern and associated β_t. For pattern i the predictions are relabeled such that for pattern i we have:

$y_i^{(1)} < y_i^{(2)} <, ..., < y_i^{(T)}$ Next, we sum the $\log(1/\beta_t)$ until we reach the smallest t so that the inequality is satisfied. The prediction from that machine t we take as the ensemble prediction. If the β_t were all equal, this would be the median.

Intuitively, the effect of varying the weight w_i to give more emphasis to difficult examples means that each subsequent machine has a disproportionately harder set of examples to train on. Thus, the average loss tends to increase as we iterate through the algorithm and ultimately the bound on L is not satisfied and the algorithm terminates [82, 83]

3.5.2 Gradient Boosting

Gradient Boosting [84, 85] produces a prediction model in the form of an ensemble of weak prediction models. It builds additive regression models by sequentially fitting a simple parameterized function to current pseudo residuals by least squares at each iteration. The pseudo residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data point, evaluated at the current step.

Formally, in the function estimation problem one has a system consisting of a random output variable y and a set of random input variables $x = \{x_1, ..., x_n\}$. Given a training sample $\{y_i, x_i\}_1^N$ of known (y, x) values, the goal is to find a function $F^*(x)$ that maps x to y, such that over the joint distribution of all (y, x) values, the expected value of some specified loss function $L(y, F(x))$ is minimized: $F^*(x) = \text{argmin}_{F(x)} E_{y,x} L(y, F(x))$

Boosting approximates $F^*(x)$ by an additive expansion: $F(x) = \sum_{m=0}^{M} \beta_m h(x; a_m)$ where the functions $h(x; a)$ which are base learners are usually chosen to be simple functions of x with parameters $a = \{a_1, a_2, ...a_k\}$. The expansion coefficients $\beta_{m_0}^M$ and the parameters $a_{m_0}^M$ are jointly fit to the training data in a forward stage-wise manner. One starts with an initial guess $F_0(x)$, and then for $m = 1, 2, ..., M$:

$$(\beta_m, a_m) = \text{argmin}_{\beta,a} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h(x_i; a))$$

and

$$F_m(x) = F_{m-1}(x) + \beta_m h(x; a_m)$$

Gradient boosting approximately solves (β_m, a_m) for arbitrary loss functions $L(y, F(x))$ with a two step procedure. First, the function $h(x; a)$ is fit by least-squares:

$$a_m = \text{argmin}_{a,\rho} \sum_{i=1}^{N} [\tilde{y}_{im} - \rho h(x_i; a)]^2$$

to the current pseudo residuals:

$$\tilde{y}_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

Then, given $h(x; a_m)$, the optimal value of the coefficient β_m is determined:

$$\beta_m = \text{argmin}_{\beta} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h(x_i; a_m))$$

This strategy replaces a potentially difficult function optimization problem by one based on a least-squares, followed by a single parameter optimization based on a general loss criterion L.

3.6 Experimental Study

This section presents the evaluations of our ETA prediction system.

3.6.1 Setup

We evaluate the performance of our prediction system on 10 major flight routes in Spain, presented in Table 3.1. We chronologically order each dataset for each route, normalize and standardize, and use the first 80% for modeling and the remaining 20% for validation. Table 3.5 lists the prediction models used in this study. We use 10-fold cross-validation and evaluate the algorithms using Root Mean Squared Error (RMSE) metric. We create a baseline of performance on this problem and spot-check a number of algorithms to address the regression problem. The first round of algorithms we use are linear and non-linear regression methods, in addition to historical

Tab. 3.5: A list of prediction models used in this study.

Method	**Algorithm**
Linear	Linear Regression (LR)
	Lasso Regression (LASSO)
	Elastic Net Regression (EN)
Non-linear	Classification and Regression Trees
	Support Vector Regression (SVR)
	k-Nearest Neighbors (KNN)
Ensemble	Adaptive Boosting (AdaBoost)
	Gradient Boosting (GBM)
	Random Forest Regression (RF)
	Extra Trees Regression (ET)
Recurrent Neural Network	Long Short-Term Memory (LSTM)

average (HA), which simply averages all historical flight times for the same routes for the same time period. The algorithms all use default tuning parameters. The second round of algorithms we use are ensemble methods comprised of two boosting (AB and GBM), and two bagging (RF and ET) methods. These algorithms also use default tuning parameters. The final algorithm we use is the Long Short-Term Memory (LSTM), which is a type of RNN. Given the flight times for the previous 10 most recent subsequent days, we compute the time for the next day's flight, provided that all flights share the same defining features such as flight number, departure and arrival airports, aircraft type, etc. In order to perform LSTM on our dataset, we first transform our multivariate time series data into a supervised learning problem. Next, we define the LSTM with 50 neurons in the first hidden layer and 1 neuron in the output layer for predicting the flight time. The input shape is one time step with over 300 features (the number of features depends on the average flight time). The model is fit for 50 training epochs with a batch size of 72.

Tab. 3.6: The RMSE values of all algorithms.

Model	LEBL LECO	LEBL LEMG	LEBL LEVX	LEBL LEZL	LEMD LEAM	LEMD LECO	LEMD LEJR	LEMD LEMH	LEMD LEPA	LEMD LEVX
HA	5.590140	5.498889	5.555261	4.911651	3.265500	4.000898	3.311410	3.953404	3.787092	4.666325
LR	6.565834	5.583422	6.669648	5.240185	4.848514	4.612558	4.167661	4.433904	4.943242	5.183345
LASSO	5.423176	5.328147	4.566031	4.620595	2.979159	3.939358	3.061945	3.618408	3.830357	4.676572
EN	5.129900	5.328147	4.510600	4.598869	2.979159	3.972221	3.078638	3.618059	3.759207	4.562907
KNN	4.315834	4.041417	4.080471	3.768015	3.768015	3.472992	2.837109	3.109983	3.633360	3.409278
CART	6.208306	5.228796	5.098875	4.717167	4.527573	4.439111	3.729169	3.790538	4.691645	4.175965
SVR	4.850719	5.036133	4.276852	4.320527	2.914718	3.575904	3.001464	3.199346	3.536542	4.151692
AB	**3.991038**	3.790171	4.244822	**3.620588**	**2.840969**	**3.128822**	2.602778	2.987388	3.347820	**3.092941**
GBM	3.993338	**3.430164**	4.068700	3.630516	3.045774	3.276929	**2.589360**	**2.908439**	**3.344821**	3.174045
RF	4.308547	3.841439	4.103479	3.706125	3.016808	3.345249	2.740151	2.910654	3.5721842	3.437595
ET	4.171390	3.538316	4.266809	3.731383	3.081153	3.396700	2.630542	3.028200	3.6377407	3.436975
LSTM	7.586650	5.674340	**3.708545**	4.673223	2.932544	3.714577	2.604398	3.988213	3.789673	4.311236

Our experiments are conducted on a computer with Intel Core i7-6820HQ CPU @ 2.70GHz and 16GB memory, running on Linux Ubuntu 16.04.1 64-bit Operating System. All the algorithms in our system are implemented in Python 3.6.4.

3.6.2 Results

Table 3.6 shows the RMSE value for each algorithm. The best score for each route is highlighted. In a quick glance, it seems clear that HA and linear models (LR, LASSO, EN) perform poorly on all routes. KNN and SVR seem to achieve the best scores among the linear and non-linear methods. In fact, KNN generates 8, SVR generates 2 best scores out of 10 routes among the linear and non-linear methods. The default value for the number of neighbors in KNN is set to 7. We obtain slightly better results when we use a grid search to try different numbers of neighbors. Unlike the previous work [60, 62] which emphasize the accuracy of bagging methods (RF and its extensions), our results indicate that boosting methods (AB and GBM) dominate the bagging methods (RF and ET) on all routes. LSTM yields relatively unstable scores, generating only one best score on all 10 routes. This may not be a surprise as LSTM is better suited for sequence prediction than autoregression.

Overall, the boosting methods dominate the entire list, generating the best scores among all including the linear, non-linear, ensemble and RNN methods. In fact, the scores obtained by boosting methods can be further improved with tuning. The default number of boosting stages to perform (n estimators) is originally set to 100. We define a parameter grid n estimators values from 50 to 400 in increments of 50. Each setting is evaluated using 10-fold cross-validation. By parameter tuning, we reach even better scores.

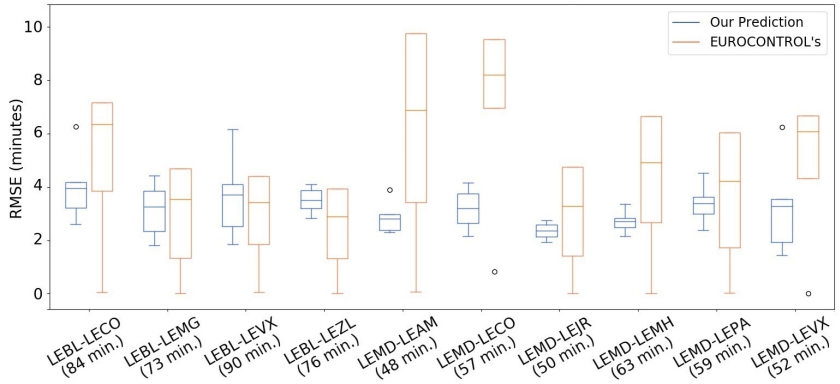
We complete our comparative study among the models listed in Table 3.5 and select the best score for each route. Next, we compare our final results with the ETA values, EUROCONTROL

uses [75]. Figure 3.5 illustrates RMSE values in minutes for each route between our prediction versus EUROCONTROL's prediction. Note that Figure 3.5 presents both results at two different scales. Figure 3.5a is a closer look at the box plots, where the median values are visible. However, the full extent of the boxplots are missing in Figure 3.5a due to outliers. Hence, we provide Figure 3.5b, where the full extent of the boxplots including the outliers are visible. From the results, we make the following observations: i) Our prediction yields better median scores on eight routes, while the EUROCONTROL's ETA shows better median scores on two routes (LEBL-LEVX and LEBL-LEZL). ii) The standard deviation values in EUROCONTROL's ETAs are much larger, resulting in larger windows of predictability at arrival times. This may have a significant adverse impact on airlines' ground resource management as well as connecting flights' scheduling. iii) Boxplots representing EUROCONTROL's ETAs in Figure 3.5b. show extreme outliers. In summary, our prediction system offers more accurate ETA prediction with far smaller standard deviation than those of EUROCONTROL.

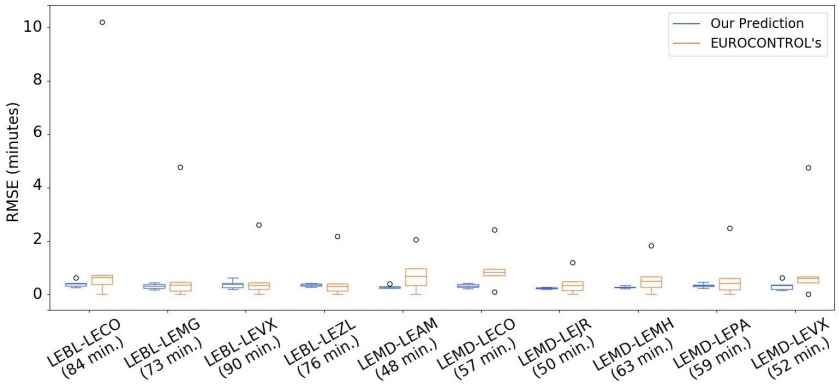
To evaluate the effectiveness of our features, we rank them based on their relative importance. The top 10 features are presented in Table 3.7. Evidently, the arrival airport is of critical importance on accurate ETA prediction. Aside from that, meteorological features in the form of a time series along the potential trajectory make a significant impact during regression. This insight validates our approach that meteorological and airspace sector features along the potential trajectories should be taken into account to achieve accurate ETA prediction.

3.7 Conclusion

We have presented a novel ETA prediction system for commercial flights. Our experiments verify that our system can predict any commercial flight's ETA in Spain within 4 minutes of RMSE on average regardless of the flight length. Our system outperforms EUROCONTROL's



a) RMSE by our prediction vs. EUROCONTROL's prediction at a larger scale.



b) RMSE by our prediction vs. EUROCONTROL's prediction at a smaller scale.

Fig. 3.5: RMSE by our prediction vs EUROCONTROL's on all flight routes.

Tab. 3.7: Top 10 features ranked by their relative importance.

Rank	**Feature**	**Score**
1	Arrival airport	1.0
2	Atmospheric pressure	0.67854
3	Atmospheric wind speed	0.66231
4	Atmospheric wind direction	0.65224
5	Atmospheric humidity	0.63331
6	Atmospheric temperature	0.61314
7	Airport congestion rate	0.53212
8	Sector congestion rate	0.31153
9	Flight no	0.29192
10	Aircraft type	0.13221

ETA prediction by offering not only a higher accuracy but also a far smaller standard deviation, resulting in increased predictability of flight arrival times. This enables airlines to better coordinate the action of ground handling personnel and equipment, thereby reducing the waste of time and energy. Likewise, air traffic flow management, airport runway and gate assignment, and ground support equipment usage optimization are all processes that can benefit from a more accurate ETA prediction for commercial flights.

Chapter 4

Trajectory Clustering

Whether descriptive, predictive or prescriptive, most analytics applications pertaining to aircraft trajectory data require clustering to group similar trajectories and discovering a representative trajectory for all as a single entity. During the process, considering a trajectory as a whole may mislead, resulting in overfitting and a failure to discover a representative trajectory.

In this chapter, we describe a novel clustering framework; divide-cluster-merge, *DICLERGE*, for the aircraft trajectory data, that divides trajectories into three major flight phases: climb, en-route, and descent. It clusters each phase in isolation, then merges them together. Our unique approach also discovers a representative trajectory, the model for the entire trajectory set. Our experiments use a real trajectory dataset with pertinent weather observations to demonstrate the effectiveness and efficiency of the *DICLERGE* algorithm to the aircraft trajectory clustering problem.

4.1 Introduction

Air Traffic and Capacity Management constantly monitors aircraft operating within the National Airspace System (NAS) and aims to safely and efficiently manage its flow. Maintaining

safe separation and sequencing of aircraft is a challenging task due to fact that the current approach of procedural control assumes that aircraft will fly along routes designated by *waypoints* or *fixes*. This approach is easy for air traffic controllers to understand; however, it can be inefficient for aircraft. To more effectively and efficiently address the pertinent challenges in the airspace with increasing volume, Trajectory Based Operations (TBO) is investigated in the context of *NextGen* in the U.S. [86] and *SESAR* in Europe [87]. The TBO concept uses four-dimensional aircraft trajectories as the base information for managing safety and capacity. In the heart of TBO resides the trajectory prediction process that relies on accurate clustering of aircraft trajectories, when a stochastic approach is used. Hence, clustering of aircraft trajectories is a key building block in probabilistic trajectory prediction process.

Although there is plenty of work in the area of data mining with regards to trajectory clustering in general [88–94], the vast majority address the issues pertaining to clustering of road networks [95–99]. Among the rest, only a handful of them deal with clustering of aircraft trajectories [100–102]. In fact, of those, to the best of our knowledge, there is no previous work addressing the clustering of aircraft trajectories taking three or more dimensions into account. Note that we do not deal with the issues of how the trajectories are generated (e.g., [65]), queries on the individual objects or waypoints (e.g., [66, 68–70]), and matching (e.g., [71–73]).

In this chapter, we propose a novel framework for clustering aircraft trajectories that is based on divide, cluster, and merge tasks. The framework consumes a set of multidimensional trajectory points pertaining to a particular flight, divides them based on flight phases, clusters them in isolation, and merges them. By performing lateral and vertical smoothing, the framework generates a representative trajectory, the model, capturing the behavior of aircraft.

In summary, the contributions of this chapter are as follows:

- Our unique approach enables accurately clustering of three or more dimensional aircraft tra-

jectories by employing a divide-cluster-merge framework. Whereas the very limited amount of previous research address clustering of aircraft trajectories with two dimensions, only.

- We present a formal aircraft trajectory clustering algorithm based on well-known aviation domain facts. We also present a representative trajectory algorithm that reveals the trajectory model based on lateral and vertical smoothing.

- We demonstrate by using a real aircraft trajectory dataset that our framework effectively performs clustering and discovers the representative trajectory.

The rest of this chapter is organized as follows. Section 4.2 discusses related work. Section 4.3 describes our novel algorithm for the aircraft trajectory clustering problem. Section 4.4 presents the results of experimental evaluation. Conclusions directions for future work are outlined in Section 4.5.

4.2 Related Work

Although there is a vast amount of literature in the data mining area, with regards to clustering trajectories, there is only a limited number of research on clustering aircraft trajectories. Of this research, there is no work addressing the clustering of aircraft trajectories taking three or more dimensions into account, to the best of our knowledge.

Clustering algorithms can be considered in four major categories: partitioning based methods (e.g., k-means [91]), density-based methods (e.g., DBSCAN [89]), hierarchical methods (e.g., BIRCH [94]), and grid-based methods (e.g., STING [93]). Once the aircraft trajectories are divided into flight phases, our *DICLERGE* framework uses k-means to perform clustering and merges them together.

One of the previous research, TRACLUS [90] proposes Partition-and-Group framework

that is similar to ours in a sense that it partitions the trajectories into subtrajectories. However, in their approach, subtrajectories are represented by line segments and grouped together using a distance function. Besides, unlike ours, their clustering algorithm is density based.

In their research, Gariel et al. [101] proposed a way point based trajectory clustering to monitor the airspace and its conformance. Although their approach was an efficient way to determine the compliance of flown trajectories with published trajectories, they focused on the 2-D coordinates of the way points in the (x,y) plane, disregarding the 3rd and higher dimensions of the trajectory points. Eckstein [103] presented an automated flight taxonomy. With their approach, they first resampled the trajectories and then clustered using k-means on a reduced order model, attained by Principal Component Analysis (PCA). Eckstein used only first two PCs, omitting the others.

Leiden et al. [102] presented two trajectory clustering algorithms for metroplex applications. The first algorithm they developed used a well-known image processing technique, ridge detection. One of the major shortcomings of their first algorithm was that it was unable to identify a backbone route where no ridge existed. Given the shortcomings of their first algorithm, they implemented a second algorithm that was based on k-means clustering. However, they only evaluated their results qualitatively. They never performed a quantitative evaluation using the model itself.

In their effort, Brinton et al. [100] investigated the automated partitioning of airspace into sectors so that sufficient resources could be allocated to meet the demand. They used Dynamic Density metrics and utilized a weighted euclidean distance function to address the constrained clustering problem. However, they never compared their results with a true optimal solution. So, the outcome appeared to be inconclusive.

In their study, Hansen et al. [104] perform aircraft trajectory clustering to consolidate

historical flight tracks into a small set, which were used as ground truth when predicting aircraft trajectory choice.

4.3 Aircraft Trajectory Clustering

In this section, we present an overview of our design. Section 4.3.1 formally describes the problem statement. Section 4.3.2 provides a skeleton of our trajectory clustering framework, *DICLERGE*. Section 4.3.3 elaborates on the AircraftTrajectoryClustering algorithm, which is a subtask of *DICLERGE*. Section 4.3.4 discusses the RepresentativeTrajectory algorithm, which is another subtask of *DICLERGE*, generating a representative trajectory.

4.3.1 Problem Statement

We implement an algorithm, *DICLERGE* that is based on the idea of divide, cluster, and merge. Given a set of multidimensional trajectory points pertaining to a particular flight, our framework executes a number of efficient and effective algorithms to generate a set of clusters as well as a representative trajectory. Aircraft trajectory, phases of flight, cluster and representative trajectory are defined as follows:

Definition 4.1: **Aircraft trajectory** An aircraft trajectory is a set of multidimensional points, where each point is defined by its 4-dimensions, latitude, longitude, altitude, and time. Additional dimensions such as airspeed and weather conditions may also be used.

Definition 4.2: **Phases of flight** Although there are at least five phases of flight which are taxi, climb, enroute, descent, and landing from the aviation standpoint, we will consider only three here as we are interested in wheels-up phases, only. These phases are climb, enroute, and descent. Climb phase refers to an aircraft's climbing to a certain altitude (typically 30,000 ft). Enroute is

the level portion of aircraft between climb and descent phases. A descent during flight is the phase where an aircraft decreases altitude to approach landing.

Definition 4.3: **Cluster** A cluster is a set of multidimensional points pertaining to aircraft trajectories with commonality that are grouped together.

Definition 4.4: **Representative trajectory** A representative trajectory is a set of cluster centroids, filtered and connected together to best represent the underlying trajectories. It is an imaginary trajectory for representation purposes only.

Figure 4.1 shows the overall procedure of aircraft trajectory clustering in the divide-cluster-merge framework. To better illustrate the flight phases, a vertical profile of a flight is used. First, a set of aircraft trajectories are divided into climb, enroute, and descent phases. Then, points per phase are clustered in isolation. Finally, the clusters and representative centroids are accumulated together, forming the entire set for the original trajectories.

4.3.2 DICLERGE Algorithm

Algorithm 4.1 illustrates the skeleton of our trajectory clustering framework, *DICLERGE*. It goes through three phases by executing a number of algorithms to perform divide, cluster, and merge subtasks. In addition, Representative Trajectory is generated by applying lateral and vertical smoothing processes. These algorithms are elaborated in the following sections.

4.3.3 Clustering Algorithm for Air Traffic Data

We now present our clustering algorithm for air traffic data. Given a set D of multidimensional trajectory points, our algorithm generates a set of clusters, defined by their centroids. The input parameters for the algorithm are multidimensional observation set and a number of clusters

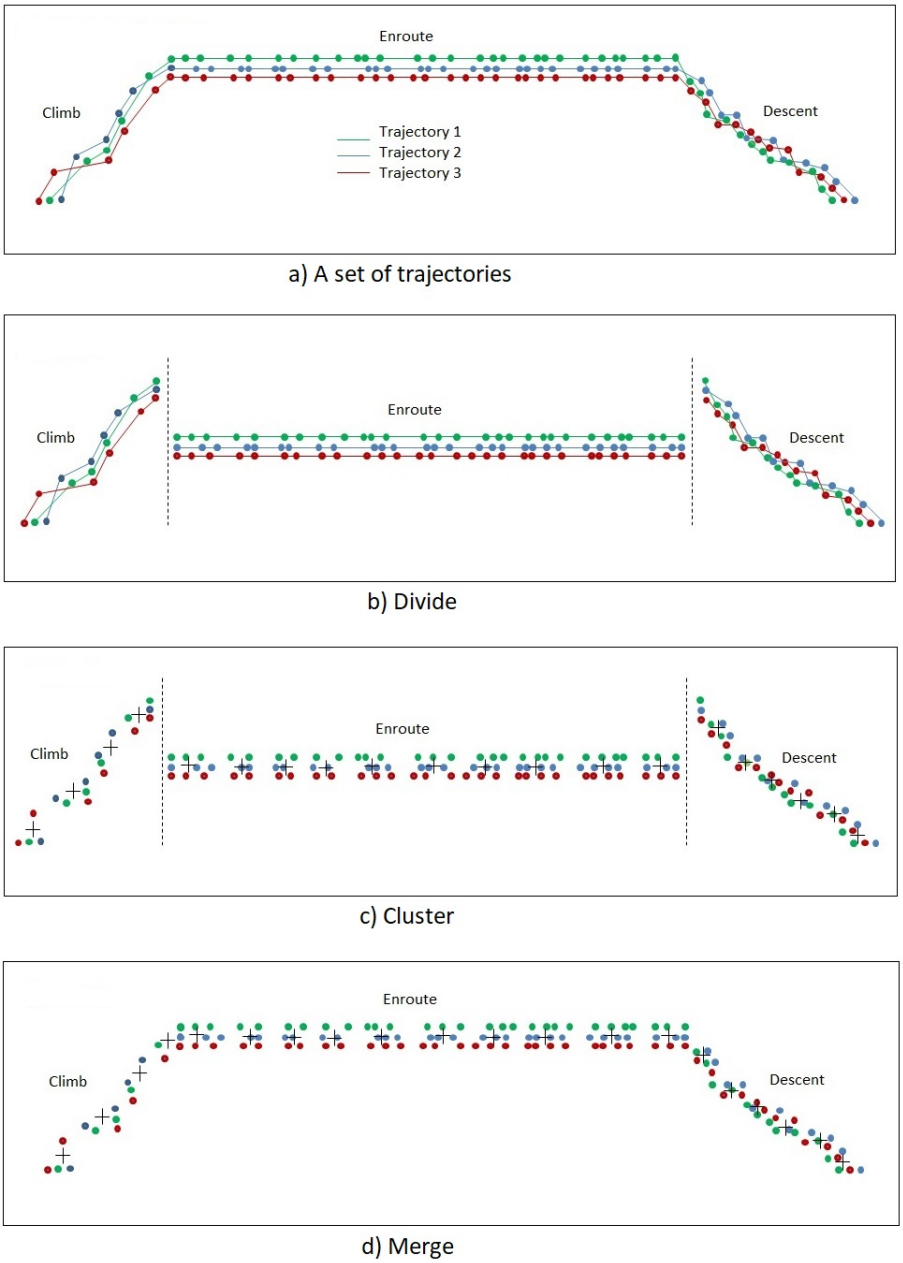


Fig. 4.1: The overall procedure of aircraft trajectory clustering using *DICLERGE* framework

Algorithm 4.1: DICLERGE (DIvideCLustermERGE)

Result: Representative trajectory RTR

Input : A set of multidimensional trajectory points D, number of clusters per flight phase $kClimb, kEnroute, kDescent$, max delta bearing $maxDBearing$

Output: A set of clusters O, a set of centroids CP, a representative trajectory RTR

/* Algorithm AircraftTrajectoryClustering */

/* DIVIDE */

1: **for each** $P \in D$ **do**

2: Divide D into a set of TR

3: Divide each trajectory tr into three phases

4: Accumulate each phase

5: **end for**

Get three sets of points

/* CLUSTER */

6: Perform k-means clustering on each set

Get sets of clusters per point set

Get a centroid per cluster

/* MERGE */

7: Accumulate sets of clusters, centroids

Get clusters, *O*

Get centroids, *CP*

/* Algorithm RepresentativeTrajectory */

8: Apply lateral smoothing

9: Apply vertical smoothing

Get a representative trajectory, *RTR*

per flight phase. Our algorithm is based on k-means clustering. However, instead of feeding in the entire dataset, the algorithm divides the input data into three chunks, each representing one of three flight phases and performs clustering on each chunk in isolation. The output clusters and representative centroids per chunk are then merged together.

4.3.4 Representative Trajectory

Our algorithm shares partitioning characteristic with the Mini Batch k-means algorithm [105]. However, unlike Mini Batch k-means, which randomly splits the dataset into smaller subsets, we divide the dataset with a well-known domain fact in mind; flight consists of three phases: climb, enroute and descent. Due to fact that aircraft's altitude increases at each observation during the climb, remains fixed during the enroute, and decreases during the descent phase, and that points per phase are spatially correlated, feeding the entire dataset at once to k-means results in overfitting, generating erroneous centroids.

Algorithm 4.2 shows AircraftTrajectoryClustering. Initially, all the points are assumed to be unclustered. As the algorithm progresses, clusters are formed and a centroid per cluster is computed. The algorithm consists of 3 steps: In the first step (lines 1∼7), the algorithm determines the first and last data recording for each trajectory by comparing its timestamp, as the timestamp of the first data point of the first trajectory will always be less than the timestamp of the first data point of the following trajectory (of the same flight which is usually scheduled for the next day). By splitting each sorted trajectory data in half, enroute altitude is determined, which is basically the median altitude of all trajectory points'. In the second step (lines 8∼20), the algorithm identifies the start and end indices of the enroute phase. This goal is attained by performing binary search on the sorted (by timestamp) list of data points. The leftmost occurrence of enroute altitude is the start, and the rightmost occurrence of enroute altitude is the end of the enroute phase of the flight.

Algorithm 4.2: AircraftTrajectoryClustering

Result: A set of cluster centroids

Input : A set of multidimensional trajectory points D, number of clusters per flight phase $kClimb$, $kEnroute$, $kDescent$

Output: A set of clusters O, a set of centroids CP

```
1  flight ← ∅, flightClimb ← ∅, flightEnroute ← ∅, flightDescent ← ∅
2  previousDate ← 01011001
3  p ← latitude, longitude, altitude, dateTime
4  foreach p ∈ D do
5      if p.date = previousDate then
6          previousDate ← p.date
7          if flight ≠ ∅ then
8              enrouteAltitude ← p(length(flight)/2).altitude
9              enrouteStart ← BinSearchLeftmost(flight, enrouteAltitude)
10             enrouteEnd ← BinSearchRightmost(flight, enrouteAltitude)
11             flightClimb.add(flight.slice(0, enrouteStart − 1))
12             flightEnroute.add(flight.slice(enrouteStart, enrouteEnd))
13             flightDescent.add(flight.slice(enrouteEnd, length(flight) − 1)
14             flight ← ∅
15             flight.add(p)
16         end
17         else
18             previousDate ← p.date
19             flight.add(p)
20         end
21     end
22 end
```

23 $centroidClimb \leftarrow kmeans(flightClimb, kClimb)$;

24 $centroidEnroute \leftarrow kmeans(flightEnroute, kEnroute)$;

25 $centroidDescent \leftarrow kmeans(flightDescent, kDescent)$;

26 $centroids \leftarrow centroidsClimb + centroidsEnroute + centroidsDescent$;

This enables the algorithm to compute the climb and descent phases of the flight, as index 0 to start of the enroute phase identifies the climb and the end of the enroute phase to the end of the flight identifies the descent phase. The algorithm retains a global array per phase, by adding the relevant data points per flight. This way, all data points for each phase are accumulated and stored in a relevant array. In the final step (lines 21∼24), the algorithm uses the array of data points along with a number of clusters per phase as input to the k-means clustering. The process fits the model using Euclidean distance and determines to which cluster each data point belongs. Upon convergence, the final set of centroids per flight phase are computed, which are merged to form the final list.

The representative trajectory of a cluster captures the overall movement of the aircraft based on trajectory data records. Once centroids are generated upon clustering, consecutive centroids are connected to model the behavior of the aircraft. This requires omitting the abnormal centroids while connecting the valid ones to most accurately represent the overall movement of the aircraft with a single trajectory.

Algorithm 4.3 generates the representative trajectory in two steps. During the first step, (lines 1∼18), the algorithm performs the lateral smoothing. This goal is attained by keeping a domain fact in mind that aircraft usually fly straight and keep the delta bearing at minimal when they make turns. In the second step, (lines 19∼23), the vertical smoothing is done. This step takes advantage of the aircraft trajectory property: aircraft usually maintain the cruise level at or above

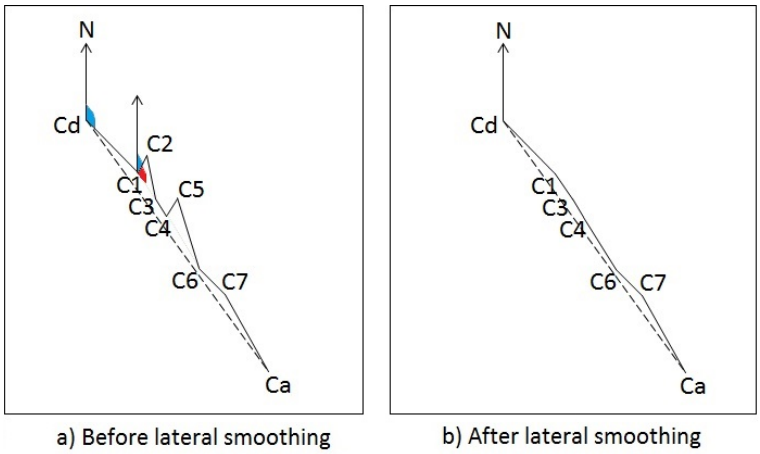


Fig. 4.2: Lateral smoothing

30,000 feet between the top of climb and the top of descent.

The input to the algorithm is a set of centroids sorted in the direction of departure to arrival airports. The default maximum delta bearing the aircraft is allowed to turn is a constant, *maxDBearing*, which is an input to the algorithm. The algorithm basically computes the bearing between two centroids at a time, as it traverses them. If the absolute value of the difference between the current and previous bearing is less than or equal to the maximum delta bearing, which is the lateral smoothing factor, the current centroid is appended to the representative trajectory. Otherwise, the current centroid is skipped and the next centroid is considered. The bearing is computed using the the *Haversine* equation [106] equation and the initial bearing is calculated using the latitude, longitude pairs of the departure and arrival airports.

The simplistic process is illustrated in Figure 4.2. The bearing between the departure (Cd) and arrival airports (Ca) is 154 degrees. The bearing between departure airport and the next centroid (C1) is 143 degrees. Given the delta bearing is 11 degrees, which is less than *maxDBearing*, 30 degrees, C1 is appended to the Representative Trajectory. The next centroid to be considered is C2. Hence, the algorithm computes the bearing between C1 and C2. Since the bearing between

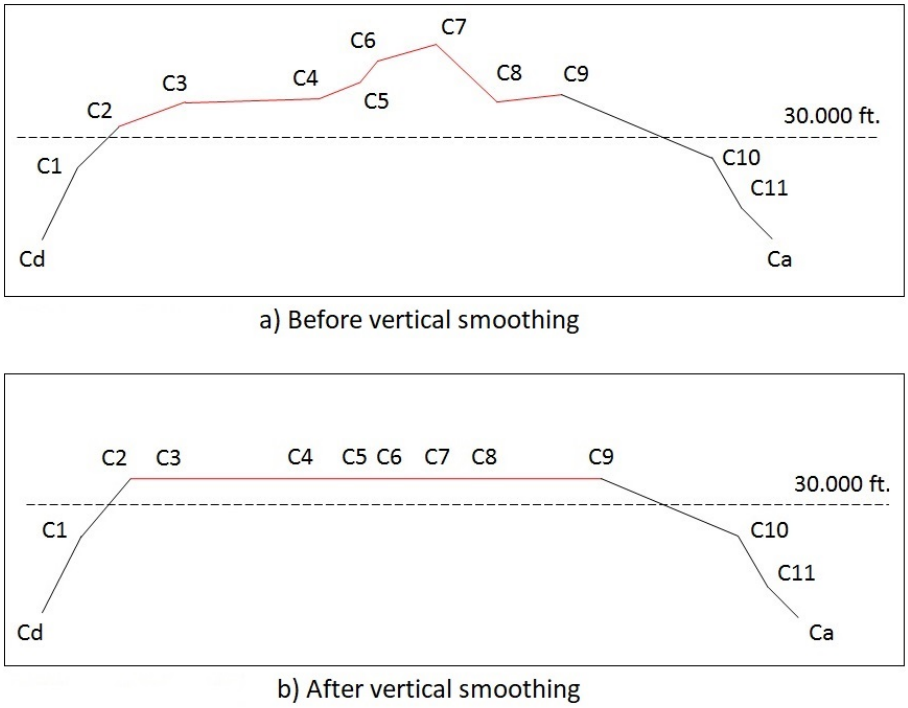


Fig. 4.3: Vertical smoothing

the two centroids is 65 degrees, the delta bearing is 143-65=78 degrees, which is greater than the maximum delta bearing. Therefore, the algorithm skips C2 and computes the bearing between C1 and C3. Due to fact that the bearing is 157 degrees, which means that the delta bearing is 157-143=14 degrees, the third centroid is appended to the Representative Trajectory. The process stops when the arrival airport is reached. The resulting set of centroids forms the lateral profile of the Representative Trajectory.

Vertical smoothing is done in the next step. The sorted centroids with regards to their lateral profile in the direction of departure to arrival airport are fed into a binary search function, where the function searches for the leftmost occurrence of altitude that is greater than or equal to 30,000 feet. The index of the leftmost occurrence is stored. The rightmost index is also searched and stored the same way. All centroids in between these two indices are sliced, forming the enroute phase of the Representative Trajectory. The mean value for their altitudes is computed. The mean altitudes

Algorithm 4.3: RepresentativeTrajectory

Result: A representative trajectory

Input : A set of centroids, each representing a cluster CP, max delta bearing $maxDBearing$

Output: A representative trajectory RTR

$initial \leftarrow True, repTrajectory \leftarrow \emptyset, maxDeltaLateral \leftarrow 30.0, enrouteAltitude \leftarrow 30000, prevPos \leftarrow 0.0, 0.0, 0.0, prevBearing \leftarrow haversine(deptAirportLon, deptAirportLat, arrAirportLon, arrAirportLat)$

foreach $ct \in CP$ **do**

  if $initial$ **then**

    $prevPos \leftarrow ct.latitude, ct.longitude, ct.altitude$

    $repTrajectory.append(prevPos)$

    $initial \leftarrow False$

    $continue$

  end

  $bearing \leftarrow haversine(prevPos.lon, prevPos.lat, ct.longitude, ct.latitude)$

  if $abs(bearing - prevBearing) \leq maxDBearing$ **then**

    $prevBearing \leftarrow bearing$

    $prevPos \leftarrow ct.latitude, ct.longitude, ct.altitude$

    if $prevPos \notin repTrajectory$ **then**

      $repTrajectory.append(prevPos)$

    end

  end

end

18 $enrouteStart \leftarrow BinSearchLeftmostGT(repTrajectory, enrouteAltitude)$;

19 $enrouteEnd \leftarrow BinSearchRightmostGT(repTrajectory, enrouteAltitude)$;

20 $avgEnrouteAltitude \leftarrow$

$avg(repTrajectory.slice(enrouteStart, enrouteEnd).altitude)$;

21 $setAlt(repTrajectory.slice(enrouteStart, enrouteEnd)) \leftarrow$

$averageEnrouteAltitude$;

22 $repTrajectory \leftarrow repTrajectory.slice(0, enrouteStart - 1) +$

$repTrajectory.slice(enrouteStart, enrouteEnd) +$

$repTrajectory.slice(enrouteEnd, length(flight) - 1)$;

for each of the enroute centroids are replaced with the original altitudes. This process completes the vertical smoothing and yields the final Representative Trajectory. Figure 4.3 captures the before and after view of vertical smoothing process in a simplistic way. The left figure shows ever changing cruise altitude in the form of a rugged line in the vertical profile. In the right figure, the cruise altitude is fixed, presenting a realistic vertical profile.

4.4 Experimental Study

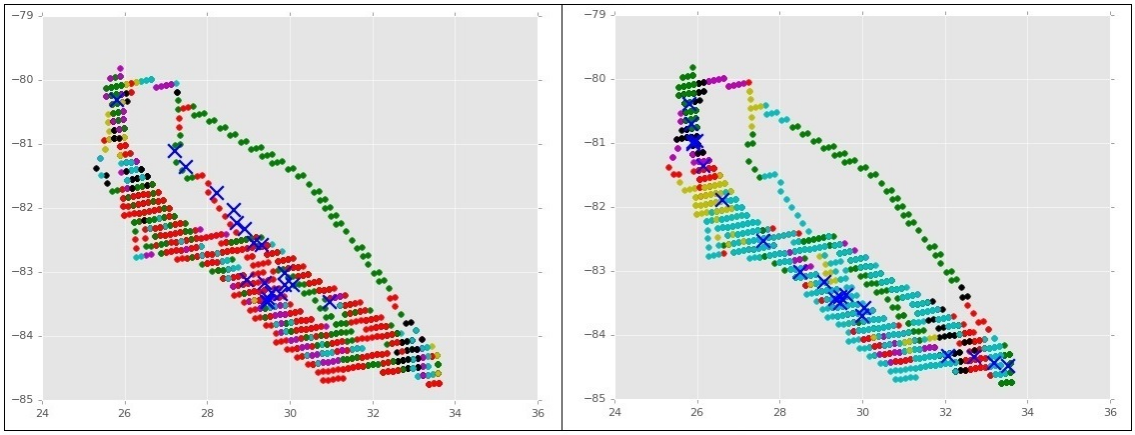
In this section, we conduct experiments to validate the effectiveness of our aircraft trajectory clustering algorithm, *DICLERGE*. We describe the experimental dataset preparation and environment in Section 4.4.1. We discuss the results for the clustering process as well as the effects of parameter values in Section 4.4.2.

4.4.1 Dataset Preparation

We used a real aircraft trajectory dataset: The Delta Airlines' flight DAL1865, departing from Hartsfield-Jackson Atlanta International Airport (KATL) and arriving at Miami International Airport (KMIA) for the duration of January through June 2015. The dataset has a total of 168 trajectories and 25036 points. ATL to MIA is one of the major routes in the NAS due to fact that the departure airport is the busiest airport in the United States and the flights are prone to frequent convective weather in the airspace controlled by Miami Air Route Traffic Control Center (ARTCC).

The source of the surveillance data is Aircraft Situation Display to Industry (ASDI) [107] provided in near real-time by the FAA. The surveillance part of the data was generated by merging a subset of various ASDI message types including: departure information, track information, and flight management information. In order to fuse weather parameters with surveillance data during the scheduled hours, we also obtained weather data. The source of the weather data was National Oceanic and Atmospheric Administration (NOAA) National Centers for Environmental Prediction (NCEP) Rapid Refresh product [108], an hourly-updated modeling system operational at NCEP. This way, each trajectory point for flight DAL1865 was associated with weather parameters yielding the following attributes per trajectory point: source center, source date, source time, aircraft id, aircraft speed, latitude, longitude, altitude, temperature, wind speed, and wind direction.

Our experiments were conducted on a Oracle VM VirtualBox v4.3.20 virtual machine running on Linux Ubuntu 14.04.2 64-bit LTS Operating System hosted by a computer with Intel Core i7-3840QM CPU @ 2.80GHz and 16GB memory, running on Microsoft Windows 7 Operating System. All the algorithms in our system were implemented in Python v2.7.



a) k-means clustering of aircraft trajectories, k=20

b) k-means clustering of aircraft trajectories using *DICLERGE* framework, k=20

Fig. 4.4: Clustering of aircraft trajectories with and without *DICLERGE* framework

$s_{regular}$	s_{climb}	$s_{enroute}$	$s_{descent}$
0.678	0.557	0.702	0.565

Tab. 4.1: Silhouette Coefficients

4.4.2 Results for the Aircraft Trajectory Data

Our assessment included running k-means clustering with and without the *DICLERGE* framework on the DAL1865 trajectory data. For the clustering process, we used k=20 for both executions. Our attempt was to measure the clustering results quantitatively and qualitatively for each execution.

Unfortunately, there is no well-defined measure for quantitative evaluation due to fact that the ground truth labels are not known. Hence, we performed the evaluation using the model, itself. The *Silhouette Coefficient* is a way of such an evaluation, where a higher coefficient score relates to a model with better defined clusters [109].

We computed the coefficient for regular k-means clustering ($s_{regular}$). We also computed

Fig. 4.5: Representative trajectory

the coefficients for the *DICLERGE* framework on climb (s_{climb}), enroute ($s_{enroute}$), and descent ($s_{descent}$) phases. Table 4.1 indicated that the clustering quality of regular k-means was higher than the *DICLERGE* framework's climb and descent phases. However, it was lower than the *DICLERGE* framework's enroute phase.

For the qualitative measure, we used visual inspection and domain knowledge. The outcome of the execution without the *DICLERGE* framework is captured on Figure 4.4a. This execution treated the entire trajectory as a whole, generating centroids concentrated in the center. Whereas *DICLERGE* framework divided the trajectories into 3 phases first, clustered each set in isolation and merged them together at the end. For the *k*=20, we used *kClimb*=4, *kEnroute*=10, and *kDescent*=6 as the optimal parameters. The outcome of this execution is illustrated on Figure 4.4b.

It is obvious that the Figure 4.4 on the right presents a better clustering as the centroids follow a more linear pattern from departure to arrival airport, lining up toward the bottom of the trajectory set where the trajectory points are denser.

To generate representative trajectory, we ran our algorithm and obtained the result illustrated

in Figure 4.5. The representative trajectory shown with red on Google Earth [110] is overlaid on top of the trajectory set, providing a model for the movement of the flight over the period of 6 months.

The heuristic in selecting the number of clusters per flight phase depends on the number of trajectory points per phase. We maintain the ratio of the number of trajectory points in each phase to the total trajectory points and map that to the pertinent parameters, *kClimb*, *kEnroute*, and *kDescent*. Our heuristic for selecting the lateral smoothing parameter is based on the level of smoothing we would like to perform. Due to fact that our trajectory is prone to convective weather, which means that aircraft may perform drastic turns to avoid the convection, we set our lateral smoothing to a relatively larger value which was 30 degrees.

4.5 Conclusion

This chapter presented the *DICLERGE* framework for clustering multidimensional aircraft trajectory data and discovering the representative trajectory upon lateral and vertical smoothing. Our experiments with the real trajectory dataset showed the effectiveness and efficiency of *DICLERGE* framework and its advantages over the regular k-means clustering for the aircraft trajectory data.

Note that this study is just a milestone for trajectory prediction and we plan to utilize our work toward predictive and prescriptive analytics of multi-dimensional aircraft trajectory data.

Chapter 5

Time Series Clustering

Reliable trajectory prediction is paramount in Air Traffic Management (ATM) as it can increase safety, capacity, and efficiency, and lead to commensurate fuel savings and emission reductions, minimizing the environmental impact of aviation. Inherent inaccuracies in forecasting winds and temperatures often result in large prediction errors especially when a deterministic approach is used. A stochastic approach can address the aircraft trajectory prediction problem by taking environmental uncertainties into account and exploiting machine learning techniques to train a model using historical trajectory data along with weather observations. With this approach, weather observations are assumed to be realizations of hidden aircraft positions that is trajectory segments and the transitions between the underlying hidden segments follow a Markov model. However, this approach requires input observations, which are unknown, although the weather parameters overall are known for the pertinent airspace. We address this problem by performing time series clustering on the current weather observations for the relevant sections of the airspace.

In this chapter, we present a novel time series clustering algorithm that generates an optimal sequence of weather observations used for accurate trajectory prediction in the climb phase of the flight. Our algorithm computes a cost value for each weather observation based on its frequency

per time period, ranks, and stores them in a matrix. Using *Dynamic Programming (DP)*, the algorithm computes the optimal sequence of weather observations, a set with the minimum cumulative cost that satisfies the *continuity* constraint. Our experiments use a real trajectory dataset with pertinent weather observations and demonstrate the effectiveness of our algorithm over time series clustering with a *k-Nearest Neighbors (k-NN)* algorithm that uses *Dynamic Time Warping (DTW)* Euclidean distance.

5.1 Introduction

Trajectories are usually discussed for cars along roads [64] with an emphasis on queries (e.g., [66, 67, 72, 73]), Here we are interested in aircraft trajectory prediction which is a crucial process for Air Traffic Control (ATC) as the more accurate and reliable the predicted trajectories the more safe and efficient the deconfliction of flights. In order for ATC to attain this goal, the trajectory prediction tools should be able to provide high levels of accuracy for an adequate horizon of time. Although prior trajectory prediction research and development activities have been able to meet the challenge to some extent, dealing with congested airspace and environmental factors still remains the challenge when a deterministic approach is used in trajectory prediction process [111–115]. Hence due to the uncertainties contributing to trajectory prediction errors, we take a stochastic approach to address these issues. The past efforts addressing aircraft trajectory prediction using probabilistic methods are similar to our approach in a way that they aim at modeling uncertainties to describe potential changes in the future trajectory of an aircraft. However, many of them [116–121] either lack empirical validation or use a simulated set of aircraft trajectories instead of actual track data in their evaluations.

We define airspace as a set of spatio-temporal cuboids where each cuboid is considered as an atomic unit. Weather parameters such as temperature, wind speed, wind direction, and humid-

ity considered homogeneous within the cuboid during a period of time. Other parameters that describe the cuboid include its center coordinates *(latitude, longitude, altitude)* along with a time stamp. These spatio-temporal cuboids form the overall airspace. Next, we adapt historic raw trajectories to the cuboids. Using adapted trajectories in the form of 4D joint cuboids, we train our model with the historical data and choose a state sequence that best explains the current weather observations. Due to the nature of interconnected cuboid centroids forming a trellis, we use the Viterbi algorithm [122] to efficiently generate the optimal trajectory by joining the multiple segments together, where one segment is only dependent on the previous segment. However, the Viterbi algorithm requires input observations, which are unknown in our problem space, although the weather parameters overall are known for the pertinent airspace. One way to address this problem is by performing clustering on the current weather observations that yields the input observations. However, standard clustering techniques generate clusters that are largely independent of the time series from which they originate [123].

We use time series clustering approach to this problem. We basically perform time series clustering on the current weather observations for the cuboid centroids that were traversed by historic trajectories. Before the process, we split the current weather observations into buckets and then execute our algorithm. Our time series clustering algorithm computes a cost value for each weather observation based on its frequency per time period, and ranks them. The process yields a matrix, where each matrix element corresponds to a weather observation with its cost. Using *DP*, our algorithm computes the optimal sequence of weather observations, a set with the minimum cumulative cost that satisfies the *continuity* constraint. We finally feed the output cluster into the Viterbi process to probabilistically generate the best sequence of trajectory segments.

Our experiments use a real trajectory dataset with pertinent weather observations and demonstrate that our algorithm outperforms time series clustering with a *k-Nearest Neighbors (k-NN)*

algorithm that uses *DTW* Euclidean distance [77]. The results suggest that our algorithm can be of significant value for predicting aircraft trajectories even during the climb phase of the flight. Trajectory prediction for the climb and descent phases of flight is difficult because of the uncertainty of rates of climb and descent due to the effects of outside air temperature on engine power and therefore climb/descent rates. Hence accurate trajectory prediction of climb phase is more challenging than the cruise phase of the flight due to fact that the wind in the vertical direction varies more and the length of time in the varying winds is difficult to forecast. Also, parameters such as mass and flight management system settings have a significant impact on vertical motion but little on the horizontal.

In summary, the contributions of this study are as follows:

- We propose a novel time series clustering algorithm that is based on a cost value computed by the frequency of weather observations at each time period. Next, we feed the series of cluster centroids as observations into our aircraft trajectory prediction system that is based on a stochastic model, HMM to predict trajectories for the climb phase of a flight.

- We conduct experiments based on actual track data and weather observations and demonstrate that our time series clustering algorithm outperforms time series clustering with *k-NN* algorithm that uses *DTW* Euclidean distance by yielding cluster centroids that result in more accurate trajectory predictions for the climb phase of a flight.

The rest of the chapter is organized as follows. Section 5.2 presents generic time series clustering and discusses our algorithm. Section 5.3 details experiments by comparing our time series algorithm against the state-of-the-art *k-NN* algorithm that uses *DTW* Euclidean distance. Section 5.4 concludes the chapter.

5.2 Time Series Clustering

The time series clustering aids in the trajectory prediction process. Section 5.2.1 presents the background in time series clustering and Section 5.2.2 discusses our algorithm in detail.

5.2.1 Background

Clustering is the general problem of partitioning a set of observations into a number of homogeneous groups where the similarity is minimized within the group and the dissimilarity is maximized between the groups [124]. Han et al. classified clustering methods into five categories: partitioning, hierarchical, density-based, grid-based, and model-based methods [125].

A partitioning method generates k partitions from n unlabeled data tuples, where $k \leq n$. Major heuristics in partitioning methods are *k-means* [126], *k-medoids* [127], *fuzzy c-means* [128], and *fuzzy c-medoids* [129]. A hierarchical method groups data into a tree of clusters. There are two major hierarchical clustering methods: agglomerative and divisive. Chameleon [130], CURE [131], and BIRCH [132] are part of the hierarchical methods. Density-based methods keep growing a cluster until the density in the neighborhood reaches some threshold. DBSCAN [133] and OPTICS [134] belong to density-based methods. Grid-based methods divide the object space into a set of grid cells on which the clustering operations are performed. STING [135] is part of the grid-based clustering methods. Model-based clustering methods assume a model for each of the clusters and attempt to best fit the data to the model. There are two major model-based methods: statistical and neural network. AutoClass [136], ART [137], and self-organizing maps [138] are three approaches to the model-based clustering methods.

A time series is a sequence of real numbers collected at regular intervals over a period of time. Time series clustering is a critical analysis technique which can be used as a preprocessing

step for further data mining and it mostly relies on classic clustering methods, either by replacing the default distance measure with an alternative one or by transforming time series into static data so that classic clustering methods can be directly used [124]. Liao et al. [124] classified time series clustering approaches into three major categories: *i)* raw-based [139–141], *ii)* feature-based [142–144], and *iii)* model-based [145–147] approaches, depending on whether they work directly with raw data, indirectly with features extracted from the raw data, or indirectly with models built from the raw data. In this study, we follow a raw-based approach.

Although many clustering criteria to identify the similarity and dissimilarity have been proposed, the minimum within the cluster sum of squared distances is the most commonly used measure [148]. To represent a cluster, a centroid is used that is defined as the data point that minimizes the sum of squared distances to all other points. Hence, using the distance measure is critical in computing centroids. Finding such a centroid is known as the Steiner's sequence [149]. The centroid is computed with the arithmetic mean property, when Euclidean distance is used [150]. In the realm of time series clustering, *DTW* is the most widely used measure to compare time-series sequences with alignment [140]. Hence, we evaluate our time series clustering algorithm by comparing it against the state-of-the-art *k-NN* algorithm that uses *DTW* Euclidean distance. Unfortunately, using an arithmetic mean to find a single representative data point for each time interval along the time series often doesn't generate an accurate centroid as we present in Section 5.3.

5.2.2 Time Series Clustering Algorithm

In this section, we propose a novel time series clustering algorithm for multivariate weather observations of the same length to aid in the aircraft trajectory prediction process.

Suppose we have a time series of weather observations between departure and arrival air-

ports for the same flight over a period of time. We can represent this time series with an n x m matrix, where each row contains a set of periodic weather observations recorded along the aircraft's trajectory as the flight progresses. As the flight takes place the next time, a new row is added to the matrix. Assuming that the flight occurs once a day, each matrix cell corresponds to an observation recorded at a particular time interval during the flight in a particular day. Note that each observation is composed of the major weather parameters affecting the aircraft trajectory, that is *temperature, wind speed, wind direction,* and *humidity*. Given those observations, we want to trace an optimal weather observation path that best represents the underlying observation set. This objective translates to finding a single representative centroid for each time interval along the time series and optimally concatenating them as we move forward.

Formally, given n observation sets, each with length of m, $O_1 = [o_{1,1}, o_{1,2}, ..., o_{1,m}]$, $O_2 = [o_{2,1}, o_{2,2}, ..., o_{2,m}]$, ... $O_n = [o_{n,1}, o_{n,2}, ..., o_{n,m}]$, we want to come up with the optimal observation path, $OOP = [o_{i,1}, o_{j,2}, ..., o_{k,m}]$, which is subject to *continuity* constraint that is defined as:

Continuity: Given $o_{i,j} = (a, b)$, then $o_{i,j-1} = (a', b')$, where $a - a' \geq 0$ and $b - b' \geq 0$. This restricts the allowable steps in the observation path to adjacent cells (including diagonal adjacent cells).

There are exponentially many observation paths that satisfy the *continuity* constraint. However, we are only interested in the path, OOP that minimizes the cumulative cost denoted by

$$OOP = \arg\min_{c} \sum_{i=1}^{m} c_i$$

where cost of a cell $c_{i,j}$ is computed based on the pertinent observation's frequency at that time period.

To solve the optimization problem, we perform preprocessing by computing: i) Column-wise observation counts Coc, and column-wise maximum observation counts $Cmoc$, ii) A costs

Algorithm 5.1: OBSERVATIONPATHS

Input: observation matrix Om

Output: observationPaths op

1: $Coc \leftarrow \{\}$, $Cmoc \leftarrow \{\}$, $Cm \leftarrow \{\}$, $Pm \leftarrow \{\}$ // columnwise observation counts, columnwise max observation counts, costs matrix, and pointers matrix, respectively

/* STEP 1: Compute counts */

2: **for** $c \in Om.cols$ **do**

3: $Coc.o \leftarrow$ COUNT($o \in c$)

4: **end for**

5: **for** $c \in Om.cols$ **do**

6: $Cmoc.c \leftarrow$ MAX(Coc)

7: **end for**

/* STEP 2: Build pointers and costs matrices */

8: **while** $c \neq 0$ ($c \leftarrow |Om.cols|$-1) **do**

9: **for** $r \in Om.rows$ **do**

10: Cm[r][c] $\leftarrow$ $Cmoc$[c] - COUNT[Om[c]]

11: **if** $c \neq |c_{n-1}|$ **then**

12: **if** r = 0 **then**

13: $mn \leftarrow$ MIN(Cm[r][c+1], Cm[r+1][c+1])

14: Cm[r][c] $\leftarrow$ Cm[r][c] + mn

15: **if** mn = Cm[r][c+1] **then**

16: Pm[r][c].concat(r, Om[r][c+1])

17: **end if**

```
18:     if mn = Cm[r+1][c+1] then
19:         Pm[r][c].concat(r+1, Om[r+1][c+1])
20:     end if
21: else if r = |Om.rows|-1 then
22:     mn ← MIN(Cm[r-1][c+1], Cm[r][c+1])
23:     Cm[r][c] ← Cm[r][c] + mn
24:     if mn = Cm[r-1][c+1] then
25:         Pm[r][c].concat(r-1, Om[r-1][c+1])
26:     end if
27:     if mn = Cm[r][c+1] then
28:         Pm[r][c].concat(r, Om[r][c+1])
29:     end if
30: else
31:     mn ← MIN(Cm[r-1][c+1], Cm[r][c+1], Cm[r+1][c+1])
32:     Cm[r][c] ← Cm[r][c] + mn
33:     if mn = Cm[r-1][c+1] then
34:         Pm[r][c].concat(r-1, Om[r-1][c+1])
35:     end if
36:     if mn = Cm[r][c+1] then
37:         Pm[r][c].concat(r, Om[r][c+1])
38:     end if
```

```
            if mn = Cm[r+1][c+1] then
                Pm[r][c].concat(r+1, Om[r+1][c+1])
            end if
        end if
    end if
  end for
  c ← c-1
end while
/* STEP3: Compute observation paths */
procedure ObsPathsDp(psf, r, c, cc)
  Procedure is detailed in the next algorithm
end procedure
ic ← s ∈ MIN(Cm.cols[0])
for ic ∈ ic[0] do
  p ← [Om[ic][0]]
  ObsPathsDp(p, ic, 0, Cm[ic][0])
end for
```

Algorithm 5.2: OBSERVATIONPATHS DP

/* STEP3: Compute observation paths */

1: **procedure ObsPathsDp**(psf, r, c, cc)

Input: pathSoFar psf, row r, colmn c, cumulativeCount cc

Output: observationPaths op

2: **if** c = $|Om.cols|$-1 **then**

3: **return** $op \leftarrow psf$, (c + $Cm[r][c]$)

4: **end if**

5: **for** $p \in P_m[r][c]$ **do**

6: psf.concat(p[1])

7: **ObsPathsDp**(psf, p[0], c+1, cc + $Cm[r][c]$)

8: psf.pop(-1)

9: **end for**

10: **end procedure**

matrix Cm of size n x m where each matrix cell corresponds to a cost value, and a pointers matrix Pm of size n x m where each matrix cell corresponds to the adjacent observation (including diagonally adjacent), the original observation can connect to. Next, we perform dynamic programming in reverse order *i.e. beginning with the last column* to iteratively select the next weather observation that satisfies both conditions, *continuity* and *minimum cumulative cost* at each time period of the series.

Algorithm 5.1 illustrates the preprocessing steps in lines 1 through 46 such that:

- For each column c, count the number of occurrences $o(i, c)$ of each instance i in the column, and find the instance i_max occurring the maximum number of times $o(i\_max, c)$ in the column,
- For each cell (r, c) in column c, let it contain instance i. Then, compute the cost $Cm(r, c)$ as $o(i\_max, c) - o(i, c)$.
- Starting from the last but one column, for each row r in each column c, add to the cost Cm $P(r, c)$, the minimum value of $P(r-1, c+1), P(r, c+1), P(r+1, c+1)$. Store a pointer in cell (r, c) to the minimum of three cells.

In lines 49 through 53, Algorithm 5.1 iteratively calls the OBSPATHSDP procedure detailed in Algorithm 5.2 to dynamically find the next optimal sequence such that:

- Find all paths by following the pointers, starting from the cell in the first column with minimum cost, and ending in the last column.

To put in a better context, Figure 5.1 illustrates two time series of weather observations. Clustering the time series on the left is relatively straight forward as the matrix cell with the minimum cost per time period always connects to the next matrix cell with the minimum cost until

	Minute 1	Minute 2	Minute 3	Minute 4	Minute 5	Minute 6	Minute 7	Minute 8
Day 1	Obs. 1	Obs. 1	Obs. 2	Obs. 3	Obs. 4	Obs. 1	Obs. 3	Obs. 2
Day 2	Obs. 2	Obs. 1	Obs. 2	Obs. 3	Obs. 3	Obs. 2	Obs. 3	Obs. 2
Day 3	Obs. 3	Obs. 4	Obs. 4	Obs. 4	Obs. 3	Obs. 4	Obs. 3	Obs. 2
Day 4	Obs. 1	Obs. 1	Obs. 1	Obs. 2	Obs. 3	Obs. 4	Obs. 3	Obs. 2
Day 5	Obs. 1	Obs. 2	Obs. 2	Obs. 3	Obs. 4	Obs. 4	Obs. 3	Obs. 3

a) Observation set 1

	Minute 1	Minute 2	Minute 3	Minute 4	Minute 5	Minute 6	Minute 7	Minute 8
Day 1	Obs. 1	Obs. 1	Obs. 3	Obs. 3	Obs. 4	Obs. 1	Obs. 3	Obs. 2
Day 2	Obs. 3	Obs. 3	Obs. 1	Obs. 3	Obs. 3	Obs. 2	Obs. 3	Obs. 2
Day 3	Obs. 4	Obs. 4	Obs. 2	Obs. 3	Obs. 3	Obs. 4	Obs. 3	Obs. 2
Day 4	Obs. 2	Obs. 2	Obs. 4	Obs. 4	Obs. 3	Obs. 4	Obs. 3	Obs. 2
Day 5	Obs. 1	Obs. 1	Obs. 2	Obs. 1	Obs. 4	Obs. 4	Obs. 3	Obs. 3

b) Observation set 2

Fig. 5.1: Observation sets.

the algorithm reaches the end. Hence there are many observation paths that satisfy both conditions, resulting in 0 cumulative cost. The optimal observation path is formed by the observation sequence of [*Obs.1, Obs.1, Obs.2, Obs.3, Obs.3, Obs.4, Obs.3, Obs.2*]. Clustering the time series on the right is less straightforward as the matrix cell with the minimum cost per time period connects to the next matrix cell with the minimum cost sequentially in the last-to-first column order until the algorithm reaches *Minute 3*. Since the last observation in the sequence *(Day 3,Minute 3)* doesn't connect to neither of *(Day 1,Minute 2)* and *(Day 5,Minute 2)* which are the matrix cells with the minimum cost, the algorithm relaxes its *cumulative cost* constraint to meet the *continuity* constraint, and selects one of the *Obs.2, Obs.3* for *Minute 2* time period. Hence, the optimal observation paths that satisfy both *continuity* and *minimum cumulative cost* conditions are [*Obs.1, Obs.2, Obs.2, Obs.3, Obs.3, Obs.4, Obs.3, Obs.2*] and [*Obs.1, Obs.3, Obs.2, Obs.3, Obs.3, Obs.4, Obs.3, Obs.2*]. Note that the observations with the minimum cost are illustrated with solid cells in Figure 5.1.

Feeding the output cluster centroids into the Viterbi algorithm finalizes the process to prob-

abilistically generate the best sequence of trajectory segments.

5.3 Experiments

In this section, we describe the experiments performed using our algorithm versus *k-NN* clustering with *DTW* on the weather observations along the time series. To validate the effectiveness, we conduct test cases and feed the output from both algorithms into the Viterbi process to predict climb phase of aircraft trajectories.

Our experiments used real trajectory and weather data: The Delta Airlines' flights DAL2173, departing from Hartsfield-Jackson Atlanta International Airport (ATL) and arriving at Miami International Airport (MIA) for the period of May 2010 through December 2015 were studied. The dataset has a total of 1624 trajectories and 183797 points. Figure 6.3 shows 3D raw trajectories of flight DAL2173 between Atlanta and Miami for the period of May 2010 through December 2015.

The main data sources for our Aircraft Trajectory Prediction System are the FAA's Aircraft Situation Display to Industry (ASDI) and NOAA's RAP data, detailed in [46]. Before training data processing, time series data interpolation and filtering processes took place that reduced the number of historical recordings from 183797 to 137689. In the training data processing, we split the weather parameters into buckets as shown in Table 5.1, 5.2, and 5.3. Next, the following HMM parameters were computed:

- 7292 distinct states, S were generated.
- A sparse transition matrix, A of size 7292 x 7292 = 53173264 was generated.
- An emission matrix, B of size 495 x 7292 = 3609540 was generated.
- An initial matrix, π of size 38 x 1 was generated.

Fig. 5.2: 3D raw trajectories of flight DAL2173 between Atlanta and Miami for the period of May 2010 through December 2015 in Google Earth. Climb phase is colored white, cruise and descent phases are colored gray.

To evaluate our prediction system, we performed bootstrapping by drawing many trajectory samples with replacement from the historical trajectories. The trajectory sample for May 18, 2015 was chosen at random to be used as an example of the process for trajectory samples.

In test data processing, we performed time series clustering, using *DTW* and our algorithm. The input to the process was 1206 time series of 78 weather observations, where each observation contained *temperature, wind speed, wind direction*, and *humidity* parameters. Our first round evaluation tested the *k-NN* clustering with *k*=1 that used *DTW* Euclidean distance for the similarity measure. The resulting set of cluster centroids for the first round evaluation identified by weather parameters defined the first observation sequence Y_{s1}. The second round evaluation tested our time series clustering algorithm with the identical input. This defined the second observation sequence Y_{s2}.

As the final step of the process, we fed the first and second observation sequences, Y_{s1}, Y_{s2}, respectively along with pertinent HMM parameters into the Viterbi algorithm, one at a time. The algorithm returned the optimal state sequence with the maximum probability per observation sequence.

Our experiments were conducted on an Oracle VM VirtualBox v4.3.20 virtual machine running on Linux Ubuntu 14.04.2 64-bit LTS Operating System hosted by a computer with Intel Core i7-3840QM CPU @ 2.80GHz and 16GB memory, running on Microsoft Windows 7 Operating System. All the algorithms in our system were implemented in Python v2.7.

5.3.1 Results

Our evaluation was based on bootstrapping by drawing 7 trajectory samples with replacement from the historical trajectories. This way, we performed two series of comparisons:

- We compared the cluster centroids generated by both time series algorithms against the

Tab. 5.1: Buckets for temperature

Temperature ($temp$)		
No	**Bucket**	**Value** ($kelvin$)
1	$temp \leq 220$	220
2	$220 < temp \leq 240$	240
3	$240 < temp \leq 260$	260
4	$260 < temp \leq 280$	280
5	$280 < temp \leq 300$	300
6	$300 < temp \leq 350$	350

Tab. 5.2: Buckets for wind speed

Wind speed (ws)		
No	**Bucket**	**Value** ($knots$)
1	$ws \leq 30$	30
2	$30 < ws \leq 60$	60
3	$60 < ws \leq 90$	90
4	$90 < ws \leq 120$	120
5	$120 < ws \leq 150$	150

Tab. 5.3: Buckets for wind direction

Wind direction (wd)		
No	**Bucket**	**Value** ($degrees$)
1	$wd \leq 45$	45
2	$45 < wd \leq 90$	90
3	$90 < wd \leq 135$	135
4	$135 < wd \leq 180$	180
5	$180 < wd \leq 225$	225
6	$225 < wd \leq 270$	270
7	$270 < wd \leq 315$	315
8	$315 < wd \leq 360$	360

Tab. 5.4: Buckets for humidity

Humidity ($hmdty$)		
No	**Bucket**	**Value** ($percent$)
1	$hmdty \leq 20$	20
2	$20 < hmdty \leq 40$	40
3	$40 < hmdty \leq 60$	60
4	$60 < hmdty \leq 80$	80
5	$80 < hmdty \leq 100$	100

ground truth, the weather observations along the sample trajectories.

- We compared climb phases of the predicted trajectories aided by both algorithms against the ground truth, flight DAL2173's aligned sample trajectories.

Then, we rank ordered the means to estimate the 2.5 and 97.5 percentile values for 95% CI.

Time series clustering with *k-NN* algorithm that uses *DTW* Euclidean distance correctly found 34 centroids on the average out of full 78 original centroids formed by weather observations *temperature, wind speed, wind direction,* and *humidity* of ground truth. This corresponds to 43.6% accuracy. Time series clustering with our algorithm correctly found 55 centroids on the average out of full 78 original centroids formed by weather observations of ground truth, which corresponds to 70.5% accuracy. Hence, our time series clustering algorithm outperformed time series clustering with a *k-NN* algorithm that uses *DTW* Euclidean distance by 61.8%, in accuracy.

Figure 6.4 shows a lateral view of the actual aligned trajectories in white overlaid on top of spatio-temporal cuboids for flight DAL2173's climb phase on May 18, 2015, generated by our trajectory prediction system. On the left, cuboids in red were generated upon time series clustering of weather observations using *k-NN* with *DTW* algorithm. On the right, cuboids in blue were generated upon time series clustering using our own algorithm. The predicted trajectory in the form of spatio-temporal cuboids in red on the left substantially deviates from the aligned trajectory in white as soon as the aircraft departs. The deviation from the aligned trajectory during the departure is relatively less significant for the spatio-temporal cuboids in blue on the right. Both sets of spatio-temporal cuboids in red and blue align well with the actual trajectories in white as the flight progresses.

Figure 5.4 shows vertical profile of the actual aligned trajectory in yellow versus vertical profiles of predicted trajectories for flight DAL2173's climb phase on May 18, 2015. The vertical profile in red was generated upon time series clustering with a *k-NN* algorithm that uses *DTW*

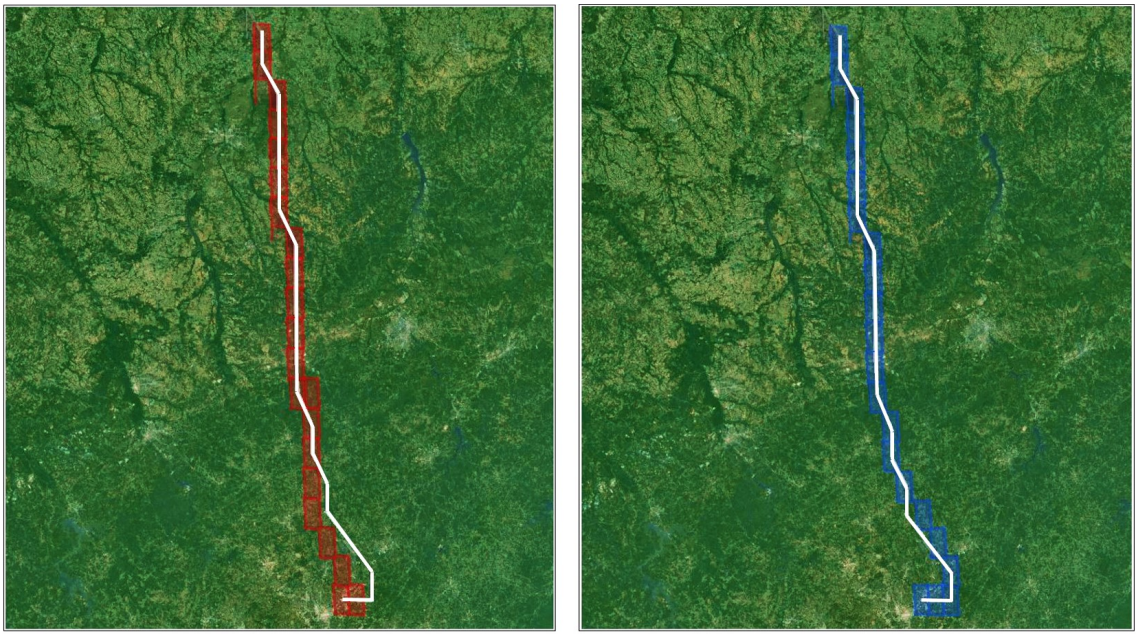


Fig. 5.3: Actual flown lateral trajectory in white overlaid on top of spatio-temporal cuboids for flight DAL2173's climb phase on May 18, 2015.(left) k-NN with DTW, (right) our algorithm.

Euclidean distance. The vertical profile in blue was generated upon our time series clustering algorithm. Deviation of the vertical profile in red from the actual aligned trajectory in yellow is apparent. The vertical profile in blue aligns relatively well with the actual aligned trajectory in yellow.

Our quantitative evaluation is based on trajectory prediction accuracy metrics, including cross-track and vertical errors, as outlined in [151,152]. The cross-track error, denoted by e_{cross} is computed based on the actual position of the aircraft AC, predicted trajectory segment containing the previous predicted position TR_1, current predicted position TR_2, and the angle θ between the two vectors, $|\overrightarrow{TR_1AC}|$ and $|\overrightarrow{TR_1TR_2}|$. Hence, $e_{cross} = |\overrightarrow{TR_1AC}|sin\theta$. The vertical error represents the difference between the actual altitude of the aircraft AA and predicted altitude of the aircraft PA. Hence, $e_{vert} = AA - PA$.

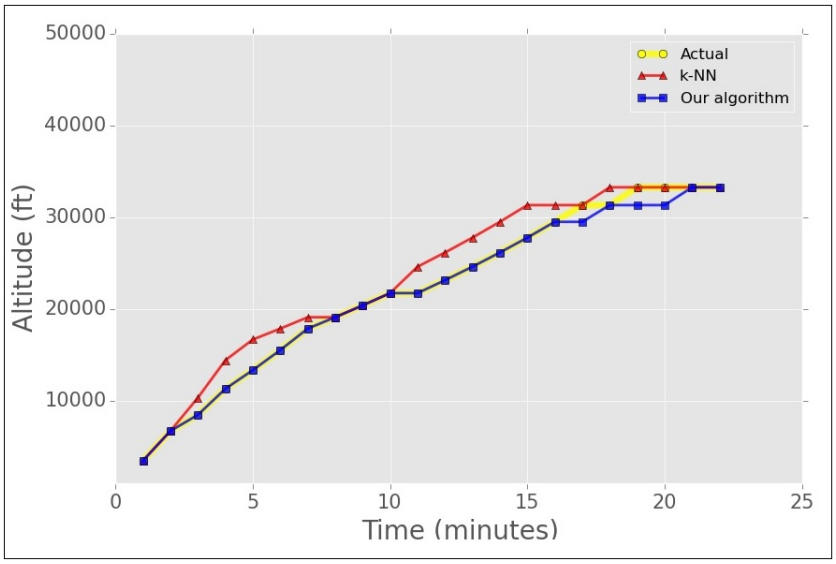


Fig. 5.4: Vertical profile of the actual flown trajectory in yellow versus vertical profiles of predicted trajectories for flight DAL2173's climb phase on May 18, 2015.

Table 6.4 captures mean error μ and standard deviation σ values for the entire trajectory samples. The first two columns correspond to cross-track and vertical errors generated upon time series clustering of weather observations using *k-NN* with *DTW* algorithm. The last two columns correspond to cross-track and vertical errors generated upon time series clustering of weather observations using our own algorithm. Note that both the cross-track and vertical errors are signed errors. However, to better represent, the sign of the errors are omitted in the computation of the mean values along the climb phase of the trajectory.

Figure 5.5 illustrates the histograms for cross-track errors for flight DAL2173's climb phase on May 18, 2015. On the left, the graph in red illustrates the histogram generated upon time series clustering of *k-NN* with *DTW* algorithm. On the right, the graph in blue illustrates the histogram generated upon time series clustering of our own algorithm. The area of the histograms provides an indication of overall performance.

To evaluate the behavior of trajectory prediction error over time, we also computed cross-

Tab. 5.5: Mean and standard deviation values for cross-track and vertical errors for flight DAL2173 on trajectory samples.

k-NN		**Our algorithm**	
Mean error (μ)			
$\mu(e_{cross})$	$\mu(e_{vert})$	$\mu(e_{cross})$	$\mu(e_{vert})$
$1.490nm$	$1697.042ft$	$0.982nm$	$259.588ft$
Standard deviation (σ)			
$\sigma(e_{cross})$	$\sigma(e_{vert})$	$\sigma(e_{cross})$	$\sigma(e_{vert})$
$3.100nm$	$1299.931ft$	$3.296nm$	$653.612ft$

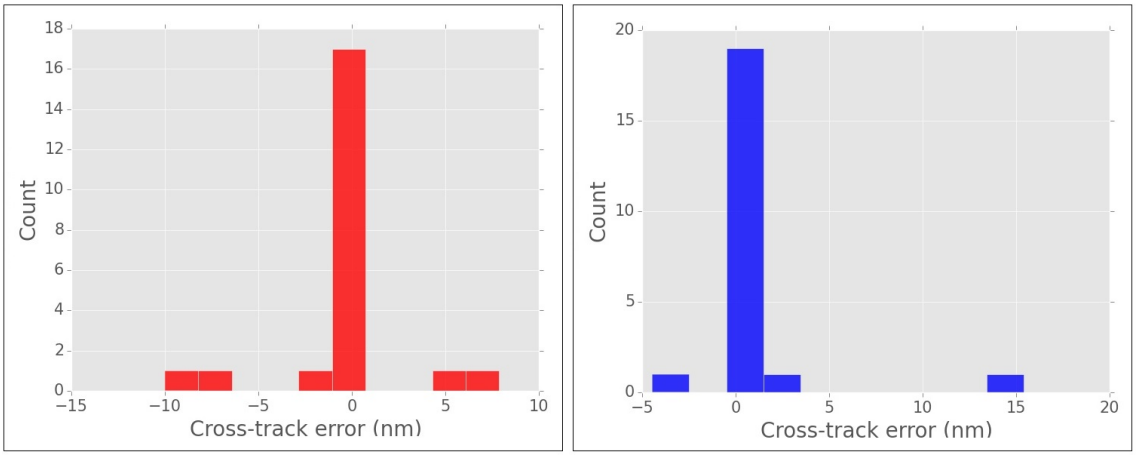


Fig. 5.5: Histograms for flight DAL2173's climb phase on May 18, 2015. (left) k-NN with DTW, (right) our algorithm.

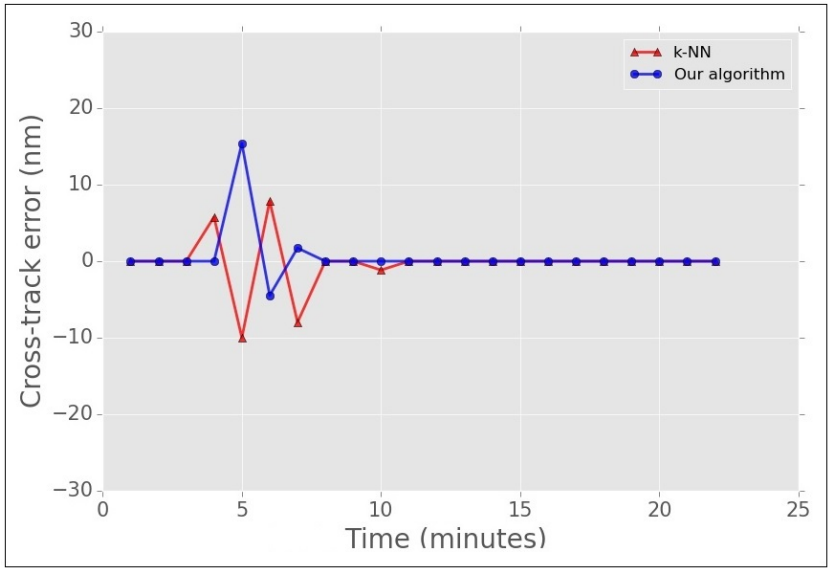


Fig. 5.6: Cross-track errors over look-ahead time for flight DAL2173's climb phase on May 18, 2015.

track errors based on look-ahead time for flight DAL2173's climb phase on May 18, 2015, as shown in Figure 6.7. Lines in red depicts the cross-track error over look-ahead time generated upon time series clustering of weather observations using *k-NN* with *DTW* algorithm. Lines in blue depicts the cross-track error over look-ahead time generated upon time series clustering of weather observations using our own algorithm. This figure confirms the fact that the errors are introduced not too long after the aircraft departs as illustrated in Figure 6.4.

5.4 Conclusion

The role and performance of a trajectory prediction system is critical to the success of the ATC decision support functions which have substantial impact on ATM and airspace flow management. In this chapter, we presented a novel time series clustering algorithm that generates an optimal sequence of weather observations used in accurate trajectory prediction for the climb phase of the flight. Our algorithm computes a cost value for each weather observation based on its frequency per time period, and ranks them. The process generates a matrix, where each matrix

element corresponds to a weather observation with its cost value. Using *DP*, the algorithm computes the optimal sequence of weather observations, a set with the minimum cumulative cost that satisfies the continuity constraint. We evaluated our algorithm using a real trajectory dataset with pertinent weather observations and demonstrated the effectiveness of it over time series clustering with *k-NN* algorithm that uses *DTW* Euclidean distance.

Part IV

Aircraft Trajectory Prediction, Conflict Detection and Resolution

Chapter 6

Aircraft Trajectory Prediction

At the heart of Air Traffic Management (ATM) lies the Decision Support Systems (DST) that rely upon accurate trajectory prediction to determine how the airspace will look like in the future to make better decisions and advisories. Dealing with airspace that is prone to congestion due to environmental factors still remains the challenge especially when a deterministic approach is used in the trajectory prediction process. In this chapter, we describe a novel stochastic trajectory prediction approach for ATM that can be used for more efficient and realistic flight planning and to assist airspace flow management, potentially resulting in higher safety, capacity, and efficiency commensurate with fuel savings thereby reducing emissions for a better environment.

Our approach considers airspace as a 3D grid network, where each grid point is a location of a weather observation. We hypothetically build cubes around these grid points, so the entire airspace can be considered as a set of cubes. Each cube is defined by its centroid, the original grid point, and associated weather parameters that remain homogeneous within the cube during a period of time. Then, we align raw trajectories to a set of cube centroids which are basically fixed 3D positions independent of trajectory data. This creates a new form of trajectories which are 4D joint cubes, where each cube is a segment that is associated with not only spatio-temporal attributes

but also with weather parameters. Next, we exploit machine learning techniques to train inference models from historical data and apply a stochastic model, a Hidden Markov Model (HMM), to predict trajectories taking environmental uncertainties into account. During the process, we apply time series clustering to generate input observations from an excessive set of weather parameters to feed into the Viterbi algorithm. Our experiments use a real trajectory dataset with pertaining weather observations and demonstrate the effectiveness of our approach to the trajectory prediction process for ATM.

6.1 Introduction

The goals of Air Traffic Control (ATC) are the safe and efficient management of aircraft. The Air Navigation Service Provider (ANSP) in the USA, the FAA, is concerned with providing optimal airspace capacity and efficiency within the National Airspace System (NAS). These goals require the separation and sequencing of airborne aircraft by controllers who often use DSTs to monitor the progress of each aircraft and resolve conflicts when necessary. Other controllers monitor the future aggregate flows of aircraft traffic and using a DST implement flow management decisions to ensure that the traffic density always remains within safe limits. At the core of these automation tools resides the Trajectory Prediction (TP) tool, that given the aircraft initial state, its flight plan and the wind and temperatures, computes where the aircraft will be in the future in four dimensions. Inaccurate trajectory prediction can have a substantial impact on the performance of the DST and the ATM system, resulting in

- larger separation standards that limit the number of aircraft that can be allowed in controllers' sectors and/or
- increased numbers of conflicts increasing the workload of the controllers and reducing their

sector capacity and/or

- inefficient fuel consumption due to continual deviations to avoid other aircraft and/or
- adverse impact on the environment due to more fuel burn leading to more emissions.

As a consequence, the performance of the TP tool which supports the DSTs is critical to the success of the DST functions.

Although prior trajectory prediction research and development activities have been able to address the challenge to some extent, dealing with increasingly congested airspace and new environmental concerns still remains the challenge when deterministic approach is used in trajectory prediction process. Hence due to the nature of uncertainties that contribute to trajectory prediction errors, in our research, we have taken a stochastic approach to address the trajectory prediction issues.

We define airspace as a set of spatio-temporal data cubes where each cube is considered as an atomic unit. Weather parameters such as temperature, wind speed, and wind direction remain homogeneous within the cube during a period of time. Other parameters that describe the cube include its center coordinates (latitude, longitude, altitude) along with a time stamp. These spatio-temporal data cubes form the overall airspace. Then, we adapt raw trajectories to the cubes, by aligning each trajectory vertex to the nearest cube centroid, inspired by the *Realm* method [153]. The process yields 4D joint cubes that can be considered as piecewise trajectory segments. Using adapted trajectories in the form of 4D joint cubes, we train our model with the historical data and choose a state sequence that best explains the current observations, the pertinent weather parameters. This process corresponds to the Problem No.2 described in [154] where it attempts to uncover the hidden part of the model, which is the optimal state sequence, given the observations. Note that in the process, each segment is chosen from one of the many hidden segments and the

occurrence of a hidden segment that underlines an observation is dependent on the hidden state of the previous segment. Since our objective is to find the most likely trajectory, we use HMM that is nothing more than a probabilistic function of a Markov process. Due to the nature of interconnected cube centroids forming a trellis, we use the Viterbi algorithm [122] to efficiently generate the optimal trajectory by joining the multiple segments together, where one segment is only dependent on the previous segment.

Now, a critical question arises: *What constitutes the input observations?* or more importantly, *among so many weather observations in the airspace volume of interest during the time period of flight, which ones should be passed to the Viterbi process?* We address this problem by clustering the current weather observations for the cube centroids that were historically traversed. Before the process, we split the current weather observations into buckets and then perform time series clustering [77]. We finally feed the output cluster into the Viterbi algorithm to probabilistically generate the best state sequence.

In summary, the contributions of this study are as follows:

- We propose a novel way of representing aircraft trajectories, a set of 4D joint cubes generated upon an alignment and fusion process.
- We build a stochastic model, HMM that learns from the combination of historical trajectories and aircraft specifications, and their correlation with the pertinent weather parameters. Then, we perform time series clustering on the trajectory segments that were historically traversed. Finally, we feed the series of cluster centroids as observations into the Viterbi algorithm in order to predict trajectories that can be used as more realistic pre-departure flight plans.
- We conduct experiments based on real host track and aircraft specification data and demon-

strate that our system effectively predicts aircraft trajectories.

Although the proposed solution can be adapted for both tactical and strategic trajectory prediction, our experimental study focuses more on a ground-based tactical trajectory prediction system that can be used by both AOC dispatchers to file more realistic flight plans and/or flow managers at the Air Traffic Control System Command Center (ATCSCC) to approve the proposed flight plans based on their conformance to the relatively more realistic predicted trajectories, right before the aircraft departs.

The rest of this chapter is organized as follows. Section 6.2 reviews the related work. Section 6.3 introduces the preliminary concepts, presents the problem, and overviews the proposed system. Section 6.4 discusses in detail the aircraft trajectory prediction system, that we propose. Section 6.5 presents the results of an experimental evaluation. Section 6.6 concludes the chapter.

6.2 Related Work

There has been much work on trajectories in the spatial domain for motor vehicles along roads [64] with an emphasis on their generation (e.g., [65]), queries (e.g., [66–70]), and matching (e.g., [71–73]). This is not our subject here. On the other hand, there has been a vast amount of research and abundant literature with regards to predicting aircraft trajectories. Methodologies to attain this goal can be divided into deterministic and probabilistic approaches. Deterministic approaches are made up of nominal and worst-case techniques, and probabilistic techniques include Sequential Monte Carlo (SMC), Hidden Markov Models (HMM) and others.

The nominal technique gives the aircraft position by propagating estimated states into the future along a single trajectory without considering uncertainties of the state estimate and the prediction [114, 155]. Worst-case techniques assume that an aircraft will perform any of a set of maneuvers and the worst case, defined by the application, is considered for aircraft trajectory

prediction. Algorithms based on this idea are too conservative since civilian aircraft rarely perform extreme maneuvers [156]. Overall, deterministic techniques [111–113, 115] suffer from degraded accuracy due to fact that they don't account for uncertainties and address only specific phase(s) of the flight when predicting aircraft trajectories.

Past efforts addressing aircraft trajectory prediction using probabilistic methods are similar to our approach in a way that they aim at modeling uncertainties to describe potential changes in the future trajectory of an aircraft. However, many of them [116–121] either lack empirical validation or use a simulated set of aircraft trajectories instead of real host track data in their evaluations.

Aside from accounting for uncertainties in the computation, there is considerable research in identifying potential sources of uncertainties in trajectory prediction [157–159]. In addition to purely deterministic and probabilistic methods, a hybrid method combining the SMC and the worst-case method was also proposed [160]. Although the majority of the research on trajectory prediction is used toward detecting and resolving conflicts with other aircraft, there is also a number of research papers on the same topic from the standpoint of convective weather and Special Use Airspace (SUA) avoidance [161–165].

Our work is closer in spirit to research [166–168] that studied the aircraft trajectory prediction problem with a machine learning approach, in which they train the model using historical surveillance data and make predictions using various observations. Choi et al [166] present an approach to predict future motion of a moving object based on its past movement. The approach exploits the similarities of short-term movement behaviors by modeling a trajectory as a concatenation of short segments. Although their approach shares some common ground with ours, they have never applied it to aircraft trajectory prediction. Leege et al. [167] specifically propose a machine learning approach to aircraft trajectory prediction. However, unlike our study, they use

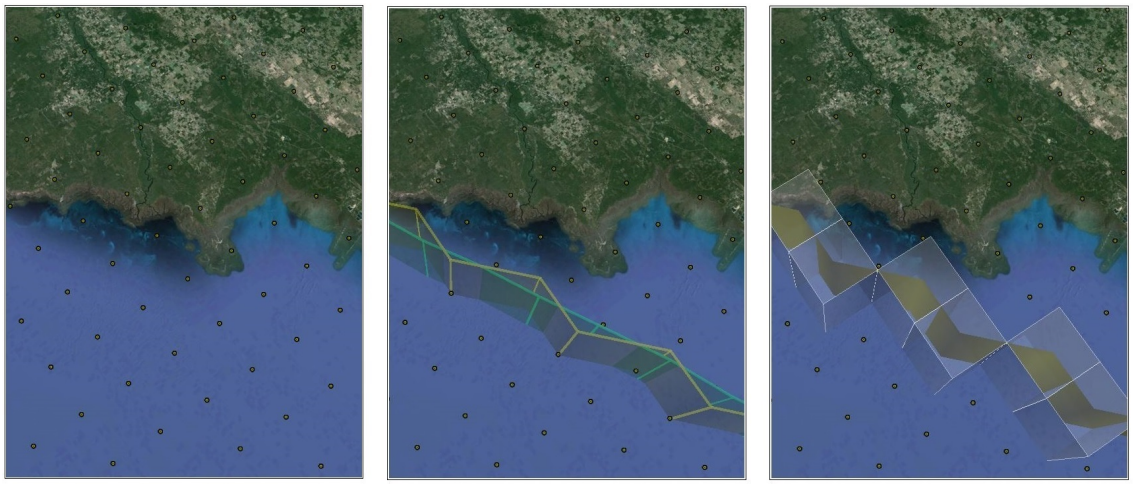
a stepwise regression approach to systematically determine which input variables to include in the models based on explanatory power. In his PhD. dissertation, Winder et al. [168] present a framework for designing a hazard avoidance alerting system that is based on a Markov decision process model.

The machine learning approach also inspires our study as with our stochastic technique, we leverage the historical Aircraft Situation Display to Industry (ASDI) [15] track data and aircraft specifications to select among the possible intermediate reference points and predict the aircraft trajectory based on meteorological observations. During the process, we assume that the weather observations are realizations of hidden trajectory segments and the transitions between the underlying hidden segments follow a Markov model.

To summarize, our approach is distinguished from past efforts in at least one of the following four respects: *i)* We use a probabilistic approach by taking the uncertainties into account, which yields higher accuracy *ii)* We consider trajectories as a set of 4D joint cubes, which helps us to unify trajectory segments and associate them with pertinent weather parameters, *iii)* We perform time series clustering on the excessive set of current observations to generate time series to feed into the Viterbi algorithm, which is an efficient way to find the optimal state sequence, and *iv)* We use real host track and aircraft specifications data along with weather observations to validate the effectiveness of our approach.

6.3 Preliminary and Overview

In this section we introduce some preliminary concepts, present the problem, and give an overview of the proposed aircraft trajectory prediction system.



a) 2D view of RAP weather model's grid network.

b) 3D view of aligned (yellow) and raw (green) trajectories with the reference system.

c) A set of spatio-temporal data cubes defining an aligned trajectory.

Fig. 6.1: Partial illustration of reference system, raw and aligned trajectories, and spatio-temporal data cubes on top of the western Florida area in Google Earth.

6.3.1 Concepts

The EUROCONTROL/FAA Action Plan 16 (AP16) [169] defines a(n) (aircraft) trajectory as *"the path a moving aircraft follows through the airspace and can be mathematically described by a time-order set of trajectory (state) vectors or the geometry of the flight path"*. The International Civil Aviation Organization (ICAO)'s definition of trajectory is slightly different; *"the description of the movement of an aircraft, both in the air and on the ground, including position, time, and at least via calculation, speed and acceleration"* [170]. Our definition of a trajectory, which we call "original trajectory", inspired by Ayhan et al. [74] is slightly different:

Definition 6.1: An **original trajectory** of an aircraft is a continuous representation of its motion with 4D spatio-temporal parameters *(latitude, longitude, altitude, and time)*, indicating the exact path, traveled by the aircraft.

Due to the fact that there exists no system that continuously records and stores exact positions of an aircraft's original trajectory, only a discrete set of sample data are recorded and stored which presumably represent a close approximation of the original trajectory. We call this a raw trajectory, which we formally define below.

Definition 6.2: A **raw trajectory T** of an aircraft is a finite sequence of positions with timestamps sampled from the original trajectory. $T = [p_1, p_2, ..., p_n]$, where each point *p* is defined by its 4D spatio-temporal parameters *(latitude, longitude, altitude, and time)*.

Depending on the sampling strategy, which is how the position recording is triggered, discrete set of spatio-temporal parameters are captured as the aircraft moves. Among others, the most widely used sampling strategies are time-driven (e.g once every minute), distance-driven (once every mile), and geometry-driven (e.g. once the aircraft deviates from its heading more than 15 degrees). Our study uses ASDI track data which is a form of time-driven raw trajectory, described in detail in Section 6.4.

Definition 6.3: A **reference point r** is a fixed spatial location in the 3D space, that is independent of the trajectory data source. A set of evenly distributed reference points form a *reference system R*.

Our study uses the Global Forecast System (GFS) Rapid Refresh (RAP) weather model's 3D grid network [108] as a reference system, described in detail in Section 6.4. Figure 6.1.a shows a 2D partial view of the RAP weather model's grid network, which is the reference model overlaid on top of the western Florida area in Google Earth [110].

Definition 6.4: An **aligned trajectory** $\bar{T}$ is a set of reference points into which the raw trajectory points are transformed. More formally, given a *reference system R*, the aligned trajectory $\bar{T}$ for the *raw trajectory* $T = [p_1, p_2, ..., p_n]$ is $\bar{T} = [r_1, r_2, ..., r_n]$, where $r_i \in R$.

Aligned trajectories must preserve the original trajectories as much as possible. Note that erroneous adjustments are introduced each time original trajectories are turned into raw trajectories and raw trajectories are turned into aligned trajectories. Hence, preserving original trajectories in aligned trajectories is obviously a challenging task. This study uses Euclidean distance as a distance function and searches for the nearest neighboring grid point in the reference system to transform raw trajectory points into reference points. Figure 6.1.b illustrates 3D partial view of both aligned and raw trajectories with the reference system on top of the western Florida area in Google Earth.

Definition 6.5: A **spatio-temporal data cube** is an atomic trajectory unit in space, defined by its reference point coordinates *(latitude, longitude, altitude)* where weather parameters (temperature, wind speed, and wind direction) are considered to remain constant within a period of time.

Aligning raw trajectories to reference points is a critical preparation step where all necessary parameters are aggregated per point. However, due to uncertainties, we hypothetically create cubes around reference points, building an entire airspace composed of 3D cubes, where each data cube has homogeneous weather parameters within during a certain period of time. This way, trajectories are defined by a set of cubical segments. In our study, the lateral resolution of each cube is $13km$ and temporal resolution is $1hr$. Figure 6.1.c is an illustration of a set of spatio-temporal data cubes defining an aligned trajectory transformed from a partial raw trajectory.

Definition 6.6: An **airspace volume of interest** is a 4D airspace volume in which a number of spatio-temporal data cubes are stacked up horizontally and/or vertically.

6.3.2 Problem Statement

Given a set of historical raw trajectories for specific aircraft types along with pertinent historical weather observations, we aim at learning a model that reveals the correlation between

weather conditions and aircraft positions and predicts trajectories in the form of a time series. In this problem space, we assume that the weather observations are realizations of hidden aircraft positions i.e. trajectory segments and the transitions between the underlying hidden segments follow a Markov model. This assumption considers a finite set of states, each of which is associated with a probability distribution over all possible trajectory segments. Transitions among the states are managed by a set of probabilities. The states are not visible, but the pertinent observations are. Given a sequence of observations, we want to learn an HMM, a statistical Markov model, and derive a sequence of hidden states that correspond to the sequence of observations.

The Viterbi algorithm [122] is an efficient method to compute this. However, in our problem space, the observation sequence for which we want to predict the state sequence is unknown. In other words, given a set of weather observations for the full set of spatio-temporal cubes covering the airspace volume of interest and approximate time period of flight, we need to identify the ones that we can input into the Viterbi process.

6.3.3 System Overview

Figure 6.2 shows an overview of the proposed aircraft trajectory prediction system, which generates the optimal state sequence in three steps:

- In the first step, we perform training data processing by generating the HMM parameters based on a set of historical trajectories and weather parameters.
- In the second step, we perform test data processing by executing time series clustering on weather observations for the entire airspace volume of interest which yields the observation sequence.
- In the final step, we use the output of the first two steps in the Viterbi algorithm to generate the optimal state sequence based on the observation sequence.

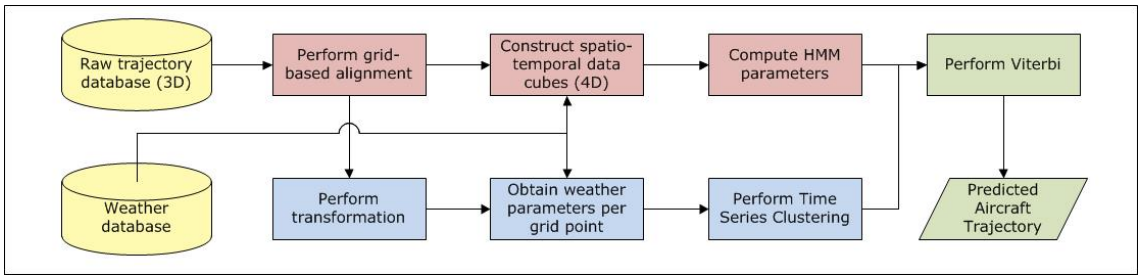


Fig. 6.2: System overview. Data storage is colored yellow, training data processing is colored red, test data processing is colored blue, and the Viterbi process with the final output is colored green

In Figure 6.2, data storage is colored yellow, training data processing is colored red, test data processing is colored blue, and the Viterbi process with the final output is colored green.

6.4 Aircraft Trajectory Prediction

This section presents our approach to aircraft trajectory prediction and elaborates in detail each step of the process.

6.4.1 Training Data Processing

This step enables us to accurately fuse weather parameters per sample point of a raw trajectory. To attain this goal, we need a weather model with the highest spatio-temporal resolution available and that offers both current and historical weather data. The National Oceanic and Atmospheric Administration (NOAA) GFS RAP product operational at the National Center for Experimental Prediction (NCEP) meets these requirements. Hence, we use GFS RAP weather model's 3D grid network as our reference system.

Training data processing, colored red in Figure 6.2, is performed as follows: Based on the reference system, grid-based alignment is performed on the historical raw trajectories. The process

is based on the simple idea of finding the nearest 3D reference point for each sample point of a raw trajectory and then mapping the original sample point to a nearest 3D reference point, inspired by the *Realm* method [153]. More precisely, each sample point in a raw trajectory is aligned to a nearest 3D reference point. This process generates aligned trajectories for historical trajectory data. Although, this seems to suffer from accuracy, it allows to form a unified set resulting in increased similarity between aligned trajectory points. Spatio-temporal data cubes are formed in the next step. During this step, weather parameters for the pertinent time window are retrieved from the weather database and resampled to generate N buckets with distinct ranges. Then, the weather parameters in distinct buckets are fused with spatial data for each grid point along the aligned trajectory. The process yields training data where each historical raw trajectory becomes a set of 4D joint data cubes.

6.4.2 Test Data Processing

In order for us to compute the maximum probability of HMM generating the optimal state sequence of $s_1, s_2, ..., s_m$, the observation sequence $o_1, o_2, ..., o_m$ is needed. Although the reference points covering the entire airspace volume of interest are known, we don't know which one of these should be fed into the Viterbi process. To answer this question, we perform time series clustering with Dynamic Time Warping (DTW) on an excessive set of observations and generate the input time series.

Test data processing, colored blue in Figure 6.2, enables us to address this question. In this step, among all reference points between the departure and arrival airports, we consider only the ones that were traversed in the historical trajectories. The first step in the process is to perform transformation on these aligned historical trajectories by replacing the original date per reference point with the current date, presuming that the flight is about to depart. In other words, during this

step, all aligned historical trajectories are treated as if they are current. In the second step, based on new timestamps, weather parameters are retrieved from the weather database and resampled to generate N buckets with distinct ranges. Next, the weather parameters in distinct buckets are fused with spatial data for each grid point along the aligned trajectory. This step generates hypothetical cubes, in the same way as in the training data processing. The output of this step is an excessive set of spatio-temporal data cubes forming distinct trajectories in the form of a time series.

In the final step of the test data processing, spatial data is omitted and *k-Nearest Neighbors (k-NN)* clustering is performed using DTW on the weather parameters along the time series. The algorithm enforces a *locality* constraint and uses a *LB Keogh* lower bound.

Note that in time series analysis, DTW is an algorithm for measuring similarity between multiple temporal sequences which may vary in time or speed. For simplicity, suppose there are two temporal sequences, Q and C, of length n and m respectively, such that $Q = q_1, q_2, ..., q_i, ..., q_n$ and $C = c_1, c_2, ..., c_i, ..., c_n$. To align two sequences using DTW, an n-by-m matrix is constructed where the (i^{th}, j^{th}) element of the matrix contains the distance $d(q_i, c_j)$ between the two points q_i and c_j. Each matrix element (i, j) corresponds to the alignment between the points q_i and c_j. A *warping path* W is a contiguous set of matrix elements that defines a mapping between Q and C. During the process, the warping path is subject to a *locality* constraint. There are exponentially many warping paths that satisfy this constraint. We are only interested in the path that minimizes the warping cost. This path can be found using dynamic programming to evaluate the following recurrence, which defines the cumulative distance $\gamma(i, j)$ as the distance $d(i, j)$ found in the current cell and the minimum of the cumulative distances of the adjacent elements:

$$\gamma(i, j) = d(q_i, c_j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}$$

By enforcing a *locality* constraint through a threshold window, the algorithm ensures that it is unlikely for d_i and t_j to be matched if i and j are too far apart. In this algorithm, the number

of clusters is set apriori to 1 and similar time series are clustered together. The process yields an observation sequence in the form of a set of weather parameters which are fed into the Viterbi process, in the next step.

6.4.3 HMM Processing and Viterbi

We approach the HMM problem by identifying the parameter set computed based on training data as follows:

- States $S = \{S_1, S_2, ..., S_K\}$ are represented by reference points' coordinates *(latitude, longitude, altitude)* that form aligned trajectories.
- Transition probabilities $A = \{a_{ij}\}$, $1 \leq i, j \leq K$, i.e. a_{ij} is the probability of an aircraft discretely transitioning from one state S_i to another S_j along its aligned trajectory, $\bar{T}$.
- Emission probabilities $B = \{b_i(o)\}$, $1 \leq i \leq K$ is the probability of discrete weather parameters having been observed at a specific state, S_i.
- Initial probabilities $\pi = \{\pi_i\}$, $1 \leq i \leq K$ is the probability of an aligned trajectory beginning at a specific state, S_i.

These parameters form an HMM, denoted by $\lambda = \{S, A, B, \pi\}$. The next step in the process is to choose a corresponding state sequence $s = \{s_1, s_2, ..., s_m\}$, that best explains the observation sequence, $O = \{o_1, o_2, ..., o_m\}$. To answer this question, we use the Viterbi algorithm [122], which has a recursive approach that works in parallel for all states in a strictly time synchronous manner. The key component in the algorithm is the *optimal probability*, denoted as $\delta_m(i)$, which is the maximal probability of HMM generating the observation segment $o_1, o_2, ..., o_m$, along the optimal state sequence $s_1, s_2, ..., s_m$, in which $s_m = i$. Hence, we compute:

Fig. 6.3: 3D raw trajectories of flight DAL2173 between Atlanta and Miami for the period of May 2010 through December 2011 in Google Earth

$$\delta_m(i) = \max_{s_1,\ldots,s_{m-1}} \pi_{s_1} b_{s_1}(o_1) \prod_{j=2}^{m} (a_{s_{j-1},s_j} b_{s_j}(o_j))$$

The process colored green in Figure 6.2 yields the predicted aircraft trajectory.

6.5 Experimental Evaluation

In this section, we conduct experiments to validate the effectiveness of our aircraft trajectory prediction system. Section 6.5.1 introduces the experimental dataset. Section 6.5.2 presents the implementation details and our environment. Section 6.5.3 discusses the results for our prediction system, and Section 6.5.4 outlines the discoveries gained by our trajectory prediction system.

6.5.1 Experimental Dataset

Our empirical evaluation used real host track data along with real weather data: The Delta Airlines' flights DAL2173, departing from Hartsfield-Jackson Atlanta International Airport (ATL) and arriving at Miami International Airport (MIA) for the period of May 2010 through December 2011 were studied. The dataset has a total of 594 trajectories and 56752 points. ATL to MIA is one of the major routes in the NAS due to fact that the departure airport is the busiest airport in the U.S. and the flights are prone to frequent convective weather in the airspace controlled by three Air Route Traffic Control Centers (ARTCC), Atlanta (ZTL), Jacksonville (ZJX), and Miami (ZMA). Figure 6.3 shows 3D raw trajectories of flight DAL2173 between Atlanta and Miami for the period of May 2010 through December 2011.

Two main data sources to our Aircraft Trajectory Prediction System are the FAA's ASDI and NOAA's RAP data which are colored yellow in Figure 6.2 and presented next.

ASDI

The source of the raw trajectory data is ASDI, which is a continuous stream of messages delivered over a TCP/IP network socket. Note that a total of over 30 million ASDI messages are processed in any single day. ASDI messages can be flight plan related data, oceanic reports, or host track reports. The host track data is recorded once in every 60 seconds, and provided in near real-time by the FAA. Upon processing, the data is stored in a relational database and made available for use by our trajectory prediction system. Note that surface data is not included in the ASDI feed, although departure and arrival airports remain the same, the positions of the first and last track records may differ over time. The raw trajectory data is generated by joining various ASDI message types including: Track Information and Flight Plan Information. The process generates *source center, date, time, aircraft ID, speed, latitude, longitude, altitude* values from the Track

Information and *special aircraft qualifier* from the Flight Plan Information message type.

RAP

The source of the weather data is NOAA National Centers for Environmental Prediction (NCEP) Rapid Refresh (RAP) [108] which is the continental-scale NOAA hourly-updated assimilation/modeling system. RAP covers North America and is comprised primarily of a numerical forecast model and an analysis/assimilation system to initialize that model. It has $13km$ horizontal resolution with 50 vertical levels, ptop=10hPa, sigma vertical coordinate. Although there is an experimental $3km$ hourly updated nest inside of the $13km$ Rapid Refresh, it is not widely available yet. The RAP weather data is stored as a set of *grib2* files [171] each hour and made available for use by our prediction system.

6.5.2 Implementation

The ASDI host track data is composed of radar recordings at an approximate rate of once every 60 seconds by each Air Route Traffic Control Center (ARTCC) for any aircraft operating under Instrument Flight Rules (IFR) within the confines of an ARTCC's airspace. Our airspace volume of interest is controlled by three separate ARTCCs, due to fact that an aircraft traveling between Atlanta and Miami airports uses Atlanta ARTCC (ZTL), Jacksonville ARTCC (ZJX), and Miami ARTCC (ZMA). The ASDI host track data contains multiple recordings when the aircraft crosses a boundary between multiple ARTCCs. Hence, our initial implementation step filtered these multiple recordings by taking the timestamp and source center (ARTCC) information into account. The filtering process reduced the number of historical recordings from 56752 to 48694. As in rare cases, ARTCCs recorded multiple track data within a relatively short period of time, such as 20 seconds, we also filtered these recordings. The process brought the number of historical

Tab. 6.1: Buckets for temperature

Temperature ($temp$)		
No	**Bucket**	**Value** ($kelvin$)
1	$temp \leq 220$	220
2	$220 < temp \leq 240$	240
3	$240 < temp \leq 260$	260
4	$260 < temp \leq 280$	280
5	$280 < temp \leq 300$	300
6	$300 < temp \leq 350$	350

recordings down to 44824.

The remaining historical raw trajectory points went through alignment process as elaborated in Section 6.4.1. During the process, we used GFS RAP weather model's 3D grid network as our reference model, and mapped the raw trajectory points to the nearest reference points. This process yielded aligned historical trajectories. In the next step, spatio-temporal data cubes were formed and reference points along aligned trajectories were fused with weather parameters. However, due to lack of historical weather data for the flight time window of interest of some aligned trajectories, we were not able to expand their dimensions. Subsequently these aligned historical trajectories were eliminated bringing the total number of reference points down to 37849. Fusing positional data with weather parameters yielded the following attributes per spatio-temporal data cube: *source center, date, time, aircraft id, speed, latitude, longitude, altitude, aircraft type, temperature, wind speed, wind direction*. As the final step of the training data processing, we split the weather parameters into buckets as shown in Table 6.1, 6.2, and 6.3.

Next, we computed the following HMM parameters:

Tab. 6.2: Buckets for wind speed

Wind speed (ws)		
No	**Bucket**	**Value** ($knots$)
1	$ws \leq 30$	30
2	$30 < ws \leq 60$	60
3	$60 < ws \leq 90$	90
4	$90 < ws \leq 120$	120
5	$120 < ws \leq 150$	150

Tab. 6.3: Buckets for wind direction

Wind direction (wd)		
No	**Bucket**	**Value** ($degrees$)
1	$wd \leq 45$	45
2	$45 < wd \leq 90$	90
3	$90 < wd \leq 135$	135
4	$135 < wd \leq 180$	180
5	$180 < wd \leq 225$	225
6	$225 < wd \leq 270$	270
7	$270 < wd \leq 315$	315
8	$315 < wd \leq 360$	360

- 3698 distinct states, S were generated.
- A sparse transition matrix, A of size 3698 x 3698 = 13675204 was generated.
- An emission matrix, B of size 117 x 3698 = 432666 was generated.
- An initial matrix, π of size 28 x 1 was generated.

Aside from these HMM parameters, we also needed to feed the observation sequence into the Viterbi algorithm.

To evaluate our prediction system, we performed bootstrapping by drawing many trajectory samples with replacement from the historical trajectories. In the meantime, we removed the pertinent track records from the training dataset. The process allowed us to plot 95% confidence interval (CI) for the mean error and standard deviation of our trajectory samples. Note that one of the trajectory samples we draw happened to be May 14, 2011. Hence, we present the process for this particular trajectory sample. The test data processing repeated itself for other trajectory samples.

Test data processing for this particular trajectory sample is as follows: We used aligned trajectories for the same historical data. However, this time we treated each trajectory as if it was flown on May 14, 2011. This required a simple transformation of replacing date value per aligned trajectory point. The next step in the process required that we obtain weather parameters for the positional data. However, in order for us to do that, we needed to identify the time period of the flight. Given the median duration of the flight DAL2173 being 78 minutes, we decided to use two *RAP grib2* weather files, one targeting weather observations recorded at 15:00 UTC and the other targeting weather observations recorded at 16:00 UTC on May 14, 2011. By using these files, we retrieved the pertinent weather parameters per aligned trajectory point. Next, we split the weather parameters into buckets as shown in Table 6.1, 6.2, and 6.3. In the final step of test

data processing, we performed time series clustering, using DTW. The input to the process was 474 time series of 78 weather observations, where each observation contained *temperature, wind speed, wind direction* parameters. The *k-nearest neighbors (k-NN)* clustering with *k*=1 used DTW Euclidean distance for similarity measure. To speed up the process, we used *LB Keogh* lower bound [77]. The resulting set of cluster centroids identified by weather parameters defined the observation sequence Y_s.

As the final step of the process, we fed the observation sequence Y_s and HMM parameters, $\lambda = \{S, A, B, \pi\}$ into the Viterbi algorithm. The algorithm worked in parallel for all states in a strictly time synchronous manner and returned the optimal state sequence with the maximum probability.

Our experiments were conducted on a computer with Intel Core i7-3840QM CPU @ 2.80GHz and 16GB memory, running on Linux Ubuntu 14.04.2 64-bit LTS Operating System. All the algorithms were implemented in Python v2.7.

6.5.3 Results

We evaluated the effectiveness of our prediction system based on bootstrapping by drawing 23 trajectory samples with replacement from the historical trajectories. This way, we compared our prediction with the ground truth, flight DAL2173's aligned trajectory. Next, we computed the mean error and standard deviation per trajectory sample. Then, we ranked order the means to estimate the 2.5 and 97.5 percentile values for 95% CI.

Figure 6.4 shows actual flown trajectory in red overlaid on top of spatio-temporal data cubes in white predicted by our system for one of the trajectory samples drawn, May 14, 2011 as part of the bootstrapping. Major prediction errors are introduced around top of climb and descent phases.

Our quantitative evaluation is based on trajectory prediction accuracy metrics, including

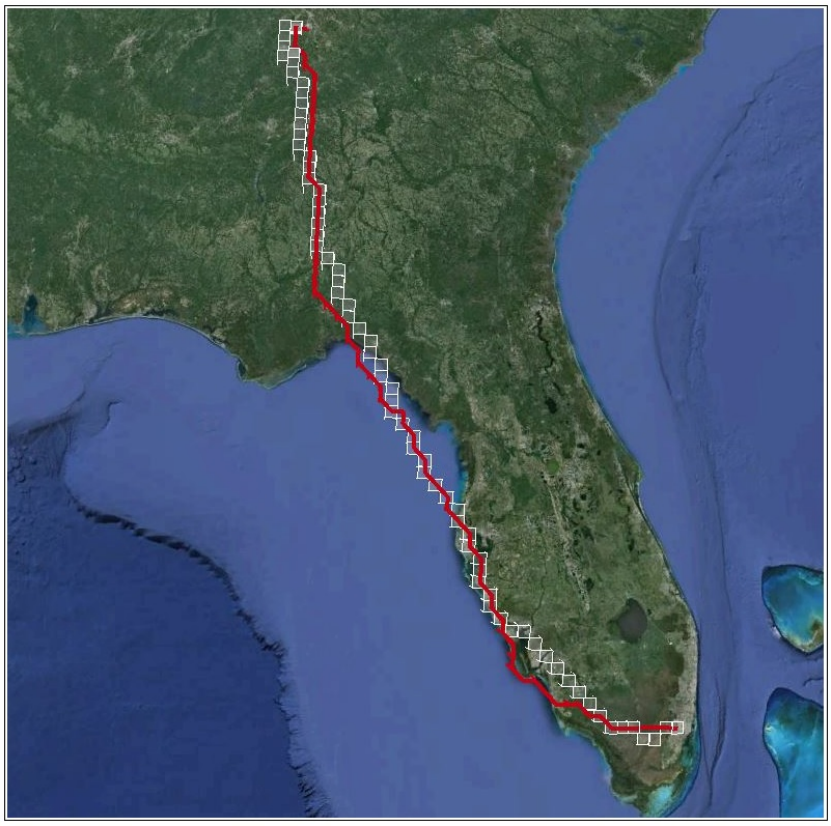


Fig. 6.4: Actual flown trajectory overlaid on top of spatio-temporal data cubes generated for flight DAL2173 on May 14, 2011 by our prediction system

horizontal, along-track, cross-track, and vertical errors, as outlined in [151, 152]. The errors illustrated in Figure 6.5 are based on the coordinates of the aircraft, denoted as AC, and a trajectory segment containing the points TJ_1 and TJ_2.

Table 6.4 captures mean error μ and standard deviation σ for horizontal, vertical, along-track, and cross-track errors for the entire set of trajectory samples with 95% CI. The errors were computed based on our prediction versus flight DAL2173's raw trajectory per trajectory sample.

The horizontal error is unsigned whereas the along-track and cross-track errors are signed errors. Note that the mean value for the cross-track error $\mu(e_{cross})$ along the entire trajectory of flight DAL2173 is 6.804nm (=12.601km), when the sign is omitted. Due to fact that spatial resolution of our trajectory prediction system is 13km, which is the horizontal length of each spatio-temporal cube, we can conclude that the mean value $\mu(e_{cross})$ for the cross-track error of 12.601km is within the boundaries of our spatial resolution.

Figure 6.6 illustrates the pertinent histograms for horizontal, vertical, along-track and cross-track errors for the entire set of trajectory samples with 95% CI. Due to fact that the along-track and cross-track errors are signed errors, they generate a relatively symmetric distribution about the zero, as shown in Figure 6.6. The area of the histograms provides an indication of overall performance.

To evaluate behavior of trajectory prediction errors over time, we also computed horizontal, along-track, and cross-track errors for the entire set of trajectory samples with 95% CI based on look-ahead time, as shown in Figure 6.7. Generally, the longer the look-ahead time, the greater the uncertainty, yielding larger errors. However, this is not the case for our system, as shown in Figure 6.7.

As part of the time-based evaluation, we compared estimated time of arrival initially entered by AOC, predicted time of arrival computed by our system and the actual time of arrival for flight

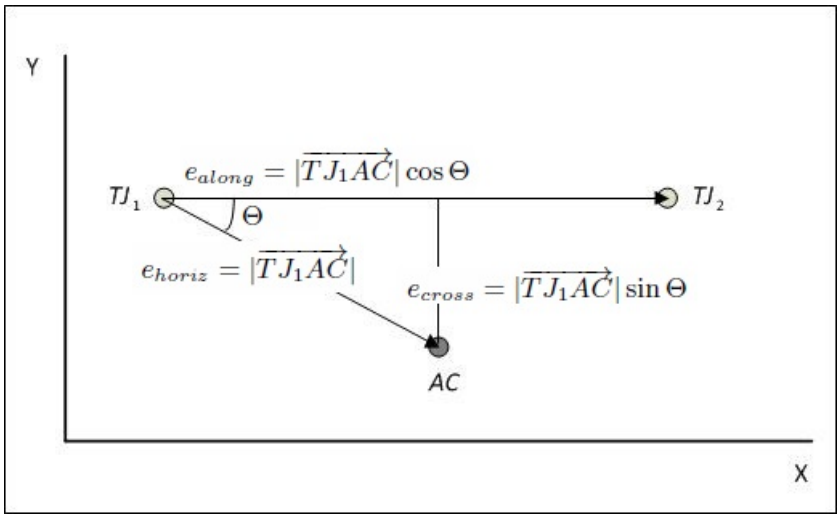


Fig. 6.5: Trajectory prediction accuracy metrics

DAL2173.

Although our prediction system considers time of last ASDI track record as arrival time, there is an additional 4 minutes in average for the aircraft to actually land. By taking this offset into account, our predicted arrival time virtually matches the actual arrival time. Accurate arrival time prediction is a critical capability in maintaining safe separation and sequencing of aircraft.

6.5.4 Discoveries

In this section, we present the value our system offers in an operational view of flight planning.

The majority of other ground-based tactical TP tools communicate with the aircraft periodically throughout the flight, obtain and update the following parameters, and predict trajectories: $i)$ initial conditions (aircraft's 4D position), $ii)$ flight plan, $iii)$ aircraft performance model, $iv)$ weather observations, and $v)$ aircraft intent.

As with proprietary implementations, their prediction accuracy depends on correct initial state values and look-ahead time. Hence, they require frequent parameter updates from the aircraft.

Tab. 6.4: Mean and standard deviation values for horizontal, vertical, along-track, and cross-track errors for flight DAL2173 on trajectory samples

Mean error (μ)			
$\mu(e_{horiz})$	$\mu(e_{along})$	$\mu(e_{cross})$	$\mu(e_{vert})$
12.965nm	1.454nm	-1.859nm	687.497ft
Standard deviation (σ)			
$\sigma(e_{horiz})$	$\sigma(e_{along})$	$\sigma(e_{cross})$	$\sigma(e_{vert})$
12.093nm	13.833nm	10.836nm	4910.691ft

Each update requires datalink communication throughout the flight, resulting in high communication costs. To give a notional idea on communication cost, it is safe to assume that a TP tool communicates with an aircraft once every 10 seconds for position and once every 2 minutes for aircraft intent data where each message costs $0.50, although airlines pay yearly flat rates to data link service providers such as ARINC and SITA that are kept confidential. Based on the assumption, for a single flight of 80-minute duration, the data link communication cost can add up to $260. Depending on the fleet size and frequency and distance of flights, data link communication cost can easily become millions of dollars per year.

Unlike other TP tools, our system inputs only weather observations and predicts a trajectory, based on an HMM that was learned from the historical trajectories for particular aircraft types. Our prediction system is also ground-based and runs offline requiring no datalink communication with the aircraft. As presented in the analysis of mean error versus look-ahead time, the accuracy of a trajectory predicted by our system is independent of look-ahead time. Our system saves the airlines data communications costs, if deployed.

Initial pre-departure flight plans are filed by the AOC and approved by the FAA, when validated. As presented in the analysis of mean errors and standard deviation values, our system

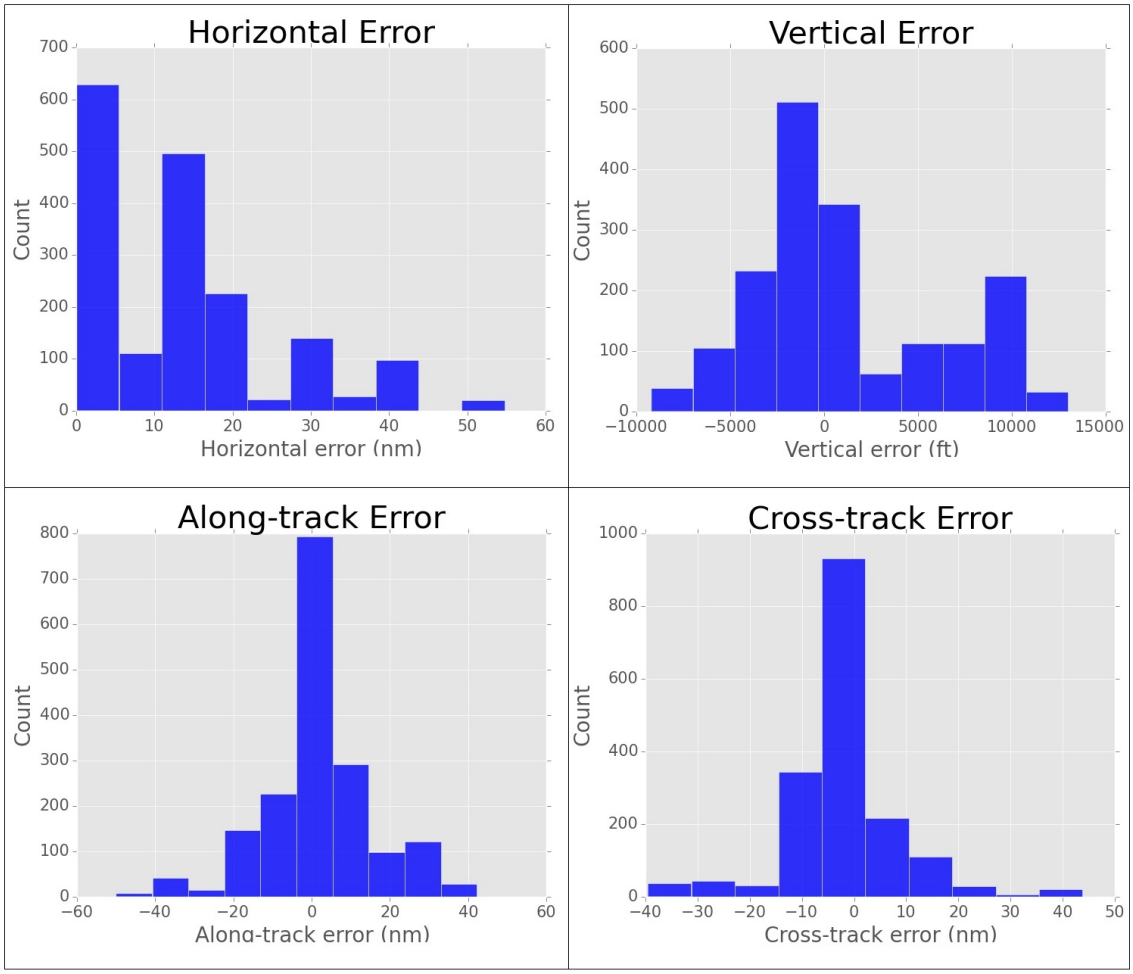


Fig. 6.6: Histograms for horizontal, vertical, along-track, and cross-track errors

predicts aircraft trajectory within the boundaries of our horizontal spatial resolution of $13km$. Hence, when deployed by the airlines, AOC can generate an accurate flight plan and file for approval. In return, the FAA can run our system and validate the filed flight plans, when deployed.

As presented in the analysis of arrival times, our system predicted virtually the same arrival time of flight DAL2173 as the actual arrival time. The accuracy would help in maintaining separation and sequencing of aircraft in the NAS. Better maintained separation and sequencing means: i) higher safety and capacity, ii) accurate arrival metering which translates to efficiency and so fuel savings and reduced emissions. An erroneously predicted arrival time would lead the Air Traffic Controller to allot more time or insufficient time and airspace for the flight, and generate the incorrect landing sequence potentially reducing runway acceptance rate and causing other aircraft to be perturbed in their operations resulting in reduced efficiency, waste of resources, and possibly reducing safety. Note that the aircraft type Delta Airlines flown for DAL2173 on May 14, 2011 is Boeing 757-200 and operational cost of this particular aircraft per hour is $8,383.15 [172], or $139.72 each minute of extra flight time due to erroneous arrival prediction plus the cost of the impact on other perturbed dependent flights.

6.6 Conclusion

The role and performance of trajectory prediction system is critical to the success of the DST functions which have substantial impact on ATM and airspace flow management. In this chapter, we have proposed a novel approach to aircraft trajectory prediction that can be used for more efficient and realistic flight planning by aircraft operators and to assist airspace flow management, potentially achieving higher safety, capacity, and efficiency and commensurate fuel savings and so emission reductions for a better environment.

Our evaluation on a real trajectory dataset verified that our prediction system achieved hor-

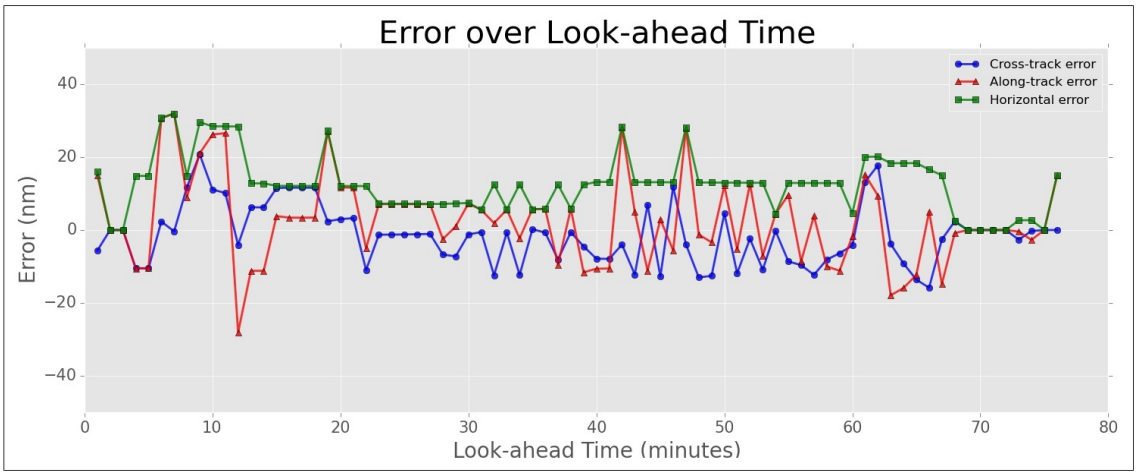


Fig. 6.7: Horizontal, along-track, and cross-track mean errors over look-ahead time

izontal accuracy of 12.601km which is defined as a mean cross-track error $\mu(e_{cross})$ that is within the boundaries of highest spatial resolution, our trajectory prediction system has to offer. In order for us to achieve higher accuracy, we need weather observations of higher spatial resolution than 13km, which is the current highest spatial resolution available. We plan to investigate the feasibility of resampling and intelligent interpolation of weather data to generate higher spatial resolution. Another option would be to use the recent High-Resolution Rapid Refresh (HRRR) weather product, which offers 3km of spatial resolution for a smaller geographic region, yet still experimental by the NOAA. The introduction of a browsing capability similar to what is available in the spatial domain (e.g., [173, 174]) is also of interest.

Chapter 7

Long-Range Aircraft Conflict Detection and Resolution

At the present time, there is no mechanism for Air Navigation Service Providers (ANSPs) to probe new flight plans filed by the Airlines Operation Centers (AOCs) against the existing approved flight plans to see if they are likely to cause conflicts or bring sector traffic densities beyond control. In the current Air Traffic Control (ATC) operations, aircraft conflicts and sector traffic densities are resolved tactically, increasing workload and leading to potential safety risks and loss of capacity and efficiency.

In this chapter, we propose a novel Data-Driven Framework to address a long-range aircraft conflict detection and resolution (CDR) problem. Given a set of predicted trajectories, the framework declares a conflict when a protected zone of an aircraft on its trajectory is infringed upon by another aircraft. The framework resolves the conflict by prescribing an alternative solution that is optimized by perturbing at least one of the trajectories involved in the conflict. To achieve this, the framework learns from descriptive patterns of historical trajectories and pertinent weather observations and builds a Hidden Markov Model (HMM). Using a variant of the Viterbi algorithm,

the framework avoids the airspace volume in which the conflict is detected and generates a new optimal trajectory that is conflict-free. The key concept upon which the framework is built is the assumption that the airspace is nothing more than a horizontally and vertically concatenated set of spatio-temporal data cubes where each cube is considered as an atomic unit. We evaluate our framework using real trajectory datasets with pertinent weather observations from two continents and demonstrate its effectiveness for strategic CDR.

7.1 Introduction

In the present Air Traffic Management (ATM) system, Airline Operations Center (AOC) personnel file a flight plan for a particular flight. The filed flight plan usually contains 2D coordinates of the fixed way points and the planned initial cruise speed and cruise altitude in addition to their speed and level changes along the route. It does not include the time information for the way points and the existing information is not routinely updated by the ATM systems. Moreover, the filed flight plan is not checked against other flight plans to probe potential interference with other aircraft or identify sector traffic complexity before the aircraft departs. Airspace sector complexity and aircraft separation assurance are dealt with tactically, usually just a few minutes before the aircraft enters the sector or is likely to have a conflict with other aircraft.

It is estimated that by 2040 the USA alone can expect an increase of more than 68% in the commercial air traffic [1]. Hence, a new concept of operations is needed to accommodate this increase in volume. To meet this challenge ahead of us, new technologies and procedures for next generation ATM are being developed in the context of the Next Generation Air Transportation System (NextGen) [6] in the USA and the Single European Sky ATM Research (SESAR) [7] in Europe. The SESAR and NextGen concept of operations requires a paradigm shift from a highly structured and fragmented system that is heavily reliant on tactical decision making and

with few strategic planning functions based on uncertain information, to an integrated one based on collaborative strategic management of trajectories and information sharing [12]. In the future ATM systems to be built under NextGen and SESAR, the trajectory becomes the fundamental element of new sets of operating procedures collectively referred to as Trajectory-Based Operations (TBO). The underlying idea behind TBO is the concept of a business trajectory that the airline agrees to fly and the ANSP and airports agree to facilitate, given the safe separation provision. A business trajectory is the representation of an airline's intention with respect to a given flight that will best meet the airline's business interests. The TBO concept of operations and the notion of a business trajectory will result in increased overall predictability of air traffic, reduced costs and emissions due to lowered fuel consumption and/or time, and increased capacities. Thus, the future ATM system should offer flexibility to accommodate business trajectories, while bearing the primary goal of safety in mind. Overall, the next generation ATM paradigm shift requires more safe and efficient CDR due to fact that maintaining the separation minima among the vast volume of aircraft that fly their versions of most optimal business 4D trajectories becomes more challenging.

Hence, we introduce a Data-Driven Framework to address the long-range aircraft CDR problem. The framework performs CDR in two steps: In the first step, given a set of predicted trajectories in the form of a 4D joint spatio-temporal data cubes on a 3D grid network, the framework declares a conflict if one of the aircraft's predicted trajectory segment overlaps with the other's protected zone at the same time interval in the future. Obviously, the more accurate the predicted trajectories, the more accurate the detection of the conflict as the separation minima will be on the containment volume of each predicted trajectory. In the second step, the framework builds a stochastic model, HMM that learns from the historical trajectories and their correlation with the pertinent weather parameters. Using a variant of the Viterbi algorithm, the framework prescribes an optimal solution by perturbing at least one of the 4D trajectories involved in the

conflict.

In summary, the contributions of this chapter are as follows:

- We propose a cube-shaped protected zone surrounding each aircraft that can be expanded by joining the neighboring cubes horizontally and vertically, yielding a protected zone with a desired size. This idea resonates with the assumption that an airspace is nothing more than a set of concatenated spatio-temporal data cubes around a 3D grid network, where each cube is considered as an atomic unit.

- We propose a scalable Data-Driven Framework to strategically address the aircraft CDR problem. Given a set of predicted trajectories, the framework declares a conflict when a protected zone of an aircraft on its trajectory is infringed upon by another aircraft at the same time interval in the future. Upon a conflict, the framework executes our conflict resolution algorithm and prescribes a solution that resolves the conflict by perturbing at least one of the 4D trajectories involved in the conflict.

- We conduct extensive experiments based on real trajectory and weather data from two continents and demonstrate that our framework can detect and resolve conflicts with lateral and vertical accuracies that are within the boundaries of conventionally accepted minimum separation values, set by the ANSPs. This translates to the fact that, ANSPs can now detect and resolve potential conflicts before the aircraft depart, resulting in safer and greener airspace with more efficiency and capacity, and thereby reducing the air traffic controller workload.

In the previous chapter [46], we presented our Aircraft Trajectory Prediction System that shares a common ground with the current framework in a way that both the previous system and the current framework attempt to find the most likely sequence of aircraft positions given a set of weather observations. However, they differ in the following respects: *i)* The previous system

[46] aimed at predicting aircraft trajectories without considering any potential obstacles along the route. The proposed framework detects and resolves conflicts among the aircraft. In fact, the framework we propose in this work uses predicted trajectories generated by the previous system [46] as input. *ii)* The previous system [46] utilized historical trajectories and pertinent weather observations to build an HMM which was fed into the Viterbi algorithm. Although our current framework uses the same input for an HMM during the conflict resolution phase, it employs a variant of the Viterbi algorithm. Unlike the regular Viterbi algorithm, the variant generates optimal trajectories bypassing the spatio-temporal data cubes in which the conflict is detected. *iii)* Unlike the previous system [46], the current framework is scalable. It can detect and resolve conflicts among multiple aircraft.

Our Data-Driven Framework can be used as a ground-based strategic CDR system by air traffic flow managers to resolve potential interference among large volume of aircraft and identify high density and complex sector traffic before the aircraft depart. The set of optimized resolutions should improve ATM automation and reduce the workload of air traffic controllers. The rest of the chapter is organized as follows. Section 7.2 reviews related work. Section 7.3 introduces preliminary concepts followed by Section 7.4 where our Data-Driven Framework is described. Section 7.5 presents the results of our experiments, while Section 7.6 discusses the results. Concluding remarks are drawn in Section 7.7.

7.2 Related Work

The problem of aircraft CDR remains an active area of research in the spatial domain and in the aviation community and has attracted the attention of many researchers. An excellent survey of various CDR methods is presented in [175]. In this survey, Kuchar et al. propose a taxonomy to categorize the basic functions of CDR modeling methods. The proposed taxonomy

includes: dimensions of state information (lateral, vertical, or three-dimensional); method of state propagation (nominal, worst-case, or probabilistic); conflict detection threshold; conflict resolution method (prescribed, optimized, force field, or manual); maneuvering options (speed change, lateral, vertical, or combined maneuvers); and management of multiple aircraft conflicts (pairwise or global).

Conflict detection methods can be classified as nominal, worst-case, and probabilistic techniques. The nominal technique projects the current states into the future along a single trajectory without taking uncertainties into account [156, 176–179]. The worst-case technique assumes that an aircraft will perform any of a set of maneuvers and a conflict is predicted if any of the maneuvers could cause a conflict [180–183]. The disadvantage of the worst-case technique is that it can declare a conflict as soon as there is a minimum likelihood of a conflict within the definition of the worst-case trajectory model thereby leading to false positives. The probabilistic approach offers a balance between relying on either a single trajectory model as in the nominal technique or a set of worst-case maneuvers. Instead it models uncertainties to describe potential changes in the future trajectory [184–191].

Once a conflict has been detected, the next step in the CDR process is to initiate the conflict resolution phase by determining the course of action. The conflict resolution method and the maneuvering options are two major factors in defining the course of action. Conflict resolution methods can be categorized as *a)* prescribed, *b)* optimized, *c)* force-field, and *d)* manual. The prescribed resolution method provides a fixed maneuver based on a set of predefined procedures [192]. Hence, depending on the nature of the conflict, the predefined resolution maneuver is automatically performed, minimizing the response time. However, it does not compute the optimal resolution path for the aircraft, thereby resulting in a less efficient trajectory. The optimized method provides a conflict resolution strategy with the lowest cost based on a certain cost function

(separation, fuel, time, workload, etc.) [176]. In the force-field resolution method, each aircraft is treated as a charged particle and the resolution maneuvers are defined using repulsive forces between the aircraft [193]. Although it is practical when properly applied, it may require a high level of guidance on the flight deck especially when the aircraft vary their speed over a wide range. The manual resolution method allows users to generate potential resolution options and provide feedback if the option is viable [188, 191].

During the resolution phase, some CDR approaches only offer a single maneuver such as speed change [181, 194, 195] or lateral maneuver [196] or vertical maneuver [176, 197], while others offer a combination of these maneuvers [184, 198, 199]. Obviously, the more maneuvering options the CDR approach offers, the more likely an efficient solution can be provided to a conflict. Our CDR approach has some similarities with [194, 200] due to fact that both approaches consider the airspace as a set of cubic cells, called the grid model, declare a conflict if one of the aircraft's predicted trajectory segments overlap with the other's protected zone, and propose an optimized 4D trajectory for the conflict resolution. However, unlike our study, they attempt to address tactical CDR (short-term and medium-term) by changing speed profiles [194] or using Particle Swarm Optimization [200].

Summarizing the above literature and comparing to the proposed prescriptive analytics approach, we make the following remarks: *i)* Our CDR modeling method considers dimensions of state information as 4D (latitude, longitude, altitude, and timestamp), *ii)* Due to input trajectories being probabilistically computed, our conflict detection method is also considered probabilistic, *iii)* Our conflict resolution method is optimized, and it offers a combination of lateral and vertical maneuvers. *iv)* Our framework is scalable, i.e., it can address the CDR problem among multiple aircraft. Given a set of aircraft trajectories in 4D, our framework declares a conflict if any of the aircraft's predicted trajectory segments overlaps with the other's protected zone at the same time

interval in the future. In our approach, one or more cubic cells forming the airspace is considered to be a conflict detection threshold. Next, we compute and prescribe an optimized solution which is a conflict-free 4D trajectory. Our approach addresses the CDR problem strategically over a time horizon of several hours to compute conflict-free 4D trajectories for optimal flight plans and less complex sector traffic densities. In particular, the proposed data-driven approach exploits machine learning techniques to predict conflicts and prescribe an optimized solution based on constraints introduced to the framework.

7.3 Preliminaries

The primary concern of the ANSPs, for example; the FAA in the USA and EUROCONTROL in Europe is to assure safety, which is quantified by the number of resolved conflicts.

Definition 7.1: A **conflict** is an event in which two or more aircraft come closer than a certain distance to one another.

Definition 7.2: **Separation minima** are encoded by lateral and vertical separation, forming a bounding volume around each aircraft, a *protected zone(PZ)*. Currently, the minimum lateral separation for en-route airspace is 5nmi. It is 3nmi inside the terminal radar approach control (TRACON) area. The minimum vertical separation is 2000 ft above the altitude of 29000 ft (FL290) and 1000 ft below FL290. Due to fact that lateral and vertical separation are specified by single distance values, the resulting PZ becomes a cylinder. Each aircraft is assumed to be surrounded by a PZ that moves along the aircraft. An interesting idea is to use the Hausdorff distance [72] to define the cubes.

Definition 7.3: **Conflict detection** is a process that evaluates the separation between any pair of aircraft, by comparing the distance between them with the separation minima. Formally, given a

pair of predicted aircraft trajectories formed by a set of aircraft positions $T_i = [p_{i1}, p_{i2}, ..., p_{im}]$, $T_j = [p_{j1}, p_{j2}, ..., p_{jn}]$ where each point *p* is defined by its 4D spatio-temporal parameters *(latitude, longitude, altitude,* and *timestamp)*, distance values between them $d_{i,j}$ are computed and compared with the separation minima d_s and a conflict is declared if any of distance values is less than the separation minima $d_{i,j} < d_s$.

Definition 7.4: **Conflict resolution** is a process that generates a feasible safe alternative trajectory by fulfilling the separation minima criteria. Formally, given a pair of predicted aircraft trajectories formed by a set of aircraft positions $T_i = [p_{i1}, p_{i2}, ..., p_{im}]$, $T_j = [p_{j1}, p_{j2}, ..., p_{jn}]$, upon conflict resolution, all the distance values $d_{i,j}$ between the pairs of aircraft positions are greater than the separation minima $d_{i,j} > d_s$.

Definition 7.5: **Long-range CDR** is a process in which conflict detection and resolution are carried out several hours before the potential conflict occurs. Hence, long-range CDR is strategically performed before the departure for better planning, whereas mid-range and short-range CDR are tactically performed while the aircraft is airborne.

Unlike online CDR approaches in which distance between predicted future aircraft positions are constantly computed and compared with separation minima upon receipt of each new aircraft position, our approach is offline and uses a 3D grid network as a reference system. In our approach, raw trajectories are transformed into *aligned trajectories* causing aircraft to move along grid points. This results in conflict queries being computed at grid points only. In addition, unlike most other CDR approaches, our framework considers a cube shaped PZ surrounding each aircraft. This idea resonates with the fact that our approach creates virtual data cubes around grid points, forming an overall airspace. Each cube is defined by its centroid, the original grid point, and associated weather parameters that remain homogeneous within the cube during a period of

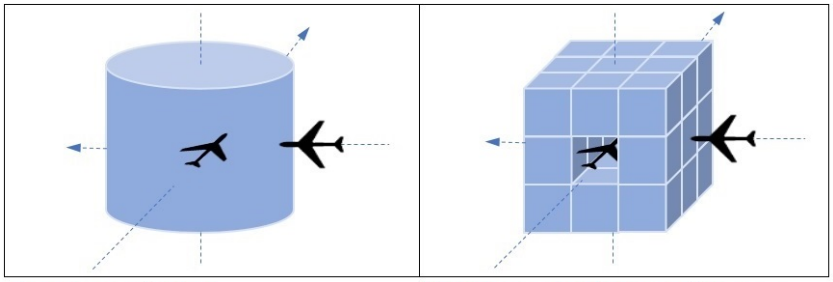


Fig. 7.1: A sample crossing conflict and a PZ. (left) regular representation, (right) our unique representation.

time. With this vision, we define trajectories as a set of 4D joint cubes.

This uncommon representation of 4D trajectories enables us to view conflicts and PZ from a unique perspective. Hence, in our view, PZ is a cube that can be expanded by joining the neighboring cubes horizontally and vertically. The process yields a PZ with a desired size. In our study, we use a PZ of variable size. It can be made up of a single cube or expanded by a number of cubes in each direction on each axis reaching a larger volume. Figure 7.1 illustrates a PZ in two different forms. The PZ on the left is formed by a cylinder. The PZ on the right is formed by 27 cubes. Figure 7.2 illustrates a sample pairwise CDR. Aircraft #1 departs before aircraft #2. Both aircraft move one cube at a time. In Figure 7.2a. aircraft #1 and #2 are located at cube $J7$ and $K6$, respectively. This causes a conflict as aircraft #2 intrudes aircraft #1's PZ outlined in red. In Figure 7.2b. the conflict is resolved by a lateral shift to cube $L6$ by aircraft #2. No conflict occurs from here on as aircraft #1 follows cubes in gray ($K7, L7, M7, N7, O7, P7, R7, S7$) and aircraft #2 follows cubes in blue ($M7, N8, N9, N10, O11, P11, R11, S11, T11$) until they land at their pertinent airports.

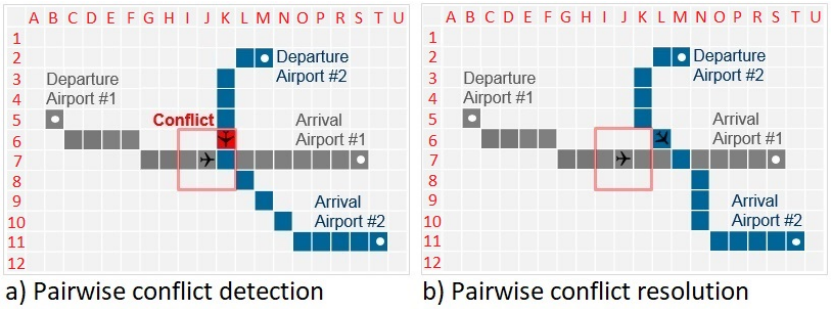


Fig. 7.2: Simplified depiction of a pairwise CDR.

7.4 Prescriptive Analytics System for CDR

Figure 7.3 shows an overview of the proposed Data-Driven Framework for a simplified pairwise CDR. Predicted trajectory #1 and #2 are generated by our previous Aircraft Trajectory Prediction System [46]. Our conflict detection algorithm takes predicted trajectories along with separation minima as input. The output of the process is a data cube defined by its 4D position, where conflict, if any, occurs. The next process in the pipeline is the conflict resolution, where a variant of the Viterbi algorithm is performed to avoid the conflicting trajectory segment. The process perturbs at least one of the trajectories involved in the conflict, generating a new optimized path and thereby resulting in conflict-free trajectories. Separation minima may be expanded to probe what-if scenarios if the conflict detection algorithm doesn't generate a conflict.

7.4.1 Pairwise Conflict Detection

The predicted trajectories along with the size of the PZ are fed into our pairwise conflict detection algorithm, presented in Algorithm 7.1. The algorithm declares a conflict if a PZ of an aircraft is infringed upon by another aircraft at the same time interval in the future.

Formally, given a pair of predicted trajectories $T_i = [p_{i1}, p_{i2}, ..., p_{im}]$ and $T_j = [p_{j1}, p_{j2}, ..., p_{jn}]$ where each trajectory is formed by a set of segments defined by their 4D spatio-temporal centroid

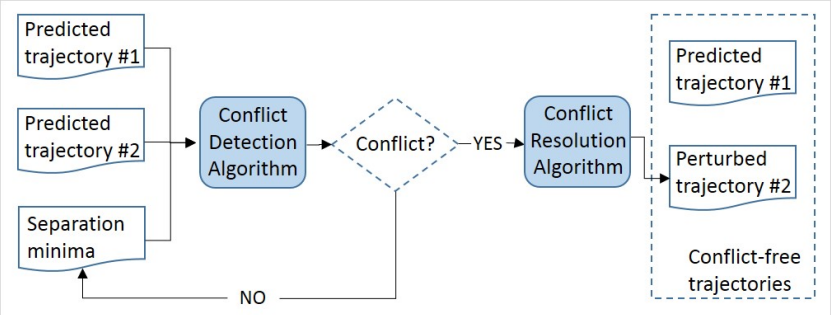


Fig. 7.3: Overview of our Data-Driven Framework.

parameters *latitude, longitude, altitude*, and *timestamp*, along with PZ for trajectory T_i in data cubes, we want to return the very first trajectory segment p_{js}, if a conflict occurs, *null* otherwise. Note that each aircraft position along predicted trajectories are recorded once every minute. To compute this, we start with the departure point of the aircraft that departs beforehand, and keep moving forward, one grid point at a time. With the departure of the second aircraft, we compare the trajectory segment p_{js} with PZ_{is} at each time instance t_s to see if p_{js} overlaps with PZ_{is}.

Note that Algorithm 7.1 assumes that all the PZs have the same definition.

7.4.2 Pairwise Conflict Resolution

Our conflict resolution approach shares a common ground with our previous Trajectory Prediction System [46] as they both attempt to address an optimization problem; *given a set of weather observations, what is the most likely sequence of aircraft positions*? However, unlike the previous system [46], the current conflict resolution algorithm avoids the spatio-temporal data cubes where the conflict is detected. To recapitulate our approach to the optimization problem, we briefly review our previous Trajectory Prediction System [46] here.

Given a set of historical trajectories along with pertinent weather observations, the system works based upon an assumption that the weather observations are realizations of hidden aircraft positions i.e. trajectory segments and the transitions between the underlying hidden segments fol-

Algorithm 7.1: Pairwise Conflict Detection

Result: Detected conflict or no conflict

Input : Trajectory pairs T_i, T_j, Protected Zone PZ_i

Output: Conflicting trajectory segment p_{js} or *null*

```
T_i ← [p_i1, p_i2, ..., p_im]
T_j ← [p_j1, p_j2, ..., p_jn]
foreach t_s ∈ (p_is ∩ p_js) do
    if p_js ⊂ PZ_is then
        return p_js
    end
end
return null
```

lowing a Hidden Markov model [154]. This assumption considers a finite set of states, each of which is associated with a probability distribution over all possible trajectory segments. Transitions among the states are managed by a set of probabilities. The states are not visible, but the pertinent observations are. Given a sequence of observations, the system trains an HMM, a statistical Markov model, and derives a sequence of hidden states, aircraft positions that correspond to the sequence of weather observations. The system computes the most likely sequence of aircraft positions in three steps:

- In the training data processing step, the system transforms raw trajectories into aligned trajectories and combines weather parameters for each grid point along aligned trajectories. To achieve this, the system uses a 3D grid network with a spatial resolution of 6km x 6km as a reference system.

- In the test data processing step, the system resamples the weather parameters to generate buckets with distinct ranges and feeds them into the time series clustering algorithm [78] to produce input observations.
- In the final step, the HMM parameters generated in the first two steps and the flight time computed by our Estimated Time of Arrival (ETA) Prediction System [201] are used as input to the Viterbi algorithm. The output is the optimal state sequence, joint 4D cubes defining aircraft trajectories.

During the conflict resolution stage, our current framework makes use of the first two steps, outlined above. However, unlike the 3rd step of the process, our current framework avoids the cubes where the conflict is detected. This translates to a perturbation of at least one of the trajectories. Hence, the framework prescribes an optimized solution by perturbing at least one of the 4D trajectories, involved in the conflict. The process executes as follows: In addition to the regular HMM parameters of transition, emission, and initial probabilities, the framework uses conflict-free probabilities where each state is assigned a probability value indicating how conflict-free it is. The parameters are fed into a variant of the Viterbi Algorithm, in which the framework computes the optimal state sequence by considering the maximum HMM probabilities. Due to fact that the trajectory segments that are part of the first aircraft's trajectory are assigned low conflict-free probabilities, the framework avoids selecting them during the Viterbi process, yielding a conflict-free trajectory for the second aircraft.

Now, we present a variant of the Viterbi algorithm. Note that our previous Trajectory Prediction System [46] characterized an HMM by the following elements:

- N, the number of states in the model. States $S = \{S_1, S_2, ..., S_N\}$ are represented by reference points' coordinates *(latitude, longitude, altitude)* that form aligned trajectories. We denote state at time t as q_t.

- M, the number of distinct observation symbols per state. Observations $V = \{v_1, v_2, ..., v_M\}$ are represented by weather parameters *(temperature, wind speed, wind direction, humidity)* recorded at grid points.

- The state transition probability distribution $A = \{a_{ij}\}$ is the probability of an aircraft discretely transitioning from one state i to another j along its aligned trajectory, where

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \leq i, j \leq N$$

- The observation symbol probability distribution in state j, $B = b_j(k)$ is the probability of discrete weather parameters having been observed at that specific state, where

$$b_j(k) = P[v_k \; at \; t | q_t = S_j], \quad \begin{matrix} 1 \leq j \leq N \\ 1 \leq k \leq M \end{matrix}$$

- The initial state distribution $\pi = \{\pi_i\}$ is the probability of an aligned trajectory beginning at a state i, where

$$\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N$$

These parameters form an HMM, compactly denoted by $\lambda = \{A, B, \text{and } \pi\}$. Now, we propose an additional parameter;

- The conflict-free probability distribution in state j, $C = c_j(k)$ is the probability of a conflict not occurring at that specific state, where

$$c_j(k) = P[v_k \; at \; t | q_t = S_j], \quad \begin{matrix} 1 \leq j \leq N \\ 1 \leq k \leq M \end{matrix}$$

Hence, the lower the conflict-free probability for a particular state, the lower likelihood of that state to be included in the most probable path. With this new parameter, C an HMM can be expanded and denoted by $\lambda = \{A, B, \pi \text{ and } C\}$

The next step in the process is to choose a corresponding state sequence $Q = q_1, q_2, ..., q_T$ that best explains the observation sequence $O = O_1, O_2, ..., O_T$ given the model λ. A variant of the Viterbi algorithm [122] that is based on dynamic programming addresses this problem. The key component in the algorithm is the *optimal probability*, $\delta_t(j)$, and is computed as follows:

$$\delta_t(j) = \max_{q_1,...,q_{t-1}} \pi_{q_1} b_{q_1}(o_1) c_{q_1}(o_1) \prod_{j=2}^{t} (a_{q_{j-1},q_j} b_{q_j}(o_j) c_{q_j}(o_j))$$

Due to their low conflict-free probabilities, a variant of the Viterbi algorithm avoids trajectory segments where the conflict is detected, and generates a new optimized path.

7.4.3 CDR for Multiple Aircraft

Our pairwise CDR solution can be scaled to address CDR for multiple aircraft which can be used towards better planning of airspace sector densities. Similar to the current flight planning procedures, we propose predicted trajectories to be processed on a first come first served basis. For the sake of simplicity, consider an empty airspace. The airspace will be fully available to the first predicted trajectory. Hence, the first trajectory's optimal set of data cubes will be reserved in the airspace. This process will introduce a set of constraints in the form of PZs by the first trajectory to the second and following trajectories during its flight time. Any conflict between the first and second trajectory will be detected and resolved. Once resolved, a new set of constraints will be introduced by the second trajectory. The next trajectory will need to satisfy the constraints introduced by the previous trajectories and so on. This also means that the next trajectory will need to avoid the PZs in the constraints list while generating its optimal set of data cubes during the conflict resolution phase. This process is repeated until one or more flights land, which will result in the removal of all pertinent constraints from the list which will free up the particular sections of airspace.

Formally, given a new predicted trajectory $T_j = [p_{j1}, p_{j2}, ..., p_{jl}]$ and a time series of existing constraints list $CL_{PZ} = [PZ_1, PZ_2, ..., PZ_i, ..., PZ_m]$, where $PZ_i = [pz_{i1}, pz_{i2}, ..., pz_{ik}]$ along their existing trajectories $T = [T_1, T_2, ..., T_i, ..., T_n]$, where $T_i = [p_{i1}, p_{i2}, ..., p_{ik}]$, we want to detect and resolve conflicts among these trajectories T_j and T (i.e. *one vs. all*). Note that PZ_i is formed by a set of data cubes, each defined by 4D spatio-temporal parameters around its centroid p_{ij}. With this approach, the implementation and management of the constraints list is of central importance. In our implementation, we map time instances to data cubes forming PZs, which are stored in an Octree data structure, where there is one Octree for each time instance. Hence, when a new trajectory comes in, the pertinent Octree is located based on the trajectory's time instance. We declare a conflict if the trajectory segment is found in the Octree. We process all pairs of possibly conflicting data cubes using spatial indexing techniques to prune the search (e.g., [202, 203]).

Our scalable conflict detection algorithm for multiple aircraft is presented in Algorithm 7.2. To keep the constraints list up-to-date, we periodically check and delete the pertinent Octree if any of the time instances has expired. Once a conflict has been detected, a variant of the Viterbi algorithm is performed where all the data cubes included in the constraints list for the pertinent time instance are avoided to find a conflict-free, optimized path for the new trajectory. To achieve this, we assign a minimum conflict-free probability value, such as 1×10^{-100} to those data cubes included in the constraints list.

Figure 7.4 is an illustration of a simplified case where two consecutive trajectory segments of aircraft #1 and #2 are inserted into two separate Octree data structures at time instances *t* and *t+1*. For the sake of simplicity, the size of aircraft #1's *PZ* is considered as a single data cube. Assuming an empty airspace, the very first *PZ* set for aircraft #1 is allocated in the airspace without being considered for a potential conflict. This translates to the fact that an Octree for time instance *t* and another Octree for time instance *t+1* are created as they don't exist yet. Next, given the

Algorithm 7.2: Conflict Detection for Multiple Aircraft

Result: Detected conflict or approved optimal trajectory

Input : A new predicted trajectory T_i, constraints list CL_{PZ}

Output: Conflicting trajectory segment p_i or updated constraints list CL_{PZ}

$T_i \leftarrow [p_{i1}, p_{i2}, ..., p_{in}]$

$CL_{PZ} \leftarrow [PZ_1, PZ_2, ..., PZ_m]$

foreach $t_s \in (p_i \& CL_{PZ})$ **do**

    PruneAndSearch

    if $p_i \subset CL_{PZ}$ **then**

        return p_i

    end

    else

        Insert p_i **into** CL_{PZ}

    end

end

Algorithm 7.3: Conflict Resolution for Multiple Aircraft

Result: A conflict-free optimized trajectory

Input : # of states N, # of distinct observations M, transition probabilities A, emission probabilities B, initial probabilities π, constraints list CL_{PZ}

Output: A conflict-free optimized trajectory T_o and updated constraints list CL_{PZ}

$S \leftarrow [p_1, p_2, ..., p_i, ..., p_n]$

$CL_{PZ} \leftarrow [PZ_1, PZ_2, ..., PZ_m]$

foreach $t_s \in ((p_i \in S)\ \&\ CL_{PZ})$ **do**

  PruneAndSearch

  if $p_i \subset CL_{PZ}$ **then**

    $c_{p_i} \leftarrow 1 \times 10^{-100}$

  end

  else

    $c_{p_i} \leftarrow 1$

  end

end

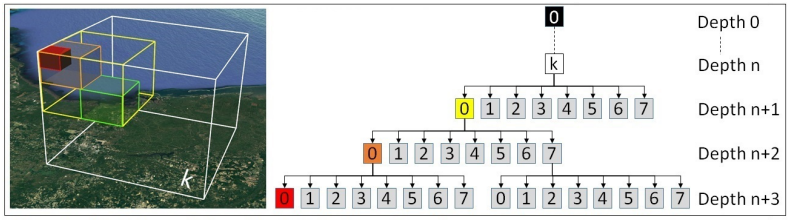
VariantOfViterbi

$T_o \leftarrow \max_{s_1,...,s_{t-1}} \pi_{s_1} b_{s_1}(o_1) c_{s_1}(o_1) \prod_{j=2}^{t} (a_{s_{j-1},s_j} b_{s_j}(o_j) c_{s_j}(o_j))$

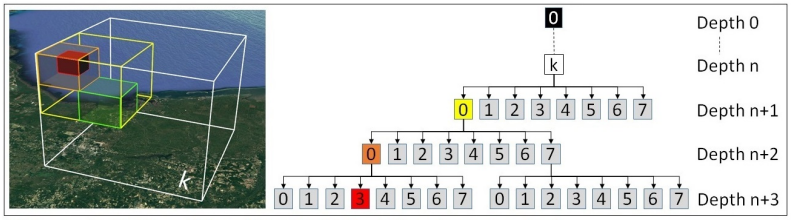
foreach $t_s \in ((p_o \in T_o)\ \&\ CL_{PZ})$ **do**

  Insert p_o **into** CL_{PZ}

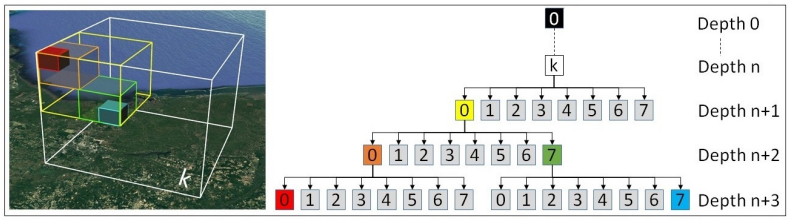
end



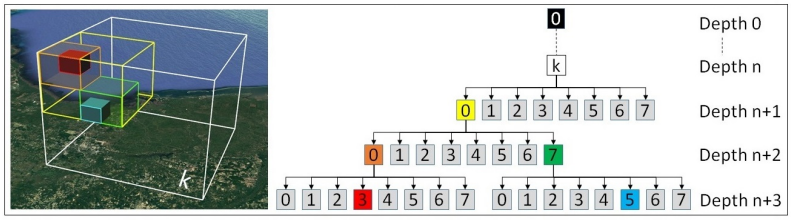
a) A single *PZ* in solid red in OCTREE at time instance *t*.



b) A single *PZ* in solid red in OCTREE at time instance *t+1*.



c) *PZ* in solid red for aircraft #1 and *PZ* in solid blue for aircraft #2 in OCTREE at time instance *t*.



d) *PZ* in solid red for aircraft #1 and *PZ* in solid blue for aircraft #2 in OCTREE at time instance *t+1*.

Fig. 7.4: Octree data structure for efficient aircraft conflict detection and resolution.

predicted trajectory for aircraft #1, each consecutive *PZ* in solid red is inserted into each Octree at time instance *t* and *t+1* as shown in Figures 7.4a and 7.4b, respectively. When the predicted trajectory segments of aircraft #2 in solid blue in Figures 7.4c, and 7.4d are received, the Octree data structure at each time instance *t* and *t+1* are searched to see if the pertinent Octree node is empty. To achieve this, each Octree is looked up based on its index. Next, the pertinent Octree is searched as each time instance is mapped to an Octree. If found empty, each consecutive trajectory segment is inserted into each Octree at time instance *t* and *t+1*, as shown in Figures 7.4c and 7.4d, respectively. The insert operation for the first trajectory segment of aircraft #2 in solid blue is performed by traversing the *node k* at *depth n* illustrated as white cube mesh, followed by the *node 0* at *depth n+1* illustrated as yellow cube mesh, followed by the *node 7* at *depth n+2* illustrated as green cube mesh, and finally reaching to the leaf *node 7* at *depth n+3* and populating it with solid blue as shown in Figure 7.4c. Upon search, if the leaf node is not found empty, which translates to a conflict, our conflict resolution Algorithm 7.3 is executed to perturb the predicted trajectory for aircraft #2 to find an alternative optimal solution that results in conflict-free trajectories.

Formally, given the number of states N, number of distinct observations per state M, state transition probability distribution A, observation probability distribution B, and initial state probability distribution π, along with a conflict-free probability distribution for all data cubes included in the constraints list at time instances $t = [t_1, t_2, ..., t_n]$, we want to form an HMM λ, and train it to find the optimized path that is conflict-free. Our conflict resolution algorithm for multiple aircraft is presented in Algorithm 7.3.

Tab. 7.1: A set of European and U.S.A. airports.

Airport Code	**Airport Name**
LEAL	Alicante–Elche Airport
LEBL	Barcelona–El Prat Airport
LECO	A Coruña Airport
LEIB	Ibiza Airport
LEMD	Adolfo Suárez Madrid-Barajas Airport
LEMG	Málaga Airport
LEMH	Menorca Airport
LEPA	Palma de Mallorca Airport
LEVC	Valencia Airport
LEVX	Vigo–Peinador Airport
LEZL	Seville Airport
KATL	Hartsfield-Jackson Atlanta International Airport
KBOS	Boston Logan International Airport
KDFW	Dallas/Fort Worth International Airport
KJAX	Jacksonville International Airport
KLGA	NewYork LaGuardia Airport
KMIA	Miami International Airport
KORD	Chicago O'Hare International Airport
KPIT	Pittsburgh International Airport

7.5 Evaluation

To evaluate our framework, we generated a number of test cases using real trajectory and weather data from Europe and USA. Table 7.1 shows the European and USA airports forming the routes we used in our evaluation. Although the ideal evaluation would use real trajectories in actual conflicts, this is infeasible due to the nature of ATC operations, where the controller would interfere and separate the aircraft as soon as they are likely to infringe upon one another, so there would be no conflicts to find. Hence, we used a total of 20 test cases formed by European and USA flights that were in close proximity to cause potential conflicts when a minimal perturbation was applied. 16 out of these 20 cases were used to test pairwise CDR, while 4 cases were used to test CDR for multiple aircraft.

Tab. 7.2: Sizes of training and test datasets for pairwise CDR.

TestCase #	**Route #1**	**TrainingSetSize**		**TestSetSize**		**Route #2**	**TrainingSetSize**		**TestSetSize**	
		$\#trjs$	$\#pts$	$\#trjs$	$\#pts$		$\#trjs$	$\#pts$	$\#trjs$	$\#pts$
1	LEAL-LEBL	1118	55116	200	9860	LEMD-LEIB	2572	125623	200	9769
2	LEAL-LEBL	1118	55116	19	937	LEMD-LEMH	1056	68141	19	1226
3	LEAL-LEBL	1118	55116	152	7493	LEPA-LEMD	5116	306128	152	9095
4	LEBL-LEMG	1704	127451	43	3216	LEMD-LEMH	1056	68141	43	2775
5	LEBL-LEMG	1704	127451	180	13463	LEPA-LEMD	5116	306128	180	10771
6	LEBL-LEZL	2404	183343	41	3127	LEMD-LEAM	1434	70128	41	2005
7	LEBL-LEZL	2404	183343	46	3508	LEMD-LEMH	1056	68141	46	2968
8	LEBL-LEZL	2404	183343	164	12508	LEMG-LEMD	1403	75408	164	8815
9	LEBL-LEZL	2404	183343	210	16016	LEPA-LEMD	5116	306128	210	12566
10	LEIB-LEBL	1360	53443	259	10178	LEPA-LEMD	5116	306128	259	15498
11	LEIB-LEBL	1360	53443	158	6209	LEPA-LEVC	1426	50467	158	5592
12	LEMG-LEBL	1563	114767	38	2790	LEMD-LEAM	1434	70128	38	1858
13	LEMG-LEBL	1563	114767	46	3378	LEMD-LEIB	2572	125623	46	2247
14	LEZL-LEBL	2380	186299	40	3131	LEMD-LEIB	2572	125623	40	1954
15	KDFW-KJAX	1355	153970	19	2159	KATL-KMIA	1296	108005	19	1583
16	KLGA-KORD	1155	131454	62	7056	KPIT-KBOS	822	57490	62	4339

Tab. 7.3: Sizes of training and test datasets for multiple aircraft CDR.

$TestCase\#$	$Route\#1$	$TestSetSize$		$Route\#2$	$TestSetSize$		$Route\#3$	$TestSetSize$	
		$\#trjs$	$\#pts$		$\#trjs$	$\#pts$		$\#trjs$	$\#pts$
17	LEBL-LEZL	164	12508	LEMG-LEMD	164	8815	LEPA-LEMD	210	12566
18	LEMG-LEBL	38	2790	LEMD-LEAM	38	1858	LEMD-LEIB	46	2247
19	LEBL-LEMG	30	2244	LEMD-LEMH	30	1935	LEMD-LEPA	30	1795
20	LEBL-LEZL	27	2059	LEMD-LEMH	27	1742	LEMD-LEPA	27	1616

7.5.1 Setup

Due to the fact that there exists no system that continuously records and stores exact positions of an aircraft's original trajectory [74], only a discrete set of sample data are recorded and stored which presumably represent a close approximation of the original trajectory. We call this a raw trajectory. The raw trajectory data from Europe was provided by Spanish ANSP, ENAIRE using a radar surveillance feed with a 5 seconds update rate. The raw data was wrangled as part of the Data-driven AiRcraft Trajectory prediction research (DART) project under the SESAR Joint Undertaking Work Programme [75]. The European trajectory data contains all commercial domestic flights for Spain, a total of 119,563 raw trajectories and 80,784,192 raw trajectory points for the period of January through November 2016. The fields of the raw trajectory data are as follows: *flight no, departure airport, arrival airport, date, time, aircraft speed* in *X, Y, Z* directions, and position information *(latitude, longitude, altitude)*. Note that, as a preprocessing step, we downsampled raw trajectory data from the original resolution of 5 seconds to 60 seconds and aligned them to our 3D reference grid [46]. The raw trajectory data from the USA was extracted from an Aircraft Situation Display to Industry (ASDI) data feed which is recorded once every 60 seconds, provided in near real-time by the FAA [14], and stored in an aviation data warehouse [59]. The USA trajectory data contains flights between 8 major airports, a total of 4,628 raw trajectories and 450,919 raw trajectory points for the period of May 2010 through December 2015. The fields of the raw trajectory data are as follows: *source center, date, time, aircraft Id, speed, latitude, longitude* and *altitude*. Both European and USA weather data were extracted from the Global Forecast System (GFS), provided by NOAA [76]. The original data has 28-km spatial and 6-hour temporal resolution and it contains over 40 weather parameters including *atmospheric, cloud* and *ground* attributes for each grid point as part of its 3D weather model. Hence, for this

study's geographic volume and time period of interest, over 160TB of weather data was collected.

Due to fact that our current framework aims at addressing the CDR problem before departure, at least a pair of predicted trajectories are needed as input. Hence, we used our previous system [46] to generate a pair set of predicted trajectories. Next, we searched and found a number of trajectory points between the two flights' trajectories in the first and second pair, where the *date, latitude, longitude,* and *altitude* values matched, and *time* mismatched. By perturbing one of the flights' departure time we *virtually* created conflicts, where both aircraft traversed the same trajectory point at the same time. Table 7.2 shows the final size of training and test data in number of trajectories ($\#trjs$) and points ($\#pts$).

To evaluate our framework on multiple CDR test cases, we generated a set of three flights that is a combination of the existing routes listed in Table 7.2. The additional test cases for multiple CDR with their final size of test data in number of trajectories ($\#trjs$) and points ($\#pts$) are listed in Table 7.3. Table 7.3 does not include training data for each test case as they are already included in Table 7.2. Note that test case #17 and #18 use two separate *virtual* conflicts in each test case, where each conflict is addressed in sequence, i.e., first the *virtual* conflict between route #1 and route #2 is detected and resolved, next the *virtual* conflict between route #1 and #3 is detected and resolved. Unlike test case #17 and #18, test case #19 and #20 address multiple CDR all at once, i.e., all three routes share a common trajectory point. Hence, to form a set of three flights for test case #19 and #20, we searched and found a number of trajectory points between these three flights' trajectories, where the *date, latitude, longitude,* and *altitude* values matched, and *time* mismatched. Assuming that the predicted trajectories were received in the order of route #1, #2, and #3 respectively, first we perturbed the 2nd flights' departure time to create a *virtual* conflict between the first and second flight, which is detected and resolved using our multiple CDR approach. As an outcome, route #2 was perturbed at the conflict location, yielding a common

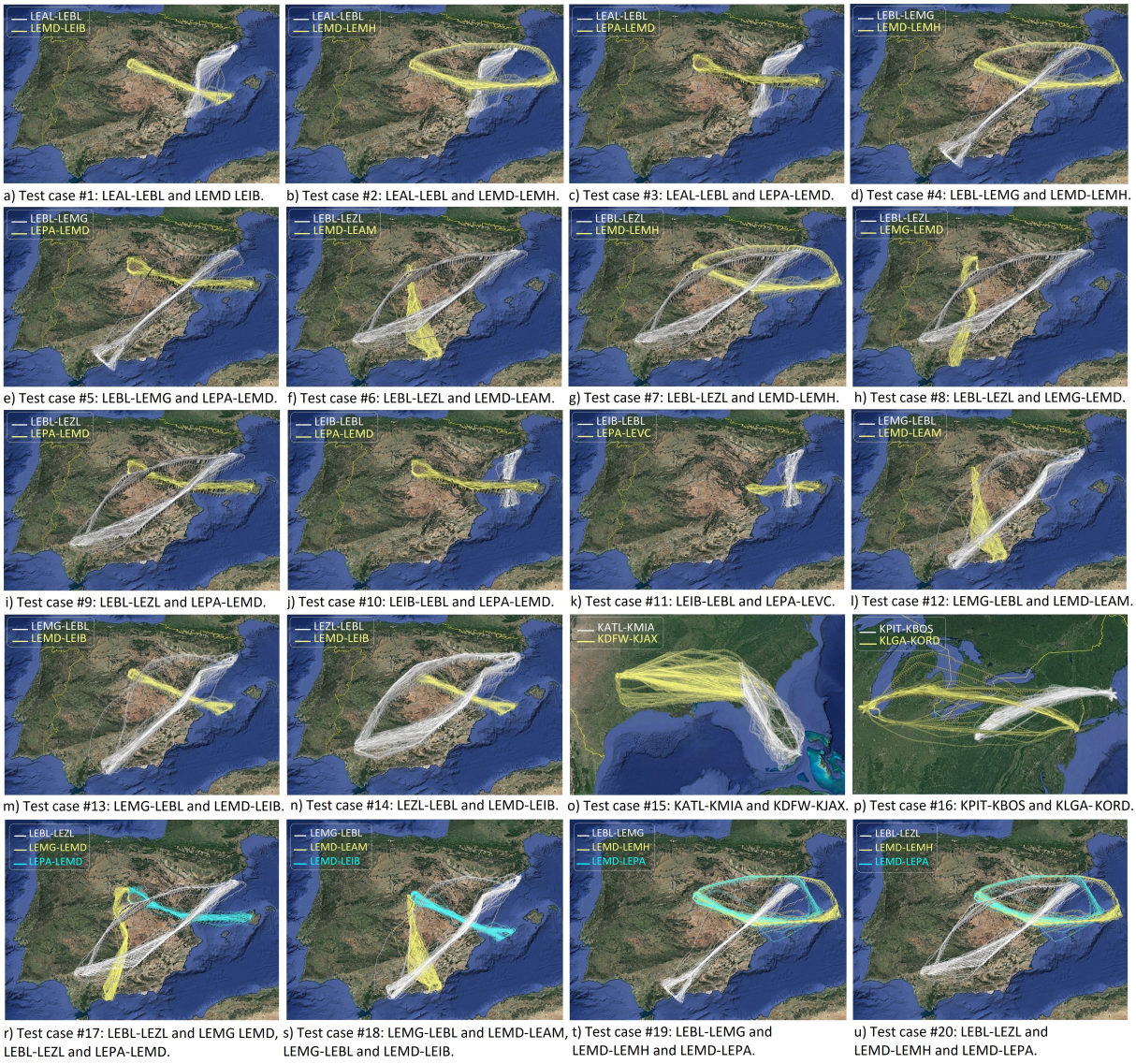


Fig. 7.5: Visual representation of training and test data for conflicting flights.

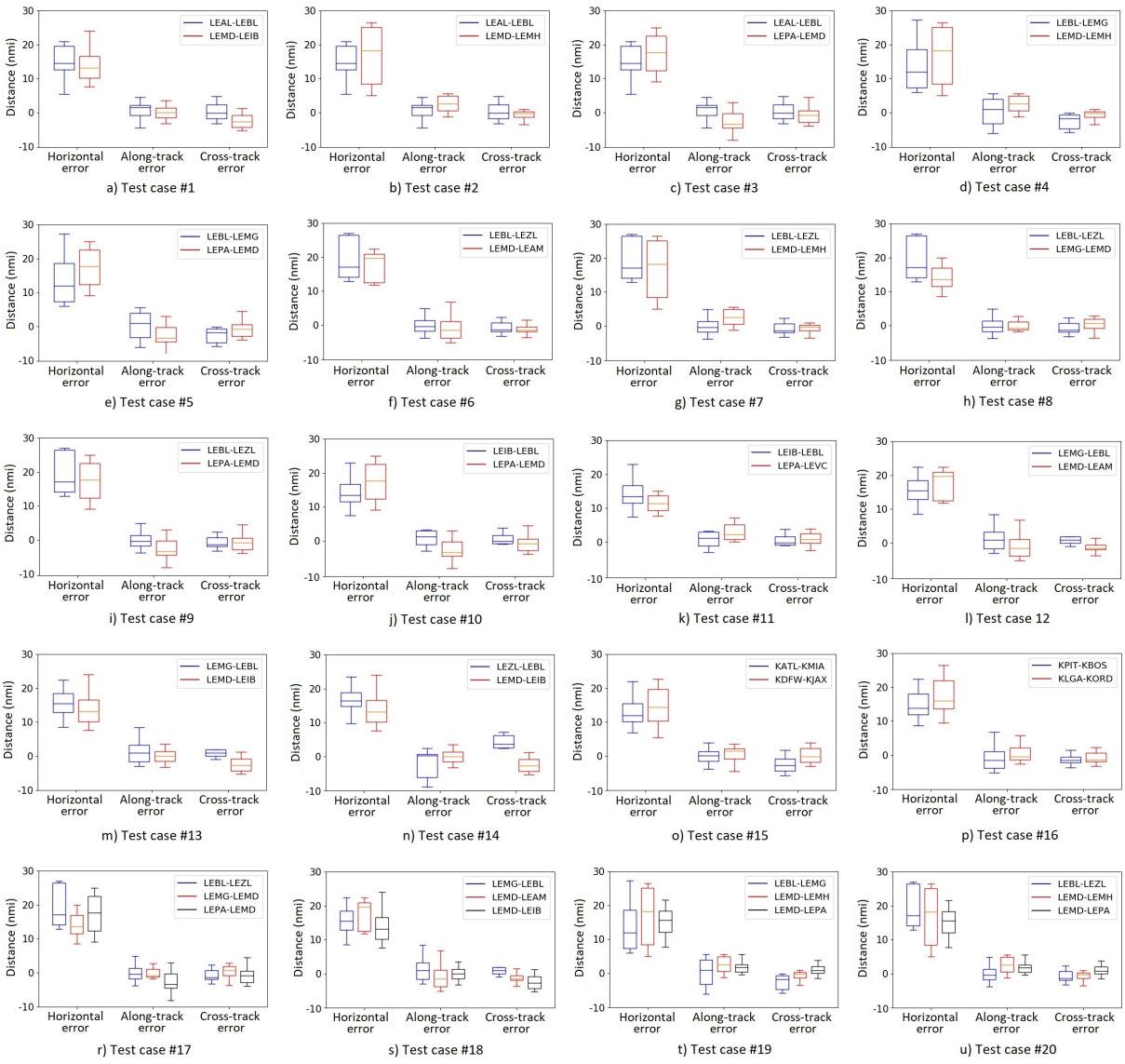


Fig. 7.6: Trajectory prediction errors in the form of box plots.

trajectory point between route #1 and route #3. Next, we perturbed the 3rd flights' departure time to create a *virtual* conflict between the first and third flight, which is also detected and resolved using our multiple CDR approach. In all multiple CDR test cases listed in Table 7.3, we used an Octree data structure and executed Algorithm 7.2 and Algorithm 7.3.

Note that overall, the trajectory data alone contains over 4 million trajectory points. In Tables 7.2 and 7.3, the test data represents the conflicting trajectories. Aside from these 1,677 trajectory pairs, we also bootstrapped by drawing 100 additional trajectory pairs with replacement from the trajectory set to test for the false positive cases. Hence, we evaluated our framework's effectiveness with a total of 3,554 (3,354 + 200) test trajectories on 16 test cases for pairwise CDR and an additional set of 1,087 (916 + 171) test trajectories formed by three flights on 4 test cases for multiple CDR.

Figures 7.5a through 7.5p illustrate the pairwise test cases with their cumulative training and test data in white and yellow, respectively, whereas Figures 7.5r through 7.5u illustrate the test cases of three flights with their cumulative training and test data in white, yellow, and blue, respectively.

7.5.2 Results

We evaluated our framework by comparing the output of each step through the framework's pipeline with the ground truth on all 20 test cases. These steps include conflict detection and conflict resolution. However, due to fact that the accuracy of conflict detection and resolution significantly depend on the accuracy of predicted trajectories, we also computed trajectory prediction errors. For that purpose, we used trajectory prediction accuracy metrics as outlined in [152] and computed horizontal and vertical errors e_{horiz}, e_{vert} based on predicted trajectories generated by our previous system [46] versus raw trajectories of pertinent flights. The box plots in Figure 7.6

Tab. 7.4: Horizontal and vertical error ranges for each bin size.

Condition	**BinSize**
0nmi $\leq e_{horiz} \leq$ 5nmi $\wedge$ 0ft $\leq e_{vert} \leq$ 2000ft	5nmi,2000ft
5nmi $< e_{horiz} \leq$ 10nmi $\vee$ 2000ft $< e_{vert} \leq$ 4000ft	10nmi,4000ft
10nmi $< e_{horiz} \leq$ 15nmi $\vee$ 4000ft $< e_{vert} \leq$ 6000ft	15nmi,6000ft
15nmi $< e_{horiz} \leq$ 20nmi $\vee$ 6000ft $< e_{vert} \leq$ 8000ft	20nmi,8000ft

capture the aggregated errors for horizontal, along-track, and cross-track errors for each flight in all 20 test cases. The horizontal error is unsigned whereas the along-track, cross-track, and vertical errors are signed errors. Note that the mean value for the cross-track error and vertical error along the entire test trajectories is 7.692nmi and 1589.452ft respectively, when the sign is omitted.

To evaluate our conflict detection capability, we used the same accuracy metrics as in [152] and computed horizontal and vertical errors by comparing the output of conflict detection and conflict resolution of our framework with the ground truth on all 20 test cases. To compute the errors, we fed a pair set of predicted trajectories for test case #1 through #16 and a set of three predicted trajectories for test case #17 through #20 into our conflict detection algorithm and compared the locations of *virtual* conflicts versus locations of conflicts detected by our framework. Next, we created 4 bin sizes, where each bin size is an integer multiple of 5nmi of lateral and 2000ft of vertical distances, conventionally accepted as minimum separation values for enroute airspace by ANSPs.

Table 7.4 presents the pertinent condition for each bin. Using horizontal and vertical error values, we counted the number of conflicts in each case and found the bin size to which they belong. The outcome is presented as a set of histograms in Figure 7.7. Our algorithm detected **87.1%** of the conflicts within the first bin size and **99%** of the conflicts within the first two bin sizes on all 20 test cases. Note that the conflict detection can only be as accurate as the predicted

trajectories. Hence, these errors are attributed to the accuracy of our previous system [46], defined by the horizontal, and vertical error of **14.981nmi** and **1589.452ft** respectively along the entire test trajectories.

To resolve the conflicts, we ran our conflict resolution algorithm. Figure 7.8 provides a closer look at one of the detected and resolved conflicts by our framework for a pairwise CDR test case. In all figures, the yellow cubes and white cubes represent the first and second flight's, respectively, predicted trajectories. The red line parallel to the white cubes represents the first fight's actual trajectory and the red line parallel to the yellow cubes represents the second flight's actual trajectory. As both aircraft move one cube at a time in the flight direction, the PZ illustrated in the cyan cube around the current position of the first flight also moves forward in the form of a sliding window. The first figure on the far left shows where each aircraft is at time t_{s-1}, represented respectively by the solid white cubes for the first and yellow cubes for the second flight. The second figure from the left captures the conflict detected at time interval t_s by our framework illustrated with a solid red cube. The actual conflict occurs where the red lines intersect. Note that both the predicted conflict position and the actual conflict position are within the first flight's PZ. The third figure from the left is the 3D view of the second figure from the left. The actual and predicted conflict positions are only 2 cube sizes away from each other, considering the center of the cube as the predicted position of the aircraft. The figure in the far right illustrates the optimized solution to the conflict by our framework. The first flight's trajectory has been perturbed vertically and the conflict has been resolved by our framework. The altitude of the second flight has been elevated resulting in conflict being resolved. Note that due to assignment of low conflict-free probabilities to the 3D grid points inside of the PZ, the framework avoids selecting them during the conflict resolution stage, generating conflict free trajectories.

As the final evaluation step, we executed our conflict resolution algorithm on all conflicts

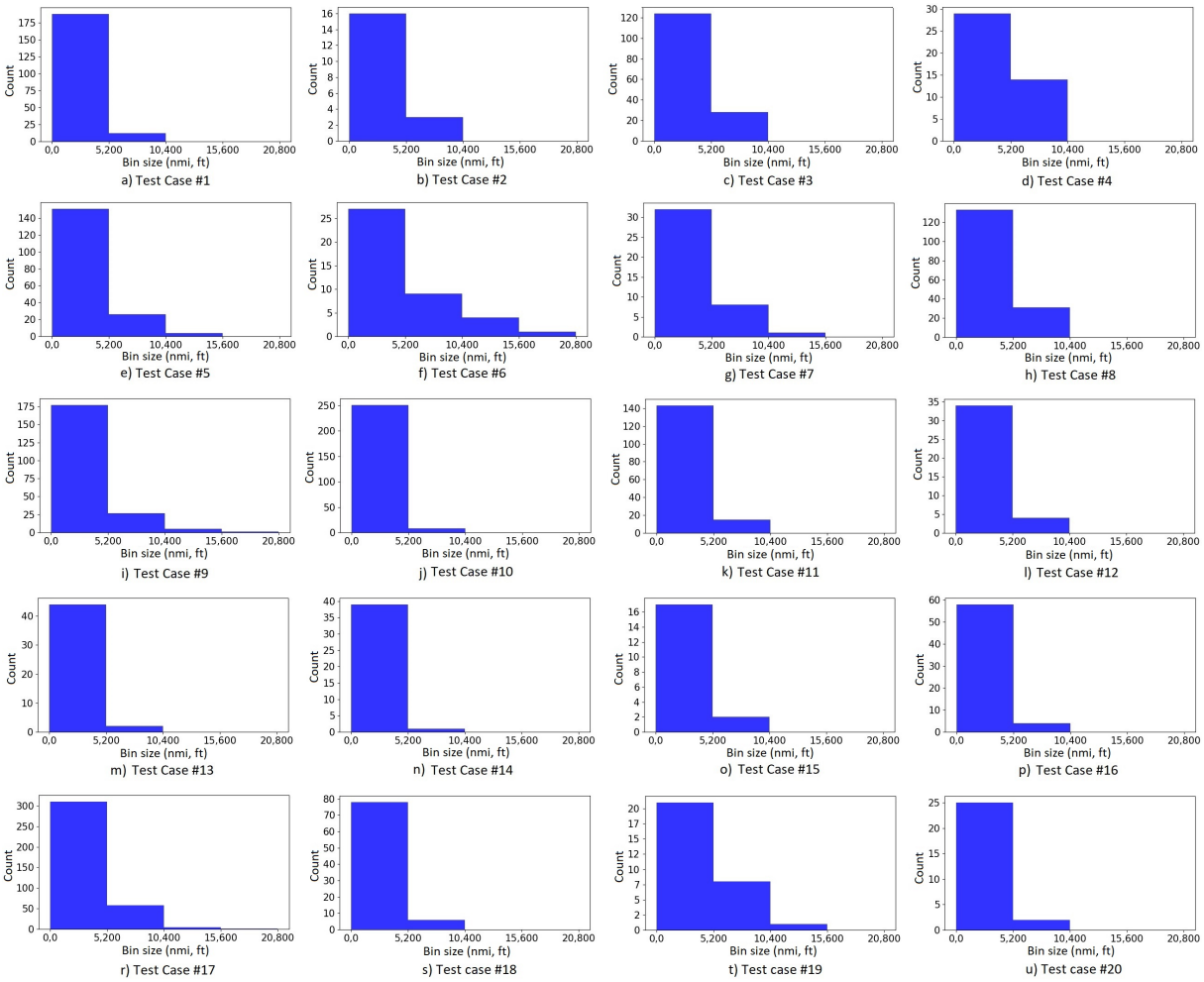


Fig. 7.7: Conflict detection histograms showing how many conflicts were captured by our framework in each bin size.

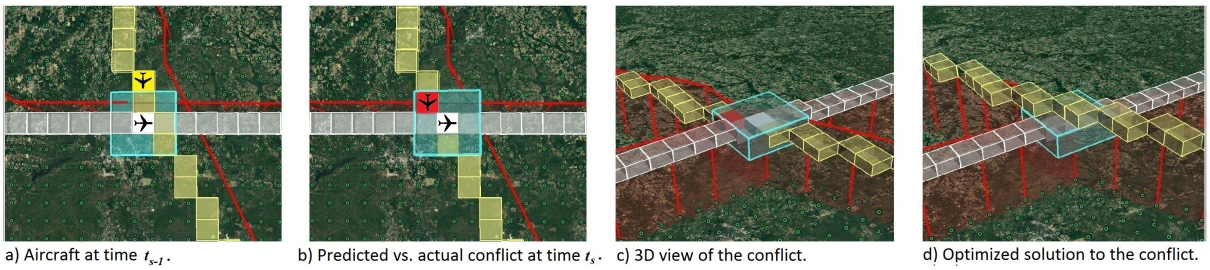


Fig. 7.8: Illustration of a detected and resolved pairwise conflict by our framework.

formed by a total of 4,641 (3,554 + 1,087) trajectories and computed accuracy values based on the number of successful resolutions, i.e. generating a feasible safe alternative trajectory by fulfilling the separation minima criteria. Due to the fact that our conflict resolution algorithm resolved the vast majority of conflicts on all test cases, we provide aggregated results overall, rather than provide results for each test case. Table 7.5 presents accuracy values for each bin size. Note that, to resolve the conflicts, we treated each bin size differently based on their varying sizes so that only relevant 3D grid points were avoided. The process perturbed the trajectory for the latter flight, generated an optimal alternative and yielded conflict-free trajectories.

7.6 Discussion

During the trajectory prediction step, errors were computed by comparing the locations of *virtual* conflicts versus locations of conflicts detected by our framework. The trajectory prediction had mean horizontal and vertical errors of **14.981nmi** and **1589.452ft**, respectively along the entire trajectory points over the full test set from two continents. Using the predicted trajectories, our conflict detection algorithm had **6.012nmi** of mean horizontal error. Note that this value is considerably less than 14.981nmi, the mean horizontal error by our previous Trajectory Prediction System [46] along the entire trajectory points including climb, cruise and descent phases of a

flight. This is due to two major facts: 1) Trajectory prediction accuracy during the cruise phase is often considerably higher than the climb and descent phases of the flight. 2) All conflicts we used in our experiments took place during the cruise phase of the flights. With 6.012nmi of mean horizontal error on the conflict positions, our conflict detection algorithm found **99%** of conflicts within the first two bin sizes of separation minima (10nmi, 4000ft). We also verified that between none of the additional 100 trajectory pairs where the *virtual* conflict never occurred was falsely detected as a conflict by our framework. Our conflict resolution algorithm's mean accuracy was over **97%** on all test cases. Though, it is interesting to see that it was unable to reach 100% accuracy, given the fact that all it had to do was avoid the selected 3D grid points when generating the new conflict-free optimal trajectory. The reason for that was the sparse distribution of data cubes, i.e. lack of data over the 3D grid network. The algorithm was unable to connect the new trajectory segments from start to end due to disconnection. Hence, our framework craves for more data to reach higher accuracy values.

These results validate the effectiveness of our framework on the long-range CDR problem. However, this is not to say that strategic CDR will detect and resolve conflicts once for all and no more conflicts will occur during the flight. There will likely be some convective weather patterns shaping after departure. These sudden changes causing potential conflicts should be addressed tactically by short and or medium-range CDR systems while the aircraft is airborne.

Although we were not able to find, there may also likely be some exceptional cases where false positive conflicts may be found. These cases should also be addressed tactically by short and or medium-range CDR systems. Note that the larger the selected PZ value, the higher probability of finding and resolving the conflicts between trajectories. However, that also means less denser sectors resulting in inefficient use of airspace. Hence, the tradeoff should be handled carefully by the ANSPs.

Tab. 7.5: Accuracy of our conflict resolution algorithm based on each bin size.

BinSizeId	Size	Accuracy
1	5nmi,2000ft	98.3%
2	10nmi,4000ft	97.5%
3	15nmi,6000ft	93.7%
4	20nmi,8000ft	100.0%

Overall, we propose our framework to be used by AOCs to file more realistic flight plans. Our framework can also be used by ANSPs to validate that the filed flight plans do not cause conflicts with previously approved flight plans or increase any sector traffic complexities. Air traffic density and complexity (a count of aircraft predicted to be in conflict with another) are two major factors defining the metric of air traffic controller workload [13]. These goals can be achieved in the planning phase before the aircraft depart, resulting in improvement in the four ATM key performance areas; safety, capacity, efficiency, and environmental impact, and thereby improving ATM automation and reducing the air traffic controller workload.

7.7 Conclusion

We have presented a novel Data-Driven Framework addressing a long-range CDR problem. Using a set of predicted trajectories, the framework delivers two major capabilities; *i)* conflict detection, and *ii)* conflict resolution before the flight depart. In the conflict detection stage, the framework declares a conflict when a PZ of an aircraft on its predicted trajectory is infringed upon by another aircraft at the same time interval in the future. In the conflict resolution stage, upon a conflict, the framework executes our conflict resolution algorithm that is derived from the Viterbi algorithm and prescribes a solution that resolves the conflict by perturbing at least one of the 4D trajectories.

Our experiments on real trajectory and weather datasets from two continents verify that our framework achieves lateral and vertical accuracies that are within the boundaries of conventionally accepted minimum separation values, set by the ANSPs. This translates to the fact that, ANSPs can now detect and resolve potential conflicts long before the aircraft depart, resulting in safer and greener skies with higher efficiency and capacity, and thereby reducing the air traffic controller workload.

Part V

Concluding Remarks

Chapter 8

Conclusion and Future Directions

8.1 Conclusion

In this thesis, we studied the characteristics of aircraft trajectories and developed several data-driven models for (1) ETA prediction (2) aircraft trajectory clustering (3) aircraft trajectory prediction, and (4) aircraft conflict detection and resolution, long before aircraft depart. The suite of analytics models runs on a scalable data management system that continuously processes streaming massive flight data to achieve the strategic airspace planning for optimal capacity, efficiency, and safety.

In Chapter 2, we looked at streaming massive flight data provided by the FAA, which is the major data source of our data-driven models. We presented a design and implementation of an efficient data management system including a scalable aviation data warehouse, which is the underlying infrastructure for our descriptive, predictive, and prescriptive analytics solutions presented in this thesis. We emphasized the fact that our efficient data management system processed an average of 34 million ASDI messages each day, yielding over ten billions of messages each year being stored in our aviation data warehouse.

In Chapter 3, we aimed at addressing ETA prediction problem for commercial flights, which is a major input to our data-driven model for aircraft trajectory prediction. Unlike other systems that collect and use features only for the arrival airport, we used a richer set of features along the potential route, such as weather parameters and air traffic data in addition to those that were particular to the arrival airport. Our feature construction process generated an extensive set of multidimensional time series data which went through Time Series Clustering with Dynamic Time Warping (DTW) to generate a single set of representative features at each time instance. We also presented algorithms for two major features that highly contributed to the accurate ETA prediction: airspace sector congestion rate, and airport congestion rate. Next, we performed a comparative analysis using a number of regression models and an RNN and presented their performances. Using the best performing model, we compared our results with those generated by the EUROCONTROL. Our experiments on real trajectory, weather, air traffic and airport data verified that our system outperformed EUROCONTROL's ETA prediction by offering not only a higher accuracy but also a far smaller standard deviation, resulting in smaller prediction windows of flight arrival times.

In Chapter 4, we studied historical aircraft trajectories, grouped similar trajectories and discovered a representative trajectory for all as a single entity. During the process, we found out that considering a trajectory as a whole may mislead, resulting in overfitting and a failure to discover a representative trajectory. Hence, we clustered aircraft trajectories based on a well-known aviation domain fact: each flight is made up of three major phases: climb, enroute, and descent. We divided trajectories into these three phases and clustered each phase in isolation, then merged them together. We also presented a representative trajectory algorithm that revealed the trajectory model based on lateral and vertical smoothing.

In Chapter 5, we analyzed timeseries of weather observations for the airspace volume of

interest. To predict aircraft trajectories, HMM requires a set of input observations, which are unknown, although the weather parameters overall are known for the pertinent airspace. Hence, given an extensive set of weather parameters for the overall airspace of interest, we computed time series clustering on the current weather observations and generated one cluster centroid for each time instance along the potential path. Unlike previous research, which used standard clustering techniques to generate clusters that are largely independent of the time series from which they originate, we generated an optimal sequence of weather observations used in accurate trajectory prediction. Our algorithm computed a cost value for each weather observation based on its frequency for each time instance, ranked, and stored them in a matrix. Using Dynamic Programming (DP), the algorithm computed the optimal sequence of weather observations, a set with the minimum cumulative cost that satisfied the continuity constraint.

In Chapter 6, we created a new conception of airspace to accurately predict aircraft trajectories. We considered airspace as a 3D grid network, where each grid point was a location of a weather observation. We hypothetically built cubes around these grid points, so the entire airspace could be considered as a set of cubes. Each cube was defined by its centroid, the original grid point, and associated weather parameters that remained homogeneous within the cube during a period of time. Then, we aligned raw trajectories to a set of cube centroids which were basically fixed 3D positions independent of trajectory data. This created a new form of trajectories which were 4D joint cubes, where each cube was a segment that was associated with not only spatio-temporal attributes but also with weather parameters. Next, we exploited machine learning techniques to train inference models from historical data and applied a stochastic model, an HMM, to predict trajectories taking environmental uncertainties into account. During the process, we leveraged our time series clustering algorithm to generate input observations from an excessive set of weather parameters to feed into the Viterbi algorithm. The experiments with real trajectory and weather

data showed that our trajectory prediction system generated mean horizontal error of less than 13 nautical miles (nmi) and mean vertical error of less than 700 feet (ft).

Lastly, in Chapter 7, we proposed a cube-shaped protected zone surrounding each aircraft that could be expanded by joining the neighboring cubes horizontally and vertically, yielding a protected zone with a desired size. This idea resonated with the assumption, which we presented in the previous chapter that an airspace was nothing more than a set of concatenated spatio-temporal data cubes around a 3D grid network, where each cube was considered as an atomic unit. We also proposed an algorithm for pairwise conflict detection. We used an Octree data structure to store and manage 4D aircraft trajectories, and presented a conflict detection algorithm for two or more conflicting aircraft. Finally, we presented a conflict resolution algorithm for two or more conflicting aircraft, which used the same Octree data structure. Hence, overall, we presented a scalable Prescriptive Analytics System to strategically address the aircraft CDR problem. Given a set of predicted trajectories, the system declared a conflict when a protected zone of an aircraft on its trajectory was infringed upon by another aircraft at the same time interval in the future. Upon a conflict, the system executed our conflict resolution algorithm and prescribed a solution that resolved the conflict by perturbing at least one of the 4D trajectories involved in the conflict. Our experiments showed that our long-range conflict detection and resolution system achieved lateral and vertical accuracies that were within the boundaries of conventionally accepted minimum separation values, set by the ANSPs.

Altogether, this thesis establishes a comprehensive understanding of a new way of conceptualizing airspace and uses this understanding to plan airspace for optimal capacity, efficiency and safety. We develop a suite of five models and algorithms to effectively predict accurate 4D aircraft trajectories and detect potential conflicts among them long before they depart. This translates to the fact that ANSPs can now efficiently plan a safer and greener airspace with more efficiency and

capacity, and thereby reducing the air traffic controller workload.

8.2 Future Directions

There are several important research directions that will further improve the understanding of 4D aircraft trajectories and their allocations within a particular airspace. In this thesis, we focused on commercial air traffic mostly in class A airspace. With the expected inclusion of drones in the next few years, demand for all airspace classes including class A, B, C, D, and E will increase and that will require strategic planning of airspace for more efficiency, capacity, and safety. In 2015, the FAA has deployed En Route Automation Modernization (ERAM) [204] at 20 FAA ARTCCs nationwide to monitor conformance of enroute flights. However, ERAM supports conformance monitoring for only 1,900 flights for each ARTCC and that will highly likely be insufficient due to the estimated growth of commercial air traffic along with expected inclusion of massive number of unmanned aircraft in the NAS. Hence, this will give a rise to design and implementation of a scalable data management and analytics systems and new computational models to efficiently manage air traffic of millions of manned and unmanned aircraft in the NAS.

Bibliography

[1] Faa taf. http://taf.faa.gov/Downloads/TAFSummaryFY2015-2040.pdf, 2015. Accessed: 2019-01-19.

[2] National airspace system metrics. https://www.faa.gov/nextgen/snapshots/nas/?fuel-burn-display-type=fiscalYear, 2018. Accessed: 2019-01-19.

[3] Measuring flight efficiency in the nas. https://www.mitre.org/sites/default/files/publications/15-3106-measuring-flight-efficiency-national-airspace-system.pdf, 2018. Accessed: 2019-01-19.

[4] Michael Ball, Cynthia Barnhart, Martin Dresner, Mark Hansen, Kevin Neels, Amedeo Odoni, Everett Peterson, Lance Sherry, Antonio Trani, and Bo Zou. Total delay impact study - a comprehensive assessment of the costs and impacts of flight delay in the united states. Technical report, The National Center of Excellence for Aviation Operations Research, College Park, MD, October 2010.

[5] Faa ato. https://www.faa.gov/air_traffic/by_the_numbers/media/Air_Traffic_by_the_Numbers_2018.pdf, 2018. Accessed: 2019-01-19.

[6] Faa nextgen. https://www.faa.gov/nextgen/, 2009. Accessed: 2019-01-19.

[7] Single european sky atm research. http://www.eurocontrol.int/dossiers/single-european-sky, 1999. Accessed: 2019-01-19.

[8] Faa tfm. https://www.fly.faa.gov/Products/Training/Traffic_Management_for_Pilots/TFM_in_the_NAS_Booklet_ca10.pdf, 2009. Accessed: 2019-01-19.

[9] Jerry A. Guttman, Parimal Kopardekar, Richard H. Mogford, and S. L. Morrow. The complexity construct in air traffic control: A review and synthesis of the literature. Technical report, Federal Aviation Administration, Atlantic City, NJ, January 1995.

[10] Parimal Kopardekar and Sherri Magyarits. Dynamic density: Measuring and predicting sector complexity. In *2002 IEEE/AIAA 21st Digital Avionics Systems Conference*, Irvine, CA, October 2002.

[11] Banavar Sridhar, Kapil S. Sheth, and Shon Grabbe. Airspace complexity and its application in air traffic management. Technical report, NASA Ames Research Center, Moffett Field, CA, December 1998.

[12] Manuel Soler, Alberto Olivares, Ernesto Staffetti, and Jesus Cegarra. Multi-phase optimal control applied to 4d business trajectory strategic planning in air traffic management. In *Proceedings of the 1st Int'l Conference on ATACCS*, pages 68–78, Barcelona, Spain, May 2011.

[13] Irene V. Laudeman, Stephen G. Shelden, R. Branstrom, and J L. Brasil. Dynamic density: An air traffic management metric. Technical report, NASA Ames Research Center, Mountain View, CA, April 1998.

[14] Faa asdi. http://www.fly.faa.gov/ASDI/, 2016. Accessed: 2019-01-19.

[15] Aircraft situation display to industry: Functional description and interface control document for the xml version. http://www.fly.faa.gov/ASDI/asdi.html, 2011.

[16] Charles Bontempo and George Zagelow. The IBM data warehouse architecture. *Communications of the ACM*, 41(9):38–48, September 1998.

[17] IBM - data warehouse - infosphere. http://www-01.ibm.com/software/data/infosphere/warehouse/, 2012.

[18] IBM - websphere message broker. http://www-01.ibm.com/software/integration/wbimessagebroker/, 2012.

[19] IBM - websphere mq. http://www-01.ibm.com/software/integration/wmq/, 2012.

[20] SPSS predictive analytics. http://www-01.ibm.com/software/analytics/spss/, 2012.

[21] IBM - cognos business intelligence. http://www-01.ibm.com/software/analytics/cognos/business-intelligence/, 2012.

[22] Danyel Fisher, Rob DeLine, Mary Czerwinski, and Steven Drucker. Interactions with big data analytics. *Interactions*, 2012.

[23] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM SIGMOD Record*, 26(1):65–74, March 1997.

[24] William H. Inmon. *Building the Data Warehouse*. John Wiley and Sons, New York, 2002.

[25] Maurizio Lenzerini, Panos Vassiliadis, Matthias Jarke, and Yannis Vassiliou. *Fundamentals of Data Warehouses*. Springer-Verlag, Heidelberg, Germany, 2003.

[26] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit*. John Wiley and Sons, New York, 2002.

[27] Jennifer Widom. Research problems in data warehousing. In *Proceedings of the 4th Int'l Conference on Information and Knowledge Management*, pages 25–30, 1995.

[28] Angelo Bonifati, Fabiano Cattaneo, Stefano Ceri, Alfonso Fuggetta, and Stefano Paraboschi. Designing data marts for data warehouses. *ACM Transactions on Software Engineering and Methodology*, 10(4):452–483, October 2001.

[29] Sergio Luján-Mora and Juan Trujillo. Physical modeling of data warehouses using uml. In *Proceedings of the 7th ACM Int'l Workshop on Data Warehousing and OLAP*, pages 48–57, 2004.

[30] Subi Arumugam, Alin Dobra, Christopher M. Jermaine, Niketan Pansare, and Luis Perez. The datapath: A data-centric analytic processing engine for large data warehouses. In *Proceedings of the 2010 ACM SIGMOD Int'l Conference on Management of Data*, pages 519–530, 2010.

[31] Jin Li, Kristin Tufte, Vladislav Shkapenyuk, Vassilis Papadimos, Theodore Johnson, and David Maier. Out-of-order processing: A new architecture for high-performance stream systems. In *Proceedings of the VLDB Endowment*, pages 274–288, 2008.

[32] Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: A new model and architecture for data stream management. *ACM Transactions on Software Engineering and Methodology*, 12(2):120–139, August 2003.

[33] Vinayak Borkar, Michael J. Carey, and Chen Li. Inside 'big data management': Ogres, onions, or parfaits. In *Proceedings of the 15th Int'l Conference on Extending Database Technology*, pages 3–14, 2012.

[34] Florin Rusu and Alin Dobra. Glade: A scalable framework for efficient analytics. *ACM SIGOPS Operating Systems Review Archive*, 46(1):12–18, January 2012.

[35] Dongwon Lee and Wesley W. Chu. Constraints-preserving transformation from xml document type definition to relational schema. In *Proceedings of the 19th Int'l Conference on Conceptual Modeling*, pages 323–338, 2000.

[36] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, and Jeffrey Naughton. Relational databases for querying xml documents: Limitations and opportunities. In *Proceedings of the 25th Int'l Conference on Very Large Databases*, pages 302–314, 1999.

[37] Daniela Florescu and Donald Kossman. Storing and querying xml data using rdbms. *IEEE Data Engineering Bulletin*, 1999.

[38] Matteo Golfarelli, Stefano Rizzi, and Boris Vrdoljak. Data warehouse design from XML sources. In *Proceedings of the 3rd Int'l Workshop on Data Warehousing and OLAP*, pages 40–47, 2001.

[39] Jeffrey Cohen, Brian Dolan, Mark Dunlap, Joseph M. Hellerstein, and Caleb Welton. MAD skills: New analysis practices for big data. In *Proceedings of the VLDB Endowment*, pages 1481–1492, 2009.

[40] Michael Stonebraker, Ugur Çetintemel, and Stan Zdonik. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, 34(4):42–47, December 2005.

[41] Sailesh Krishnamurthy, Michael J. Franklin, Jeffrey Davis, Daniel Farina, Pasha Golovko, Alan Li, and Neil Thombre. Continuous analytics over discontinuous data streams. In *Proceedings of the 2010 ACM SIGMOD Int'l Conference on Management of Data*, pages 1081–1092, 2010.

[42] Ricardo Jorge Santos and Jorge Bernardino. Optimizing data warehouse loading procedures for enabling useful-time data warehousing. In *Proceedings of the 2099 Int'l Database Engineering and Applications Symposium*, pages 292–299, 2009.

[43] Surajit Chaudhuri. What next?: A half-dozen data management research goals for big data and the cloud. In *Proceedings of the 31st symposium on Principles of Database Systems*, pages 1–4, 2012.

[44] Aviation weather center. http://www.aviationweather.gov/adds/metars/, 2012.

[45] Rapid refresh. http://rapidrefresh.noaa.gov/, 2012.

[46] Samet Ayhan and Hanan Samet. Aircraft trajectory prediction made easy with predictive analytics. In *Proceedings of the 22nd ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 21–30, San Francisco, CA, August 2016.

[47] Benjamin Letham, Lydia M. Letham, and Cynthia Rudin. Bayesian inference of arrival rate and substitution behavior from sales transaction data with stockouts. In *Proceedings of the 22nd ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, August 2016.

[48] Wei-Hua Lin and Jian Zeng. Experimental study of real-time bus arrival time prediction with gps data. *Journal of the Transportation Research Board*, 1666:101–109, April 1999.

[49] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, Linhong Zhu, Rose Yu, and Yan Liu. Latent space model for road networks to predict time-varying traffic. In *Proceedings of the 22nd ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, August 2016.

[50] Ioannis Parolas, Ron Van Duin, Ioanna Kourounioti, and Lorant A. Tavasszy. Prediction of vessels' eta using machine learning a port of rotterdam case study. Technical report, Transportation Research Board, Washington, DC, January 2017.

[51] Jimmy Krozel, Changkil Lee, and Joseph Mitchell. Estimating time of arrival in heavy weather conditions. In *AIAA GNC Conference and Exhibit*, Portland, OR, August 1999.

[52] Benjamin Levy and David Rappaport. Arrival time estimation (eta) from on-final to gate. In *AIAA ATIO Conference*, Belfast, Northern Ireland, September 2007.

[53] Paul T. Wang, Craig R. Wanke, and Frederick P. Wieland. Modeling time and space metering of flights in the national airspace system. In *Proceedings of the Winter Simulation Conference*, Washington, DC, December 2004.

[54] Joel Klooster, Keith Wichman, and Okko Bleeker. Strategic aircraft trajectory prediction uncertainty and statistical sector traffic load modeling. In *AIAA GNC Conference and Exhibit*, Honolulu, HI, August 2008.

[55] Xiaoli Bai, Lesley A. Weitz, and Stephanie Priess. Evaluating the impact of estimated time of arrival accuracy on interval management performance. In *AIAA GNC Conference and Exhibit*, San Diego, CA, August 2016.

[56] Jian Wei, Jooyoung Lee, and Inseok Hwang. Estimated time of arrival prediction based on state-dependent transition hybrid estimation algorithm. In *AIAA GNC Conference and Exhibit*, Kissimmee, FL, August 2015.

[57] Haomiao Huang, Kaushik Roy, and Claire Tomlin. Probabilistic estimation of state-dependent hybrid mode transitions for aircraft arrival time prediction. In *AIAA GNC Conference and Exhibit*, Hilton Head, SC, August 2007.

[58] Kaushik Roy, Benjamin Levy, and Claire Tomlin. Target tracking and estimated time of arrival (eta) prediction for arrival aircraft. In *AIAA GNC Conference and Exhibit*, Keystone, CO, August 2006.

[59] Samet Ayhan, Johnathan Pesce, Paul Comitz, Gary Gerberick, and Steve Bliesner. Predictive analytics with surveillance big data. In *Proceedings of the 1st ACM SIGSPATIAL Int'l Workshop on Analytics for Big Geospatial Data*, November 2012.

[60] Christian S. Kern, Ivo P. de Medeiros, and Takashi Yoneyama. Data-driven aircraft estimated time of arrival prediction. In *Proceedings of the 9th IEEE Int'l Systems Conference (SysCon)*, Vancouver, BC, April 2015.

[61] Gabor Takacs. Predicting flight arrival times with a multistage model. In *IEEE Int'l Conference on Big Data*, Keystone, CO, October 2014.

[62] Yan Glina, Richard Jordan, and Mariya Ishutkina. A tree-based ensemble method for prediction and uncertainty quantification of aircraft landing times. In *Proceedings of the 10th Conference on Artificial and Computational Intelligence and its Applications to the Environmental Sciences*, New Orleans, LA, January 2012.

[63] Karthik Gopalakrishnan and Hamsa Balakrishnan. A comparative analysis of models for predicting delays in air traffic networks. In *12th USA/Europe Air Traffic Management Research and Development Seminar*, Seattle, WA, June 2017.

[64] Jagan Sankaranarayanan and Hanan Samet. Roads belong in databases. *IEEE Data Engineering Bulletin*, 33(2):4–11, June 2010.

[65] Jagan Sankaranarayanan, Hanan Samet, and Houman Alborzi. Path oracles for spatial networks. *PVLDB*, 2(1):1210–1221, August 2009.

[66] Sarana Nutanong and Hanan Samet. Memory-efficient algorithms for spatial network queries. In *Proceedings of the 29th IEEE Int'l Conference on Data Engineering*, pages 649–660, Brisbane, Australia, April 2013.

[67] Shangfu Peng, Jagan Sankaranarayanan, and Hanan Samet. SPDO: High-throughput road distance computations on spark using distance oracles. In *Proceedings of the 32nd IEEE Int'l Conference on Data Engineering*, pages 1239–1250, Helsinki, Finland, May 2016.

[68] Jagan Sankaranarayanan, Houman Alborzi, and Hanan Samet. Efficient query processing on spatial networks. In *Proceedings of the 13th ACM Int'l Symposium on Advances in Geographic Information Systems*, pages 200–209, Bremen, Germany, November 2005.

[69] Jagan Sankaranarayanan and Hanan Samet. Distance oracles for spatial networks. In *Proceedings of the 25th IEEE Int'l Conference on Data Engineering*, pages 652–663, Shanghai, China, March 2009.

[70] Jagan Sankaranarayanan and Hanan Samet. Query processing using distance oracles for spatial networks. *IEEE Transactions on Knowledge and Data Engineering*, 22(8):1158–1175, August 2010.

[71] Edwin Jacox and Hanan Samet. Metric space similarity joins. *ACM Transactions on Database Systems*, 33(2):7, June 2008.

[72] Sarana Nutanong, Edwin Jacox, and Hanan Samet. An incremental Hausdorff distance calculation algorithm. *PVLDB*, 4(8):506–517, August 2011.

[73] Jagan Sankaranarayanan, Houman Alborzi, and Hanan Samet. Distance join queries on spatial networks. In *Proceedings of the 14th ACM Int'l Symposium on Advances in Geographic Information Systems*, pages 211–218, Arlington, VA, November 2006.

[74] Samet Ayhan and Hanan Samet. Diclerge: Divide-cluster-merge framework for clustering aircraft trajectories. In *Proceedings of the 8th ACM SIGSPATIAL IWCTS*, Seattle, WA, November 2015.

[75] SESAR. *Data-Driven Aircraft Trajectory Prediction Research*, 2017.

[76] NCEP global forecast system. https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs, 2016. Accessed: 2019-01-19.

[77] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3):358–386, March 2005.

[78] Samet Ayhan and Hanan Samet. Time series clustering of weather observations in predicting climb phase of aircraft trajectories. In *Proceedings of the 9th ACM SIGSPATIAL IWCTS*, pages 25–30, San Francisco, CA, November 2016.

[79] Samet Ayhan, Brendan Fruin, Fan Yang, and Michael O. Ball. Normstad flight analysis: Visualizing air traffic patterns over the united states. In *Proceedings of the 7th ACM SIGSPATIAL IWCTS*, pages 1–10, Dallas/Fort Worth, TX, November 2014.

[80] Samet Ayhan, Paul Comitz, and Ron LaMarche. Implementing geospatially enabled aviation web services. In *2008 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pages 1–8, Herndon, VA, May 2008.

[81] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York Inc., New York, 2001.

[82] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997.

[83] Harris Drucker. Improving regressors using boosting techniques. In *Proceedings of the 14th Int'l Conference on Machine Learning*, San Francisco, CA, July 1997.

[84] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, October 2001.

[85] Jerome H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, February 2002.

[86] Next generation air transportation system, nextgen. https://www.faa.gov/nextgen/. [Online; accessed 10-October-2015].

[87] Single european sky aair traffic management research, sesar. http://www.sesarju.eu/. [Online; accessed 10-October-2015].

[88] Mihael Ankerst, Markus Breunig, Hans-Peter Kriegel, and Jórg Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, New York, NY, June 1999.

[89] Martin Ester, Hans peter Kriegel, Jorg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, OR, August 1996.

[90] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, Beijing, China, June 2007.

[91] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, September 2006.

[92] Cynthia Sung, Dan Feldman, and Daniela Rus. Trajectory clustering for motion prediction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012*, Vilamoura, Portugal, October 2012.

[93] Wei Wang, Jiong Yang, and Richard Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, San Francisco, CA, August 1997.

[94] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, June 1996.

[95] Binh Han, Liu Ling, and Edward Omiecinski. Neat: Road network aware trajectory clustering. In *IEEE 32nd International Conference on Distributed Computing Systems (ICDCS), 2012.*, Macau, China, June 2012.

[96] Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. Spatio-temporal similarity analysis between trajectories on road networks. In *Perspectives in Conceptual Modeling*, pages 280–289. Springer Berlin Heidelberg, Berlin, Germany, 2005.

[97] Gook-Pil Roh and Seung won Hwang. Nncluster: An efficient clustering algorithm for road network trajectories. In *Database Systems for Advanced Applications*, pages 47–61. Springer Berlin Heidelberg, Berlin, Germany, 2010.

[98] Yalin Wang, Qilong Han, and Haiwei Pan. A clustering scheme for trajectories in road networks. In *Advanced Technology in Teaching - Proceedings of the 2009 3rd Int'l Conference on Teaching and Computational Science (WTCS 2009)*, pages 11–18. Springer Berlin Heidelberg, Berlin, Germany, 2012.

[99] Jung-Im Won, Sang-Wook Kim, Ji-Haeng Baek, and Junghoon Lee. Trajectory clustering in road network environment. In *IEEE Symposium on Computational Intelligence and Data Mining, 2009. CIDM '09.*, Nashville, TN, March 2009.

[100] Chris Brinton and Stephen Pledgie. Airspace partitioning using flight clustering and computational geometry. In *IEEE/AIAA 27th Digital Avionics Systems Conference, 2008. DASC 2008.*, St. Paul, MN, December 2008.

[101] Maxime Gariel, Ashok Srivastava, and Eric Feron. Trajectory clustering and an application to airspace monitoring. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1511–1524, December 2011.

[102] Kenneth Leiden and Stephen Atkins. Trajectory clustering for metroplex operations. In *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, Virginia Beach, VA, September 2011.

[103] Adric Eckstein. Automated flight track taxonomy for measuring benefits from performance based navigation. In *Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS '09.*, Arlington, VA, May 2009.

[104] Yulin Liu, Mark Hansen, David J. Lovell, and Michael O. Ball. Predicting aircraft trajectoroy choice - a nominal route approach. In *Int'l Conference on Research in Air Transportation*, Barcelona, Spain, June 2018.

[105] Mini batch k-means. http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html. [Online; accessed 10-October-2015].

[106] Haversine formula. http://rosettacode.org/wiki/Haversine_formula. [Online; accessed 10-October-2015].

[107] Aircraft situation display to industry. http://www.fly.faa.gov/ASDI/asdi.html. [Online; accessed 10-October-2015].

[108] Ncep wmo grib2 documentation. http://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc.shtml. [Online; accessed 10-October-2015].

[109] Silhouette coefficient. http://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient. [Online; accessed 10-October-2015].

[110] Google Earth. https://www.google.com/earth//, 2017. Accessed: 2017-02-07.

[111] Giulio Avanzini. Frenet-based algorithm for trajectory prediction. *Journal of Guidance, Control, and Dynamics*, 27(1):127–135, January 2004.

[112] Jose Benavides, John Kaneshige, Shivanjli Sharma, Ramesh Panda, and Mieczyslaw Steglinski. Implementation of trajectory prediction function for trajectory based operations. In *AIAA Atmospheric Flight Mechanics Conference*, Atlanta, GA, 2014.

[113] Jesper Bronsvoort, Greg McDonald, Javier Lopez-Leones, and Hendrikus Visser. Improved trajectory prediction for air traffic management by simulation of guidance logic and inferred aircraft intent using existing data-link technology. In *AIAA GNC Conference and Exhibit*, Minneapolis, MN, August 2012.

[114] Jimmy Krozel and Dominick Andrisani. Intent inference and strategic path prediction. In *AIAA GNC Conference and Exhibit*, San Francisco, CA, August 2005.

[115] Hollis Ryan and Mike Paglione. State vector based near term trajectory prediction. In *AIAA GNC Conference and Exhibit*, Honolulu, HI, 2008.

[116] Yu Liu and X Rong Li. Intent based trajectory prediction by multiple model prediction and smoothing. In *AIAA GNC Conference and Exhibit*, New Orleans, LA, January 2015.

[117] Ioannis Lymperopoulos, John Lygeros, and Andrea Lecchini. Model based aircraft trajectory prediction during takeoff. In *AIAA GNC Conference and Exhibit*, Keystone, CO, August 2006.

[118] Stephane Mondoloni. A multiple-scale model of wind-prediction uncertainty and application to trajectory prediction. In *6th AIAA Aviation Technology, Integration and Operations Conference*, Wichita, KS, September 2006.

[119] Charles Schultz, David Thipphavong, and Heinz Erzberger. Adaptive trajectory prediction algorithm for climbing flights. In *AIAA GNC Conference and Exhibit*, Minneapolis, MN, August 2012.

[120] Javier Lovera Yepes, Inseok Hwang, and Mario Rotea. An intent-based trajectory prediction algorithm for air traffic control. In *AIAA GNC Conference and Exhibit*, San Francisco, CA, August 2005.

[121] Javier Lovera Yepes, Inseok Hwang, and Mario Rotea. New algorithms for aircraft intnet inference and trajectory prediction. *Journal of Guidance, Control, and Dynamics*, 30(2):370–382, March 2007.

[122] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[123] Eamonn Keogh, Jessica Lin, and Wagner Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *Proceedings of the Third IEEE International Conference on Data Mining*, Washington, DC, November 2003.

[124] T. Warren Liao. Clustering of time series data-a survey. *Pattern Recognition*, 38(11):1857–1874, November 2005.

[125] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 2011.

[126] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[127] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in data : An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[128] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, 1981.

[129] Raghu Krishnapuram, Anupam Joshi, Olfa Nasraoui, and Liyu Yi. Low-complexity fuzzy relational clustering algorithms for web mining. *IEEE Transactions on Fuzzy Systems*, 9(4):595–607, 2001.

[130] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer Society Press*, 32(8):68–75, August 1999.

[131] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: An efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, WA, June 1998.

[132] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Quebec, June 1996.

[133] Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, OR, August 1996.

[134] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, Philadelphia, PA, June 1999.

[135] Wei Wang, Jiong Yang, and Richard Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, Athens, Greece, August 1997.

[136] Peter Cheeseman and John Stutz. Advances in knowledge discovery and data mining. chapter Bayesian Classification (AutoClass): Theory and Results, pages 153–180. American Association for Artificial Intelligence, Menlo Park, 1996.

[137] Gail A. Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37(1):54–115, April 1987.

[138] T. Kohonen, M. R. Schroeder, and T. S. Huang. *Self-Organizing Maps*. Springer-Verlag, Secaucus, NJ, 2001.

[139] Mahesh Kumar, Nitin R. Patel, and Jonathan Woo. Clustering seasonality patterns in the presence of errors. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, August 2002.

[140] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. *SIGMOD Rec.*, 45(1):69–76, June 2016.

[141] Robert H. Shumway. Time-frequency clustering and discriminant analysis. *Statistics and Probability Letters*, 63(3):307–314, June 2003.

[142] C. Guo, H. Jia, and N. Zhang. Time series clustering based on ica for stock data analysis. In *In Proc. 4th International Conference on Wireless Communications, Networking and Mobile Computing*, China, October 2008.

[143] G. Verdoolaege and Y. Rosseel. Activation detection in event-related fmri through clustering of wavelet distributions. In *IEEE International Conference on Image Processing*, Hong Kong, September 2010.

[144] Michail Vlachos, Jessica Lin, Eamonn Keogh, and Dimitrios Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In *In Proc. Workshop on Clustering High Dimensional Data and Its Applications*, San Francisco, May 2003.

[145] S. Gao, Y. He, and H. Chen. Wind speed forecast for wind farms based on arma-arch model. In *In Proc. Int'l Conference on Sustainable Power Generation and Supply*, Nanjing, China, April 2009.

[146] Yupei Lin and Yiwen Yang. Stock markets forecasting based on fuzzy time series model. In *IEEE International Conference on Intelligent Computing and Intelligent Systems*, Shanghai, China, November 2009.

[147] Y. Yang and K. Chen. Time series clustering via rpcl network ensemble with different representations. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(2):190–199, March 2011.

[148] Pierre Hansen and Brigitte Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79(1-3):191–215, October 1997.

[149] François Petitjean and Pierre Gançarski. Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment. *Theory Computer Science*, 414(1):76–91, January 2012.

[150] Evgenia Dimitriadou, Andreas Weingessel, and Kurt Hornik. A combination scheme for fuzzy clustering. In *Proceedings of the 2002 AFSS International Conference on Fuzzy Systems. Calcutta: Advances in Soft Computing*, London, UK, February 2002.

[151] Chester Gong and David McNally. A methodology for automated trajectory prediction analysis. In *AIAA GNC Conference and Exhibit*, Providence, RI, August 2004.

[152] Mike Paglione and Robert Oaks. Implementation and metrics for a trajectory prediction validation methodology. In *AIAA GNC Conference and Exhibit*, Hilton Head, SC, August 2007.

[153] Ralf Hartmut Güting and Markus Schneider. Realm-based spatial data types: The rose algebra. *The VLDB Journal*, 4(2):243–286, April 1995.

[154] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann, San Francisco, CA, 1990.

[155] Gano Chatterji. Short-term trajectory prediction methods. In *AIAA GNC Conference and Exhibit*, Portland, OR, August 1999.

[156] Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on Intelligent Transportation Systems*, 43(4):509–521, April 1998.

[157] Enrique Casado, Colin Goodchild, and Miguel Vilaplana. Identification and initial characterization of sources of uncertainty affecting the performance of future trajectory management automation systems. In *Proceedings of the 2nd Int'l Conference on ATACCS*, pages 170–175, Toulouse, France, May 2012.

[158] K. Tysen Mueller, John A. Sorensen, and George J. Couluris. Strategic aircraft trajectory prediction uncertainty and statistical sector traffic load modeling. In *AIAA GNC Conference and Exhibit*, Monterey, CA, August 2012.

[159] Sip Swierstra. Common trajectory prediction capability for decision support tools. In *Proceedings of ATM 2003, 5th USA/Europa R&D Seminar*, Budapest, Hungary, June 2003.

[160] E. Crisostomi, A. Lecchini-visintini, and J. Maciejowski. Combining monte carlo and worst-case methods for trajectory prediction in air traffic control: A case study. *Journal of Guidance, Control and Dynamics*, 43(4), 2008.

[161] Stephane Mondoloni. A genetic algorithm for determining optimal flight trajectories. In *AIAA GNC Conference and Exhibit*, Boston, MA, August 1998.

[162] Hok Kwan Ng, Shon Grabbe, and Avijit Mukherjee. Design and evaluation of a dynamic programming flight routing algorithm using the convective weather avoidance model. In *AIAA GNC Conference and Exhibit*, Chicago, IL, August 2009.

[163] Tim Stewart, Lucy Askey, and Mary Hokit. A concept for tactical reroute generation, evaluation and coordination. In *12th AIAA ATIO Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Indianapolis, IN, September 2012.

[164] Tim Stewart, James DeArmon, , and David Chaloux. Using flight information to improve weather avoidance predictions. In *Aviation Technology, Integration, and Operations Conference*, Los Angeles, CA, August 2013.

[165] Christine P. Taylor and Craig Wanke. Improved dynamic generation of operationally acceptable reroutes using network optimization. *Journal of Guidance, Control, and Dynamics*, 34(4):961–975, August 2011.

[166] Pak Yan Choi and Martial Hebert. Learning and predicting moving object trajectory: a piecewise trajectory segment approach. Technical report, Carnegie Mellon University School of Computer Science, Pittsburgh, PA, August 2006.

[167] Arjen de Leege, Marinus van Paassen, and Max Mulder. A machine learning approach to trajectory prediction. In *AIAA GNC Conference and Exhibit*, Boston, MA, August 2013.

[168] Lee F. Winder and James K. Kuchar. *Hazard Avoidance Alerting with Markov Decision Processes*. PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, August 2004.

[169] Action plan 16: Common trajectory prediction capability. https://acy.tc.faa.gov/data/_uploaded/Publications/tjm/TP_Requirements_Engineering_Methodology.pdf, 2004. Accessed: 2019-01-19.

[170] Definition of an aircraft trajectory by icao. http://ap16.atmrt.org/.

[171] Grib2 documentation. http://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc.shtml.

[172] Aircraft operating expenses. http://www.opshots.net/2015/04/aircraft-operating-series-aircraft-operating-expenses/.

[173] Claudio Esperança and Hanan Samet. Experience with SAND/Tcl: a scripting tool for spatial databases. *Journal of Visual Languages and Computing*, 13(2):229–255, April 2002.

[174] Hanan Samet, Houman Alborzi, Frantisek Brabec, Claudio Esperança, Gisli R. Hjaltason, Frank Morgan, and Egemen Tanin. Use of the SAND spatial browser for digital government applications. *Communications of the ACM*, 46(1):63–66, January 2003.

[175] James K. Kuchar and Lee C. Yang. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):179–189, December 2000.

[176] Traffic alert and collision avoidance system (tcas ii). https://www.faa.gov/other_visit/aviation_industry/airline_operators/airline_safety/info/all_infos/media/2012/InFO12010.pdf, 2012. Accessed: 2019-01-19.

[177] Alfonso Valenzuela and Damian Rivas. Conflict detection and resolution in converging air traffic. In *9th AIAA ATIO Conference*, pages 1–13, Hilton Head, SC, September 2009.

[178] Vu N. Duong and Eric G. Hoffman. Conflict resolution advisory service in autonomous aircraft operations. In *Proceedings of the 16th DASC. AIAA/IEEE Digital Avionics Systems Conference*, pages 10–17, Irvine, CA, October 1997.

[179] Huabin Tang, John Robinson, and Dallas Denery. Tactical conflict detection in terminal airspace. *Journal of Guidance, Control, and Dynamics*, 34(2):403–413, March 2011.

[180] Karl Bilimoria, Banavar Sridhar, and Gano Chatterji. Effects of conflict resolution maneuvers and traffic density on free flight. In *AIAA GNC Conference and Exhibit*, pages 1–11, San Diego, CA, August 1996.

[181] David Rey, Christophe Rapine, and Nour-Eddin El Faouzi. Subliminal speed control in air traffic management: Optimization and simulation. *Transportation Science*, 50(1):240–262, Februrary 2016.

[182] Marc J. Shewchun, Jae-Hyuk Oh, and Eric Feron. Linear matrix inequalities for free flight conflict problems. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pages 2417–2422, San Diego, CA, December 1997.

[183] Alex Vink, Seppo Kauppinen, Jos Beers, and Koen de Jong. Medium-term conflict detection in eatchip phase iii. In *Proceedings of the 16th DASC. AIAA/IEEE Digital Avionics Systems Conference*, pages 45–52, Oct 1997.

[184] Daniel J. Brudnicki, Kenneth S. Lindsay, and Alvin L. McFarland. Assessment of field trials, algorithmic performance, and benefits of the user request evaluation tool (uret) conflict probe. In *Proceedings of the 16th DASC. AIAA/IEEE Digital Avionics Systems Conference*, pages 35–44, Irvine, CA, October 1997.

[185] Todd Lauderdale. Probabilistic conflict detection for robust detection and resolution. In *12th AIAA ATIO Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, pages 1–12, Indianapolis, IN, September 2012.

[186] Weiyi Liu and Inseok Hwang. Probabilistic trajectory prediction and conflict detection for air traffic control. *Journal of Guidance, Control, and Dynamics*, 34:1779–1789, November 2011.

[187] Yoshinori Matsuno. *Probabilistic Conflict Detection in the Presence of Uncertainty*, pages 17–33. Springer Japan, Tokyo, Japan, 2013.

[188] David McNally, Ralph Bach, and William Chan. Field test evaluation of the ctas conflict prediction and trial planning capability. In *AIAA GNC Conference and Exhibit*, pages 1686–1697, Boston, MA, August 1998.

[189] Maria Prandini, Jianghai Hu, John Lygeros, and Shankar Sastry. A probabilistic approach to aircraft conflict detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):199–220, December 2000.

[190] Yang Yang, Kai quan Cai, and Maria Prandini. Fast algorithm based on computational geometry for probabilistic aircraft conflict detection. In *Proceedings of the 2017 International Conference on Robotics and Artificial Intelligence*, pages 66–70, Shanghai, China, December 2017.

[191] Lee C. Yang and James K. Kuchar. Using intent information in probabilistic conflict analysis. In *AIAA GNC Conference and Exhibit*, pages 797–806, Boston, MA, August 1998.

[192] Brenda Carpenter and James Kuchar. Probability-based collision alerting logic for closely-spaced parallel approach. In *Proceedings of the 35th Aerospace Sciences Meeting and Exhibit*, pages 1–8, Reno, NV, January 1997.

[193] Karim Zeghal and Eric Hoffman. Design of cockpit displays for limited delegation of separation assurance. In *Proceedings of the 18th Digital Avionics Systems Conference*, pages 1–8, St. Louis, MO, October 1999.

[194] José A. Cobano, David Alejo, Anibal Ollero, and Antidio Viguria. Efficient conflict resolution method in air traffic management based on the speed assignment. In *Proceedings of the 2nd Int'l Conference on Application and Theory of Automation in Command and Control Systems*, pages 54–61, London, UK, May 2012.

[195] Gerard B. M. Heuvelink and Henk A. P. Blom. Analysis and optimization of systems. chapter An Alternative Method to Solve a Variational Inequality Applied to an Air Traffic Control Example, pages 617–628. Springer-Verlag, London, UK, 1998.

[196] Maria Prandini, John Lygeros, Arnab Nilim, and Shankar Sastry. A probabilistic framework for aircraft conflict detection. In *AIAA GNC Conference and Exhibit*, pages 1047–1057, Portland, OR, August 1999.

[197] Gilles Dowek and Cesar Munoz. Conflict detection and resolution for 1,2,...,n aircraft. In *7th AIAA Aviation Technology, Integration and Operations Conference*, pages 1–13, Belfast, Northern Ireland, September 2007.

[198] Richard Irvine. Gears conflict resolution algorithm. In *AIAA GNC Conference and Exhibit*, pages 786–796, Boston, MA, August 1998.

[199] Padmanabhan K. Menon, Gregory D. Sweriduk, and Banavar Sridhar. Optimal strategies for free-flight air traffic conflict resolution. *Journal of Guidance, Control, and Dynamics*, 22:202–211, March 1999.

[200] José A. Cobano, David Alejo, Guillermo Heredia, and Aníbal Ollero. 4d trajectory planning in atm with an anytime stochastic approach. In *Proceedings of the 3rd Int'l Conference on ATACCS*, pages 1–8, Naples, Italy, May 2013.

[201] S. Ayhan, P. Costas, and H. Samet. Predicting estimated time of arrival for commercial flights. In *Proceedings of the 24nd ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 33–42, London, UK, August 2018.

[202] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, CA, 2006.

[203] H. Samet and M. Tamminen. An improved approach to connected component labeling of images. In *Proceedings of Computer Vision and Pattern Recognition'86*, pages 312–318, Miami Beach, FL, June 1986.

[204] En route automation modernization. https://www.faa.gov/air_traffic/technology/eram/, 2018. Accessed: 2019-01-19.