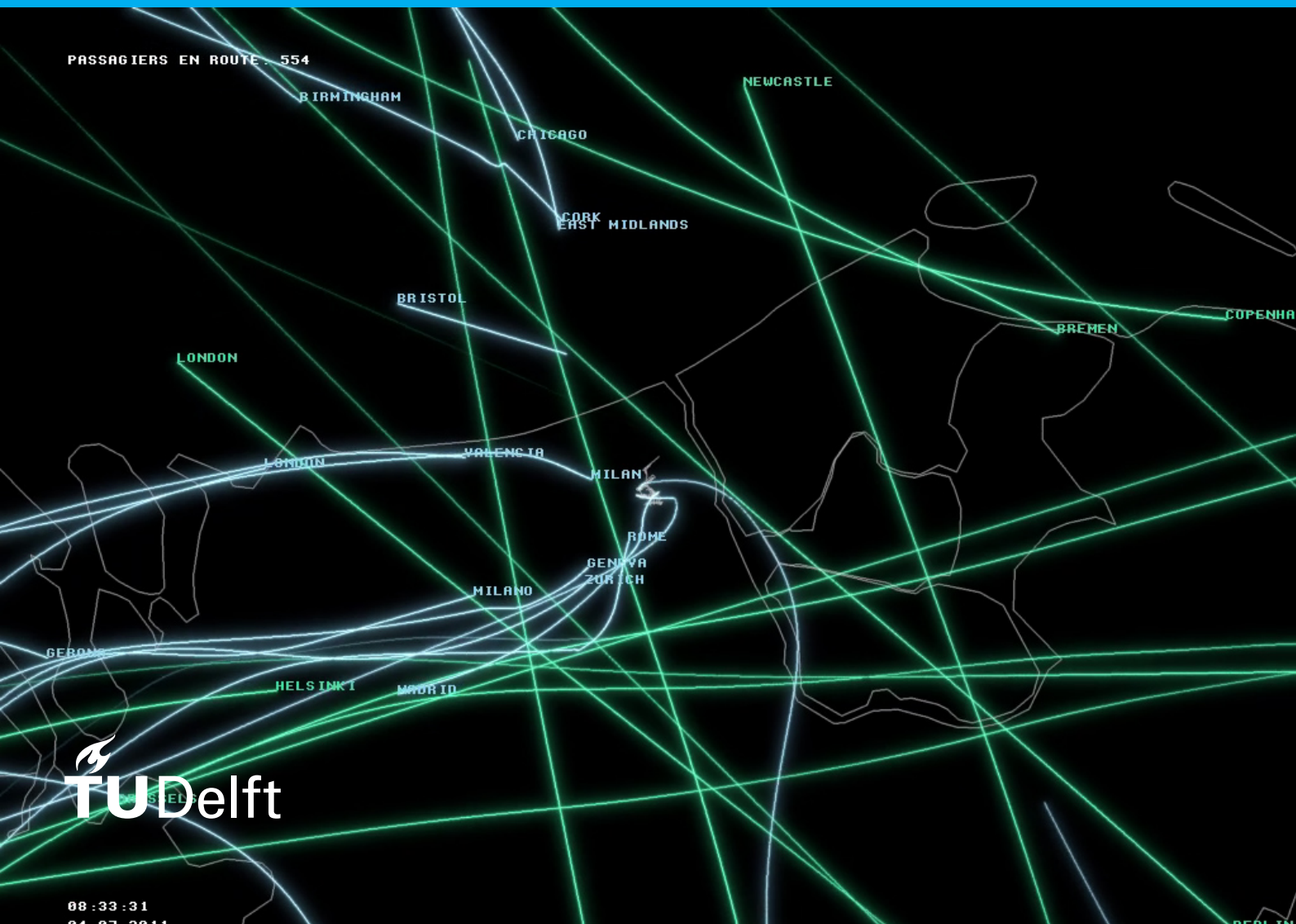
Detecting ATC commands in ADS-B data

An exploratory research into ADS-B data to see if detecting ATC commands is possible

Brent Kool

Research Paper & Preliminary Report

Student Number: 4270738



Preface

Brent Kool
October 2021

I would like to thank everybody who have helped me brainstorm, listened to me, or checked my report for any errors. I could not have done this without you all.

[0]Cover image source: http://www.minddesign.info/nederlandvanboven.html

Contents

List of Figures

List of Tables

Nomenclature

AC	Aircraft
ADS-B	Automatic Dependent Surveillance-Broadcast
AIRAC	Aeronautical Information Regulation and Control
ATC	Air Traffic Control
ATCo	Air Traffic Controller
ATM	Air traffic management
AUA	ATC Unit Airspace
BADA	Base of aircraft data
CPDLC	Controller Pilot Data Link Communications
DBSCAN	Density-based spatial clustering of applications with noise
DME	Distance measuring equipment
eAIP	Electronic Aeronautical Information Package
FIR	Flight Information Region
FRA	Free route airspace
GPS	Global positioning system
ICAO	International Civil Aviation Organization
ICAO24	ICAO unique 24-bits address
LVNL	Luchtvekeersleiding Nederland
MUAC	Maastricht Upper Area Control
Navaids	Navigational aids
NN	Neural Network
PC	Principal component
PCA	Principal component analysis
VAE	Variational autoencoder
VOR	VHF omni-directional range

I

Scientific Paper

Detection of ATC commands using ADS-B data

B. Kool
Supervised by: J. M. Hoekstra J. Ellerbroek
Control & Simulation, Faculty of Aerospace Engineering
Delft University of Technology, Delft, The Netherlands

***Abstract*—Automatic Dependent Broadcast-Surveillance (ADS-B) was introduced to increase the safety of air travel. Since then it has proven to be a useful source of air traffic data for academic research because it is publicly available. Another source of information are the communications between air traffic control (ATC) and aircraft. Methods are (being) developed to enhance the existing radio communications with digital text messages. These are not publicly available while access to these could further the research in this field.**

The lack of publicly available machine-readable ATC data hinders research in the field of air traffic management. Therefore the objective of this study is to develop a method that uses open-source data to infer ATC commands. This two-part exploratory study is presented to address this objective.

The first part explains the rule-based analysis used to infer ATC commands. This method uses a set of rules to detect lateral and vertical changes by approaching the ADS-B data as a time series. This approach can detect changes using the turn rate and rate of climb which can correspond to 'heading', 'direct to' and 'altitude' commands. It was not possible to detect changes in speed due to the variability in the wind.

The second part of this paper tests the found method in an air traffic simulator. For 111 flights, ADS-B data was gathered using the OpenSky network. The previous method was applied to this data to create a list of possible ATC commands. For every unique flight, two flight plans were made. The first was based on the found list of commands. The second flight plan used the original ADS-B data. These flight plans were simulated and differences between trajectories were used to test the fidelity of the rule-based analysis.

The results showed that the mean absolute horizontal error was below 15km for 75% of the flights and the mean absolute vertical error was below 100m for 75% of the flights. A remarkable correlation was found between the availability of ADS-B data points along the trajectory and the error in the horizontal distance. When more datapoints are available a smaller error in horizontal distance is observed.

It was found that a method could be developed using a rule-based analysis that produces a list of possible ATC commands. The could be done for both lateral and vertical commands, but not speed commands.

***Index Terms*—Air Traffic Control (ATC), Air Traffic Management (ATM), ATC command detection, ADS-B, change detection, aircraft trajectories.**

I. Introduction

Automatic Dependent Surveillance-Broadcast (ADS-B) was introduced to enhance safety while flying and has been mandatory in Europe since 2020. It sends out aircraft state information every second to inform others of its location and direction. Everyone with an ADS-B antenna can pick up this signal and upload the data to one of the existing ADS-B data networks. The open nature of the data makes it a commonly used resource for academic research, enabling the estimation of specific aircraft parameters[1] or identifying anomalous flights[2].

However, not much research has been performed using this data in the context of air traffic control. Air traffic control produces large amounts of data. While part of this data is accessible to the public, like ATC radio communication, most data is not open source. In particular, controller-pilot datalink communication (CPDLC)[1], a communication system that allows ATC and pilots to communicate using data messages, is not available. This data could be used to feed machine learning models to predict ATC controller commands. Since the CPDLC is not available for public use, other methods have to be developed to obtain publicly available ATC commands.

Having publicly available ATC command data allows further research on the topic of ATC. Methods could be developed to detect possible or likely collision areas, hidden ATC habits, cross-airspace effectiveness, or a method could be developed to filter ATC commands from trajectory data. To address the issues described earlier, the objective of this research is formulated as follows: is it possible to infer ATC commands from ADS-D data?

This study consist of two main parts. The first part explains in detail how a rule-based analysis is used to detect changes in the lateral and vertical plane. Further explanation is given on how the changes can infer 'heading' and 'altitude' commands and how further analysis can infer 'direct to' commands. 'Speed' commands were not detectable due to the variability in wind conditions. The second part uses the found ATC commands to perform a simulation to test their accuracy. This experiment compares two flight plans. The first is made based on the original ADS-B data and the second is made using the commands found via the rule-based analysis. Differences between the two trajectories are used to evaluate the methodologies.

For this study, only the high-altitude cruise phases of flights are considered. At these altitudes changes in trajectories are most likely to be ATC commands due to the class A designation for those airspaces.

This paper starts with summarising related work in section II. Section III and section IV present the data sources and preprocessing steps taken respectively. Section V examines three different methods and presents the results found during the examination. The experiment to test the accuracy

[1] https://www.eurocontrol.int/function/datalink

is explained in section VI. The results of the experiment are presented in section VII and are discussed in section VIII. Section IX presents the conclusion and section X suggests recommendations. The paper closes with two examples of the presented work in section XI.

II. Related Work

This section elaborates on related work of the scientific use of ADS-B in general and other methods to detect air traffic commands or changes in data. The first section focuses on extracting information from raw ADS-B data. Section II-B explains the practices of air traffic control. The next sections, section II-C and section II-D, cover clustering techniques and anomaly detection methods, respectively, and their applications to trajectory data in aviation. Section II-E presents change point analysis and its application to trajectory data. The last section discusses the latest research for converting ATC radio automatically into machine-readable data.

A. Extracting information from ADS-B data

Raw ADS-B data can be processed to infer flight information that is not explicitly available in the data. Two examples are discussed which are relevant for the current study, both presented by Sun, Ellerbroek and Hoekstra [3][4]. The first method paper uses fuzzy logic to determine the flight phase of the given datapoint. Based on the height, velocity and rate of climb, a flight phase can be determined. These flight phases can be then be used to determine change points for the vertical movement of aircraft. The same papers also provide a method to distinguish unique flights from raw ADS-B data using clustering algorithms. It was found that DBSCAN[5] provides the best method to extract flights from a large set of ADS-B data.

B. ATC practices

ATC practices need to be taken into consideration during this study. Therefore a summary of heading, speed and altitude commands is given. This summary is simplified to the core messages that are used in basic ATC messages. Note that this is based on the common practices of ATC in the Netherlands [6].

Heading changes are commonly given in two different formats. 1) Aircraft can be given the 'direct to xxx' command. This command directs an aircraft to fly directly to a waypoint. 2) Aircraft can be commands such 'fly heading xxx' or 'turn left/right heading xxx'. This command tells the pilot to fly a heading between 0 and 360°. In this study, the assumption will be made that these two commands are the only heading commands given. It will also be assumed that the 'direct to' command will most likely be given, as this study focuses on the cruise phase of flights.

Speed changes can be only be given in two formats. These are either 'at pilots discretion', meaning that the pilot is allowed to fly at the desired speed. Or ATC may state 'increase/reduce speed xxx knots', directing the pilot to fly at a set speed. In practise, speed change commands are rare during the cruise phase. Therefore the assumption is made that aircraft will fly close to the most efficient speed (in relation to the wind).

Altitude commands are commonly given in the 'climb/descent to FLxxx' format. Since aircraft are always assigned to a flight level to maintain vertical separation, it can be assumed that any change in altitude is either directed by or approved by ATC.

C. Trajectory Clustering

Clustering methods are applied throughout the field of air traffic studies. These clustering methods have the ability to group similar trajectories to infer flows, based on known or unknown parameters. Yuan et al. [7] surveyed and summarised several clustering algorithms applied for moving object trajectories and trajectory similarity or distance metrics. Two clustering techniques have been selected.

The first is DBSCAN, which is previously used to cluster trajectories for various purposes [2, 8–10]. This clustering method has already seen a lot of use in the field and is readily available in the scikit-learn python package[11]. However, it cannot take trajectory data directly as an input. Therefore, Principal Component Analysis (PCA) is often used to alter the trajectory data such that it can be used with DBCAN. PCA is also available in the scikit-learn package.

The second trajectory clustering method is spectral clustering. Enriquez [12] has created an algorithm based on spectral clustering to work directly with trajectory data, omitting the need to modify the data beforehand.

It was found that the detection of ATC commands was not possible using clustering methods. Individual ATC commands have a too small effect to be noticeable using clustering. Furthermore, all trajectories have been altered by ATC and thus, no comparison can be made between altered and unaltered trajectories.

D. Anomaly detection

Anomaly detection focuses on identifying data deviating from the standard. A paper published by Basora, Olive and Dubot [13] reviewed the latest techniques in the field of aviation. The research focused mostly on techniques that work with unlabelled data, as this is mostly the case when working with aircraft trajectory data. Olive et al. [14] use anomaly detection methods to detect unusual flight behaviours on the approach to Toulouse airport. These anomalies were then placed into context and from there it was assessed if an ATC action could correspond with the detected anomaly. Anomaly detection might be used in conjunction with clustering techniques to identify odd patterns in ADS-B data. The patterns might be linked to ATC commands. It was also found that anomaly detection does not work to detect ATC commands, since ATC commands are not anomalies and are present in all trajectories.

E. Change point analysis

Change points are point within a time series where a change in the time series occur. Change point detection algorithms

focus on detecting changes in the parameters that make up a time series. The study by Truong, Oudre and Vayatis [15] presents a selection of change point detection algorithms and a python package named Ruptures which includes these methods. The study uses a common framework to review numerous algorithms that allow the reader to select the best algorithm.

A hypothesis was made that ATC commands could be seen as change points. Applying change point detection algorithms to trajectories could detect ATC.

F. Automated ATC Voice Transcriptions

Research has also been conducted to use voice-recognition software to automatically transcribe ATC voice commands into machine-readable data. These studies are not yet finished and are therefore not used in this study. The results of these studies align with the goal of the current study and are therefore worth mentioning.

Automated ATC voice transcription would allow for the automatic detection of ATC commands by listening to ATC radio. Srinivasamurthy et al. [16] propose an iterative model that uses ATC voice and radar data to build up an increasing database of ATC commands that can be used to train their model. Their system used a semi-supervised learning framework that uses speech and radar data to iteratively update the model. Subramanian, Kostiuk and Katz [17] proposed a method to detect anomalies in the National Airspace System by combining ATV voice command and Traffic Flow Management data with IBM Watson's natural language processing capability. ATC voice commands could be used as a validation set when using supervised machine learning methods to further detect changes in the flight path of aircraft.

III. Data Sources

This section presents a clear overview of all data used and the respective sources. The first section presents the ADS-B sources and other flight data. Later sections explain data sources regarding navigational aids and airways.

A. Flight path data

Two ADS-B data sources have been used. The first source is the Opensky ADS-B network[18][2]. This organisation collects ADS-B data and stores this in a database to be used by the general public and research. In the current study, this database was used in conjunction with the ADS-B receiver located at the Faculty of Aerospace Engineering at the TU Delft, which also provides ADS-B data.

The second data source is the Eurocontrol R&D data release. This dataset consists of multiple tables, of which the *flights* and *filed flight points* tables are the most important for this study. Four sample months (March, June, September & December) for the years 2015 to 2018 (inclusive) are given in this release. The *flights* table contains all standard information regarding scheduled flights. The *filed flight points* table contains flight points of a flight in latitude and longitude coordinates and the altitude at that point.

[2]Information about the collected data can be found on https://opensky-network.org/data/impala. (Accessed March 2021)

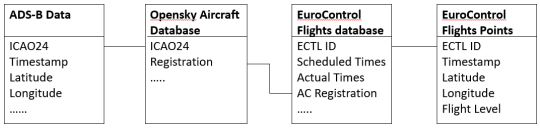


Figure 1. The initial database layout

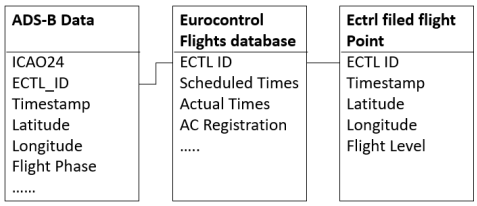


Figure 2. The final database layout

To couple these databases together, the Opensky aircraft database is used. The Opensky ADS-B dataset uses the ICAO24 identifier as unique key for aircraft, and the Eurocontrol R&D release uses the registration as unique key. The Opensky aircraft database stores both identifiers and is therefore used to couple the previous database together. A schematic overview is given in fig. 1. To make computation easier, a simpler database structure is set up, which is shown in fig. 2. This omits the need for the aircraft database by adding the Eurocontrol ID to every ADS-B message.

B. Navigational aids

Two navigation aids (navaid(s)) databases are used: The Traffic[19] database and Bluesky[20] database. Waypoints are filtered using the Eurocontrol dataset to only include waypoints used for en-route navigation. The *filed flight points* are planned according to waypoints and therefore have datapoints on the locations where these waypoints are located. Selecting only the locations where the planned altitude is above FL200 and filtering the databases based on these locations, a subset of waypoints was created. This subset contains only waypoints used during cruise. A counter for each waypoint was included to enable further filtering when necessary.

C. Airways

Airway information is contained in the traffic database, the BlueSky simulator and in the Eurocontrol R&D data release. However, the naming of the airways across the data sources is inconsistent. Therefore the Eurocontrol airspace charts[21] were used as a control dataset.

IV. Data Preprocessing

This section presents an overview of all preprocessing steps taken. Preprocessing is necessary because raw ADS-B data contains noise and possible outliers. Removing these lead to a more accurate detection of ATC commands.

In total 4 preprocessing steps are applied to the data. The first step is flight extraction, where individual flights are

selected from the ADS-B data. The second step is to discard any outliers found within the found flights. The third step is to augment the data with the flight phase. The last step is to lower the noise level by smoothing the data. Several smoothing methods and suggested and tested. The complete data processing pipeline can be found in Appendix B.

A. Flight Extraction

ADS-B was not intended to be used to study trajectories but as a safety feature. Therefore ADS-B messages do not always contain information about the flight currently being performed but do always contain their ICAO24 identifier. Since an aircraft can fly multiple flights every day, there is a need to extract flights from the ADS-B data, to study trajectories.

Flight extraction is done using data from the Eurocontrol R&D dataset. The *flights* table contains the 'Actual Off-Block Time' and the 'Actual Arrival Time'. These timestamps are used to determine when the aircraft was in the air and which flight it was performing. When the Eurocontrol data was not available, the methods using DBSCAN proposed by Sun, Ellerbroek and Hoekstra [3][4] were used.

B. Outlier Detection

The second step is to remove outliers from the data. Outliers are points where a datapoint was captured that is not within the main part of the trajectory. These points can have a negative effect on further analysis due to the large time differences between points. To remove outliers, DBSCAN is used as proposed by Sun, Ellerbroek and Hoekstra [4]. The parameters were set at 50 for $MinPTS$ and 500 for ϵ. Outlier points are not used for any further analysis.

C. Phase Identification

The third step is to add the flight phase (cruise, climb, etc.) to the ADS-B data. The proposal by Sun, Ellerbroek and Hoekstra [3] is followed to add these. By adding the flight phase to the ADS-B, identifying the en-route phase of a trajectory is quicker. This is done by identifying the first and last points of the flight which are identified as en-route and selecting everything in between. The flight phase is also used to detect changes in altitude, by detecting points where one flight phase changes to another.

D. Data Smoothing

The data gathered by the OpenSky network is real-world data and therefore it is subject to noise. To get more reliable results, a low noise level is preferred. In current literature, data is smoothed according to the needs of the research. Therefore, several smoothing functions are proposed and tested to see which is the best at removing noise from the data. The methods are given below and can be divided into two categories. The first 5 methods (1-5) are point-based smoothing functions. They do not consider the time at which the datapoint was taken. The last 3 methods (6-8) are time-based and do consider the time at which the datapoint was taken.

1) Exponential smoothing
2) Rolling mean
3) Rolling median
4) Rolling Gaussian window
5) Subsampling
6) Time-based resampling first
7) Time-based resampling mean
8) Time-based resampling median

To test these methods, a sample of all trajectories was taken and the different methods were applied to the turn rate, standardised altitude and standardised speed. For every trajectory and every smoothing function, the standard deviation was calculated. This is shown in fig. 3. Additional figures for trajectories can be found in Appendix C, fig. 22.

From this analysis, it was concluded that the time-based resampling median was the best option. This was chosen due to the low standard deviations found when applying this to the sample trajectories and the median metric has the unique property to dismiss one-off extreme outliers (1 data point with an altitude of 30 km where the local average is 10 km). A window size of 10 seconds was selected because it was the best trade-off between high temporal accuracy and minimising the amount of noise in the signal.

V. Methodology

To detect ATC in ADS-B data, a rule-based analysis was used. This method is based in the fact that air traffic is very regulated and mostly standardised. By observing and detecting the resulting patterns of these standardised methods, the original ATC command may be recovered. By setting up rules, these patterns can be detected in the ADS-B data. This section proposes a method to detect track angle changes, and in its extension, 'direct to' commands. The same method is applied to detect changes in speed. The results and why this was not possible will be discussed in section V-C. Lastly, a method using the identified flight phase is used to detect altitude changes. Other methods of detecting changes were considered, but they could not be used to detect ATC commands. These can be found in Appendix A.

A. Track angle change detection

Before changes can be detected in the data, the track angle first has to be calculated. This is done by applying the great circle equations to the resampled positional data. This gives the track angle from each datapoint to the next. The ADS-B data also contains the track angle, but this contained more noise in the dataset used in this study. Therefore the choice was made to calculate it using the aircraft coordinates. By taking the derivative with respect to time, the Δtrack (or turn rate) can be calculated. This metric is the basis of the track angle change detection. For the detection method, three algorithms were tested.

1) *K-sigma rule*. This method standardises the given signal and detects values above a given threshold, in this case, K.

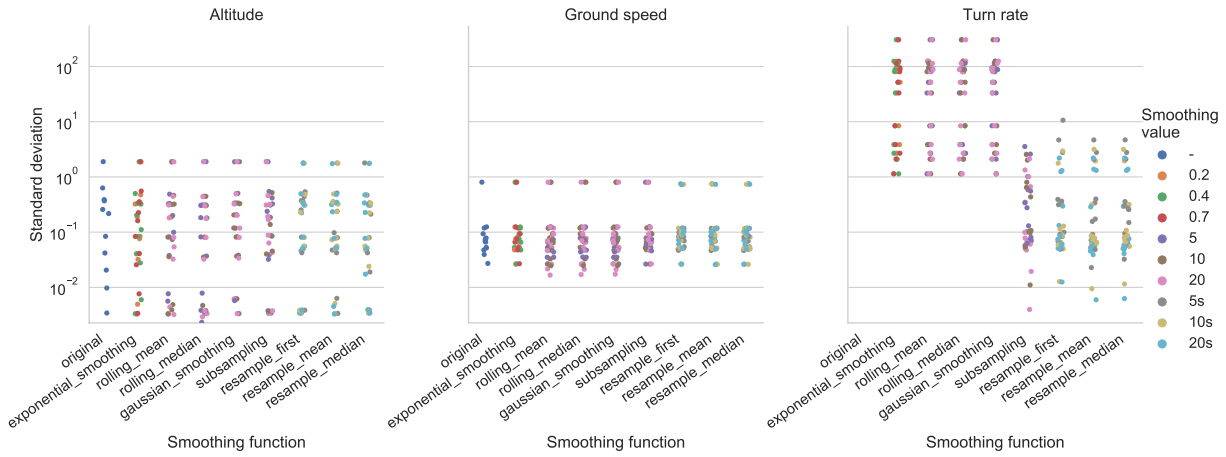


Figure 3. Standard deviations of trajectories for different values of different smoothing functions

2) *Turn rate threshold*. This method, as used by Gariel, Srivastava and Feron [9], uses a threshold value to detect turns.
3) *Change point detection*. Turns result in changes in the track angle and turn rate signals. These changes can be considered to be change points. The python packages Ruptures[15] contains several algorithms that can detect change points and thus detect turns.

Due to missing values in the data, Ruptures was unable to detect change points. The variability in track angles and turn rates across the trajectories also required that the penalty value had to be tuned for every trajectory. Therefore the choice was made to disregard Ruptures for this study. This method might still prove useful when used with consistent data or on slices of trajectories.

The other two algorithms were tested to determine a good rule for the detection of changes in the track angle. This was done experimentally using plots such as fig. 4. More are found in Appendix C. The K-sigma rule has the advantage that a false detection due to noise is filtered out, but it has trouble detecting small changes. The turn rate has the exact opposite, where it can detect small changes, but also can falsely identify noise as a turn when the signal is particularly noisy. Combining both of these metrics, a rule can be set up where noise can be ignored while detecting small changes. This rule is set at:

$$\text{Turn rate} \geq 0.2 \vee (\text{Turn rate} \geq 0.1 \wedge \text{sigma} \geq 2)$$

A check is done to identify a single turn point from multiple datapoints. This is done by checking if a turn is detected at points x_{t-1} and x_{t+1}. If so, point x_t will also be identified as a turn. The last step is to check whether the actual track angle before and after a turn has significantly changed. A difference of $1°$ was chosen to be the minimum required value. Detected turns with smaller changes are considered to be too small to be an ATC command. These neglected differences can be caused by fine-tuning of the aircraft settings, wind effects, noise in the data, or other reasons.

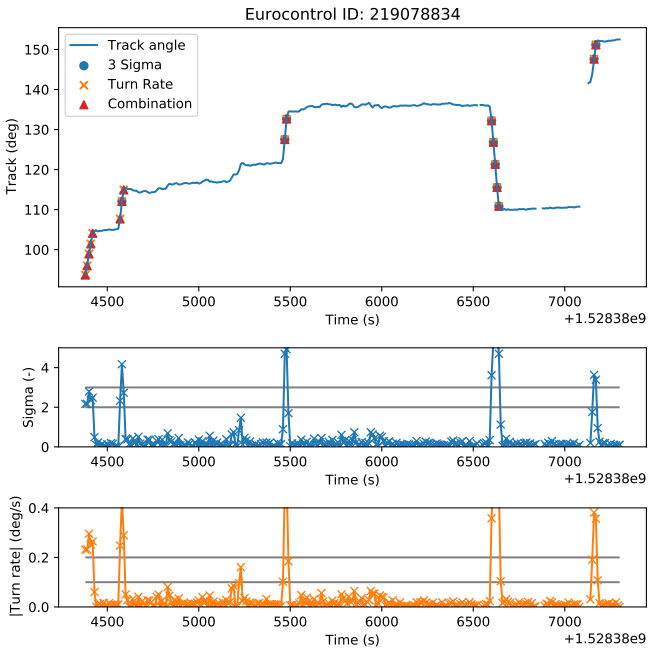


Figure 4. Top figure: The calculated track angle of a random flight. The markers show where the detection methods detect a change.
Middle figure: The standardised Δtrack, with reference lines at 2 & 3.
Bottom figure: The Δtrack, with reference lines at 0.1 & 0.2.

Information regarding the detected turn, such as Eurocontrol ID, time, latitude, longitude, track angle before, track angle after, and more, is stored.

B. Direct to detection

Now that turn points have been identified, ATC commands can be inferred. Since this study is focused on the cruise of aircraft, ATC will most likely direct aircraft with the use of waypoints. Here some assumptions are made: 1) Waypoints in a filed flight plan will take priority over regular waypoints. 2) Waypoints near a detected turn are the most likely candidate.

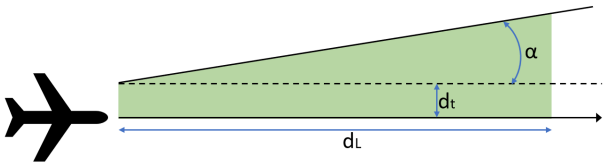


Figure 5. The search area for waypoints. The green area represents the area where waypoint are searched for. Values are: $\alpha = 1°$, $d_l = 500, 100$km, $d_t = 5$km. Only one side (left) of the search area is visualised here.

3) If no waypoint is near the turn, then the first waypoint is selected along the trajectory if no turn was taken. 4) If no waypoint is detected still, the furthest waypoint along the trajectory after the turn is selected. The following list summarises the assumptions based on the hierarchy as stated before.

1) Waypoint in flight plan near turn
2) Waypoint in flight plan in heading before next turn
3) Waypoint in flight plan in heading after current turn
4) Waypoint near turn
5) Waypoint in heading before next turn
6) Waypoint in heading after current turn
7) Heading change

This prioritization is applied at every detected turn point. To determine if a waypoint is near a detected turn, an offset of 5km is used. To speed up the search, the waypoints are preselected using a $0.2°$ offset in both latitude and longitude. To determine if a waypoint is in the heading of the aircraft before or after a turn a defined area is scanned, which is visualised in fig. 5. This area contains a transverse offset from the heading of 2km (d_t) and uses an extra angle of $1°(\alpha)$ to search. This method was chosen because the calculated heading could contain errors or small deviations. When searching far ahead of detected points, these small deviations could grow to a significant error. When a turn is detected, the track angle before and after the turn is analysed. For the before part, a d_l of 100km is used. After the turn, d_l is set at 500km.

C. Velocity change detection

To detect changes in the velocity, the same approach was taken for changes in the track angle. A test was performed to test whether the ground speed from ADS-B data or the calculated ground speed yielded the best potential for detecting changes. The results can be seen in in fig. 6. This shows that the ground speed given by the ADS-B data contains less noise. This test also shows that the resample median results in a low standard deviation. A window of 10 seconds was chosen to remain consistent with the resampling method chosen for detecting track angle changes.

After the initial analysis, it was found that the deviations in speed were too variable for any of the detection methods to work. As shown in fig. 7, no distinct changes can be seen in the data. Places, such as the sudden jump in the pink line, might indicate a speed change, but this could not be determined due to several factors. 1) Aircraft during cruise try to optimise

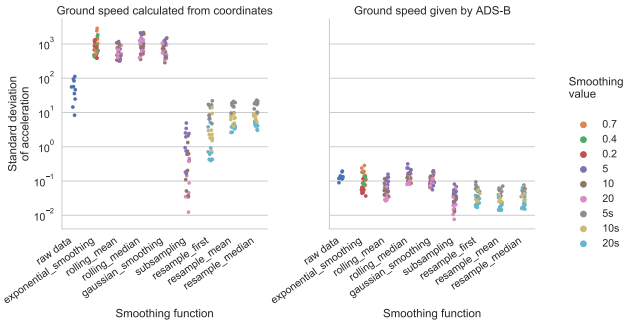


Figure 6. The calculated standard deviation of the acceleration of the aircraft.

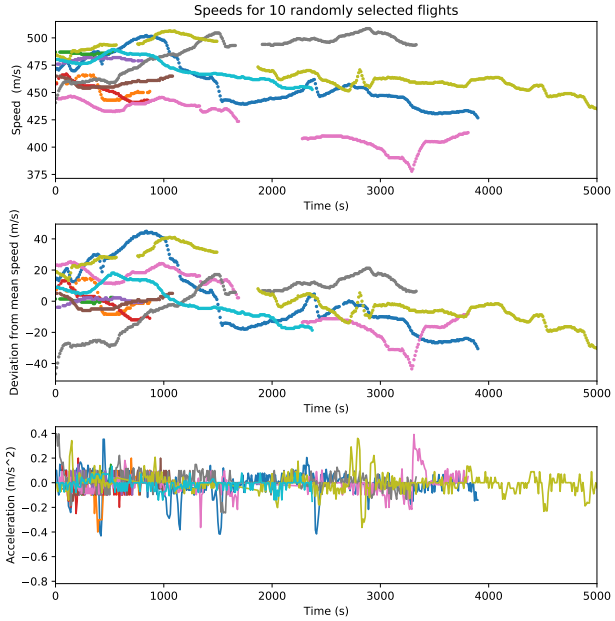


Figure 7. Top figure: Speed profile of 10 flights using ADS-B ground speed. Middle figure: the deviation of the speed from the mean speed. Lower figure: The calculated acceleration.

the fuel burn and therefore fly mostly at a constant speed. 2) Changes in the wind might speed up or slow down an aircraft. 3) Changing the altitude of an aircraft might move the aircraft into another wind region, changing the ground speed, while the aircraft flies at the same airspeed. 4) Changing the heading of an aircraft will alter the behaviour of the wind on the aircraft. Due to the broad range of possibilities, the choice was made that this direction was not to be further explored in this study.

D. Altitude change detection

To detect changes in altitude, the flight phases were used. Since a change in flight phase means the aircraft is changing its current vertical velocity, it can be used to identify when the aircraft is changing its altitude. To ensure that a transition of flight phases is not an outlier, a check was employed where at least a number datapoints have to be different in flight phase before and after the possible change. For this check, a difference of 1 to 5 datapoints was tested. Plots such as fig. 8

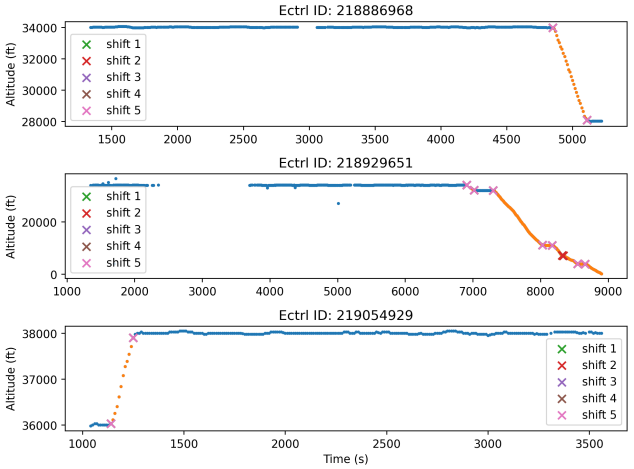


Figure 8. Several trajectories are shown with shifts of 1 to 5 applied to the phase. Blue colour means 'cruise' and orange are other phases.

Table I
TABLE STRUCTURE CONTAINING THE ALTITUDE CHANGE INFORMATION

	'From cruise'	'From non-cruise'
1	Time, Altitude, FL, etc...	Time, Altitude, FL, etc..
2	Time, Altitude, FL, etc...	Time, Altitude, FL, etc..
3	etc...	

were used to select the final value. It was found that 1 or 2 different datapoints are inadequate since these still identify too much noise as a phase shift. Therefore a value of 3 was chosen.

After the identification of the phase shifts, a table can be made where information is stored at the timestamps of the phase change, assuming an aircraft goes from 'cruise' to 'non-cruise' to 'cruise'. The table structure is given in table I. Information regarding the timestamp, the altitude, the flight level, and the current phase is stored at every detected phase change. This is augmented with metadata such as the Eurocontrol ID of the flight to allow easy storage and identification.

VI. EXPERIMENT

To test how accurate the found ATC commands are, a simulation is performed. As previously mentioned, the rule-based analysis was able to create a list of possible ATC commands. These commands are compared against a simulated flight based on the original ADS-B data.

First, the goal and set-up of the experiment are discussed. Secondly, all independent, dependent, and control variables used in the experiment will be presented. These form the basis of the results and discussion presented in sections section VII and section VIII.

A. Experiment goal & set-up

The goal of this experiment is to test the following hypothesis: A rule-based analysis can infer ATC commands using the turn rate and flight phase from ADS-B data. To test this hypothesis BlueSky is used. A comparison can be made by simulating a flight based on the ATC commands and comparing it against a simulated flight based on the ADS-B data.

Table II
DATA USED FOR SIMULATION

City pair	Carriers	Flights per carrier
EHAM, LEBL	KLM, TRA, VLG	12, 10, 18
EDDF, LPPT	DLH, RYR, TAP	8, 6, 12
EGLL, EDDM	BAW, DLH	9, 9
LFPG, EKCH	AFR, EZY, FDX, SAS	14, 3, 5, 5

For this experiment, 4 city pairs were selected where the OpenSky network has sufficient coverage[3]. A date was chosen of which data was available in the Eurocontrol R&D release. This was 26 July 2018.

To query OpenSky more effectively unique aircraft (tail numbers) are selected (OpenSky works with ICAO24 identifiers, which are unique per aircraft). Between every city pair, the number of flights (different callsigns) are counted for each tailnumber. For every carrier, the top 2 tailnumbers are selected. The carrier and the number of flight per carrier can be found in table II. Different carriers were chosen to diversify the used flight plans, making the results less dependent on flight plans. In total 111 flights were selected. An overview is presented in table II and exact flight numbers are given in Appendix F. The raw ADS-B data is visualised in fig. 9. Gaps on the map indicate that no ADS-B message from this area was captured by ADS-B receivers connected to the OpenSky Network.

The ADS-B data was preprocessed, as described in section IV. Further filtering was performed to select only the cruise period. This was done by identifying the first and last datapoint identified as cruise and selecting all datapoints in between. An additional filtering step was implemented to exclude all trajectories of which the filed flight plan (as delivered to Eurocontrol) is too different from the actual flown path. If the starting point of the cruise phase is more than 30km removed from the flight plan, the flight is discarded. This is done to ensure that flights used are without anomalies. The last step was to detect the ATC commands in the trajectories using the rule-based analysis described in section V.

B. Independent variables

The independent variables are factors that are deliberately manipulated between the different experimental setups to see the effect of that manipulated factor on the experiment. As previously mentioned, this experiment is to test how the detect ATC compare against the original ADS-B data. Therefore, every flight was simulated twice. The first flight (the simulated flight) is solely based on the gathered ADS-B data. The second

[3]https://opensky-network.org/network/facts

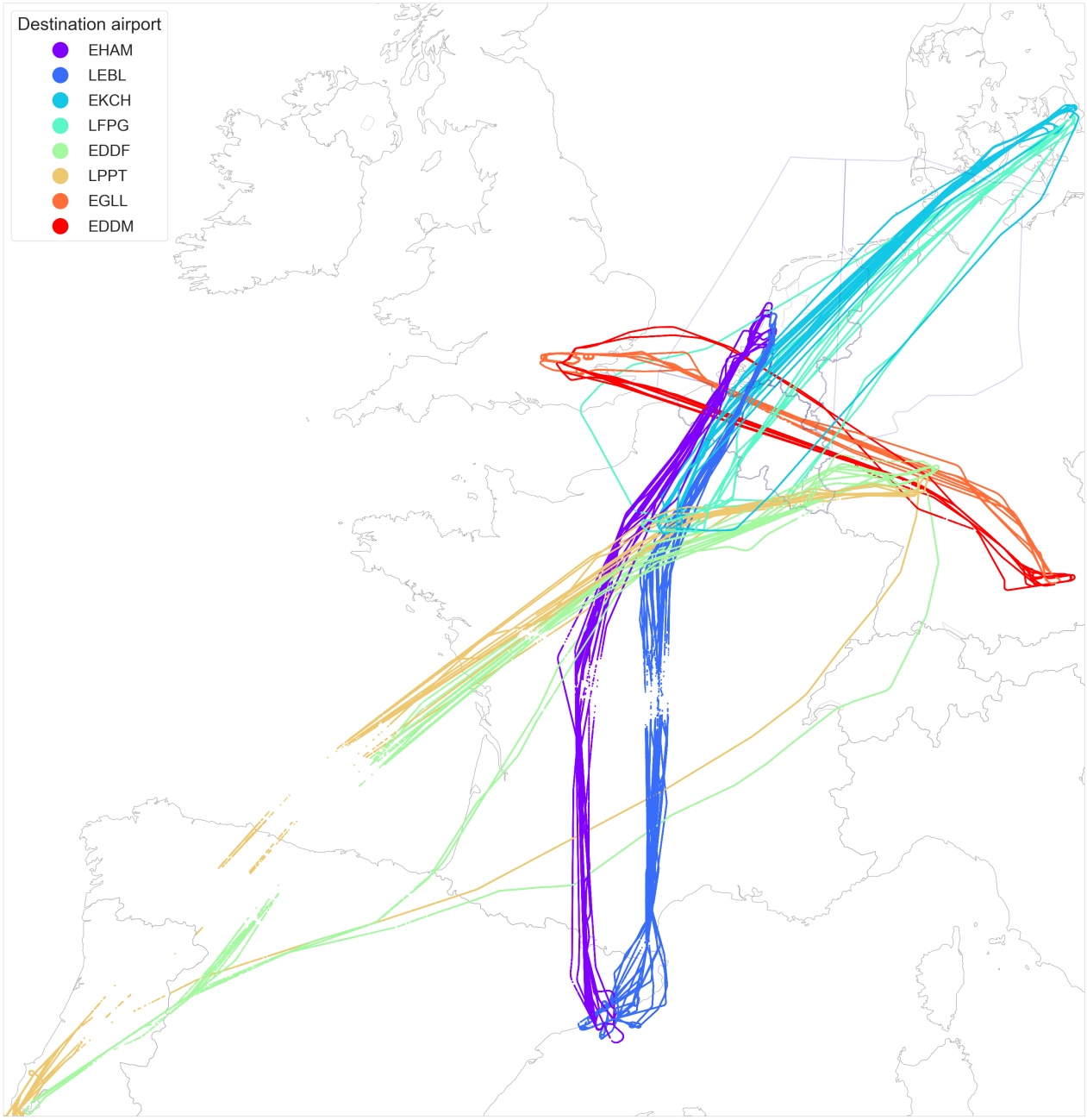


Figure 9. The augmented ADS-B data gathered from the OpenSky database for the 111 flights used in the experiment, coloured by the destination airport. Note that there are 3 gaps in the data. Above the north border of Portugal and Spain, the southern part of the Bay of Biscay, and the centre of France. In these gaps, no ADS-B messages were captured by the OpenSky network.

flight (the reconstructed flight) is made using the found ATC commands.

The following subsections explain how the two different flights are simulated. The creation of the flight plan is explained, as well as the starting and ending of a flight. The process of adding one aircraft to the simulation is explained since the process is similar for all flights.

1) Simulated flights: When the simulated flight is added to the simulation, the first datapoint in the ADS-B data is used. This contains the latitude, longitude, altitude, and track angle of the aircraft. This information is used in BlueSky (input related to speed are discusses later) to add an aircraft to the simulation.

After the creation, the flight plan is added based on the ADS-B data. To reduce the computational time, the amount of datapoints is reduced using the Ramer-Douglas-Peucker (RDP) algorithm. The algorithm is applied to the [latitude, longitude] array of ADS-B data. A reduction value of 10^{-3}[-] is used, which corresponds to a maximum deviation of $132m$ at $50°$ latitude. The remaining datapoints are used to create a flight

Table III
DEPENDENT VARIABLES AND UNITS

Variables	Unit
Horizontal difference	km
Track angle difference	°
Vertical difference	m

Table IV
CONTROL VARIABLES

Variables	Value	Comment
Aircraft speed	250 [kts]	Given as CAS
Aircraft type	B788 [-]	
Wind speed	0 m/s	
Waypoints	-	Added from Traffic package if not in BlueSky database

plan, by adding every point as a unique waypoint in the flight plan. Where ADS-B data is missing, a straight line will be flown between the last available point and the next available datapoint. The destination airport is added as final waypoint. The aircraft is removed from the simulation at the timestamp of the last ADS-B datapoint.

2) Reconstructed flights: The reconstructed flight uses the detected ATC commands to form a flight plan. These ATC commands are generated by applying the rule-based analysis, as described in section V, to the original ADS-B. The aircraft is added to the simulation in the same manner as the simulated flight, using the same parameters. After that, the flight plan is created using the found 'direct to' commands, with the destination airport being added as the last waypoint. During the flight, the aircraft is given 'direct to', 'heading', and 'altitude' commands at the timestamps at which the command was detected in the ADS-B data. The aircraft is also removed from the simulation at the timestamp of the last ADS-B datapoint.

C. Dependent variables

Dependent variables are measurements taken from BlueSky to compare the differences between experimental conditions. Data is gathered using the built-in datalogger tool, which can log data from multiple sources within BlueSky. For this experiment, only data regarding the aircraft objects is gathered. The datalogger was set to collect data every 10 seconds of the simulation. This time was chosen, such that it matches the smoothing method chosen.

The captured data, summarised in table III, allows for both lateral and vertical comparisons between the simulated flight and the reconstructed flight. The CAS was collected for verification purposes as speed is a fixed control variable (discussed later).

D. Control variables

Control variables are paramters that influence the experiment but are held constant, such that they do not affect the experimental results. An overview of all the control variables can be found in table IV. As previously mentioned, the speed variable was fixed, because this could not be detected (see section V-C). To make a fair comparison the speed of the aircraft was fixed at $250kts$ CAS. CAS is independent of altitude and can therefore be added to the aircraft at any height.

Other control variables are the aircraft type, wind, and waypoints. BlueSky requires an aircraft type to be given upon creating an aircraft. This was set as a Boeing 787-8 (B788) because the simulation needed aircraft that could reach FL400. The wind was set at $0kts$, to not affect the simulation at all. Waypoints were added to BlueSky if these were not already in the database. These were added before any aircraft were initialised.

VII. RESULTS

This section covers the results from the experiment presented in section VI. At every timestamp, the simulated ADS-B flights are compared to the reconstructed flights. First, the horizontal results will be presented followed by the vertical results. These results will be discussed in the next section, section VIII.

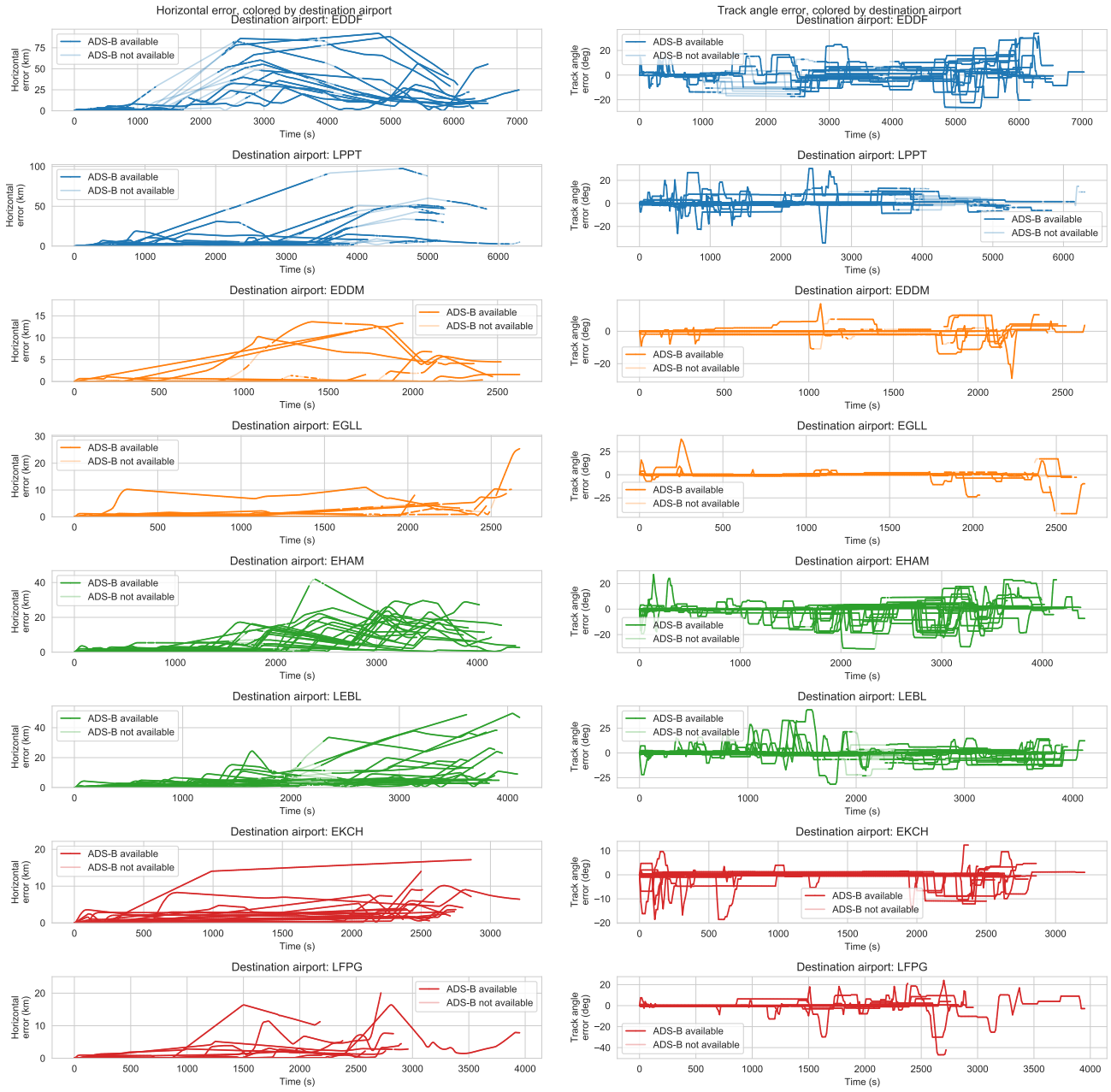
A. Horizontal

At every timestamp, the horizontal distance between the simulated and reconstructed flight is calculated. The results can be found in fig. 10a. As seen in the figure, the flights between EDDF and LPPT have the highest horizontal distance error, with errors as large as $100km$. These flights also have the biggest gap in available data as shown by the lighter coloured lines. Between 1000 & 2000 seconds, errors start to accumulate before slowly sloping back. Errors starts to grow before or during the gaps and continue to grow throughout the data gaps. The second biggest errors are between EHAM and LEBL. The other 2 city pairs have a much lower and more consistent horizontal error. These errors are around $10km$ error with an outlier peaking at $25km$.

The same plot has been made for the difference in track angle. This can be seen in fig. 10b. The large differences in horizontal distance error do not directly translate to large track angle errors. However, the variability of the track error is larger. This is due to the integrator like relationship between the track angle and the horizontal difference.

For both measurements, the mean absolute error (MAE) is calculated. The MAE for the horizontal distance error can be seen in fig. 11. This figure shows the same results as seen before. A stark difference can be observed for flights going towards EDDF. The average is twice as high as for flights to LPPT. This is further discussed in the next section (section VIII). Another observation is that the overall average MAE is around $5km$, and the 75^{th} percentile is just below the $10km$ mark.

The MAE of the track angle difference is shown in fig. 12. The overall average MAE is a little below $3°$ and the 75^{th} percentile is just above $4°$. The track angle does not follow the same pattern as the horizontal distance. Flights towards EDDF have the highest MAE, $6°$ on average, followed by the flights



(a) The horizontal error between the simulated flight and the reconstructed flight. Every colour indicates a city pair. Lighter coloured lines indicate that ADS-B was not available at these timestamps.

(b) The track angle error between the simulated flight and the reconstructed flight. Every colour indicates a city pair. Lighter coloured lines indicate that ADS-B was not available at these timestamps.

Figure 10.

towards EHAM. Since the flights towards EHAM are shorter, the erroneous track angle has less time to integrate. The extra availability of data points also provides more options for correcting errors. This effect can also be seen when comparing flights towards LPPT and LEBL. These have roughly the same track angle spread, while the horizontal distance error is twice as low for flights towards LEBL.

Fig. 13 shows the correlation between the number of detected changes in the lateral plane and the MAE of horizontal distance. It is determined that no clear correlation can be observed from this plot. Fig. 14 does the same, but for the MAE in track angle difference. This shows a positive trend, meaning that a more complex trajectory (more turns) has a larger error in the track angle difference.

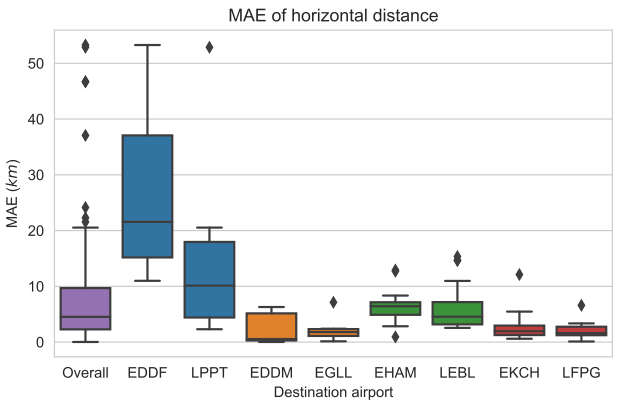


Figure 11. The mean absolute error (MAE) horizontally between the simulated flight and the reconstructed flight. Every color indicates a city pair.

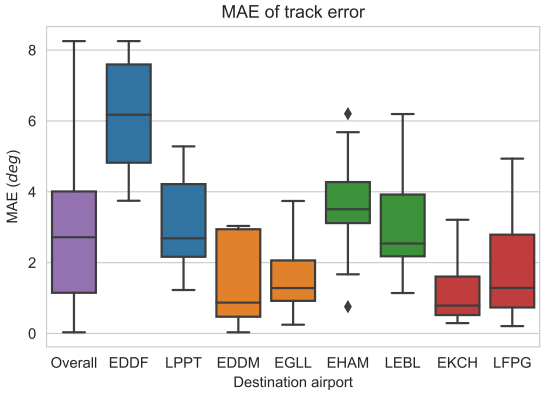


Figure 12. The mean absolute error (MAE) between the track angles of the two aircraft. Every color indicates a city pair.

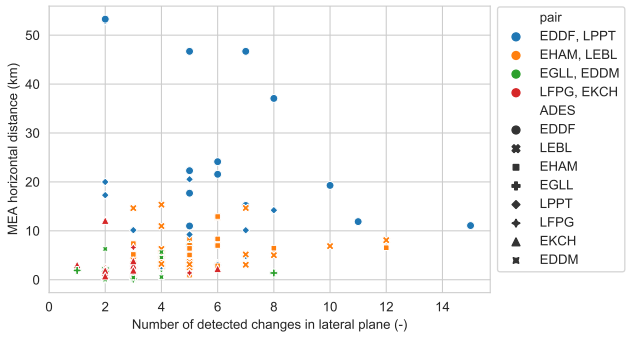


Figure 13. The mean absolute error of the horizontal distance difference against the number of detected changes in the lateral plane.

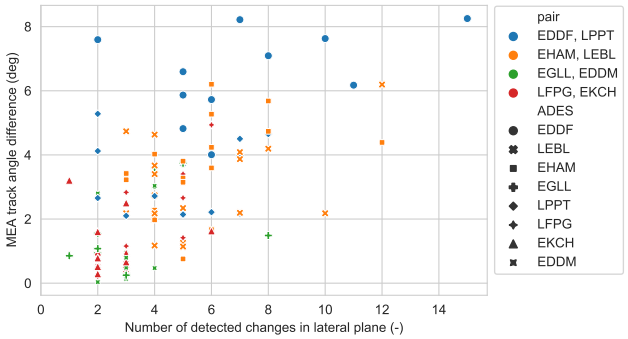


Figure 14. The mean absolute error of the track angle difference against the number of detected changes in the lateral plane.

B. *Vertical*

The vertical error is calculated by subtracting the altitude of the reconstructed flight from the simulated flight. This is also done at every timestamp. Looking at fig. 15, it can be seen that overall the errors are less sporadic when compared to the horizontal errors. There are certain time slots where errors are more common. These exist mostly towards the beginning and the end of a trajectory. The exception on this being LEBL, where the differences occur after a quarter of the flight, although these don't persist for the entire flight. These results and their causes will be discussed in the next section.

In fig. 16 the MAE of the vertical error is presented. It can be seen that differences between city pairs are less pronounced. For every destination, the 75^{th} percentile is below the 125 meter mark. On average the 75^{th} percentile is below $100m$.

Another noticeable feature is the low error scores for EDDF and especially for LPPT. This is in contrast with the results in the lateral plane. This will be discussed in the following section.

Fig. 17 shows the correlation between the number of detected changes and the vertical MAE. A positive correlation is observed from the data. This suggests that more changes in the vertical plane will lead to more MAE. This is expected since a more complex vertical profile is harder to precisely follow and more prone to errors compared to a simple profile.

VIII. Discussion

This section discusses the results presented before. First, the horizontal results are discussed. Secondly, the vertical results. For both sections, the sources of the errors will be discussed first, after which the results will be discussed along with any correlations between these two.

A. *Horizontal results*

For errors in the horizontal results, 3 general error sources have been identified. These will be explained below.

The first error source is missing data. Because the data is missing, no detection of turns can be performed, and no commands can be inferred. Because no commands are given, the reconstructed flight path will steer an aircraft towards the

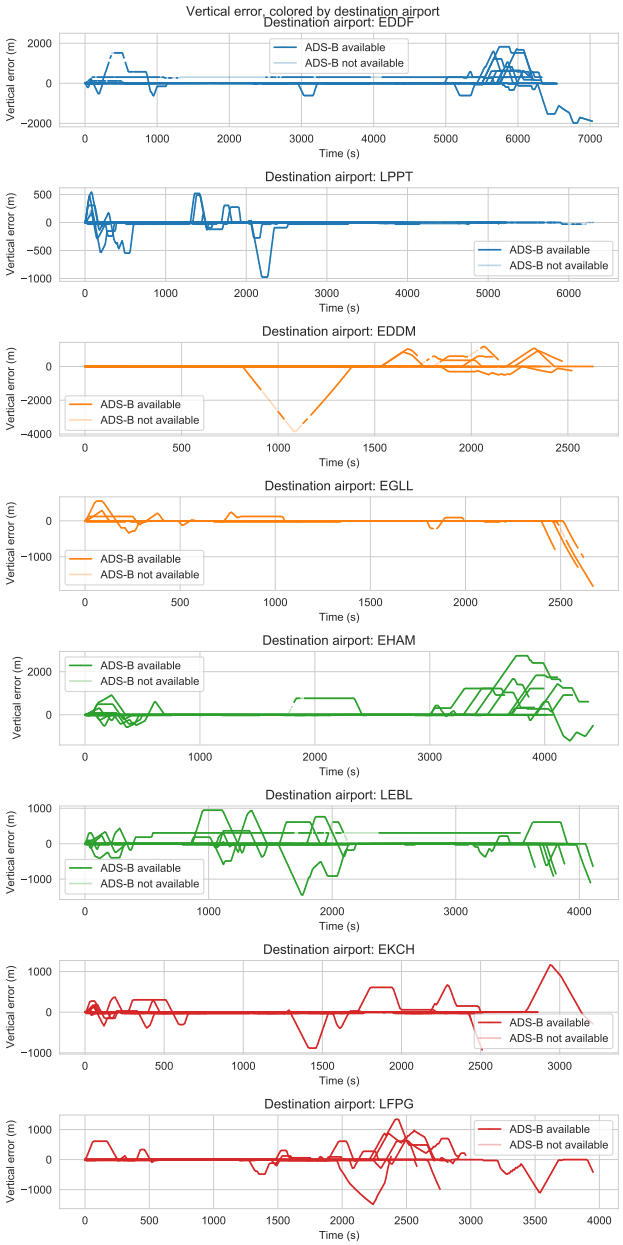


Figure 15. The vertical error between the simulated flight and the reconstructed flight. Every color indicates a city pair. Lighter colored lines indicate that ADS-B was not available at these timestamps.

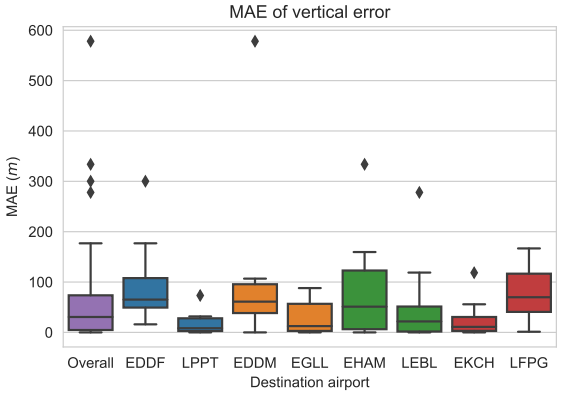


Figure 16. The mean absolute error (MAE) vertically between the simulated flight and the reconstructed flight. Every color indicates a city pair.

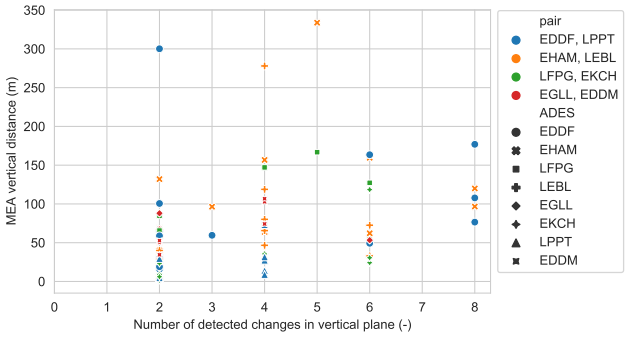


Figure 17. The mean absolute error (MAE) vertically between the simulated flight and the reconstructed flight against the number of detected changes in the vertical plane.

next detected ATC command, which may not be the actual next ATC command.

The second source is the detection of a wrong waypoint. This happens most commonly when a waypoint in the original flight plan is near the turn, but it is not the waypoint where the original aircraft is going. If this is the only waypoint (in the flight plan) that is detected, it negatively influences the error.

The third source of errors are long sections of straight flight, after which the data stops. Data is available, but no turns are detected along the straight sections. This means that no commands are given to the reconstructed flight and small errors can integrate over time.

As an example fig. 18 is given. In this case, the waypoint BSN is deemed to be the most likely candidate for the turn detected at DIDOR. This is because BSN is in the filed flight plan. However, in hindsight, the aircraft got a direct to ATLEN or UVUDO. ATLEN wasn't detected because this is outside the search area (depicted by the red lines) and UVUDO was not selected because it is not in the flight plan. This example also shows that a potential turn at ATLEN was not detected due to missing data and therefore no other commands were detected. After BSN, the aircraft flew straight towards the destination (LPPT). The simulated aircraft followed the original ADS-B points, with straight lines between the missing data. Other examples are given in Appendix E.

A broader view of the results shows that the horizontal difference error is the largest for the flights between EDDF and LPPT. The flights towards EDDF have the largest error and start to accumulate between 1000 & 2000 seconds. This can also be observed in the track angle error, where the track angles deviate from the 0 line and remain there. The return flights have a similar (although not as obvious) trend in horizontal distance after the 3000 seconds mark. This effect may partially be explained by the lack of data points over the

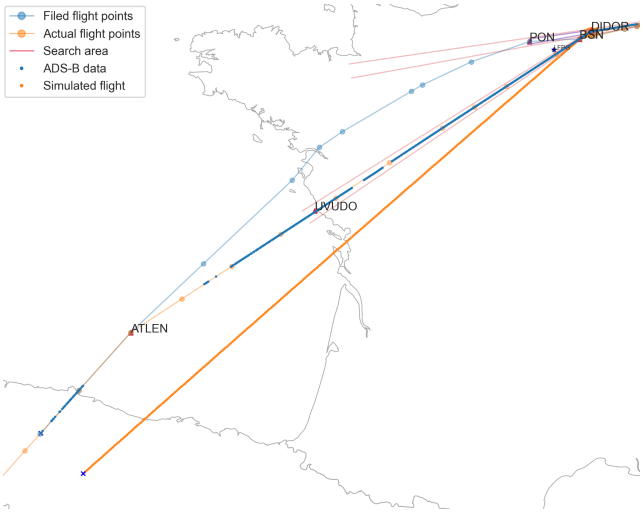


Figure 18. A flight between EDDF and LPPT and the reconstructed flight points. The blue line is the original ADS-B data. The orange line is the reconstructed flight. The light blue line with dots is the filed flight plan. The orange line with dots is the actual flown points (from Eurocontrol) and the red lines are the search area for waypoints.

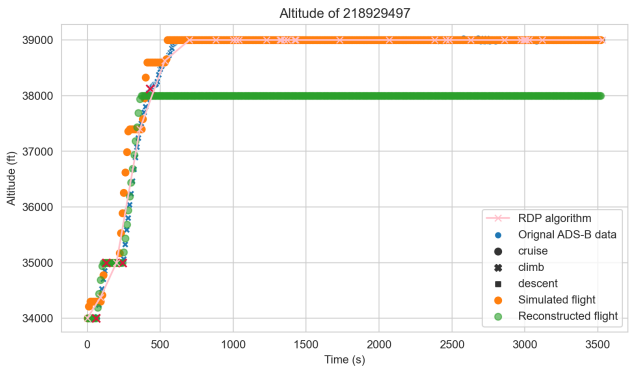


Figure 19. The altitude profile of flight 216919497. This graph shows both an error introduced by using the RDP algorithm. This is seen at $37500ft$ in the simulated flight data. It also shows a missed detection in the original data after $38000ft$.

Bay of Biscay and the north of Portugal as shown in the plots by the lighter coloured lines and in fig. 9. Two other effects at play here are the relative sparsity of waypoints over the sea and the larger amount of time spent flying. Fewer waypoints mean that a good detection is less likely. Waypoints can be further away than the $500km$ search area used, or just outside the $1°$ envelope. The longer time spent flying means that small errors in the track angle have the time to integrate into large horizontal difference errors.

Other city pairs, EDDM-EGLL & EKCH-LFPG, have smaller errors and also do not have any gaps in the original ADS-B data. Therefore changes could be detected along the entire trajectory. These changes could be formulated into possible ATC commands to create a better matching flight plan. These flights are also mostly over land, where waypoints are more abundant than above sea. This makes a good detection more likely. These flights are also shorter, meaning that small errors don't have as much time to integrate.

Fig. 13 plots the MAE of the horizontal distance against the found number of lateral changes. No clear correlation can be made between these two. This is in contrast when comparing the number of detected changes against the track angle. This is shown in fig. 14. This shows a positive trend. The reasons why the horizontal distance may not observe this behaviour are numerous. The track angle can differ to reduce the horizontal distance. There may not be enough time for track angle differences to integrate into large horizontal distance errors. Other factors, such as missing data or detection of wrong waypoints may have more influence on the horizontal distance than the track angle.

B. Vertical results

Errors in the vertical plane can be attributed to three major factors. The first is a missed detection. The flight phases can be scrambled, such that no clear distinction can be made where a phase change occurs. Since this change is missed, no command is given to the aircraft.

The second source can be due to errors in the original ADS-B data. These errors can be small, but sometimes can be large. When these points are used to make a flight plan for the simulated flight, the simulated flights will follow these. Since this is mostly one single point, the altitude will return to the correct state afterwards. This creates a triangle-shaped extrusion as seen in a trajectory towards EDDM. This can be seen in Appendix E, fig. 27.

The last source of errors is the use of the RDP algorithm. This was applied to the positional data of the ADS-B to reduce the number of data points for the simulation to make. These datapoint are used as waypoints in the simulation. BlueSky tries to optimise the altitude while being restricted to these waypoints. This combination can produce a stair pattern instead of a smooth climb or descent.

An example of the above errors can be seen in fig. 19. Here two of three examples can be seen. The RDP induced error can be seen around an altitude of $37500ft$. A datapoint here was used for the creation of the simulated flight. Therefore a plateau can be seen in the orange dots around this altitude instead of a smooth ramp as the blue. The second error is a missed detection. After FL380 the original ADS-B data does not increase as rapidly as before. The flight phases here oscillate between climb and cruise. The cruise phase was misidentified due to this oscillation and thus the command for FL380 was given to the reconstructed flight. Two more examples are given in Appendix E.

Flights toward EDDF, EDDM, EHAM, and LFPG are the flights with the most errors. These mostly occur at the end of the flight, suggesting it may be due to the busy airspace above Northern France, the Netherlands, and western Germany. This can complicate descent sequences and make it harder to reproduce a flight plan. This statement does not hold for EGLL, while the southeast of England is also considered to be a busy airspace.

The last noticeable feature is that the flight towards LEBL has errors that start after a quarter into the flight. One flight continues to have this error, while the rest of the flights being only temporary erroneous. An altitude command at this point might explain why this occurs. For some flights, this is better detected and converted into ATC commands than for others, creating this pattern.

Overall the results indicate that the found ATC commands can reproduce the original trajectory. It does become harder to follow as the altitude profiles become more complex as was shown in fig. 17. Furthermore, enough datapoints need to be available. When no datapoints are available in the original ADS-B data, no ATC commands can be detected and errors start to grow.

IX. Conclusion

This study used past ADS-B data to detect ATC commands in trajectory data. The ATC commands focused on were: altitude change commands, heading change commands, 'direct to' commands and speed change commands.

A preliminary analysis has been conducted to find which smoothing function was able to remove noise from ADS-B most effectively. It was found that time-based resampling and selecting the median of the sample performed best. This is because time-based resampling provided the best reduction in noise. Furthermore, the added benefit of the median is that single big outliers are discarded. Three window sizes were tested for resampling. The 10 second window was the best trade-off between keeping high temporal accuracy and minimising the noise level.

A rule-based analysis was able to detect lateral and vertical changes in the ADS-B data. Lateral changes were identified by using the turn rate of an aircraft. A combination of a threshold value and a K-sigma value is used to identify turns. Changes in the identified flight phase were used to detect changes in altitude. It was also found that detecting speed changes is not possible using this method. Effects of the wind on the speed produced a significant level of noise and variations in the speed data.

The rule-based analysis was further expanded upon by adding a second rule-based framework to distinguish track angle changes into 'heading' commands and 'direct to' commands. This framework searches for waypoints along the trajectory of aircraft and uses a cone-shaped area to search for waypoints. This was used before a turn started and after the turn ended.

A selection of aircraft trajectories was subjected to the rule-based framework. The found 'heading', 'direct to' and 'altitude' commands are then combined to create a flight plan. This flight plan is compared against another flight plan based on the original ADS-B data. The comparison was done by simulating both flight plans in BlueSky. Differences between the two flight plans are used to test the fidelity of the found method. The results showed that the mean absolute error (MAE) of the horizontal distance was less than $10km$ for the 75^{th} percentile of flights. It was also less than $4°$ in track angle difference and around $80m$ in vertical distance error.

It was found that that method still has several sources of errors. For the lateral plane, these include missing data, detection of wrong waypoints, and wrong sizing of the search area. Each of these can cause errors and when multiple of these occur at the same time they will enhance each other, making the reconstruction error larger. A notable feature in the lateral plane was the effect the complexity of a trajectory has on the errors. While a positive correlation was found with the track angle, this did not directly translate to the horizontal distance error. Other factors such as missing data, integration time, or a return to normal trajectory has a larger effect on this error than the complexity.

In the vertical plane, it was found that a more complex altitude profile resulted in larger reconstruction errors. Since more options for errors are available in a complex profile, more errors will be made. Sources of errors can be generalised to faulty datapoints, missed detections, or the RDP algorithm induced errors.

This paper set out to test if it was possible to detect ATC commands in part ADS-B data. It was found that a rule-based framework could form a list of possible ATC commands and that these ATC commands could successfully recreate the original aircraft trajectory if enough ADS-B data is available along the entire trajectory and the data is of sufficient quality. However, the method proposed in this paper is not perfect and several recommendations can be made to improve the proposed method.

X. Recommendations

Several recommendations can be made for further study of this topic.

- A method of increasing the accuracy over an area with sparse ADS-B data may be to adjust the parameters for detecting waypoints, as presented in section V-B. The length and width of the search area could depend on the availability of ADS-B data.
- ATC can usually only control aircraft within their own FIR. That is why a lot of waypoints are defined on the borders of FIRs. These mark certain points where aircraft can cross from one FIR to another. When determining waypoints for the 'direct to' commands, these boundary waypoints should be given prioritisation.
- Another suggestion for finding more accurate waypoints might be by using the frequency of waypoints as guidance. In section III-B a method is given to determine waypoint usage by airlines by counting how many times a waypoint is used in a flight plan. Waypoints with a higher count could be given priority when looking for 'direct to' commands.
- Further analysis may be done on temporary shifts in track angle. These temporary shifts are changes that only occur for a short amount of time before restoring to the previous angle. These occurrences may be ATC commands that

redirect an aircraft to avoid another aircraft, before being sent back to their original flight plan.

- A more in-depth look may be given to airways. In an airspace without FRA, these airways may still be useful. The original goal could be accomplished and any change from the airways could be an ATC command. These airways could also be used as a reference for clustering algorithms or anomaly detection algorithms. This baseline might reduce the amount of data needed to reach the required level of precision.

XI. Applications

This section presents two sample applications of the proposed framework and tries to show some applications that might arise from being able to detect ATC changes.

A. Connecting Live ATC radio and ADS-B data

The MUAC ATC radio was compared to the found commands by the presented framework. The Opensky database was queried to select one hour of data on the 28th of May, 2021. It was found that between 12:00 and 13:00 UTC, a total of 655 aircraft flew above 7500m in the MUAC airspace. The Opensky database was queried again to retrieve the ADS-B data from this region. The ATC recording were downloaded from *www.liveatc.net*. All recordings from the MUAC region were combined into one audio file.

The ADS-B was processed using the methods presented in this paper and was simulated in BlueSky to visualise the situation. Instead of giving the aircraft a set of commands, the detected changes in track angle and altitude are displayed in the console of BlueSky. If the audio file is played together with the BlueSky simulation a comparison can be made. The files for these can be found on Github: https://github.com/Brentebol/LiveATC_bluesky.

B. Airspace analysis for MUAC controlled airspace

The second application tested is an analysis of the MUAC upper airspace. For this analysis, the ATC commands detection framework presented in this paper is applied to ADS-B data collected in the MUAC airspace. 5 days of data (2 to and including 6 of July 2018) gathered by the TU Delft receiver was used to visualise hotspots of changes in the upper airspace. Both lateral changes and vertical changes were visualised and can be found in Appendix H.

One example of a noticeable hotspot is the waypoints located on the Dutch-English border. A lot of aircraft made turns just after or before crossing the borders. This can be due to the different routing methods. The MUAC airspace allows for free routing, where the English airspace is more strict with following waypoints.

References

[1] Junzi Sun, Jacco M. Hoekstra and Joost Ellerbroek. "Estimating aircraft drag polar using open flight surveillance data and a stochastic total energy model". In: *Transportation Research Part C: Emerging Technologies* 114 (2020), pp. 391–404. ISSN: 0968090X. DOI: 10.1016/j.trc.2020.01.026.

[2] Xavier Olive et al. "Detecting Events in Aircraft Trajectories: Rule-Based and Data-Driven Approaches". In: *Proceedings* 59.1 (2020), p. 8. ISSN: 2504-3900. DOI: 10.3390/proceedings2020059008.

[3] Junzi Sun, Joost Ellerbroek and Jacco Hoekstra. "Large-Scale Flight Phase Identification from ADS-B Data Using Machine Learning Methods". In: *7th International Conference on ...* August (2016). URL: https://www.researchgate.net/profile/Junzi_Sun/publication/305403993_Large-Scale_Flight_Phase_Identification_from_ADS-B_Data_Using_Machine_Learning_Methods/links/578dece708ae9754b7e9c6e8.pdf.

[4] Junzi Sun, Joost Ellerbroek and Jacco Hoekstra. "Flight extraction and phase identification for large automatic dependent surveillance-broadcast datasets". In: *Journal of Aerospace Information Systems* 14.10 (2017), pp. 566–571. ISSN: 23273097. DOI: 10.2514/1.I010520.

[5] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: (1996), pp. 226–231. URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.9220.

[6] Jacco M. Hoekstra and Joost Ellerbroek. *AE4321-15 Air Traffic Management*. Delft University of Technology, 2019.

[7] Guan Yuan et al. "A review of moving object trajectory clustering algorithms". In: *Artificial Intelligence Review* 47.1 (Jan. 2017), pp. 123–144. ISSN: 15737462. DOI: 10.1007/s10462-016-9477-7.

[8] C A Dek. *Predicting 4D Trajectories of Aircraft using Neural Networks and Gradient Boosting Machines*. Tech. rep. Delft University of Technology, Delft, The Netherlands, 2020, pp. 1–16.

[9] Maxime Gariel, Ashok N. Srivastava and Eric Feron. "Trajectory clustering and an application to airspace monitoring". In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 1511–1524. ISSN: 15249050. DOI: 10.1109/TITS.2011.2160628.

[10] Mayara Condé Rocha Murça et al. "Trajectory clustering and classification for characterization of air traffic flows". In: *16th AIAA Aviation Technology, Integration, and Operations Conference* June (2016), pp. 1–16. DOI: 10.2514/6.2016-3760.

[11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://scikit-learn.sourceforge.net..

[12] Marco Enriquez. “Identifying Temporally Persistent Flows in the Terminal Airspace via Spectral Clustering”. In: *Proceedings of the 10th USA/Europe Air Traffic Management Research and Development Seminar, ATM 2013* (2013).

[13] L. Basora, X. Olive and T. Dubot. “Recent advances in anomaly detection methods applied to aviation”. In: *Aerospace* 6.11 (2019). ISSN: 22264310. DOI: 10.3390/aerospace6110117.

[14] Xavier Olive et al. “Detecting controllers’ actions in past mode S data by autoencoder-based anomaly detection”. In: *SESAR Innovation Days* December (2018). ISSN: 07701268.

[15] Charles Truong, Laurent Oudre and Nicolas Vayatis. “ruptures: change point detection in Python”. In: (Jan. 2018). URL: http://arxiv.org/abs/1801.00826.

[16] Ajay Srinivasamurthy et al. “Iterative learning of speech recognition models for air traffic control”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. Vol. 2018-Septe. International Speech Communication Association, 2018, pp. 3519–3523. DOI: 10.21437/Interspeech.2018-1447.

[17] Shreyas Vathul Subramanian, Peter Kostiuk and Graham Katz. “Custom IBM watson speech-to-text model for anomaly detection using ATC-pilot voice communication”. In: *2018 Aviation Technology, Integration, and Operations Conference* (2018). DOI: 10.2514/6.2018-3979.

[18] Matthias Schäfer et al. “Bringing up OpenSky: A large-scale ADS-B sensor network for research”. In: *ACM/IEEE International Conference on Information Processing in Sensor Networks,* 2014. URL: http://www.opensky-network.org.

[19] Xavier Olive. “traffic, a toolbox for processing and analysing air traffic data”. In: *Journal of Open Source Software* 4.39 (July 2019), p. 1518. ISSN: 2475-9066. DOI: 10.21105/joss.01518. URL: http://joss.theoj.org/papers/10.21105/joss.01518.

[20] Jacco M. Hoekstra and Joost Ellerbroek. “BlueSky ATC simulator project: an open-data and open-source approach”. In: *Proceedings of the 7th International Conference on Research in Air Transportation* (2016), pp. 1–8.

[21] Eurocontrol. *Maastricht Upper Area Control Centre (MUAC) upper air route structure chart | EUROCONTROL*. Dec. 2020. URL: https://www.eurocontrol.int/publication/maastricht-upper-area-control-centre-muac-upper-air-route-structure-chart.

II

Appendices

A

Other investigated potential ATC command detection methods

A.1. Clustering

clustering of trajectories based on similarities was examined to detect ATC commands. Outliers or differences in trajectories could be due to ATC commands. Two types of clustering were tested. The first methods use clustering algorithms to cluster groups of trajectories together. The second used the data from the Eurocontrol R&D to group trajectories together based on Eurocontrol data.

A.1.1. Clustering algorithms

Two clustering methods were tested for this study. Using PCA and DBSCAN as proposed by [1, 2] and spectral clustering as proposed by [3]. Both PCA and DBSCAN are included in the Scikit-learn Python package [4]. The spectral clustering algorithm from [3] was applied since it has been adjusted to work directly on trajectory data.

A.1.2. Clustering by Eurocontrol data

Data from the Eurocontrol R&D release can be used to cluster certain flights together. The *flights* table contains information such as departure airport, destination airport, operator, aircraft type, which all can be used to group certain trajectories.

A test was performed on one day of Lufthansa flights in the MUAC airspace to establish if this method had any potential for detecting ATC commands. DBSCAN, spectral clustering and clustering by destination airport were applied to the dataset. It was found that these clustering methods did not meet the requirements to detect ATC commands. There are several reasons for these. 1) Every flight has been given ATC commands, therefore no unaltered trajectories exist and no difference can be found between altered and unaltered trajectories. 2) Clustering methods can not distinguish between the small differences that an ATC command can give a trajectory. 3) Depending on the trajectories selected, the parameters of the clustering algorithms have to be changed because of the differences between trajectory types (all same direction versus all scattered).

A.2. Airways

This method uses airways as guidelines for aircraft. The hypothesis was that aircraft will mostly likely follow their flight plan. This flight plan is in turn made using airways. Any deviations from this flight plan, may be an ATC controllers given commands to the aircraft. To test this hypothesis, the MUAC airspace was studied. Airway data was taken from sources mentioned in III-C. The Eurocontrol airspace charts [5] were used as qualitative verification.

It was found that none of the available data sources had the accuracy to be used for this kind of analysis, as seen in fig. A.1. Airways were named inconsistently, the airways did not overlap or contained errors that made these unusable. It was also concluded that, MUAC uses free routing airspace (FRA), which encouraged

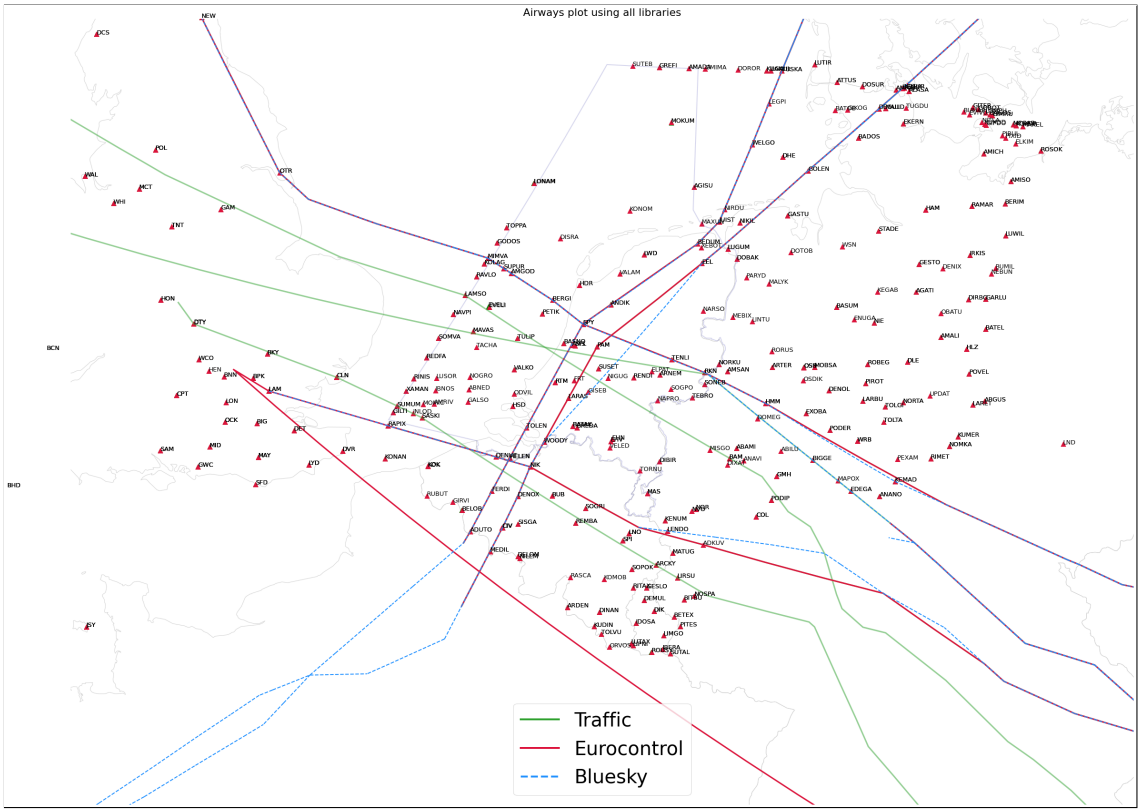


Figure A.1: Several airways (L602, l603, L604, l608, UL610, N872, N873) are plotted using three different databases.

ATC and airlines to fly direct routes through the airspace, leaving airways in this area unused. Therefore this approach was dropped for this study.

B

Data Pipeline

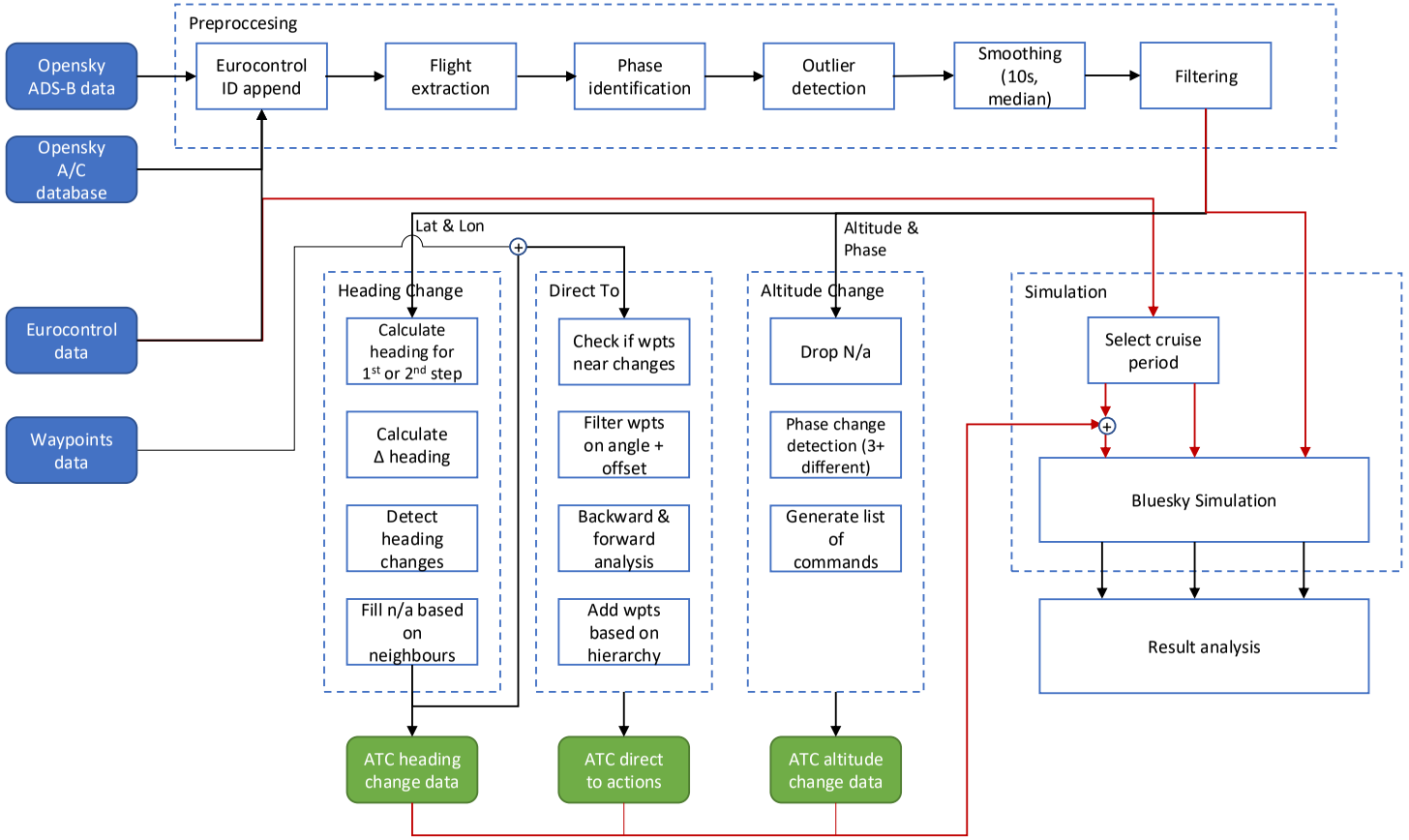


Figure B.1: The data pipeline structure of this study.

C

Smoothing Results

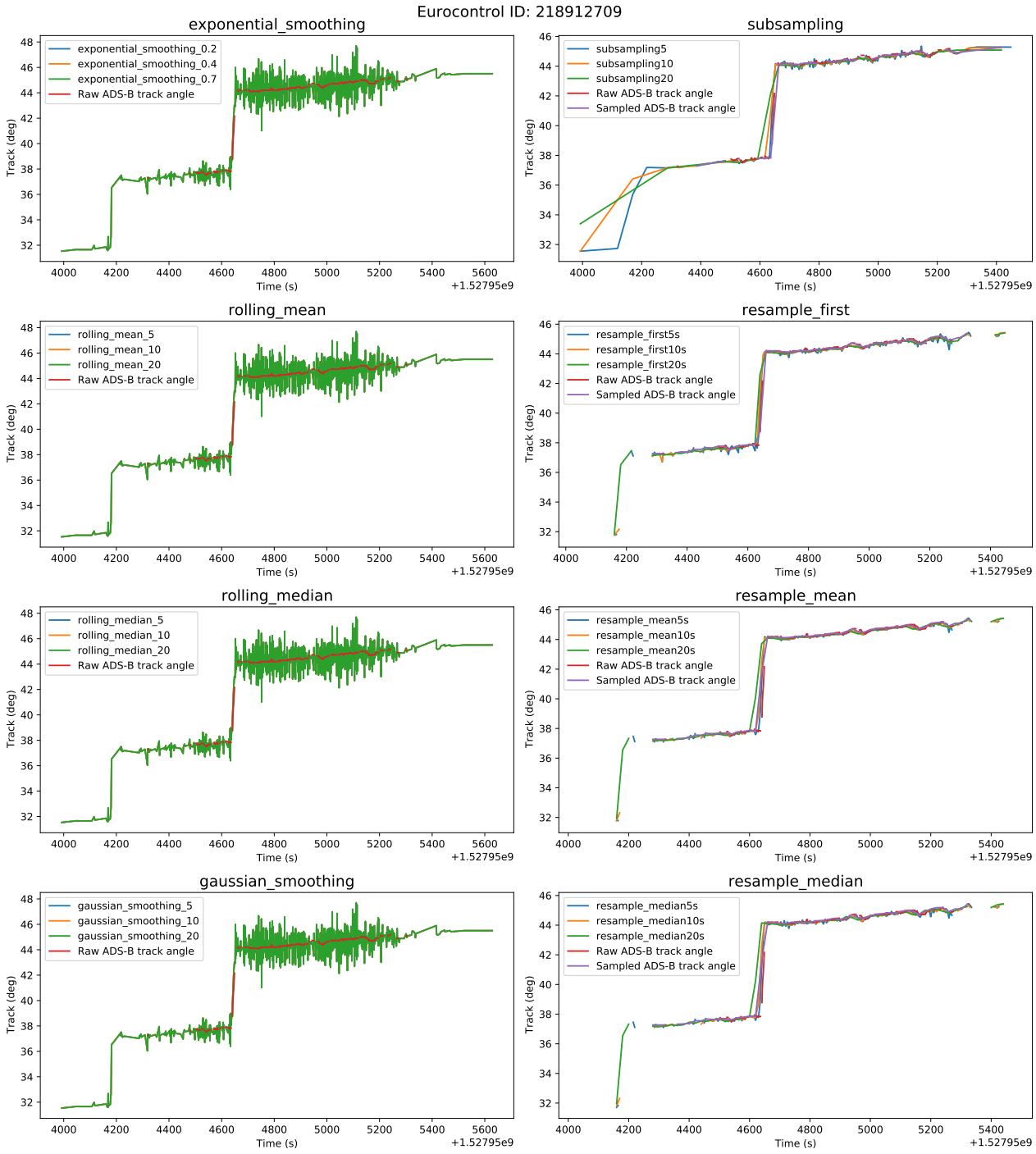


Figure C.1: Different smoothing functions are applied to a single flight (Eurocontrol ID 218912709). Data gathered by the TU Delft ADS-B receiver.

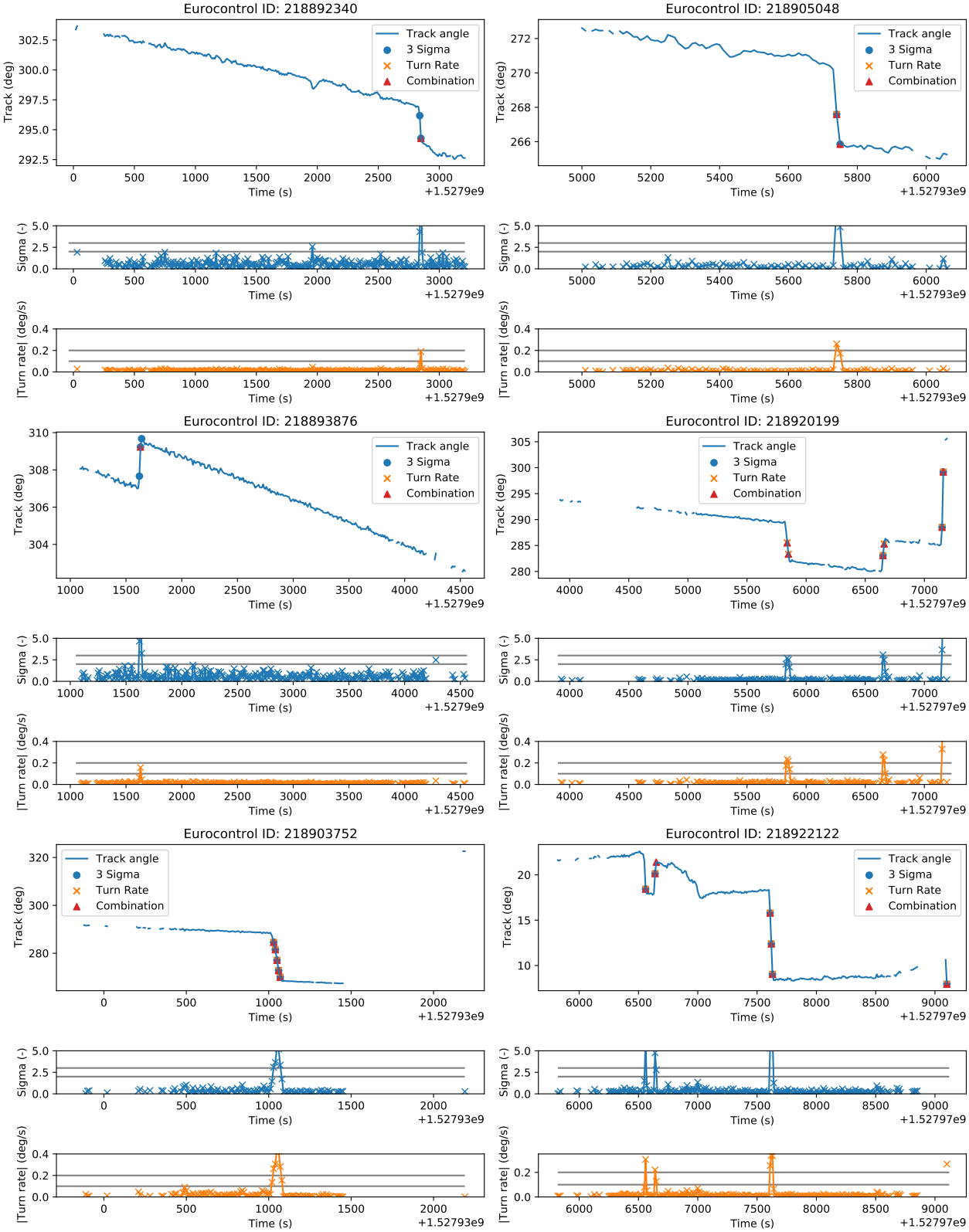


Figure C.2: A sample of 5 trajectories are used to determine the values for the detection methods. The top figure is the trajectory, with the dots being different detection methods. The middle figure is the standardize Δtrack, with guidance lines at 2 & 3 sigma. The lower figure is the Δtrack, with guidance lines at 0.1 & 0.2

D

Navaid detection

Figure D.1 visualises the search area of the used to determine the navaids as described by section V-B. In the figure below, the flight starts in France, near the border with Belgium. From here, the flight continues over the Netherlands to fly towards it end goal: Copenhagen.

The trajectory starts with a small change at ADUTO towards TOLEN. At TOLEN, the aircraft turns right towards Denmark. Here the aircraft also deviates from its planned flight path to fly a more direct route. At DEGUL (unreadable due to overlap with other waypoints) where it makes another turn towards the endpoint of its cruise phase. In total 3 turn are detected in this trajectory. For these the detection table is given in table D.1.

Table D.1: Waypoints found for flight Eurocontrol id 219079713

Turn	Most likely waypoint	Waypoint in flight plan near current turn	Waypoint in flight plan in heading before next turn	Waypoint in flight plan in heading after current turn	Waypoint near current turn	Waypoint in heading before next turn	Waypoint in heading after current turn	Track angle before turn (deg)	Track angle after turn (deg)
0	ADUTO	ADUTO	FERDI, HELEN	FERDI, HELEN, TOLEN	ADUTO	FERDI, HELEN	FERDI, HELEN, TOLEN, STD, ZANDA, BETUS, ANDIK, UNEXO, VALAM		26.5
1	TOLEN	TOLEN	TOLEN	FERDI, HELEN, TOLEN	TOLEN	TOLEN, STD, EKROS	FERDI, HELEN, TOLEN, STD, EKROS, SPY, GOMKU	26.8	24.0
2	DEGUL				DEGUL	AMRAK	EEL, KUBAT, LUGUM, GOLEN, DEGUL, MALID, LISBU	24.4	43.9
3	KOR			KOR			DEMIR, KESUR, OSKEV, IBNIL, KOR, GATMA	47.9	57.3

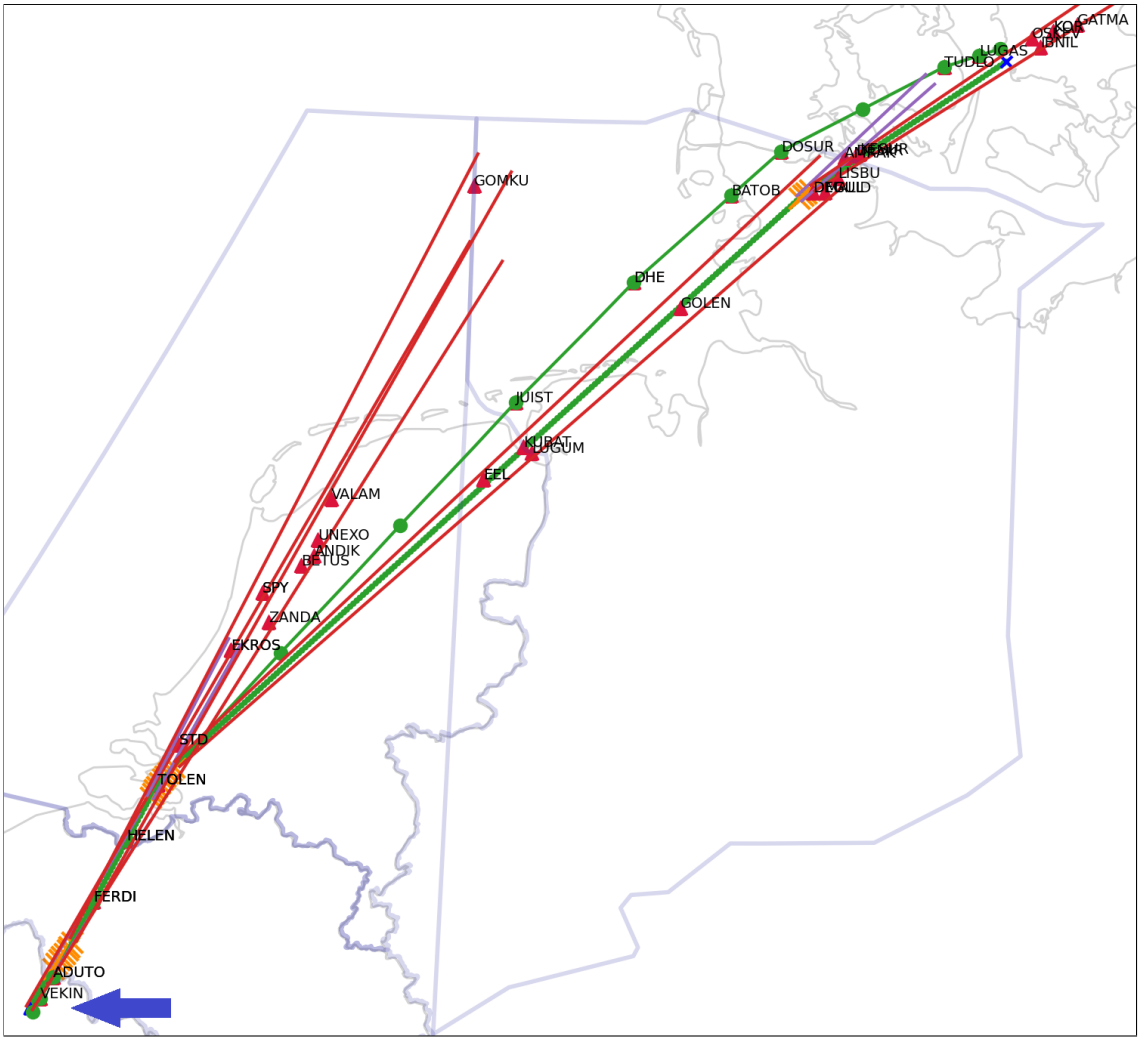


Figure D.1: A single trajectory (Ectrl id: 219079713) with the search area for navaids visualised. The blue arrow indicates the start of the trajectory. Red lines indicate the search area right after a turn. Purple lines indicate the search area before a turn. The green line represent the ADS-B data. The green line with big dots is the scheduled flight plan.
Note: The waypoints in the extended flight plan are also included. These are not shown for readability purposes.

E

Breakdown of errors in simulation

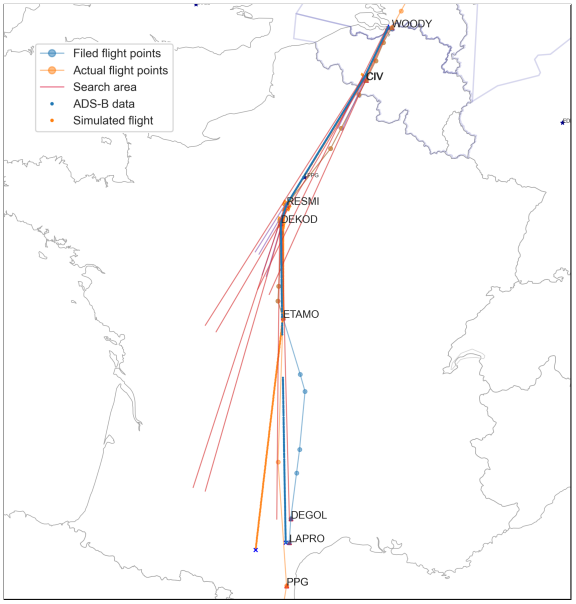


Figure E.1: Flight 219058962 towards LEBL. At the end of the original ADS-B data, a long section of straight flight is flown. In this section no turns are detected. The reconstructed flight turn towards its destination after the last waypoint has been flown. Which is ETAMO in this flight. DEGOL and LAPRO are outside of the search area and therefore not considered for direct to commands. Most likely PPG was the given direct to command, given the track angle of the original ADS-B trajectory.

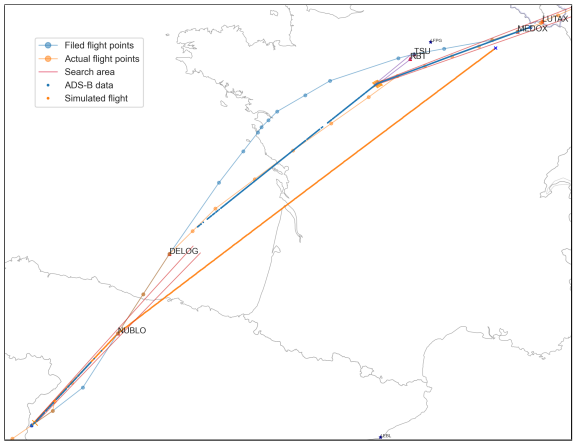


Figure E.2: flight 218929651 towards EDDF. This flight contains no datapoints over the bay of Biscay. Here two turns should be made when looking at the actual flight point (light orange line with dots). Since these turns are missing. The next ATC commands was used. This was a direct to LUTAX. Since no flight points were detected right after the turn above France, and this was the furthest when extending the turn.

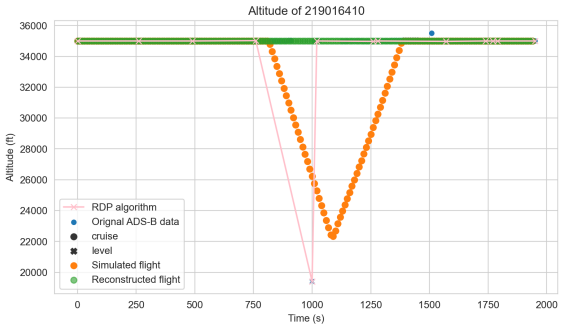


Figure E.3: Altitude profile of flight 21906410. A faulty datapoint was used to simulate the flight in BlueSky, resulting in the flight profile above.

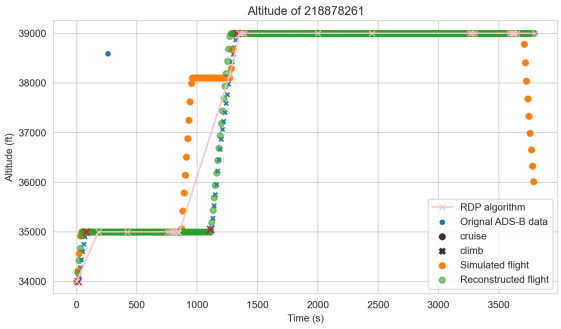


Figure E.4: Altitude profile of flight 21906410. The RDP algorithm creates a step pattern in the simulated flight (orange).

E.1. Flights for simulation

Table E.1: Flights that are used in the simulation

ADES	AC Operator	Registration	Icao code	Callsign	Eurcontrol id
EDDF	DLH	DAISF	3c6666	DLH42A	218915923
				DLH23C	218929651
		DAISN	3c666e		218947171
					218963067
	RYR	EIDAN	4ca1d4		218870080
					218966852
		EIDLX	4ca303		219067967
	TAP	CSTNU	4951d5		218869224
					218913360
				TAP576N	218977347
					219010659
		FHBAL	39840b	TAP572C	218984315
				TAP572C	219017273
LPPT	DLH	DAISF	3c6666	DLH72W	218909125
				DLH07M	218922057
		DAIWA	3c666e		218938350
					218952826
	RYR	EIDAN	4ca1d4		218863324
				RYR911J	218959666
		EIDLX	4ca303		219060908
	TAP	CSTNU	4951d5		218876321
				TAP579	218919368
					218984458
				TAP579	219017682
		FHBAL	39840b	TAP571	218992730
				TAP571	219025804
EDDM	BAW	GEUOD	40093d	BAW948M	218928654
				BAW950M	219066569
		GEUPC	400803	BAW948M	219029054
				BAW952M	219040013
				BAW960M	219078834
	DLH	DAIPF	3c6606	DLH1MF	219016410
				DLH4MC	219042097
		DAIWA	3c66e1	DLH1MF	218886814
				DLH4RJ	219033053
				DLH6TN	219063320

Table E.2: [Continued] Flights that are used in the simulation

ADES	AC Operator	Registration	Icao code	Callsign	Eurcontrol id
EGLL	BAW	GEUOD	40093d	BAW949L	218932918
				BAW951L	219072187
		GEUPC	400803	BAW949L	219034155
				BAW961L	219084335
	DLH	DAIPF	3c6606	DLH1LC	219010955
				DLH9MF	219036416
		DAIWA	3c66e1	DLH1LC	218881351
				DLH6MM	219027581
EHAM	KLM	PHBXA	484130	KLM86N	218934677
				KLM86N	219001956
				KLM88T	219081107
		PHBXC	484132	KLM1670	218939152
				KLM76T	218951487
				KLM1670	219045231
	TRA	PHHSG	484f6d	TRA13R	218885687
				TRA132K	218963887
		PHXRD	4841db	TRA13R	219047326
				TRA132K	219064324
				TRA13R	219078198
	VLG	ECMGY	344699		218866661
				VLG83NA	218899511
				VLG83NA	218929694
				VLG83NA	218994575
		ECMRF	3453c3	VLG83WG	218886968
				VLG83WG	218917857
				VLG83WG	218953949
				VLG83NA	218960644
				VLG83NA	219028019
LEBL	KLM	PHBXA	484130	KLM85J	218929497
				KLM85J	218996738
				KLM79K	219075278
		PHBXC	484132	KLM65H	218932819
				KLM75K	218945735
				KLM65H	219033809
	TRA	PHHSG	484f6d	TRA85B	218878261
				TRA89A	218958418
		PHXRD	4841db	TRA85B	219038676
				TRA89A	219058962
				TRA85B	219071455
	VLG	ECMGY	344699		218870240
				VLG83PJ	218904769
				VLG83PJ	218932785
				VLG83PJ	219000479
		ECMRF	3453c3	VLG83VF	218891694
				VLG83VF	218922113
				VLG83VF	218956533
				VLG83PJ	218967508
				VLG83PJ	219033558

Table E.3: [Continued] Flights that are used in the simulation

ADES	AC Operator	Registration	Icao code	Callsign	Eurcontrol id
EKCH	AFR	FGKXS	392af2	AFR15HH	218907555
				AFR15HH	218936708
				AFR15HH	219037697
				AFR15HH	219071491
		FGUGH	3950c7		218868752
				AFR105H	218944716
				AFR91PP	218998941
				AFR105H	219079713
	EZY	OEIJD	44015a	EZY53ZT	219051007
		OEIJZ	440128	EZY53ZT	218887890
				EZY53ZT	218981920
	FDX	N923FD	acc9d0	FDX5182	218924342
			ad8b94	FDX5182	218991236
			ad8b94	FDX5182	219057767
	SAS	OYKAS	45ac33	SAS64K	219068659
		OYKBL	45ac4c		218871752
				SAS1560	219017869
LFPG	AFR	FGKXS	392af2	AFR185R	218941691
				AFR185R	219076763
		FGUGH	3950c7		218873841
				AFR83SJ	218949669
				AFR88FE	219003935
				AFR83SJ	219084597
	FDX	N972FD	ad8b94	FDX5181	219022097
				FDX5181	219089760
	SAS	OYKBL	45ac4c		218866573
				SAS559	219013703

E.2. Additional simulation results

This section will provide the mean squared error (MSE), the root mean squared error (RMSE), the normalised root mean squared error (NRSME) and the mean absolute error (MAE) of the performed simulation. The NRSME is calculated by dividing the RMSE by the average error.
The plots are done for the horizontal error and the vertical error.

E.2.1. Horizontal distance error

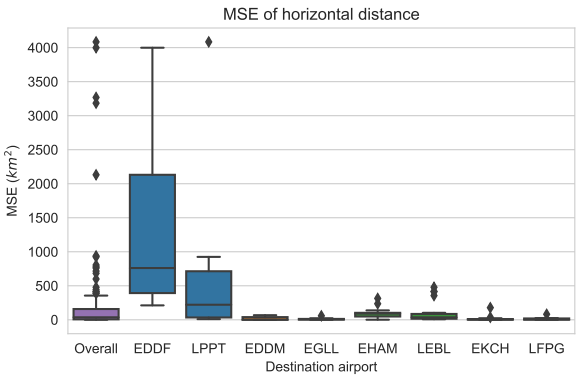


Figure E.5: Mean squared error of the horizontal distance, grouped by destination airport (ADES).

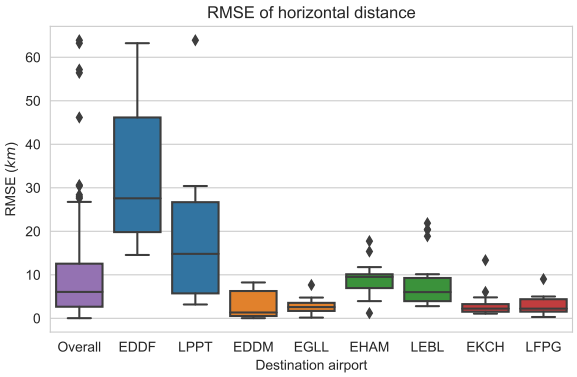


Figure E.6: Normalised root mean squared error of the horizontal distance,
grouped by destination airport (ADES).

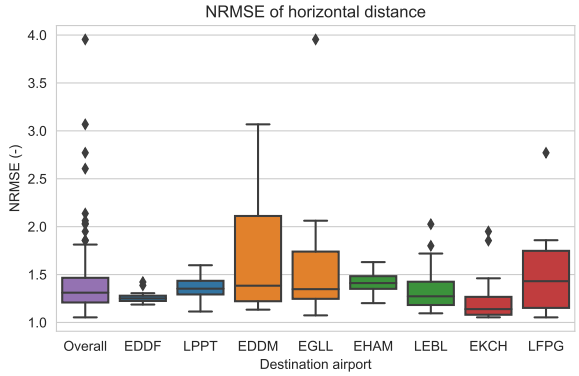


Figure E.7: Normalised root mean squared error of the horizontal distance,
grouped by destination airport (ADES).

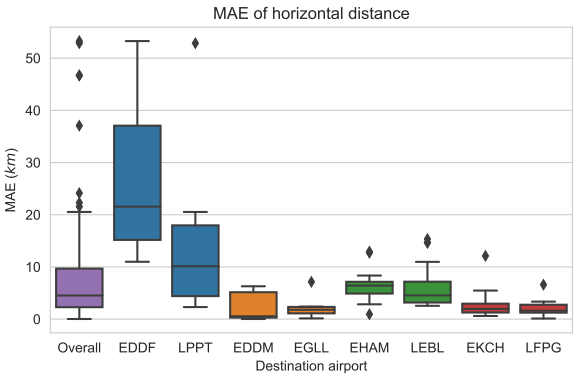


Figure E.8: Mean absolute error of the horizontal distance, grouped by destination airport (ADES).

E.2.2. Vertical error

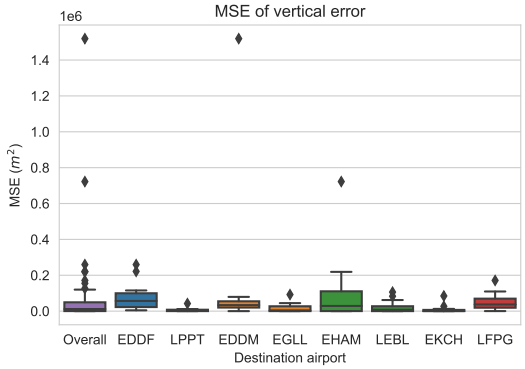


Figure E.9: Mean squared error of the vertical error, grouped by destination airport (ADES).

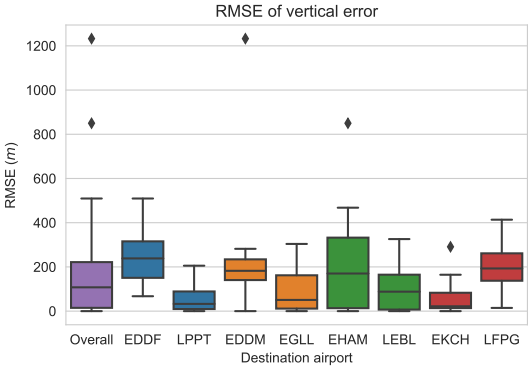


Figure E.10: Root mean squared error of the vertical error, grouped by destination airport (ADES).

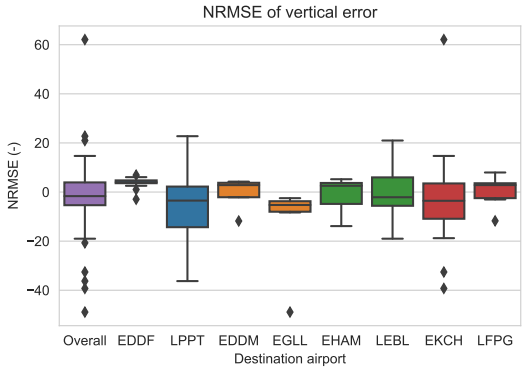


Figure E.11: Normalised root mean squared error of the vertical error, grouped by city pair.

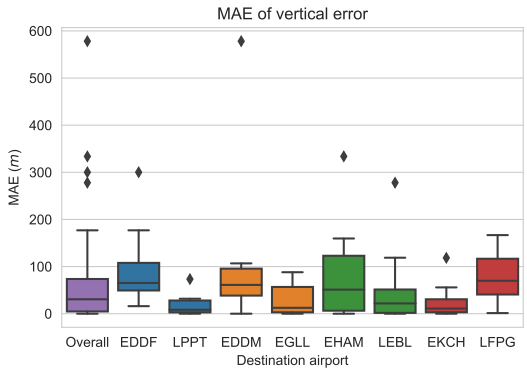


Figure E.12: Mean absolute error of the vertical error, grouped by destination airport (ADES).

E.2.3. Track angle error

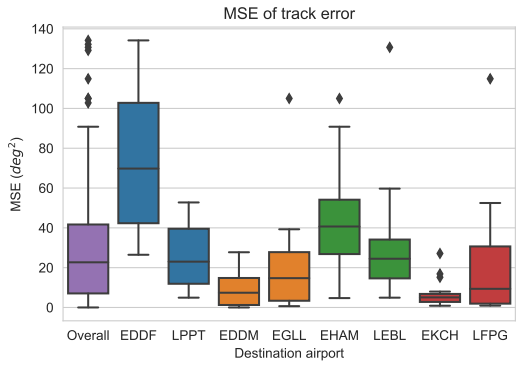


Figure E.13: Mean squared error of the track angle error.

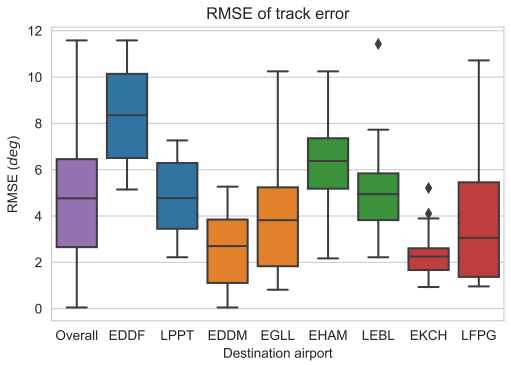


Figure E.14: Root mean squared error of the track angle error.

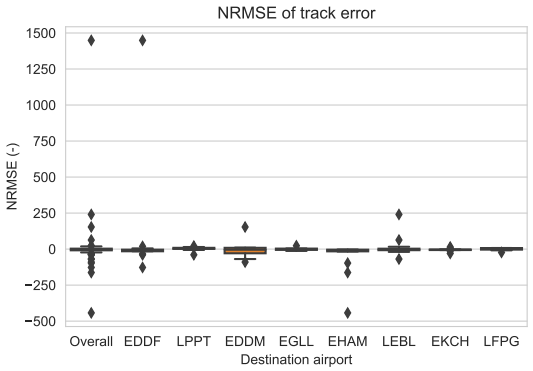


Figure E.15: Root mean squared error of the track angle error, grouped by city pair.

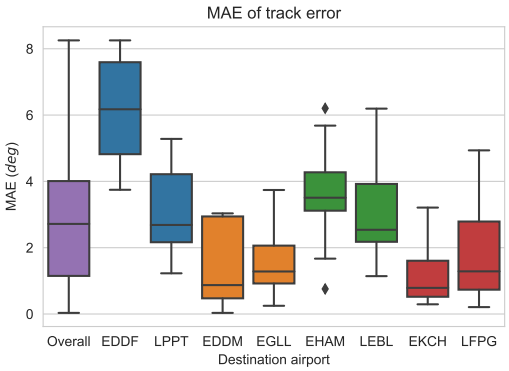


Figure E.16: Root mean squared error of the track angle error, grouped by destination airport (ADES).

E.2.4. Empirical cumulative distribution functions

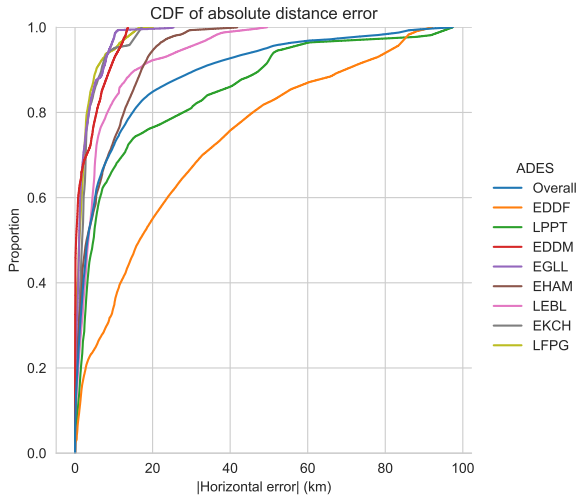


Figure E.17: The empirical cumulative distribution function of the absolute horizontal distance error, colored by city pair.

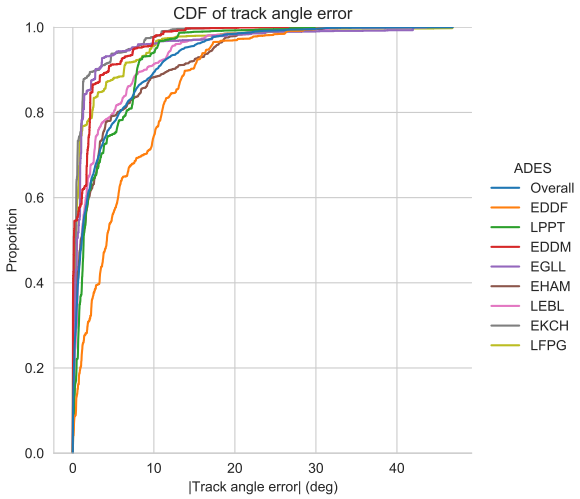


Figure E.18: The empirical cumulative distribution function of the absolute horizontal distance error, colored by destination airport.

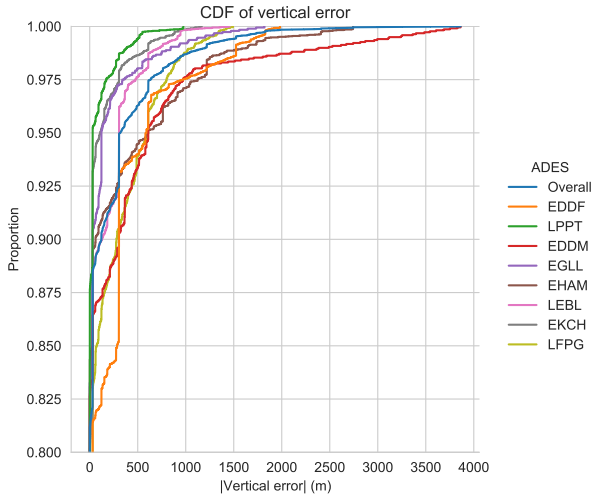


Figure E.19: The empirical cumulative distribution function of the absolute track angle error, colored by city pair.

F

Upper airspace analysis

This section presents the results when the proposed rule-based analysis is applied to several days of ADS-B in the upper airspace above the Netherlands. Both the changes in heading and changes in altitude are given in fig. F.1 and fig. F.2 respectively.

For the heading a noticeable line (PERDI to TOLEN) can be seen crossing from France into the Netherlands. Also clearly highlighted is the boundary between the Dutch FIR and the English FIR at LAMSO and RAVLO. Another highlighted square above the North sea contains REFSO, a point where a lot of aircraft turn towards EGLL.

The changes in altitude are less common. A noticeable feature is the birght colored area around FERDI, which overlap a bit with heading change map. The rest is much more spread out over the entire airspace.

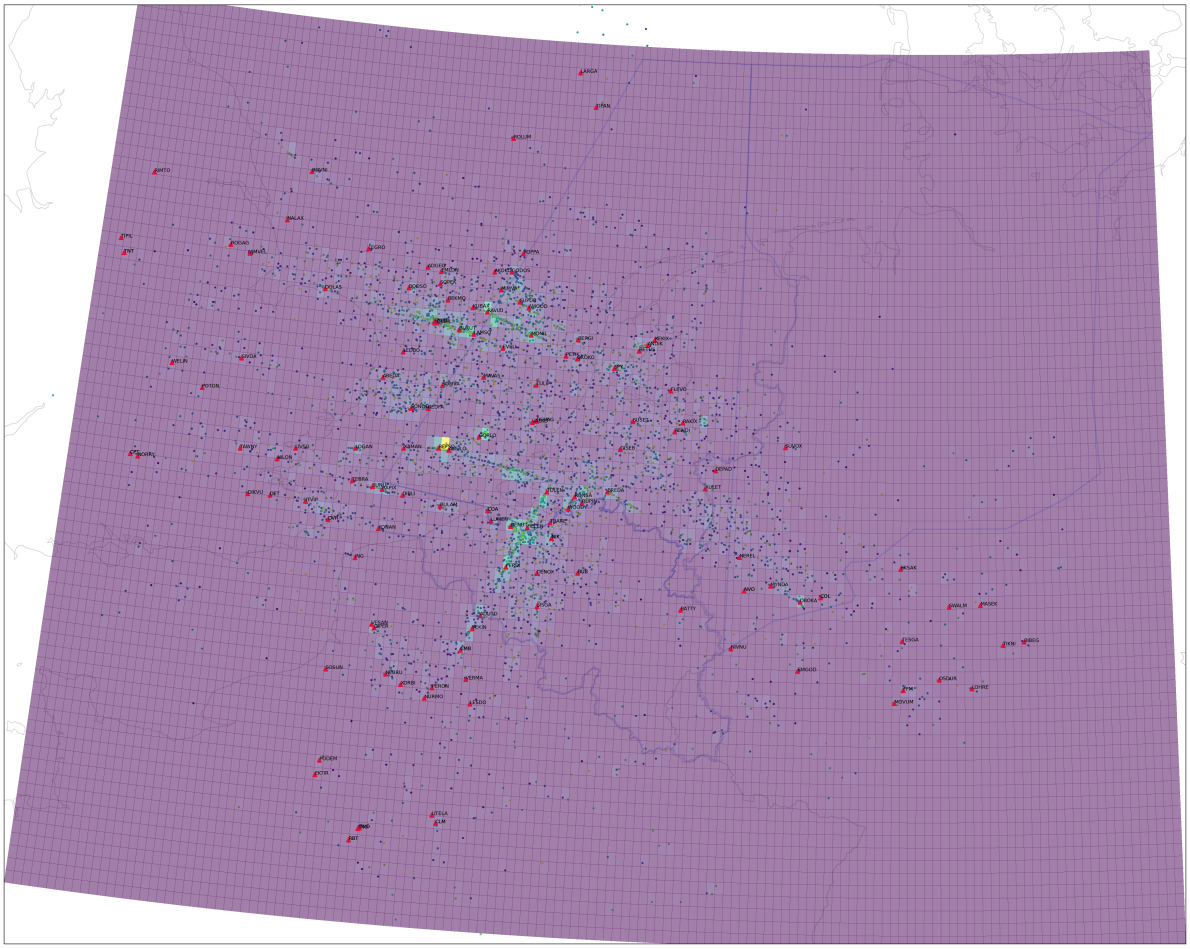


Figure F.1: Detected heading changes in the upper airspace above the Netherlands. Every square represent an area of 0.1° by 0.1° and the color represents the amount of detected changes in the area.

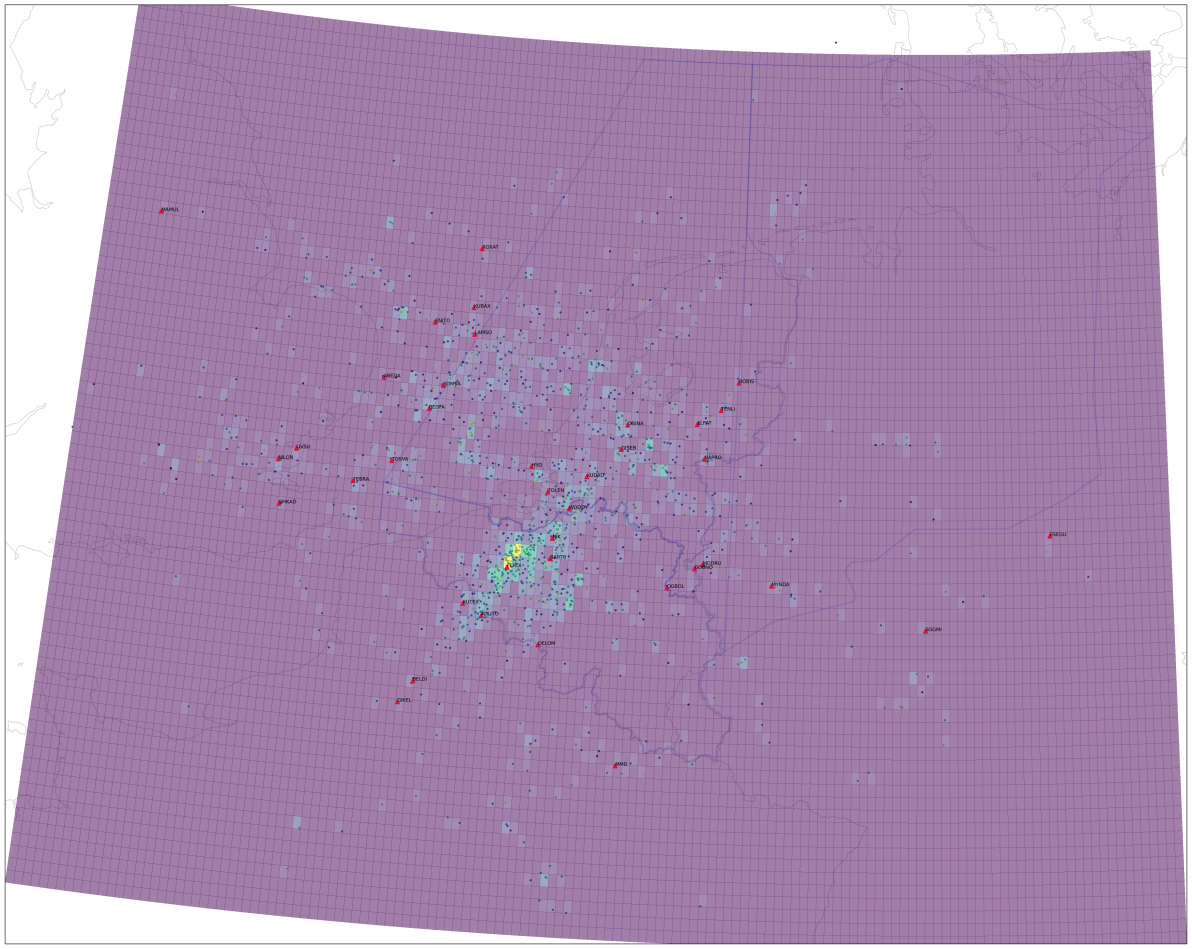


Figure F.2: Detected altitude changes in the upper airspace above the Netherlands. Every square represent an area of 0.1° by 0.1° and the color represents the amount of detected changes in the area.

III

Preliminary Report [Already graded]

1 Introduction

The last decades have seen a steady increase of air traffic and the rise of superior technologies. The ability to process large amounts of data has enabled the field of AI to become a dinner-table topic. The aviation sector embraced this increase as well and started to collect more data to improve the safety and efficiency of the sector. A major step was the introduction of ADS-B (Automatic Dependent Surveillance-Broadcast.

ADS-B data contains the location, altitude and the id of the aircraft and can be received by anyone with an ADS-B antenna. Previously, this kind of data was sent every 5 to seconds using the mode-c and mode-s transponders. With the introduction of ADS-B data, the temporal resolution has decreased to every second[6]. This data transmission has made air traffic control more precise. It also enables separation advice systems such as ACAS[1]. The increased accuracy also improved the capabilities of, among others, statistical data analysis and machine learning techniques.

One research gap that is currently getting more attention, is the detection of anomalous events in past ADS-B data. Clustering algorithms and machine learning techniques have been used to detect events such as firefighting, to go-arounds. However, as of yet, no method has been suggested that can identify basic air traffic control (ATC) commands in the past ADS-B data. Since aircraft leave this 'breadcrumb' trail, a method could be developed to detect where and what ATC commands have been given.

This report presents an exploratory research project that uses the ADS-B data to determine where, when and possibly what ATC command is given. Several approaches have been identified, including statistical time series analyses, airway usage and big data techniques. When ATC commands are identified they are compared against historical, waypoint and flight path data to predict whether such a change was an ATC command or a procedural command. This report suggests a validation strategy to determine how accurate and robust the detection method is.

Identifying commands in the airspace might be helpful in different situations. Finding ATC commands can help identify the workload or habits of the ATC controller, the complexity of the airspace, or it can provide a test and validation set for other machine learning algorithms.
Looking further into the future. A method might be developed that 'strips' these ATC commands from the ADS-B data. This could simulate an airspace where no ATC is/was present. This simulation could be used to evaluate automates separation systems or to train ATC students.

1.1. Scope

The subject of the thesis will be to investigate whether it is possible to detect ATCo actions from ADS-B data that are being sent by the aircraft.

The scope of the research will encompass the en-route flight phase of an aircraft in the Maastricht Upper Area Control (MUAC) airspace. This is due to the ADS-B data available, more on this in section 2.2. Only commercially scheduled flights will be used for this research. Furthermore, the ATCo actions that are searched for are

[1] https://www.skybrary.aero/index.php/Airborne_Collision_Avoidance_System_(ACAS)

heading changes, direct to, speed and altitude changes.

1.2. Research questions

The main research question of this thesis is:

> To what extend is it possible to identify ATCo actions from past flight data and to simulate aircraft trajectories when these commands are filtered out?

This main research question can be subdivided into multiple subquestions. Each of these questions needs to be answered for this thesis to make any significant contributions to the current literature regarding ATC.

1. Which detection method can best identify changes in ADS-B trajectory data?
2. To what level can these changes be coupled to existing ATC procedures?
3. How can the difference between an ATC command and a procedural change be identified?
4. How accurate are these found ATC commands?

1.3. Outcome & contribution

The outcome of this research project should be a method that produces a scenario where flights are passing through the MUAC airspace on their current trajectory, without any ATC intervention during the flight. This method can then be applied to any upper area control airspace to produce the same results. This scenario would allow for automatic separation systems to be tested on modified real data instead of synthetic data. The results themselves can say something about the complexity of an airspace of reveal habits in ATC.

1.4. Report structure

The section will describe the structure of the report. In chapter 2 relevant literature, current practises and additional background information is presented. Chapter 3 will describe what preprocessing steps were done to ready the data for the actual analysis. The experimental set-up is discussed in chapter 4. Here, the experimental directions are discussed and preliminary finding regarding the directions. It will also discuss the validation strategies for this research. In chapter 5 the results are presented that have been found so far. These are thus far only present for the rule-based analysis. Chapter 6 will present the next steps that will be taken after this report and the overall planning of the research. This report will be concluded in chapter 7.

2

Literature review

In this chapter, relevant literature and background information will be presented. In will start with the current practises in ATC and how changes to aircraft are relayed. Next, a more in-depth look will be taken into the main data source this thesis is going to use. Section 2.2 will explain what ADS-B data is and what currently can be achieved with it. Next, section 2.3 will describe how these ATC practises can be translated into rule-based detection methods by using time-series analyses. Section 2.4 will take a look into the current clustering techniques used to group different flights with a similar flight profile. It also discusses some of the preprocessing steps needed to use these clustering algorithms efficiently. In section 2.5 two papers have explained that automatically transcribe ATC radio commands use machine learning models. In the next section, section 2.6, several methods are described, which can detect anomalous trajectories based on an ensemble of trajectories. The last section will detail all the data and data sources, what are the sources, and where these come from. This is section 2.8.

2.1. ATC practises

When making these rules, the current ATC practices have to be taken into consideration. These practices differ for different commands to the aircraft. This thesis will stick to the basic commands and the practices that surround these commands.

First, heading changes. Heading changes are most commonly given in two different ways. Aircraft can be given the 'direct to' command. This command tells the captain of an aircraft to fly directly to a waypoint. This is usually to cut off a section of the planned trajectory that is a longer way around. The next command is the 'fly heading xxx', where x is a heading between 0 and 360. This tells the aircraft to fly that exact heading. In this thesis, the assumption will be made that these two commands are the only heading commands given. It will also be assumed that the 'direct to' command will most likely be given in the current scope of the thesis.

Next are the speed changes. Speed changes can be only be given in two ways (in the current airspace). These are either 'at pilots discretion', meaning that the pilot is able to fly at the desired speed. Or the speed is given in 'reduce/increase speed 250 knots'. This tells the pilot to adjust their speed to the specified velocity. It can be assumed that aircraft in MUAC airspace will fly at the most efficient speed (in relation to the wind) and that speed change commands are uncommon.

Lastly, the altitude change commands. Altitude commands are most commonly given in the 'climb/descent to FL200' format. The height in feet is the flight level (FL) x 100 (FL180 = 18000 foot). Flight level commands are always given as multiples of 10 (FL160, FL170, or FL 180). Since aircraft are always assigned to a flight level in order to maintain vertical separation, it can be assumed that any change in FL id due to the ATC.

2.2. ADS-B data

ADS-B is short for Automatic Surveillance Dependent Surveillance-Broadcast. It is a method of automatically transmitting data from the aircraft to nearby receiver stations.[7] The message is send encoded and can be encoded to extract the actual data. This is described by Sun et al.[8]. An addition to this is the pyMode-S,

Table 2.1: Parameters received by the TU Delft ADS-B antenna

Parameter	Unit	Accuracy	Example	Remark
ICAO24	-	-	406AE9	Hexadecimal aircraft identifier.
Unix timestamp	s	10 ms	1527897600.29	
Latitude	deg	1 mdeg	51.61093	
Longtude	deg	1 mdeg	5.73479	
Altitude	ft	25 ft	28000	
Ground speed	kts	0.01 kts	417.0	Send with extended message.
Track	deg	290	0.01 def	Send with extended message.
Rate of climb	ft/s	64 ft/s	128	Send with extended message.
Callsign	-	-	TOM23X	Send as extra value, not always included in extended message.
Flight id	-	-	cb59a5b4	Identifier for flights. Not used in this thesis because of unreliability.

which is also developed by sun et al.[9]. This is a python package that can decode captured ADS-B messages.

An entire network of ADS-B is set up by the Opensky Network[10]. This network provides open access to ADS-B and mode-s data to increase the security, reliability, and efficiency of air space usage.

For this research, a dataset was provided by the Delft University of Technology. This dataset contained decoded messages collected by an antenna atop the faculty. The information contained in the message can be found in table 2.1.

ADS-B data has been used extensively in research, because of its open nature. Several of these applications are given below.

2.2.1. Flight extraction

ADS-B data does not differentiate between different flights. It always sends the identifier of the aircraft and sometimes the callsign of the flight the aircraft is performing. However, this callsign cannot be used as an identifier for several reasons. 1) It is too sparsely available to use it directly, 2) the callsign of the aircraft can change, both due to technicalities or for other reasons. An example is when KLM changed the callsign of their 747-400 aircraft for retirement[1].

Sun et al.[11] proposed a method to use clustering algorithms to extract flights from the dataset. Sun et al. tested BIRCH and DBSCAN as clustering algorithms to separate data points based on the ICAO address and the timestamp. The DBSCAN method has been further refined by Sun et al. [12] to better tune the clustering parameters. A benchmark test was performed where the parameters of DBSCAN were varied. The clustering algorithm was applied to a dataset with the known amount of flights, in order to extract the best set of parameters.

2.2.2. Flight phase identification

ADS-B can be used to identify the flight phase of an aircraft. Sun et al.[**sun**] proposed a method using fuzzy logic to extract this data based on the altitude, RoC, and speed. This method suggests a percentage for each possible type of phase (ground, climb, descent, cruise). The type of phase with the highest score is selected for this data point.

This method is later adjusted and refined by Sun et al.[12] to also include a 'level' flight phase. Sun et al. also added extra logic that only allows for valid phase transitions. An aircraft cannot go from 'cruise' to 'ground', but has to pass 'descent' in order to get to 'ground'.

2.2.3. Aircraft performance modeling

ADS-B data can be used to estimate the performance metrics of aircraft. In his paper, Sun uses a stochastic total energy model to estimate the drag polar coefficients[13]. Hrastovec et al. used data collected by air traffic control centers to estimate aircraft performance using a machine learning model[14]. This has since

[1] https://simpleflying.com/klm-final-boeing-747-retirement/

been evolved into an open-sourced aircraft performance model that can be used by anyone in the scientific community[15].

2.3. Rule-based time serie analysis

Rule-based detection tries to determine a set of rules that can provide a basis to detect changes in trajectories. These rules are based upon the premise that trajectories are a form of time series and the changes to these time series are based on ATC practices, as described in section 2.1. The hypothesis is that a set of rules can be applied to these time series that can predict whether a change in the trajectory has happened and if it was an ATCo command or not.

Three methods have been selected to detect changes in time series. The first is using the ruptures python package [16]. This package contains multiple change point detection algorithms, which can be imported into Python. Changepoint detection focuses on detecting changes in the parameters that make up a time series. The second method is using the k-sigma rule. When applying this method to the change in direction of the aircraft, the outliers can suggest a change in aircraft heading. The third is taken from Gariel, Srivastava, and Feron [2]. In this paper, a method is mentioned for detecting turning points based on the turn rate of aircraft.

When comparing the results of the detection step with information about waypoints and historical data from the same callsign (or airport pair and carrier) an approximation can be made if a certain detection was a procedural one or this was an ATCo action.

Altitude change analysis

In addition to the time series rule-based method described above, another approach was taken for the change in altitude. This approach used the information gained from the flight phase, as described in section 2.2.2 and section 3.3. Since flight phases have already been identified in trajectories, transitions could indicate the presence of an ATC command. This has to be verified with the Eurocontrol flight path to check whether the change was planned ahead or not.

2.4. Trajectory clustering

Clustering is an unsupervised technique that can group together data points. When a group of data points is given, clustering algorithms can be used to classify data points into certain groups. This can also be applied to trajectories, although not directly. In order to cluster trajectories, they have to be altered. The considered clustering techniques and trajectory processing methods are described below.

2.4.1. Clustering algorithms

In order to obtain quantitative data from aircraft trajectories, trajectories can be grouped together in sets of trajectories that are alike. These groups are called flows. This allows comparisons between different flows or to compare single trajectories against the flow mean. Aircraft can be assigned to flows based on their current flight path and, in conjunction with other environmental data, a short-term 4D prediction can be made[17] or an ETA for landing can be estimated[18].
A review of clustering algorithms and trajectory similarity/distance measures is given by Yuan et al. [19]. In this paper, seven clustering algorithms were described and reviewed in addition to six trajectory similarity/distance metrics. Another review paper by Basora, Olive, and Dubot [20] summarizes clustering algorithms with respect to their abilities to detect anomalies.

Three clustering methods for trajectory clustering were considered. Two of these use clustering methods, DBSCAN & spectral clustering and the third uses information gained from the Eurocontrol data set.

DBSCAN

The first clustering method considered is DBSCAN. DBSCAN (Density-based spatial clustering of applications with noise) was first proposed by Ester et al. [21]. DBSCAN clusters points together based on the euclidean distance between points and the number of points within that distance. An example is shown in fig. 2.1, here three different points are identified.

1. Core points. A point is considered a core point if at least MinPts are within the reachable distance ϵ. Shown as points around A.

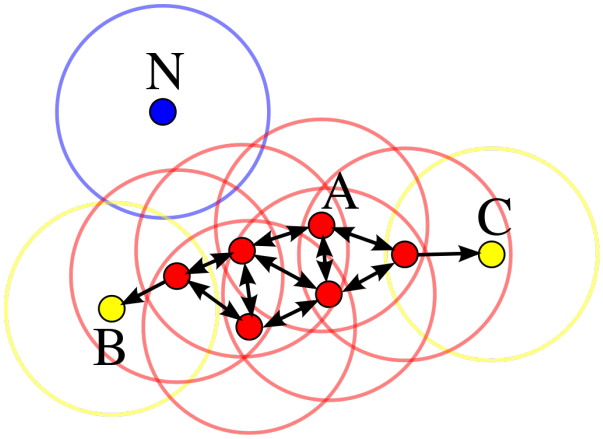


Figure 2.1: Visualisation of DBSCAN, with minPts = 3. Points around A are core points. Point B and C are not core points, but reachable from A. Point N is noise. source: [2]

2. Non-core points. Non-core points are points reachable by at least 1 core point. Non-core points are considered part of the cluster. These are point B & C in fig. 2.1.

3. Noise. If a point can not be directly reached by a core point it is considered as noise, as shown by point N.

DBSCAN is controlled by two parameters. The first is minPts. This is the minimum amount of points that need to be reachable by a point, for it to be a core point. The second is ϵ, this is the distance that is reachable from a core point. Shown by the circles in the figure.

The benefits of DBSCAN are numerous. It is used in literature as shown Murça et al. [1] and Gariel, Srivastava, and Feron [2]. It is also easily implemented using the Scikit-learn python package[22] and the understanding of the parameters is intuitive.
The downside is that trajectory data cannot be directly inputted. Therefore a dimensionality reduction such PCA has to be applied to the trajectory data before DBSCAN can be applied.

Spectral clustering
The second clustering algorithm was spectral clustering. It is used by Enriquez [3] to determine persistent flows. It also is available as a clustering algorithm from Scikit-learn[22]. This however cannot take trajectory data as input. Therefore the architecture as described by Enriquez was used. The spectral clustering method presented uses the values of the second-largest eigenvector of the Laplacian matrix L of $W_{i,j}$ to partition the different trajectories into clusters. Here, $W_{i,j}$ is a reduced representation of all the trajectories. By recursively calling the spectral cluster method on the results, until a stopping criterion is met, it divides the input into more and more clusters.
How to construct this similarity matrix is presented in the latter part of section 2.4.2. The matrix L is constructed by subtracting the diagonal matrix $W_{i,j}$ from D, with $D = \sum_{j=1}^{n} W_{i,j}$. In pseudo-code this look like the folling section:

```
def SpectralCluster(W, wmin):
    D = sum(Wij, axis=i) #sum W over i
    L = D - W
    v = second smallest eigenvector of L
    il = np.where([v < 0]) #Get indices where v < 0
    ir = np.where([v >= 0])
    if stop( W[il, il] ) > wmin:
        SpectralCluster(W[il, il], wmin)
    else:
        save il #found a cluster
    if stop( W[ir, ir] ) > wmin:
```

[2] https://commons.wikimedia.org/wiki/File:DBSCAN-Illustration.svg

```
        SpectralCluster(W[ir, ir], wmin)
    else:
        save ir #found a cluster
```

This results in an array where the indices correspond to i in T_i and the value corresponds to the cluster. The stop function can be a user-defined function. Enriquez used the ratio of maximum distances $\frac{\max(Wi.,i.)}{\max(W)}$ to determine when the algorithm should stop.

The benefit of spectral clustering is that no dimensionality reduction or feature extraction has to be performed in order to cluster flights together. Only subsampling of the trajectories.
The disadvantages of spectral clustering are that no flight can be identified as noise. Trajectories also have to be subsampled into an even amount of points in order to make a similarity matrix. The method is also slower than DBSCAN and the parameters are less intuitive.

Filtering on Eurocontrol metadata
The last clustering method considered is using the Eurocontrol data given to filter the flights. This dataset is described better in section 2.8.2. The clustering methods can be done by filtering on other data, such as airports, cities, operators, or aircraft types.

2.4.2. Dimensionality reduction & similarity parameters

Several dimensionality reduction techniques and similarity parameters are found in the literature. Both Basora, Olive, and Dubot [20] and Yuan et al. [19] have presented review papers in which these are summarized and compared. Based on their information and the current practices in other literature these conclusions were made.

Principal component analysis - DBSCAN
For the clustering using DBSCAN, principal component analysis (PCA) was chosen as reduction technique. This method is used by [2, 17] in conjunction with DBSCAN. PCA can be considered as a feature extraction method. Its goal is to identify the most relevant information and map this onto a feature space. This feature space is a lower-dimensional space than the original data. Principal components (PC) are orthogonal axes and depicts along which axes the original data has the most variance. The first PC shows the data with the most variance, the second PC (orthogonal to the first) shows where the data has the second most variance, etc... An example here is shown in fig. 2.2. The top-ranking PC will be used for the clustering algorithm.
To be more precise, the method of Wang, Liang, and Delahaye [17] was used. It is suggested to augment the trajectory data with several other parameters, such as the distance from a point to the center of the map R, the corner of the area considered D, and the angular position to the reference point Φ. Also, the sine and cosine values of the heading were included to avoid the discontinuity at 0 and π. Next, all the data is normalized to $[0,1]$ to scale everything. In the final step, the trajectories are resampled to n point. Appending all the data gives a vector representation T of a trajectory.

$$T = \left[lat^*, lon^*, alt^*, R^*, D^*, sin()^*, cos()^*\right]$$

Every parameter now consist of n normalised points. And every row consist of $n \cdot 7$ points. This is done for every trajectory and the PCA is applied on the resulting data set.

Similarity matrix - spectral clustering
For the clustering using spectral analysis, the proposed method of using similarity matrix, constructed using a Gaussian kernel, as proposed in the same paper by Enriquez [3] was used.
In the paper, the trajectory is also represented in a singular vector form T, but only the latitude and longitude parameters are used. The latitude and longitude data is normalised to $[0,1]$, creating x, y. The trajectory is then uniformly subsampled at m times. This creates a vector in the following format.

$$T_i = \left[(x_1, y_1)^{(i)}, (x_2, y_2)^{(i)}, \ldots, (x_m, y_m)^{(i)},\right]$$

Here, $(x_k, y_k)^{(i)}$, correspond to the normalised position of the aircraft i at time k.
A similarity matrix $W \in \mathbb{R}^{(n \times n)}$ is constructed, where the entries are calculated using the Gaussian kernel

[3] http://www.statistixl.com/features/principal-components/

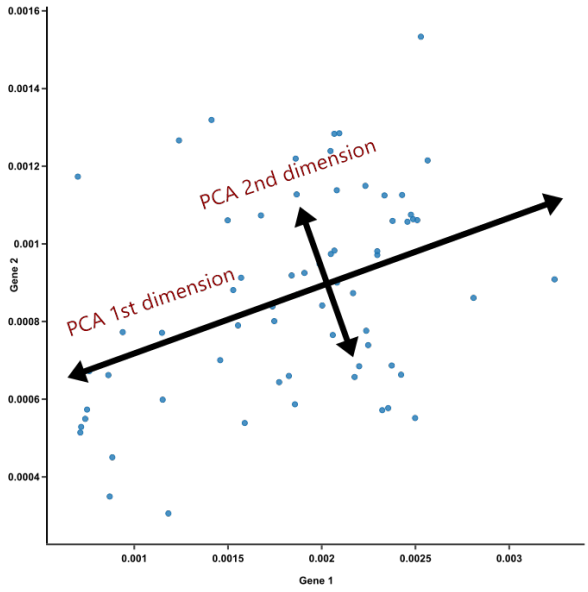


Figure 2.2: Visualisation of PCA. PCA finds the axis along which the data has the most variance. Source: [3]

below.

$$W_{i,j} = e^{-\frac{||T_i - T_j||^2}{2\sigma^2}}$$

Notice that there is a user-defined parameter, σ. This is a local scaling parameter that determines the width of each cluster. Once the similarity matrix has been made, it can be used as the input for spectral clustering.

2.5. Automated ATC voice transcriptions

Research has been conducted to use voice-recognition software to automatically transcribe ATC voice commands into machine-readable data. This would allow for the automatic detection of ATCo actions by listening to ATC radio. This can either be done using a dedicated radio or via other open sources[4].

Srinivasamurthy et al.[23] proposed an iterative model that uses ATC systems to build up an increasing database of ATC commands that can be used to train their model. Their system used a semi-supervised learning framework that uses speech and radar data to iteratively update the model.

Subramanian et al.[24] proposed a method to detect anomalies in the National Airspace System by combining ATV voice command and Traffic Flow Management data and IBM Watson's natural language processing capability.

ATC voice commands could be used as a validation set when detecting ATCo commands from ADS-B data. It can also be used as a training set when using supervised machine learning methods to further detect changes in the flight path of aircraft.

2.6. Detecting anomalies in trajectories

A more generalized concept of this thesis has already been worked on. Three papers have been found, in which different methods are used to detect anomalies in past ADS-B data. Also, a review paper has been found which state-of-the-art detection methods are summarized.

In the first paper, Olive et al. [25] uses a variational autoencoder (VAE) to detect anomalous aircraft trajectories. VEA is a type of neural network that reduces the dimensionality of the given data and tries to restore the reduced data to the original data[26]. The model reduces the data to a set of distributions in order to make the predictions more robust, hence the variational part. The difference between the reconstructed data and the original data is called the reconstruction error.

[4] Examples: www.liveatc.net, www.broadcastify.com

To start, a cluster of flights or an area has to be selected. These flights are then preprocessed and fed into the VAE to train the model. Once the network has been sufficiently trained, the flight can be tested using the network. The value of the reconstruction error represents how different the flight is from the 'normal'.

The second paper, Olive et al. [27] uses a variety of data, including ADS-B data, to identify a series of events. These events include, but are not limited to: Take-off and landing, go-arounds, firefighting missions, test flights, and important for this thesis, direct to's.
Olive et al. use flight plan data to infer where the aircraft might have received 'direct to' command, by comparing the difference in true track and aircraft bearing with respect to the waypoint. A long interval of time without a navigational point may suggest an ATC command. Also, some pitfalls have been identified. These are: 1) long-haul flights might target far away waypoints, making the criteria less reliable. 2) If two (or more) waypoints are aligned, it is difficult to know which point was detected. And 3), flight management systems are able to follow parallel routes during transatlantic flights.

The last paper, written by Basora, Olive, and Dubot [20] reviews classical and state-of-the-art anomaly detection techniques and their (if any) application to aviation. The paper includes techniques such as traditional data-driven methods, neural networks, and temporal-logic based learning. Extra focus is put on unsupervised techniques, due to their relevance in the aviation domain. The paper concludes by summarizing the anomaly detection methods and their use in the aviation domain.

2.7. Trajectory prediction

An area of interest in the ATM research field is to predict the locations of aircraft in the future. A prediction can better aid in separating aircraft or indicate to ATC when an aircraft is touching down.

Several studies have already been conducted in this field.
Wang, Liang, and Delahaye [17] proposed a short-term 4D to predict the ETA of aircraft landing at Beijing airport. The method used a neural network (NN) that is trained independently on different clusters. It was found that the NN was robust and outperformed multiple linear regression models, which were used as a baseline.

In the follow-up paper, Wang, Liang, and Delahaye [28] 5 different neural network architectures were evaluated. All of these were deep neural networks and only differ in the number of layers. It concludes that deep NN's outperform shallow NN's, but should not be too deep. The latter won't be able to generalize as well.

Dek [18] uses ADS-B data, weather data, and runway availability to predict the arrival times of aircraft landing at Amsterdam. He compares aircraft performance models, long short-term memory neural networks, deep neural networks, and gradient boosting machines to predict the ETA of aircraft at Amsterdam. It was found that the latter outperforms the other tested methods.

An entirely different method of predicting aircraft arrival times is using the Bluesky air traffic simulator by Hoekstra and Ellerbroek [29]. Bluesky is equipped with aircraft performance models, and also compatible with BADA 3. These contain aircraft performance parameters and allow Bluesky to better simulate the behavior of aircraft.

2.8. Data & data sources

The data used for this experiment comes from different sources en contains multiple different datasets. This section will explain what data is used and isn't used, where it comes from and what the data is used for.

There are 3 main databases used in this thesis. The first one is the ADS-B database from the faculty of Aerospace engineering at the TU Delft. The second database is the dataset from Eurocontrol. The third dataset is included in the Traffic python library.

2.8.1. TU Delft ADS-B data

ADS-B data (or Automatic Dependent Surveillance-Broadcast) is a message that is sent by every aircraft. The Delft University of Technology has an antenna with which it can receive ADS-B messages. These messages are stored on a server, which can be accessed when one has a TU Delft log in.

Due to the location of the receiver, it can receive most of the ADS-B signals that are given in MUAC airspace, and over the south-eastern part of the United Kingdom. A visualization is shown in fig. 2.3.

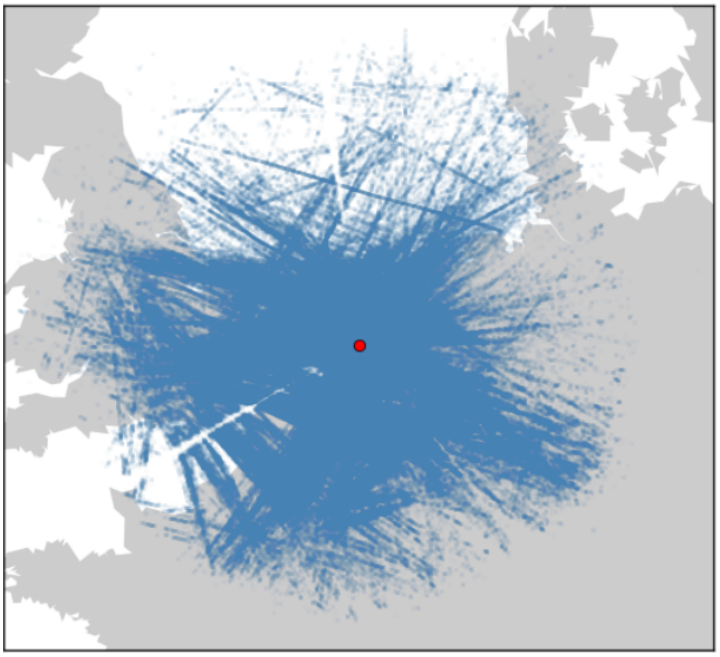


Figure 2.3: ADS-B datapoints captured over 1 day. Source:[11]

Every message contains the ICAO24 address of the aircraft, which is unique to the aircraft. The UNIX timestamp of the message, the position of the aircraft in latitude and longitude, and the altitude of the aircraft. Sometimes more information is sent which contains the ROC (Rate of Climb), the ground speed, the track angle, the flight id, and the callsign.

2.8.2. Eurocontrol flight data

The Eurocontrol dataset contains information about all commercial flights that enter or use Eurocontrol airspace. Eurocontrol has released the flight data for the years of 2015 to 2018 with 4 fixed sample months (March, June, September, December) for R&D purposes. Only for the year 2018 the month of June was available to the TU Delft.

The data consist of 4 main datasets. These are the 'flights', 'flight points', 'flight airspaces' and 'ATM environment data'. Only the first two of these are used or planned to be used (as of the time of writing) in this research.
The first dataset contains information about the flights. An overview of several parameters can be seen in table 2.2. The most important parameter here is the ECTL_ID. This parameter will be used to identify flight. An overview of the 'flight points' data is given in table 2.3. These values are available for both the filed flight points as for the actual flown flight points.
The third dataset, 'flight airspaces', contains the entry and exit times of AC for different FIR's (flight information region) and AUA (ATC unit airspace). The last dataset contains environmental data, which refers to the structure of the airspace. It contains data about the AIRAC's (aeronautical information regulation and control), the routes (or airways), and FIRs.

Since the ADS-B package provided to the TU Delft contains the entire year of 2018, the choice was made to use the month of June in 2018 as the period of interest.

2.8.3. OpenSky aircraft database

The OpenSky database[10] contains information about the aircraft itself. Most importantly, it contains the icao24 addresses and registration numbers of most aircraft. This provides a connection between the ADS-B data and the Eurocontrol data and allows these 2 datasets to be combined.

The dataset contains a lot more information that is not used for this thesis. It contains information about the owner of the aircraft, the aircraft type & model, and the manufacturer of the aircraft.

Table 2.2: Selection of parameters available in the Eurocontrol Flight dataset

Parameter	Description
ECTL_ID	Flight identifier
ADEP	ICAO code departure airport
ADES	ICAO code destination airport
Filed off-block Time	Time of departure (UTC) based on last filed flight plan
Filed arrival Time	Time of arrival (UTC) based on last filed flight plan
Actual off-block Time	Time of departure (UTC) based on ATFM-updated flight plan
Actual arrival Time	Time of arrival (UTC) based on ATFM-updated flight plan
AC Type	ICAO aircraft code
AC Operator	ICAO airline code
...	....

Table 2.3: Parameters available in the Eurocontrol flight points dataset

Parameter	Description
ECTL_ID	Flight identifier.
Sequence no.	Numeric value representing the order of points
Time Over	Time (UTC) of crossing point
Flight level	Flown flight level at point
Latitude	-
Longitude	-

2.8.4. Navigational aids data

Navigational aids (navaids) are any marker that aids an aircraft in traveling. These can be physical aid, such as VORs, DMEs, or even large arrows. Navaids can also be non-physical points, usually called 'waypoints' or 'fixes'. These points are defined by GPS coordinates. Another form of a navaid is an airway. Airways are routes in the sky along predefined waypoints. Four databases were found which contain navaids. Two of these are digital packages and can be easily accessed through Python, while the third is a digital document and cannot be accessed as easily. The final sources of data are the aeronautical maps published by Eurocontrol.

Traffic library

The traffic library[30] is a package that can be imported into the python coding language. The package comes with a database that contains data about airports, airways, waypoints, and FIRs. All of these can be readily accessed as classes in Python or the raw data can be accessed otherwise.

Bluesky

BlueSky[29] is an open-source Python-based air traffic simulator. It can visualize, analyze and simulate air traffic. It is a self-contained program that can be modified by adding plugins to customize the simulator to the needs of the user. It contains a database of waypoints, airways, and airports that can be used to better analyze air traffic data.

Electronic aeronautical information package (eAIP)

The electronic aeronautical information package (eAIP) contains information about the waypoints and airways for every country. It is a standardized form of keeping the information about everything in the airspace of the country. A reference page can be found on the website of Eurocontrol[31]. It contains the information of all waypoints, airways, airports, fir, communication lines and much more. The only information that is used from this is the waypoints database. Since it can't be accessed in bulk, every waypoint has to be looked up manually.

Eurocontrol cartography charts

Eurocontrol provides maps for the airspaces [5] it regulates. This map provides information about waypoints and the airways. Although it does not contain the information about the location of these digitally. This serves as a validation of the other databases.

Final combination
A combination of all these sources was compiled to provide a tailored solution for use in this research. This was done for several reasons. 1) The navaids databases did not allow for easy selection of a few, upper airspace, navaids. 2) Initial plots have shown that flights in MUAC airspace can have 'direct to' commands for waypoints in British airspace and vice versa. 3) MUAC is designated as a 'free route airspace'. A set of waypoints is given and each waypoint is designated as an exit, entry (or both), or intermediate point. Airlines can plan their route from any entry to any exit via any intermediate point. 4) In practice, turns have been found at waypoints that were not in this free route list and have therefore been added anyway.

This list of waypoints is a continual process. Waypoints have also been divided into different groups to allow easy selection for plotting. When this list (or selection) of waypoints is asked, the waypoints are plotted from one or both of the databases.

3

Data preprocessing

This chapter will present the step taken to preprocess the data. Firstly, section 3.1 will describe how flights are extracted from the raw ADS-B data using a clustering algorithm. It will also present how to detect outlier points using the same clustering mechanism and the use for these outlier points. Section 3.2 will present how the ADS-B data is coupled with the Eurocontrol dataset. It will also describe the data formatting of the ADS-B data after it has been coupled to allow filtering and searches based on the Eurocontrol dataset. Section 3.3 will describe how flight phases are determined for every extended ADS-B message. Section 3.4 will describe how the scope of the research is translated into a filter for flights and what parameters are used for this filter.

3.1. Finding unique flights

The raw ADS-B data itself contains no data specifying flights. It only gives us what aircraft is flying where at what time. Therefore a method has to be used to extract the flights from the data. This identification of unique flights is based on the work of Sun et al. [11][12] as described in section 2.2.1. A slight modification was used for the identification of flights. The Eps parameter was set at a value of 30 minutes to include more points that would otherwise be labeled as noise, while Min_Pts was not changed from the value of 50. The value of 30 minutes was chosen because the turnaround time of aircraft is 30 minutes or longer. The inclusion of these points increases the number of points that can be used later on to identify the 'heading to' given or the 'direct to <waypoint name>'.

3.1.1. Outlier detection

To further refine the data, outliers in each flight were determined. For this, the method of Sun et al.[12] was directly followed. DBSCAN was used with parameters set at 50 for $MinPTs$ and 500 for Eps. This gave each flight a set of core points and a set of outlier points. These core points will be used for further analysis, while the outlier points will only be used to provide extra context for the flight.

3.2. Combining the ADS-B data and the Eurocontrol data

To utilize the given ADS-B data fully and gain more contextual knowledge it can be coupled with the Eurocontrol dataset. To do so, the unique keys have to be identified first. The ADS-B data uses the ICAO24 address to identify aircraft. The Eurocontrol datasets use the $ECTL\_ID$ to identify flights. The Eurocontrol entries also contain the AC Registration of the aircraft performing the flight, thus identifying the unique aircraft. To combine these, the OpenSky Aircraft Database[10] is used. Figure 3.2 shows the connecting schematic of the simplified databases.

Several assumptions are made during the connecting process.

1. The data is not complete.
2. Each aircraft has one ICAO24 address & and one registration code
3. Each aircraft can have multiple flights

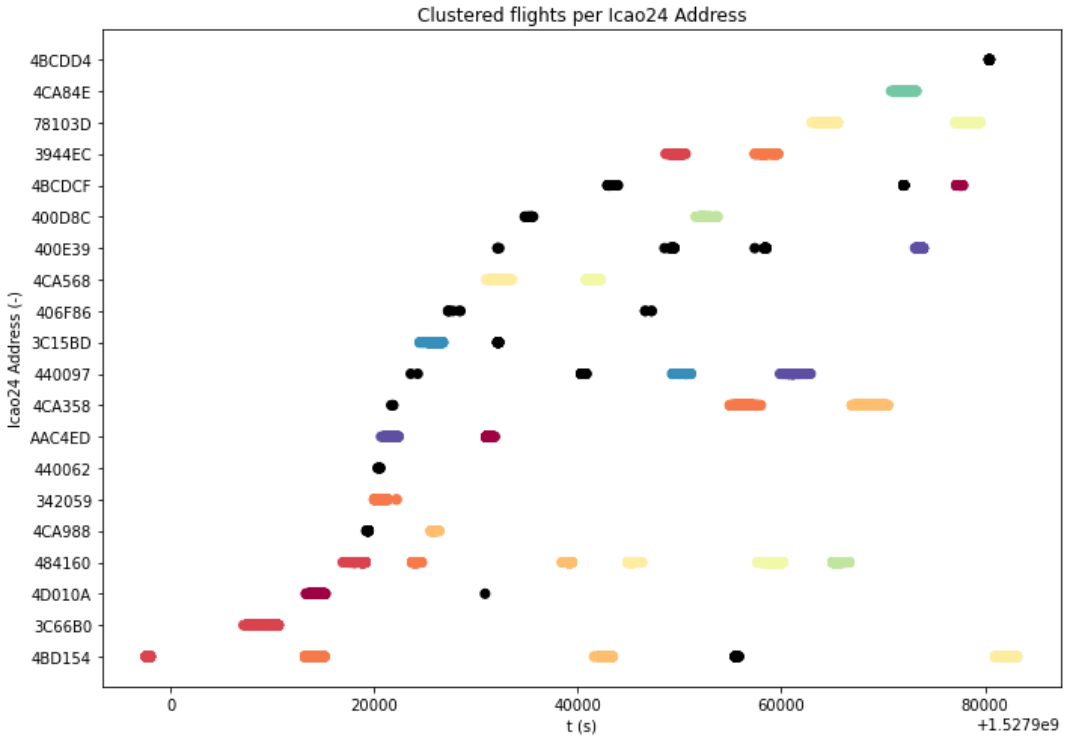


Figure 3.1: Identified flights for ICAO24 Addresses. Black dots are noise points

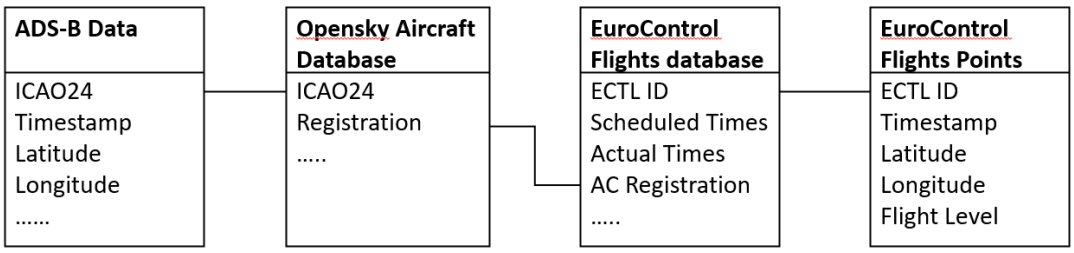


Figure 3.2: Database connections

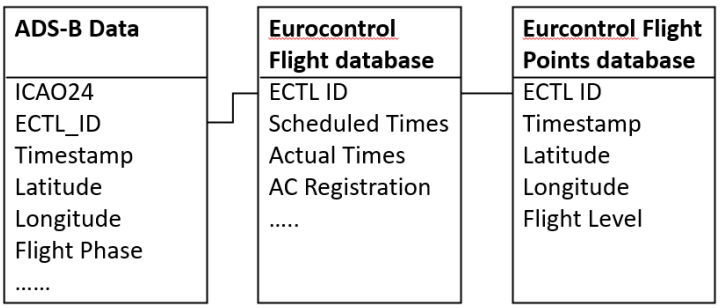


Figure 3.3: The finalized database connections

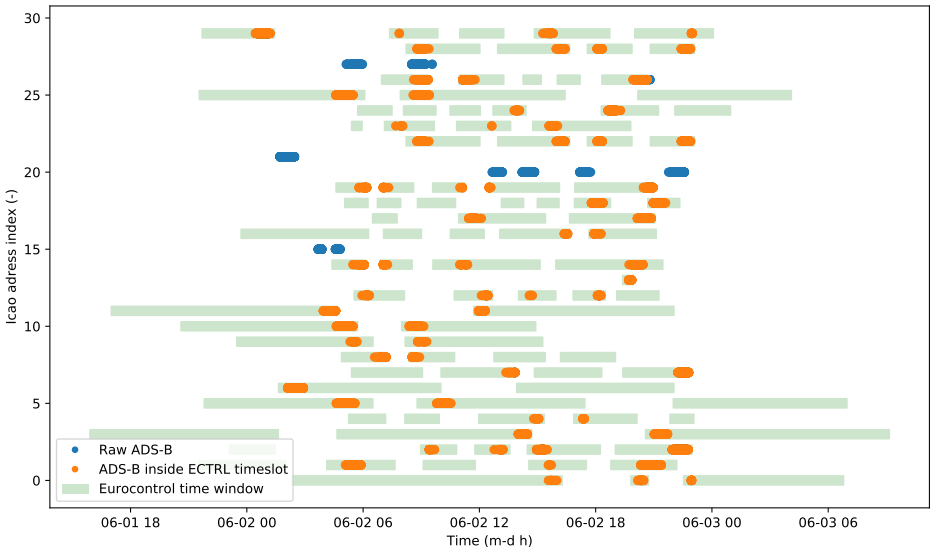


Figure 3.4: ADS-B points, Eurocontrol windows and the overlap of these two.

4. A flight cannot have more than one ICAO24 address associated with it

The effect of the first assumption is that flight identification has a chance to misidentify flights. The analysis further used flights where the ADS-B data could successfully be coupled to a $ECTL\_ID$. After the flights for a unique ICAO24 address (unique aircraft) have been found, the AC registration will be found in the aircraft database. This code will be used to select all flights from the Eurocontrol flights database, which are flown by this registration code.

Now that all identified flights from the ADS-B data and all Eurocontrol flights have been selected, they need to be coupled. This is done based on the time. The Eurocontrol flights database contains the 'Actual off-block time' and 'Actual on-block time' (actual means: including delays, postponements, or early arrivals/departures). These will be used to identify which ADS-B points are allocated to which $ECTL\_ID$.

In essence, every ADS-B timestamp will be checked if it falls in the window of time given by the Eurocontrol Flight Database. If this is true, the $ECTL\_ID$ will be added to the ADS-B entry. If not, the ADS-B will remain unaltered. To speed up the process an assumption has been made. If the first and last points of an identified flight fall within the window, all points will receive $ECTL\_ID$. This results in a database structure like fig. 3.3. Please note that the *Flight Phase* will be explained in section 3.3.

A visualization of the previous is given in fig. 3.4. A random selection of flights is shown here. Here the green windows are the Eurocontrol time windows. The blue points are the raw ADS-B points and the orange points are the ADS-B points, that are identified within a Eurocontrol window. As shown in the picture, not all identified flights are connected to a Eurocontrol time window and vice versa. The ADS-B points are gathered above the Netherlands, and therefore Eurocontrol flights that do not fly over the Netherlands don't have any ADS-B data points. Also, the missing data can be seen, as some identified flights can't be assigned to an $ECTL\_ID$.

3.3. Identifying of the flight phase

Identification of flight phases is done based on the research of Sun et al.[11]. Sun et al. use fuzzy logic to extract the flight phase of an ADS-B timestamp based on the information in the message. For this, the extended ADS-B message is used. The fuzzy logic used altitude, rate of climb, ground speed to calculate the most likely flight phase. An example is given in fig. 3.5. Once the flight phase is calculated is added to the ADS-B data as shown in fig. 3.3.

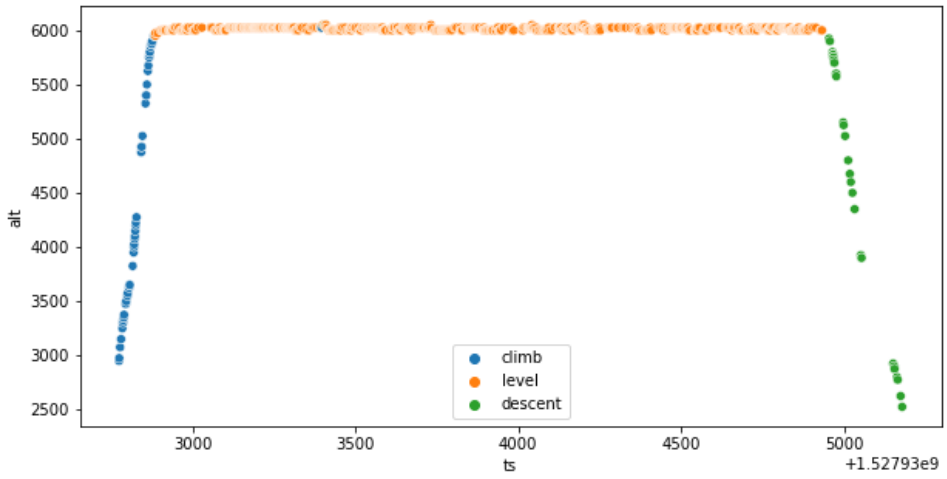


Figure 3.5: Altitude of a flight over time with different phases identified

3.4. Filtering flights

As stated before, the scope of this thesis will encompass the en-route flight phase in the MUAC airspace. Therefore the flights will have to be filtered. The MUAC airspace extends from 24,500 feet to 66,000 feet above the Netherlands, Belgium, Luxembourg, and the west part of Germany[32]. Therefore the following filters were used. The minimum altitude of a flight must be above 20,000 feet and every flight must contain over 500 data points. This is to ensure that 1) no cruise and descent stages are used, 2) the largest section of the flight is spent in the en-route phase, and 3) enough data points exist to perform analysis on.

3.5. Data analysis

This section will present some additional data analysis on the ADS-B data and other data to provide some extra context. It will take a look into the ADS-B messages received and the different types. About how much overlap all the databases have, and what the effect is of preprocessing and filtering the data.

Figure 3.6 shows the amount of data that has been collected per day. It also shows how the distribution of different kinds of messages. Of the gathered messages, 69% contained the extended messages and 10% contained the callsign of the flight.

Since multiple different data sources are used and these are not fully complete, an overview of completeness is given in fig. 3.7. This figure shows how much of a database can be found in another database.

In fig. 3.8 the amount of points is shown after the flight phase identification, the flight clustering, and Eurocontrol database coupling. More than 98.5% of all points could be allocated to a flight and around 85& of all points could be allocated to a Eurocontrol flight (or ECTL_ID). After applying the filter only 18% of all data points remained. For data points with a Eurocontrol ID, this was 20%.

Figure 3.9 presents the amount of unique values found in the ADS-B after the Eurocontrol data has been added to it. This figure shows that the amount of unique callsign is higher than the amount of detected flight, which is, in turn, more the amount of coupled Eurocontrol IDs. The filtering also has consequences on the number of unique values found in the data. This is reduced to between 11% to 14% for all parameters.

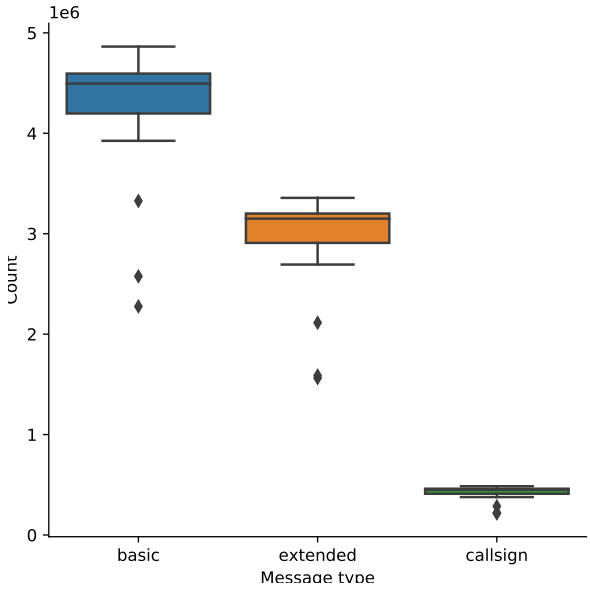


Figure 3.6: Amount of different ADS-B message received

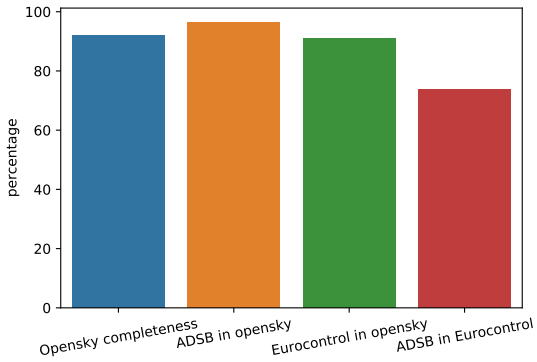


Figure 3.7: Percentages of overlap of different data sources

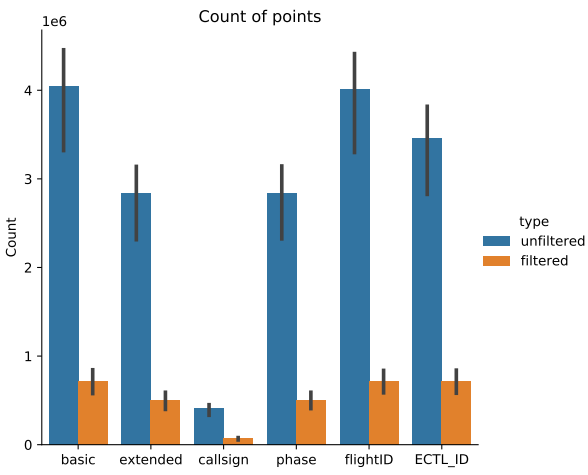


Figure 3.8: The number of points found per day

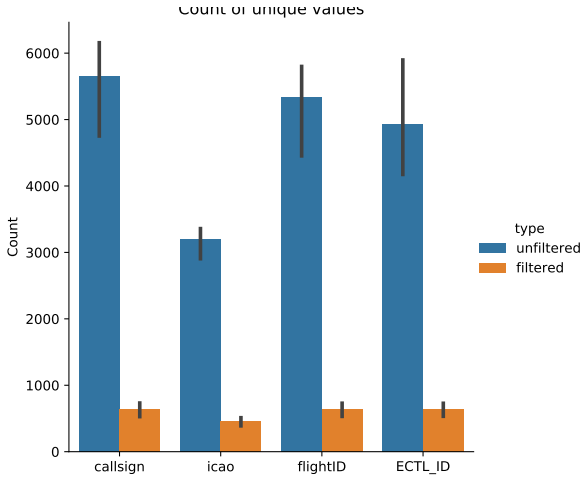


Figure 3.9: The number of unique values found per day

4

Experimental set-up

This chapter will describe the experiment and how it will be set-up. To start, section 4.1 will describe the main experimental set-up and how the research questions are translated into actionable steps. Section 4.2 will describe the main experimental directions taken and some preliminary conclusions regarding these directions. In section 4.3, two validation methods will be presented and compared. This chapter closes with the limitations of the experiment in section 4.4.

4.1. Main experiment

The main experiment proposed in this thesis will try to answer all the questions proposed in section 1.2. The main experiment is to find out if ATCo actions can be identified within past ADS-B data. This can be broken down into several actionable steps.

1. Find a detection method that can identify changes within ADS-B data.
2. Combine these changes with flight plan information to make a distinction between procedural changes and non-procedural changes.
3. Couple the non-procedural changes to existing ATC commands and procedures.

Before the experiment can be deemed a success or a failure, the results have to be validated. This is explained in depth in section 4.3, but a summary will be given. The changes will be applied to a flight plan and be simulated in Bluesky. A validation simulation will also be run using the original ADS-B data. The difference in these simulations can be calculated using similarity metrics. The resulting values can tell how similar or different the amended flight plan is to the original flight path.

4.2. Experimental directions

Several research directions are taken to find out which work and which don't. The first will be rule-based analysis. This uses set rules to determine ATCo actions. The second uses airways and the deviation of aircraft from these airways to determine if ATCo actions are given. The third concept was to determine 'company routes' from historical data and use these as airways. 'Company routes' are routes that are used quite commonly by an airline.

4.2.1. Rule-based analysis

This subsection will focus on the rule-based analysis. The goal is to try to set up a rule to detect changes in the current trajectory of the aircraft. Changes can be detected in the heading, speed, and altitude of an aircraft. Since the flight phase identification is already done, it can be used to determine the change in altitude.

Background information

Trajectory information can be seen as a time series, as it is displayed as information over time. Therefore, time series analysis methods may be used to determine changes in the trajectories. Information about these

parameters is already given in the extended ADS-B data, but can also be calculated from the standard information. To extract the best possible information, it was chosen to apply the smoothing function on the first source of data, namely the positional data or the given value in the extended data.

Not every parameter is the same and should be treated the same. Both heading and speed are continuous values, as they can be any value in a certain range. This range is (0,360) for the heading. For the speed, this value is dependent on the aircraft, the carrier, and the wind. This makes setting a definitive range undesirable. Altitude is a different parameter, as it is given in discrete values as mentioned in section 2.1.

The information received is real data and therefore subject to noise. Noise can be implemented into the system in many different ways. To reduce the noise level, a time series can be smoothed. The underlying assumption is that noise is a fast signal and that looking at a longer time-scale will reduce the effect of the noise. Therefore, smoothing functions are presented that have the property of a low-pass filter.

Smoothing functions
Six different smoothing functions are tried and tested to determine which smoothing function works best. Every smoothing function also had 3 parameters which are tested in the smoothing function itself. All of these functions exhibit the property that they act as low-pass filters. Two distinctions can be made. The first three smoothing functions keep the length of the data constant, while the latter three decreases the number of points when smoothing is applied. The second distinction is whether the time is taken into account. The first four functions don't consider time and look only at the points. The latter two resample based on the timestamp in the data. More detailed explanations are given below.

1. Exponential smoothing: $\alpha = 0.2, 0.4, 0,7$
   Exponential smoothing is a low pass filter that cancels out the higher frequency perturbation by using a formula that recalculates the next value, based on the old. The formula is: $S_t = \alpha x(t) + x_{t-1}(1-\alpha)$

2. Rolling mean: $n = 5, 10, 20$
   The rolling mean methods calculates the average value of a window with size n and puts it back in place. For every point it takes the next $n/2$ points and previous $n/2$ (rounded up), calculates the mean and places it back. It can be represented by the following equation. This method is included in the pandas rolling function. [1]
   $$x_t = \frac{1}{n} \sum_{i=-n/2}^{n/2} x_{t+i}$$

3. Rolling Gaussian: $n = 5, 10, 20$, $\sigma = 1$
   The rolling Gaussian methods is essentially a weighted rolling mean. It uses a Gaussian window instead of a uniform one to smooth the time serie. Every point is multiplied by a value based on the Gaussian distribution. This method is included in the pandas rolling function.[1]
   $$x_t = \frac{1}{n} \sum_{i=-n/2}^{n/2} x_{t+i} \cdot e^{\frac{1}{2}(\frac{i}{\sigma})^2}$$

4. Subsampling: $n = 5, 10, 20$
   Subsampling takes a sample every n steps and creates a new time serie from the original data. The new serie is smaller in size, since not every sample is used. This function smooths the data, since higher frequency disturbances are lost.

5. Time-based resampling first: $t = 5s, 10s, 20s$
   Time based resampling uses the timestamp parameter in the ADS-B data to resample the data. This is done using the pandas resample function[2]. A new column is generated based on the timestamp. Pandas then group all value that fall within window t together. The 'first' part takes the first value after t to represent the window.

6. Time-based resampling mean: $t = 5s, 10s, 20s$
   The same resampling method is used as with the 'resample first' method, but now the new value is the average of all values that are within the time window. This is also done using the pandas resample function.[2]

[1] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rolling.html
[2] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.resample.html

Detection methods
Three different detection methods were selected in section 2.3. These are described in more detail below.

1. K-Sigma rule
   The K-Simga rule is based on the statistics of the standard deviation (σ). In this method, the mean (μ) and the standard deviation are calculated from the time derivative data of the trajectory. The rule states that if a point x_t is further from the mean than $k\sigma$ a significant change is detected. Mathematically this is represented as: $(x_t - \mu) \geq k\sigma$
2. Ruptures Pelt method
   This detection method comes from the Ruptures package. The PELT package is based on the work of Killick et al.[33]. This method was chosen because it has a linear computational cost, searches for an exact solution, and requires no prior knowledge of the number of changes in the data.
3. Turn rate
   The turn rate method is used by Gariel, Srivastava, and Feron [2] to detect turning points in the San Francisco area. The method is based on a single value of the turn rate and is set experimentally. This method has been adopted and changed for this purpose. The constant value remains, but at a different level, because aircraft en-route tend to turn a lot less sharp.

Some problems were found with the ruptures package when combining it with different smoothing functions. Some smoothing functions return NaN values. The ruptures package can't handle these values and thus returns every single point as a breakpoint. Therefore it was chosen not to use the ruptures package on the smoothing functions where NaN values would be included.

4.2.2. Airway-based analysis
This direction has the hypothesis that aircraft follow mostly available airways and only deviate from these when an ATCo gives a command to change from these airways. For this, the upper airways as set out by Eurocontrol are used.

An argument why this hypothesis might not work is the free route airspace in MUAC airspace. Free route airspace means that aircraft do not have to follow specified airways, but can plan routes via 'open' waypoints. These waypoints are designated as Entry, Exit, or Intermediate points. Furthermore, when the airspace allows it, direct through flights are allowed.

A test was done to see which of the 3 online databases is the best. Seven airways were chosen as test routes. These consist of 'upper' route - indicated with start letter 'U' - and normal routes. These seven were chosen because they are long airways that cross the Dutch airspace. The airways are L602, l603, L604, l608, UL610, N872 & N873. They are shown in fig. 4.1.

As shown in fig. 4.1, these airways do not line up. Nor are they the same as in the charts issued by Eurocontrol[5]. Therefore it was chosen to not further pursue this direction.

4.2.3. Historical based analysis
This direction was enabled because a lot of ADS-B data is available. The idea was to use this data to search for deviations in the trajectories based on two different methods.

- Explain company routes
- Explain using clustering -> refer to literature -> A lot has already been done in this field, therefore this direction is less interesting.
  Also not directly in scope
- Current status of this project.

This research direction was proposed because of the large amount of ADS-B data available. A year's worth of ADS-B data was available for this thesis, in combination with a month of Eurocontrol data. Therefore several ideas sparked to use this data to detect anomalies in the trajectory data, as mentioned before in section 2.6.

Company routes
Every airline has to file a flight plan before every flight. By comparing a lot of flight plans together a 'company route' could be identified. Company routes are routes that are preferred routes by an airline and therefore

Figure 4.1: Seven different airways plotted from three different databases

often planned. By comparing the actual flight data against these routes, anomalous flights can be identified. Essentially this creates carrier-specific airways. By comparing the deviations of the aircraft from this airway anomalous trajectories can be found.

Trajectory clustering
Two detection methods that use trajectory clustering methods were hypothesized. The first method tries to identify clusters of aircraft trajectories and uses the outliers to detect anomalous flights. However, it was reasoned that this detection method could not detect small changes such as ATC commands. This is based on further analysis of current literature as shown in section 2.6.

The second method was to use clustering to identify 'flows'. Flows are collections of trajectories that can be clustered together that are persistent for longer amount of times. These flows could then be used as another alternative to airways. The middle of these flows could be used as a guide, to which other trajectories in the flow could be compared. However, the same argument as before is raised. Small deviations might not be detected at all.

4.3. Validation strategies

A ground truth is not available for this thesis. Controller Pilot Data Link Communications (CPDLC) was not available from Eurocontrol to validate the changes found directly. Therefore, other validation strategies were considered.

Transcribing ATC voice commands by hand Since ATC voice commands can be listened to freely a method of validating was to record a section of the voice commands given by MUAC. This could be manually transcribed into machine-readable data and used as a validation set. Several limitations of this came up. The LiveATC archives are accessible for free for 30 days and the Broadcastify Archive is available for one year. The ADS-B and Eurocontrol dataset available is currently June 2018, which is more than a year ago at the time of writing (March 2021). Furthermore, the voice commands are hard to hear for the untrained ear.
Current data can also be gathered using the Opensky Network. This provides ADS-B data with a resolution of 10 seconds (for non-registered users). When combining the recording of voice commands and live tracking, a new dataset can be generated that includes both ADS-B data and voice commands that can act as ground truth.

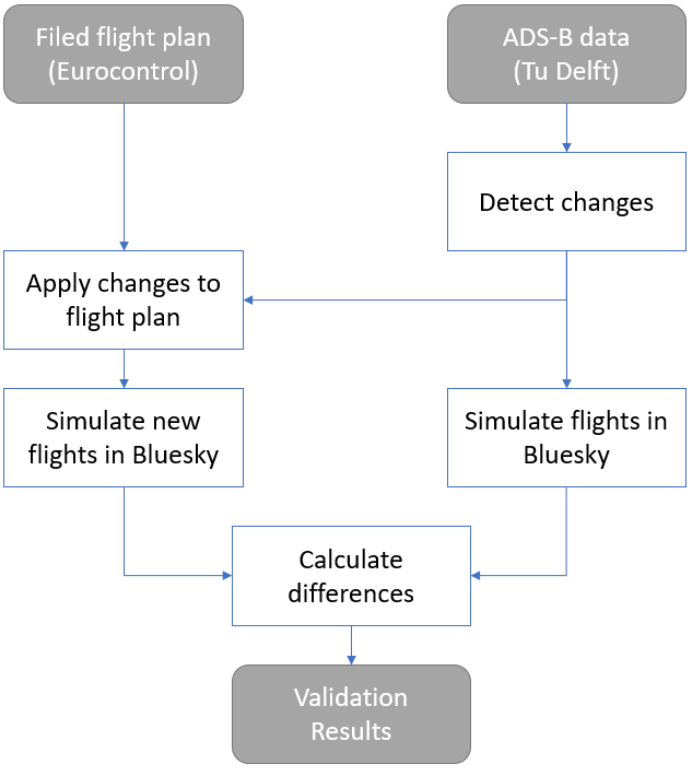


Figure 4.2: Validation strategy for applying detected changes to original flightplan

Table 4.1: List of validation metrics

Metric	**Unit**
Cross-track error	m
Along-track error	m
Altitude error	m
Horizontal error	m
Others	-

Another approach was to simulate the aircraft in Bluesky using the original flight plans and modifying these with the detected ATCo actions. The resulting flight paths could be compared to the actual ADS-B data gathered. Differences between the two tracks could be measured using metrics such as cross-track error, along-track error, altitude difference, horizontal difference, and heading difference.
Caution has to be taken when the actual flight plan is too different from the filed flight plan. When a flight plan has already significantly changed before the aircraft reaches MUAC airspace the flight is no longer useable. A flowchart of this validation strategy is given in fig. 4.2.

The second validation strategy is chosen to validate the performance of the model. This model has several advantages. The first is that idea behind the thesis is to simulate aircraft trajectories without ATC commands in Bluesky. Using Bluesky is therefore a logical choice. Secondly, the validation can be done for multiple flights or multiple days efficiently and is not dependent on manual work to prepare a different validation. The last advantage is that Bluesky is an open-source project and can therefore be used and modified by anyone, making it flexible to different airspaces. The major disadvantage of using this strategy is that no direct comparison to actual ATC commands can be made. The metrics used during the validation process can be found in table 4.1. There can still be added to this list of metrics.

4.4. Limitations

The main limitation of this thesis is that no CPDLC is available to validate the model. A ground truth dataset could serve as a training set for supervised machine learning models and as a validation set for the current approaches.

5

Preliminary results

This section will present and discuss the results found so far in this thesis. It will first describe the results found by investigating the rule-based analysis. The progress is shown for heading, altitude, and speed. It also describes the current progress for the historical based analysis in section 5.2.

5.1. Rule-based results

The results of the rule-based analysis can be found below. Several steps were taken to find a good rule or set of rules from the proposed methods in section 4.2.1.

5.1.1. Heading change detection

The first step for detecting changes in the heading included testing the smoothing function. Initial parameters set for the detection methods were also used for extra context. The result for the heading can be seen in fig. 5.1, where the k-sigma rule is shown. Other detection methods are shown in appendix A.
This figure shows one flight (Eurocontrol id: 218897382) with all 6 smoothing functions applied including all different values of the function. It also shows the results found (if any) by the K-Sigma rule detection method (where sigma = 3). This plot was made for 10 randomly selected flights.
The smoothing function was also applied to the 'trk' parameter found in the ADS-B data and on the heading data, which was calculated from the position data in the ADS-B data.

From this figure, several conclusions about the smoothing functions could be made.

1. The 'exponential smoothing', 'rolling mean' and rolling 'Gaussian smoothing' functions give too much noise to be considered as good candidates.
2. The calculated heading provides better temporal accuracy than the 'trk' data and therefore provides better accuracy in predicting changes.
3. The 'resample mean' method provides the most robust results.
4. From the 'resample mean', the resampling rate of 10 seconds provides the best results. Small enough to provide good insight, but large enough to filter out small deviation.

Therefore the 'resample mean' with a value of 10 seconds was chosen for the smoothing function.

A small conclusion could also be made for one of the detection methods. The ruptures packages cannot handle missing data. As a result, it returns a 'true' value at every data point.

To check the detection methods a new plot was made, as seen in fig. 5.2. This plot consists of three subplots. The first subplot is the heading angle with 3 different data sources: smoothed calculated heading, raw 'trk' data &smoothed 'trk' data. The latter two only act as a reference. It also contains some scatter points connecting to the figures below.
The second subplot is the standardized absolute change in heading angle. This acts as a guide plot for tuning the k-sigma parameter. It has guidelines at values 2 and 3.

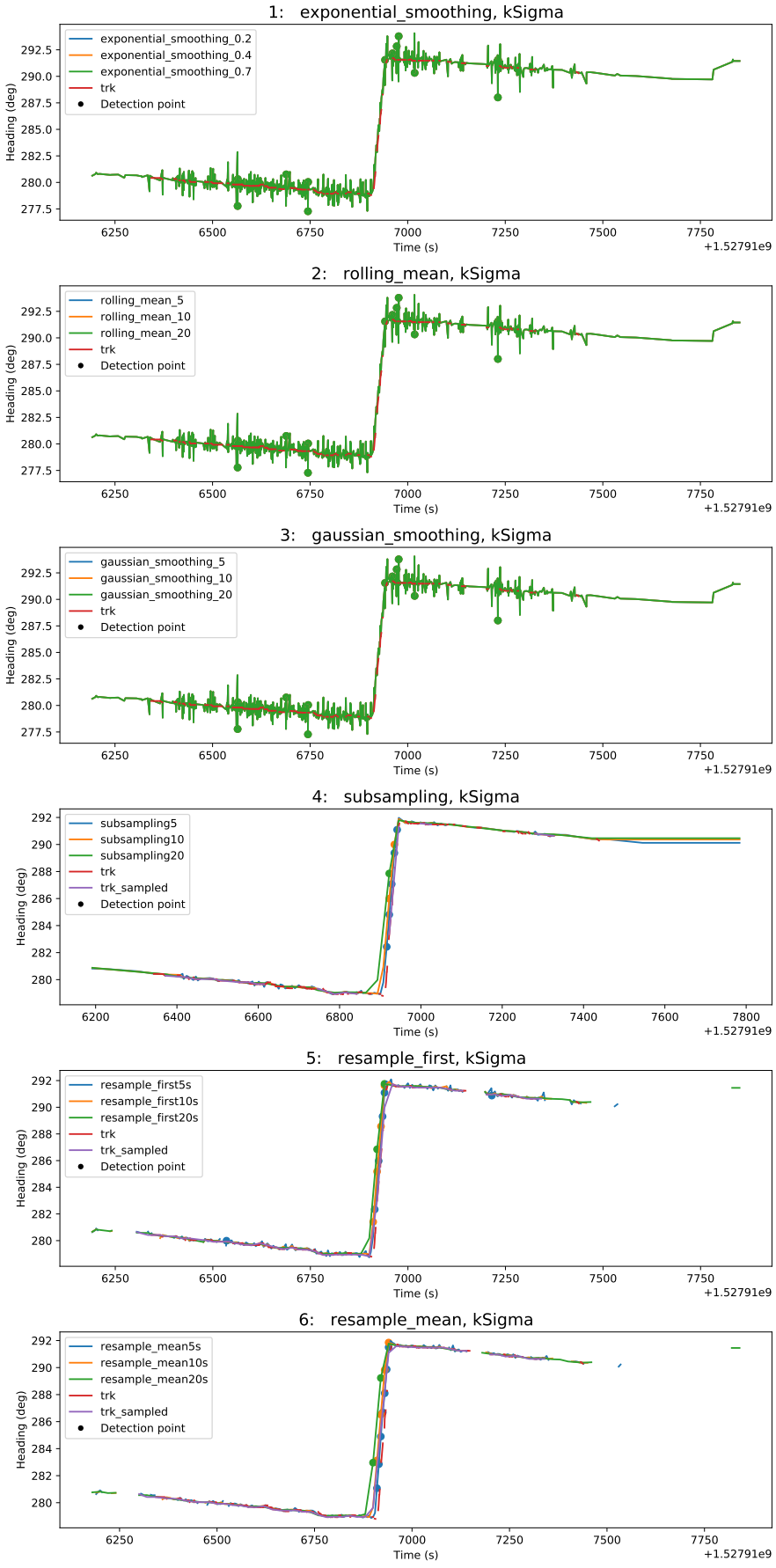


Figure 5.1: All smoothing parameter applied to the flight with Eurocontrol ID 218897382 to calculate the heading. Detection method shown is kSigma.

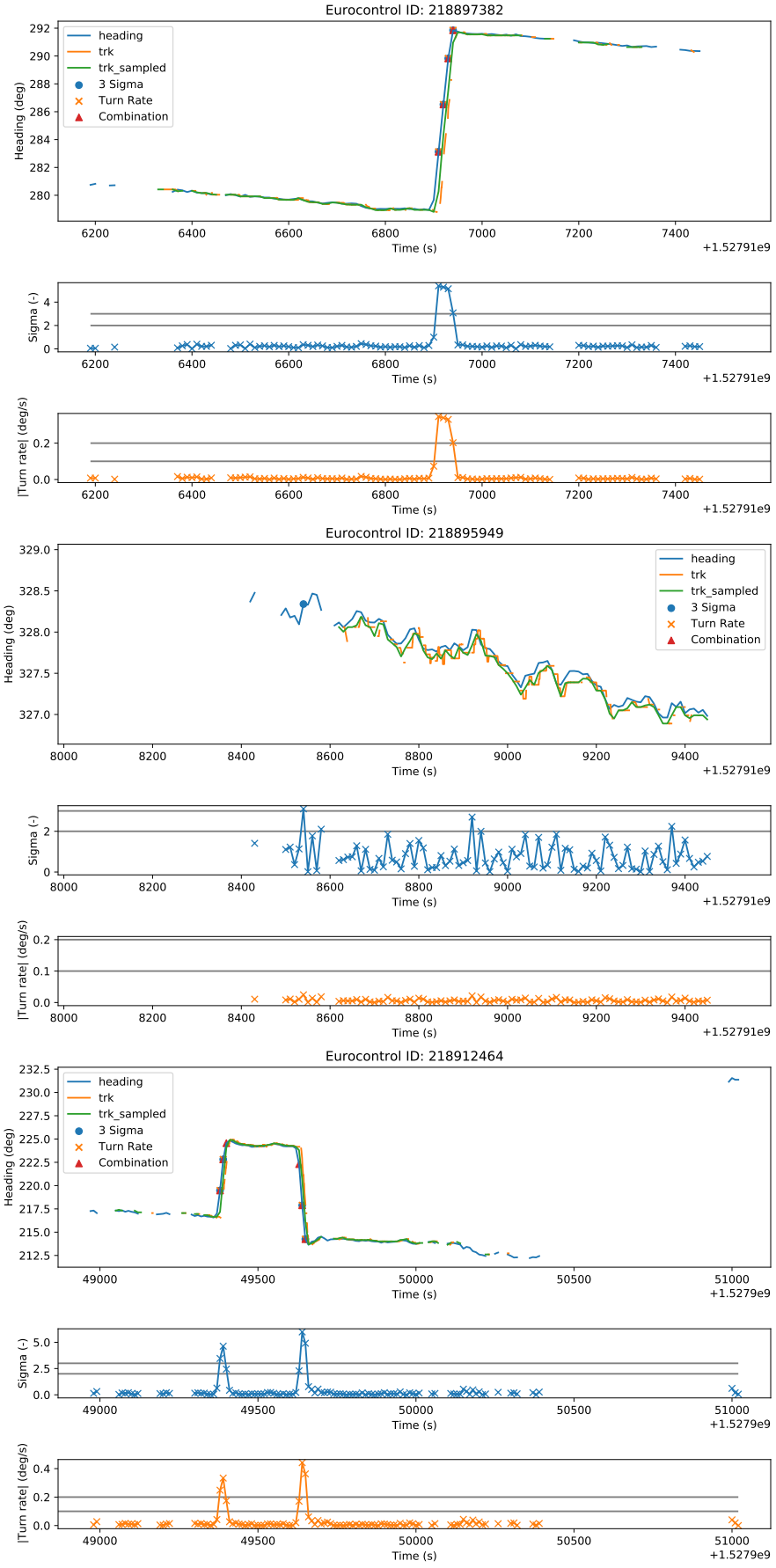


Figure 5.2: Detailed look at detection methods kSigma and Turn rate

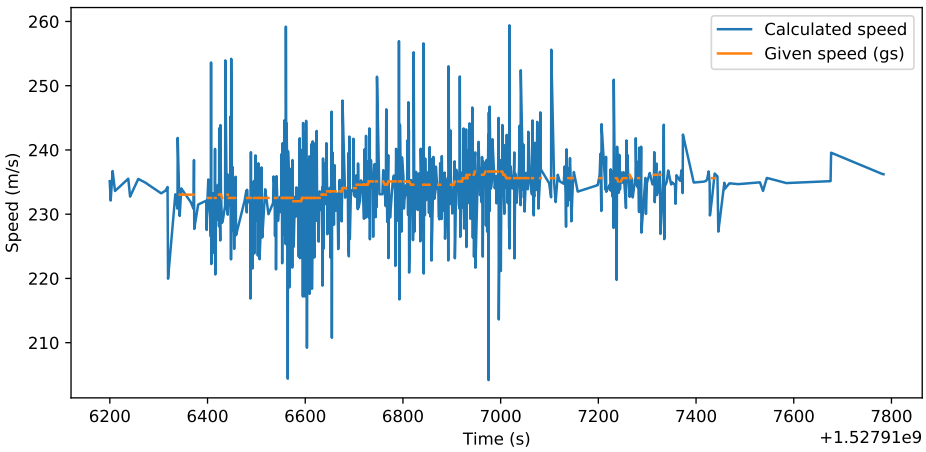


Figure 5.3: The two different methods of determining the aircraft speed. No smoothing functions are applied.

The third subplot shows the absolute change in heading angle and is used as a guide for tuning the detection parameter. The thresholds here are at 0.2 °/s and 0.1 °/s.

Shown in fig. 5.2 are three different flights. The top flight has one visible change in the middle of its flight. The second flight has no clear change in the heading and contains visible noise. This indicates that the aircraft was flying straight during this flight. The bottom flight contains 2 quick turns, first to the right then back to the left. It also contains some visible noise. In total 10 randomly selected flights were inspected to determine a set of rules that was able to detect the heading change.

The k-sigma method has a disadvantage when the aircraft is flying in a straight line. Since no real 'big' deviation is encountered, the standard deviation is small. This allows for the noise points to be considered as changes. The turn rate threshold has the disadvantage that slow corners might not be identified because the parameter is set too high. Therefore a combination was of these two was considered. A combination would allow heading changes to be detected while minimizing the effects of noise.

A rule was experimentally set. This rule consist of the following logic

$$\text{Turn rate} \geq 0.2 \vee (\text{Turn rate} \geq 0.1 \wedge \text{sigma} \geq 2)$$

5.1.2. Speed change detection

For speed changes, the same method to heading was used. The ground speed given from the data and speed calculated from the latitude and longitude positions were used. However, due to the noise in the latitude and longitude values, the calculated values were noisy. This can be seen in fig. 5.3. Smoothing functions were applied to the positional data, as had been done for calculating the heading data. These figures can be found in appendix A. Figure A.4 shows the values from the calculated speed. These values regularly exceed $1000 m/s$ and are therefore considered to be unusable. The same approach was taken for the ground speed gained from the ADS-B data. There can be found in fig. A.5. Here the smoothing function was applied to the ground speed data. This produced cleaner data. Note that the entries where no data was found are excluded from this plot. To better understand which smoothing combinations work well and which don't a simple statistical analysis was done.

In fig. 5.4 the standard deviation of the acceleration is plotted on a log scale. These show a massive difference in resulting smoothing parameters. In the left figure, the results are given for the smoothing of the latitude and longitude positions. Since the standard deviation is higher for some smoothing functions, it means that these do not smooth the result, but instead enlarges the noise of the signal. The best smoothing function for the calculated speed is the subsampling method.
For the given ADS-B data, a smaller scale was needed to better identify the results. Therefore fig. 5.5 has a y-scale limited to 0.2. Here is shown that the smoothing function have lower values than the raw data. This means that the smoothing of the raw data results in a signal with less noise. Even with the improved smoothing when the ADS-B data is used, no conclusion could be drawn. More analysis has to be done to find

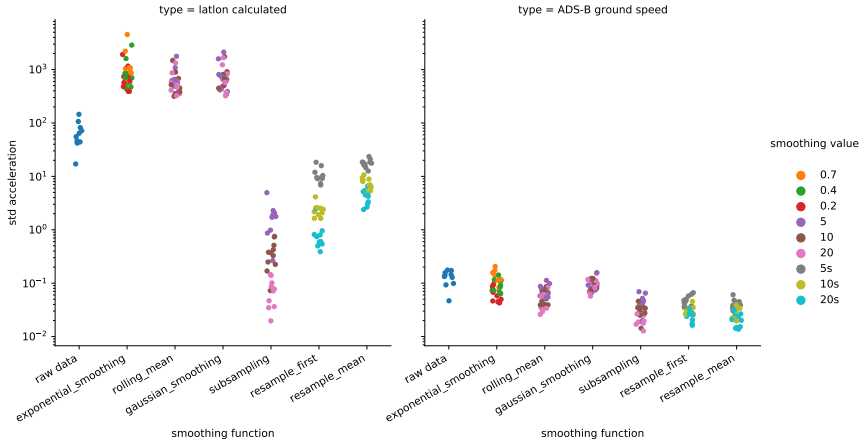


Figure 5.4: The standard deviations of 10 flights, smoothed by different combinations

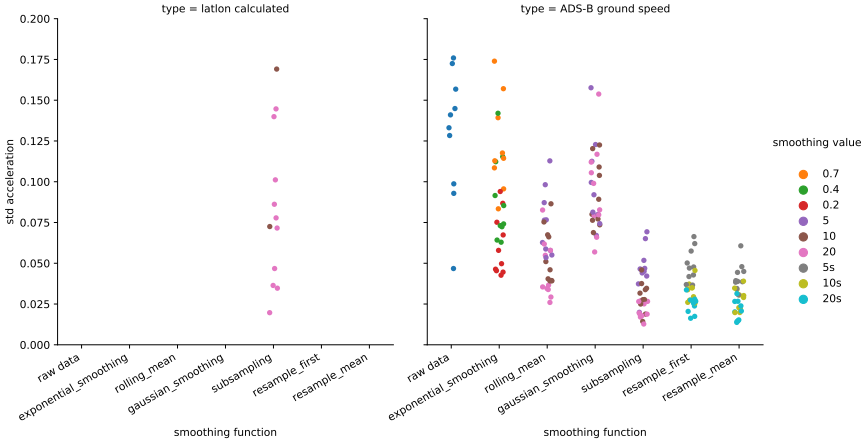


Figure 5.5: The standard deviations of 10 flights, smoothed by different combinations. Limited on the y-axis to 0.2.

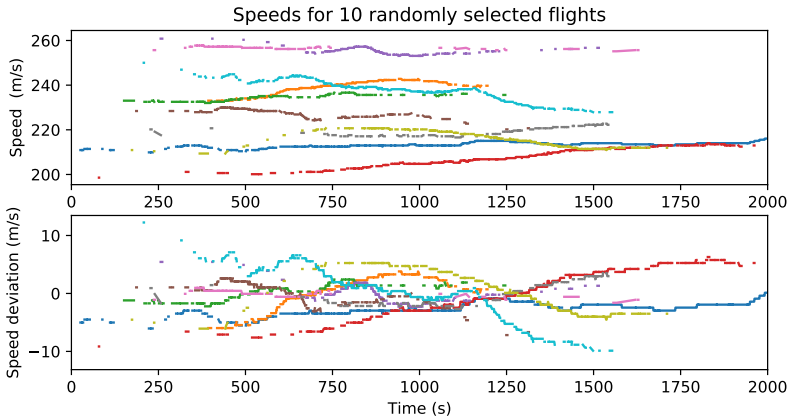


Figure 5.6: Speed and speed deviation for the 10 randomly selected flights.

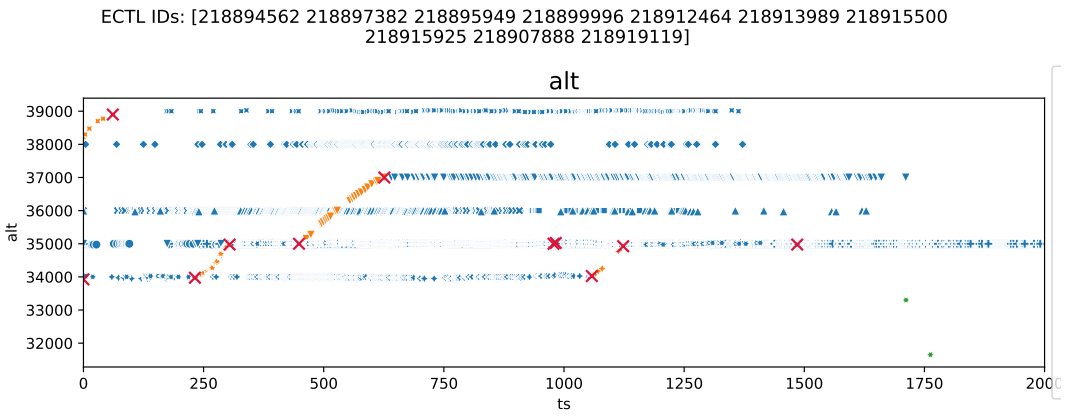


Figure 5.7: Altitudes, phases and phase shifts for 10 flights

a good combination to correctly identify changes in the speed of an aircraft. These analyses include testing smoothing functions on calculated speed and analyzing more flights.

- Apply smoothing function to calculated speed. Not altering the positional data might improve the resulting signal. This can be seen in the subsampling method already since this has the lowest standard deviation.
- Analyse more flight. The current flights might not contain speed changes. These speeds are plotted in fig. 5.6.

5.1.3. Vertical rate change detection

At the time of writing, only one of the two proposed methods for detecting changes in altitude has been implemented. This is the method where phase transitions are used as an indicator for altitude changes. An example of this is shown in fig. 5.7. The red crosses indicate where a phase change has been detected.

This method is not yet fully robust, as can be seen by several red crosses in the middle of a flight, without the flight moving to a new flight level. Some adjustments have to be made to avoid these false positives. This can be done by sampling a larger area of phases, or by using checking the altitude data and looking for distinct changes in flight level.

5.2. Historical data analysis

For the historical based analysis, only a trial with clustering was performed. Both DBSCAN and spectral clustering were implemented as a test. Flights flown by Lufthansa, departing from Frankfurt airport were clustered together to test if the clustering methods work. The results can be seen in appendix B.

Here the differences between the two algorithms can be seen. DBSCAN allows for noise. This is shown in fig. B.1 as black. These trajectories could be not assigned to a group. On the other hand, spectral clustering does not allow for noise. It groups the extra-long trajectories, as by the light blue dots in fig. B.2. These are more different than the other trajectories.

Further tuning and testing has to be done before these clustering techniques can be properly used for analysis. However, this method may be redundant, since the rule-based detection provides much promise in identifying ATC commands. Previous literature also provides several methods to identify anomalous events in ADS-B data through the use of various methods.

6

Planning

This chapter will present the planning of the thesis after this preliminary report. In section 6.1 the next steps of this thesis will be explained. Section 6.2 will present the overall timeline of this report and accompanying Gantt chart.

6.1. Steps to be taken

This section will describe the next steps for this report. It will start more detailed with the steps that are taken right after this preliminary report and work towards the broader end goals.

The next steps are to finish the rule-based analysis part of the experiment, as described in section 4.2.1. This includes the following.

- Heading change analysis
    1. After a change has been detected, retrieve the old and new heading.
    2. Search along the route if a waypoint is crossed. This may indicate a 'direct to' command.
    3. Compare change to the filed flight plan to check if the change was procedural or not.
    4. Couple the findings to the datapoint where the change was initially detected.
- Altitude change analysis
    1. Make the current method more robust to reduce the effect of single (or a small number of) false phase change(s).
    2. Find the old and new altitude.
    3. Compare change to the filed flight plan to check if the change was procedural or not.
    4. Couple the findings to the datapoint where the change was initially detected.
- Speed change analysis
    1. Apply the smoothing function to the calculated speed. Check this against the current methods to determine the best possibility for change detection
    2. Include more flights in the analysis. The current flight might not have any (significant) speed changes to be detected.
    3. If no significant changes are detected. Conclude that speed detection may not available with the current data.
    4. Else, compare to the filed flight plan, couple finding with data points.
- Check if the above commands are given at the same point in time and combine these if they do.

When these steps are completed a framework has to be made, that can quickly scan any flight trajectory and return any changes it has found.

The next step is to start the validation of the results. This is also described in section 4.3, but will be summarized here. The ADS-B data need to be converted to Bluesky scenarios. Two scenarios will be made that will simulate the raw ADS-B data and the filed flight plans, amended with the found changes. Both scenarios will be run in Bluesky and the data will be analyzed using metrics as along-track error, cross-track error, and altitude error (among others). Based on the analysis of these results, a conclusion can be drawn regarding how the change detection analysis performs.

6.2. Project planning

A Gantt chart has been made to track the progress of the thesis and to guide further work. The Gantt chart can be found in appendix C. The work has been divided into 5 major work packages, which correspond to 5 major steps taken in this research project. These are: 1) reviewing literature, 2) preprocessing data, 3) following the research directions and exploring data, 4) validating the found results, and 5) reporting the findings.
A more detailed plan has been given for the research directions since these took the most amount of time.

7

Conclusion

This report presents the literature study into the field of detecting ATC commands in past ADS-B data and the progress made at the time of writing. Using ADS-B data a lot of information can be gained. However, no method has been proposed to detect basic ATCo actions and commands in past ADS-B data. Finding these actions can help to determine the complexity of the airspace, hidden habits of ATCo. It can also be helpful for other machine learning models, as it can be used as extra input or as training or validation data. Furthermore, if a simulated airspace can be generated where these commands are not present, a new 'stripped' airspace scenario can be made. This could be used to train automated separation software or to train ATC students.

Current literature presents many different ways to extract knowledge from ADS-B data or to add extra context. A statistical model can infer the flight phase of an aircraft, state-of-the-art model has been proposed to identify anomalies in aircraft trajectories or even to predict the ETA of aircraft entering the CTA of an airport. However, no definitive tool or paper is found that addresses detecting the actions of ATC. One specific reference is found where turn rates were used to detect the start of a STAR, however, this was not researched since it wasn't the goal of the paper. To add to the current literature a tool or method shall be delivered that is accurate, easy to use, and preferably quickly computed.

Several preprocessing steps are presented to add extra context and information to the ADS-B data. Several of these steps are presented in previous literature. When combined with the Eurocontrol R&D dataset, a dataset is gained with more context. This allows to select flights based on carrier, airport (destination and/or departure), and more.

To achieve the goal of detecting ATCo actions several steps had to be taken. The first step was to limit the area of research for this thesis. It was chosen to use the en-route phase of aircraft within the Maastricht Upper Area Control. The main reason for this choice was the availability of the data and the relatively simple airspace. Secondly, several directions were identified where information about changes in trajectory could be found. These were using time series analysis, using airways, or using historical data (or big data) analysis. It was concluded that the time series analysis had more potential and was less computationally expensive, and therefore the best choice. Several smoothing functions were selected and tested to reduce the noise in the data. This allowed for better detection of changes in the ADS-B data. The next step is to use the flight plan information, provided by Eurocontrol, to check if the change detected was a procedural change or a non-procedural change.

Next, a validation strategy was proposed. The method chosen uses the BlueSky air traffic simulator to simulate a new airspace scenario. This scenario is made using the detected non-procedural change and the original flight plan. By combining these a new flight plan is made. Simulating the new flight plan(s) and comparing the results against the known ADS-B data, gives a measure if the detected ATC commands are valid. Several metrics have been proposed to calculate the error between the new flight plan and ADS-B data.

Results of the discussed smoothing functions and detection methods are presented. It was found that resampling the ADS-B data with a resolution of 10 seconds was the best smoothing function. A rule was also presented in order to detect changes in the heading. This rule is:

Turn rate ≥ 0.2 or (turn rate ≥ 0.2 and sigma ≥ 2)
For changes in the altitude, transitions in the earlier found flight phase were used. This produced positive results, but the method has to be further tuned to reduce the effect of noise.
It was found that the statistical analysis proposed does not yet work for speed changes. Several options are presented as possible alternatives.

To finish the report, several next steps have been presented and a global planning in the form of a Gantt chart. These next steps form a roadmap from the current status of the research to a finished and validated tool. These steps include finalizing the time series analysis, checking the detected change with the flight plans, starting the validation process, and analysing the final results.

Bibliography

[1] Mayara Condé Rocha Murça et al. “Trajectory clustering and classification for characterization of air traffic flows”. In: *16th AIAA Aviation Technology, Integration, and Operations Conference* June (2016), pp. 1–16. DOI: 10.2514/6.2016-3760.

[2] Maxime Gariel, Ashok N. Srivastava, and Eric Feron. “Trajectory clustering and an application to airspace monitoring”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 1511–1524. ISSN: 15249050. DOI: 10.1109/TITS.2011.2160628.

[3] Marco Enriquez. “Identifying Temporally Persistent Flows in the Terminal Airspace via Spectral Clustering”. In: *Proceedings of the 10th USA/Europe Air Traffic Management Research and Development Seminar, ATM 2013* (2013).

[4] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://scikit-learn.sourceforge.net..

[5] Eurocontrol. *Maastricht Upper Area Control Centre (MUAC) upper air route structure chart | EUROCONTROL*. Dec. 2020. URL: https://www.eurocontrol.int/publication/maastricht-upper-area-control-centre-muac-upper-air-route-structure-chart.

[6] Martin Strohmeier et al. “Realities and challenges of nextgen air traffic management: the case of ADS-B”. In: *IEEE Communications Magazine* 52.5 (May 2014), pp. 111–118. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6815901. URL: http://ieeexplore.ieee.org/document/6815901/.

[7] Airservices Australia. *How ADS-B works - Airservices*. URL: https://www.airservicesaustralia.com/about-us/projects/ads-b/how-ads-b-works/.

[8] Junzi Sun. *The 1090MHz Riddle*. 2nd ed. TU Delft OPEN Publishing, 2020. URL: https://mode-s.org/decode/adsb/introduction.html.

[9] Junzi Sun et al. “pyModeS: Decoding Mode-S Surveillance Data for Open Air Transportation Research”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.7 (July 2020), pp. 2777–2786. ISSN: 1524-9050. DOI: 10.1109/TITS.2019.2914770. URL: https://ieeexplore.ieee.org/document/8718517/.

[10] Matthias Schäfer et al. “Bringing up OpenSky: A large-scale ADS-B sensor network for research”. In: *ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2014. URL: http://www.opensky-network.org.

[11] Junzi Sun, Joost Ellerbroek, and Jacco Hoekstra. “Large-Scale Flight Phase Identification from ADS-B Data Using Machine Learning Methods”. In: *7th International Conference on ...* August (2016). URL: https://www.researchgate.net/profile/Junzi_Sun/publication/305403993_Large-Scale_Flight_Phase_Identification_from_ADS-B_Data_Using_Machine_Learning_Methods/links/578dece708ae9754b7e9c6e8.pdf.

[12] Junzi Sun, Joost Ellerbroek, and Jacco Hoekstra. “Flight extraction and phase identification for large automatic dependent surveillance-broadcast datasets”. In: *Journal of Aerospace Information Systems* 14.10 (2017), pp. 566–571. ISSN: 23273097. DOI: 10.2514/1.I010520.

[13] Junzi Sun, Jacco M. Hoekstra, and Joost Ellerbroek. “Estimating aircraft drag polar using open flight surveillance data and a stochastic total energy model”. In: *Transportation Research Part C: Emerging Technologies* 114 (2020), pp. 391–404. ISSN: 0968090X. DOI: 10.1016/j.trc.2020.01.026.

[14] Marko Hrastovec and Franc Solina. “Prediction of aircraft performances based on data collected by air traffic control centers”. In: *Transportation Research Part C: Emerging Technologies* 73 (Dec. 2016), pp. 167–182. ISSN: 0968090X. DOI: 10.1016/j.trc.2016.10.018.

[15] Junzi Sun, Jacco M. Hoekstra, and Joost Ellerbroek. “OpenAP: An open-source aircraft performance model for air transportation studies and simulations”. In: *Aerospace* 7.8 (2020). ISSN: 22264310. DOI: 10.3390/AEROSPACE7080104.

[16] Charles Truong, Laurent Oudre, and Nicolas Vayatis. “Selective review of offline change point detection methods”. In: (2020).

[17] Zhengyi Wang, Man Liang, and Daniel Delahaye. “Short-term 4D trajectory prediction using machine learning methods”. In: *SESAR Innovation Days* November (2017). ISSN: 07701268.

[18] C A Dek. *Predicting 4D Trajectories of Aircraft using Neural Networks and Gradient Boosting Machines*. Tech. rep. Delft University of Technology, Delft, The Netherlands, 2020, pp. 1–16.

[19] Guan Yuan et al. “A review of moving object trajectory clustering algorithms”. In: *Artificial Intelligence Review* 47.1 (Jan. 2017), pp. 123–144. ISSN: 15737462. DOI: 10.1007/s10462-016-9477-7.

[20] L. Basora, X. Olive, and T. Dubot. “Recent advances in anomaly detection methods applied to aviation”. In: *Aerospace* 6.11 (2019). ISSN: 22264310. DOI: 10.3390/aerospace6110117.

[21] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: (1996), pp. 226–231. URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.9220.

[22] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[23] Ajay Srinivasamurthy et al. “Iterative learning of speech recognition models for air traffic control”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. Vol. 2018-Septe. International Speech Communication Association, 2018, pp. 3519–3523. DOI: 10.21437/Interspeech.2018-1447.

[24] Shreyas Vathul Subramanian, Peter Kostiuk, and Graham Katz. “Custom IBM watson speech-to-text model for anomaly detection using ATC-pilot voice communication”. In: *2018 Aviation Technology, Integration, and Operations Conference* (2018). DOI: 10.2514/6.2018-3979.

[25] Xavier Olive et al. “Detecting controllers’ actions in past mode S data by autoencoder-based anomaly detection”. In: *SESAR Innovation Days* December (2018). ISSN: 07701268.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[27] Xavier Olive et al. “Detecting Events in Aircraft Trajectories: Rule-Based and Data-Driven Approaches”. In: *Proceedings* 59.1 (2020), p. 8. ISSN: 2504-3900. DOI: 10.3390/proceedings2020059008.

[28] Zhengyi Wang, Man Liang, and Daniel Delahaye. “Automated data-driven prediction on aircraft Estimated Time of Arrival”. In: *Journal of Air Transport Management* 88.December (2020). ISSN: 09696997. DOI: 10.1016/j.jairtraman.2020.101840.

[29] Jacco M. Hoekstra and Joost Ellerbroek. “BlueSky ATC simulator project: an open-data and open-source approach”. In: *Proceedings of the 7th International Conference on Research in Air Transportation* (2016), pp. 1–8.

[30] Xavier Olive. “traffic, a toolbox for processing and analysing air traffic data”. In: *Journal of Open Source Software* 4.39 (July 2019), p. 1518. ISSN: 2475-9066. DOI: 10.21105/joss.01518. URL: http://joss.theoj.org/papers/10.21105/joss.01518.

[31] Eurocontrol. *Aeronautical information services around the world (@IS Online) | EUROCONTROL*. URL: https://www.eurocontrol.int/articles/ais-online.

[32] Eurocontrol. *About our Maastricht Upper Area Control Centre | EUROCONTROL*. 2021. URL: https://www.eurocontrol.int/info/about-our-maastricht-upper-area-control-centre.

[33] R. Killick, P. Fearnhead, and I. A. Eckley. “Optimal detection of changepoints with a linear computational cost”. In: *Journal of the American Statistical Association* 107.500 (2012), pp. 1590–1598. ISSN: 01621459. DOI: 10.1080/01621459.2012.737745.

A

Figures from Rule Based detection

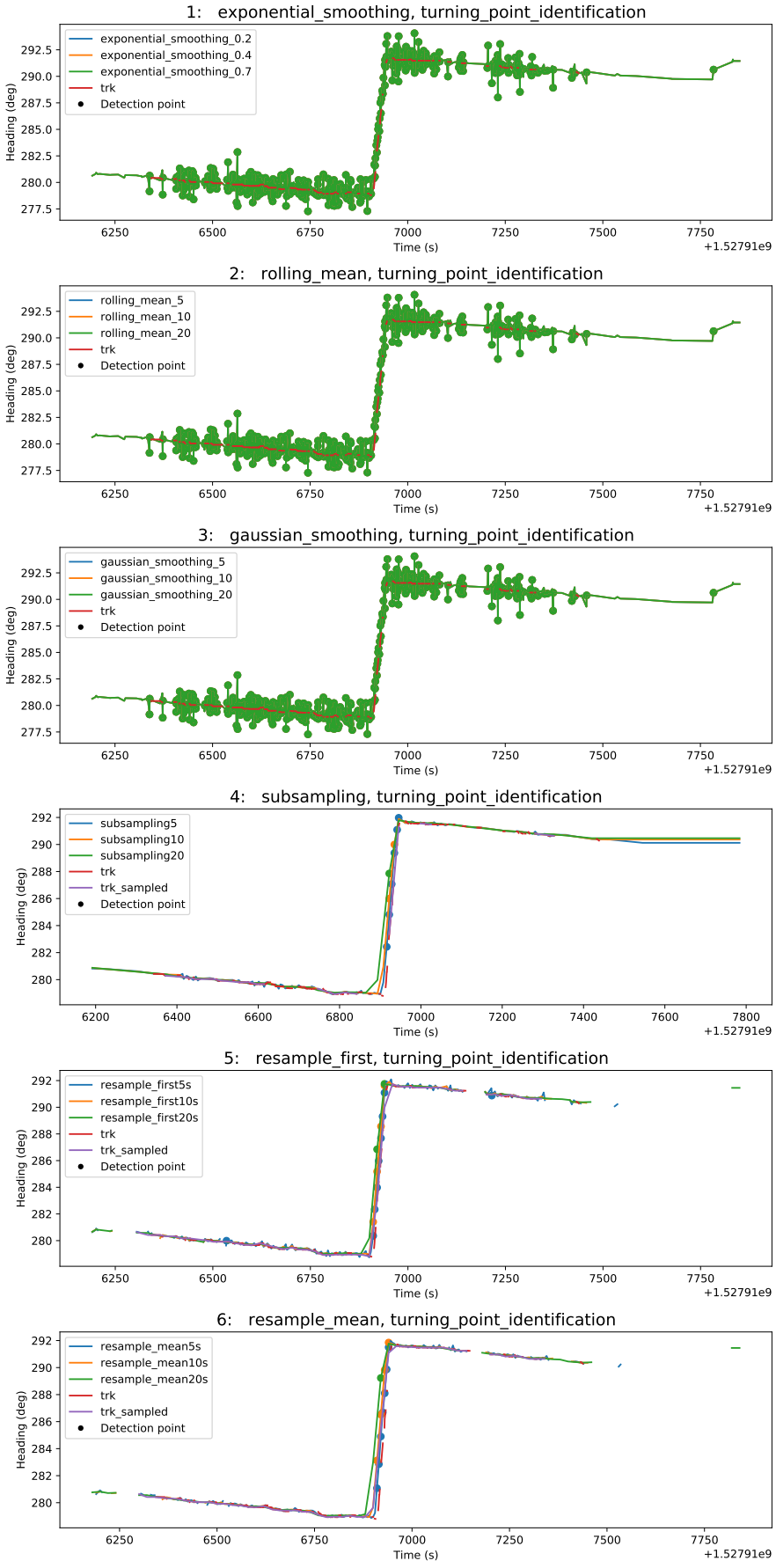


Figure A.1: All smoothing parameter applied to the flight with Eurocontrol ID 218897382 to calculate the heading. Detection method shown is turn rate method.

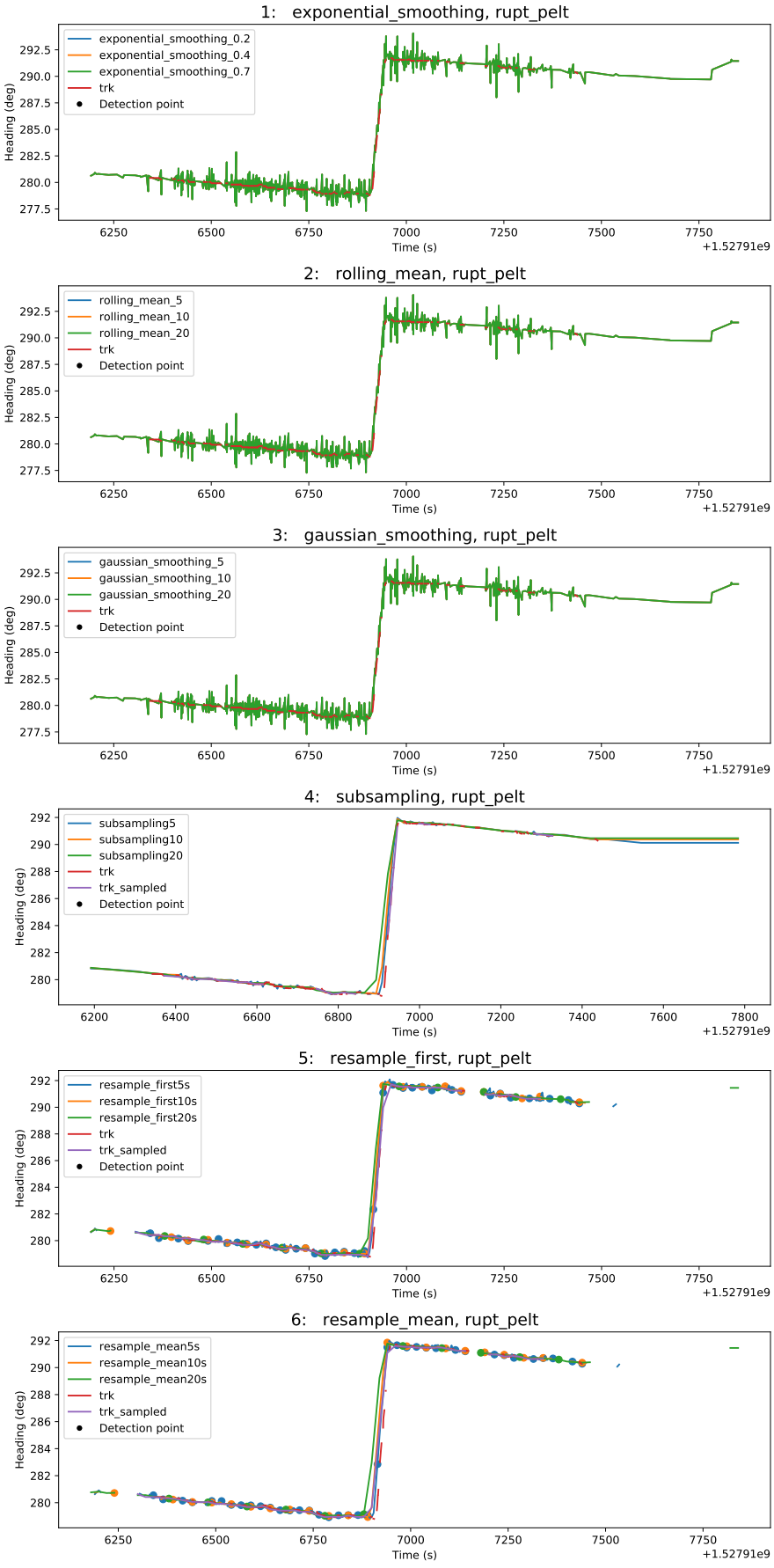


Figure A.2: All smoothing parameter applied to the flight with Eurocontrol ID 218897382 to calculate the heading. Detection method shown is the ruptures pelt method.

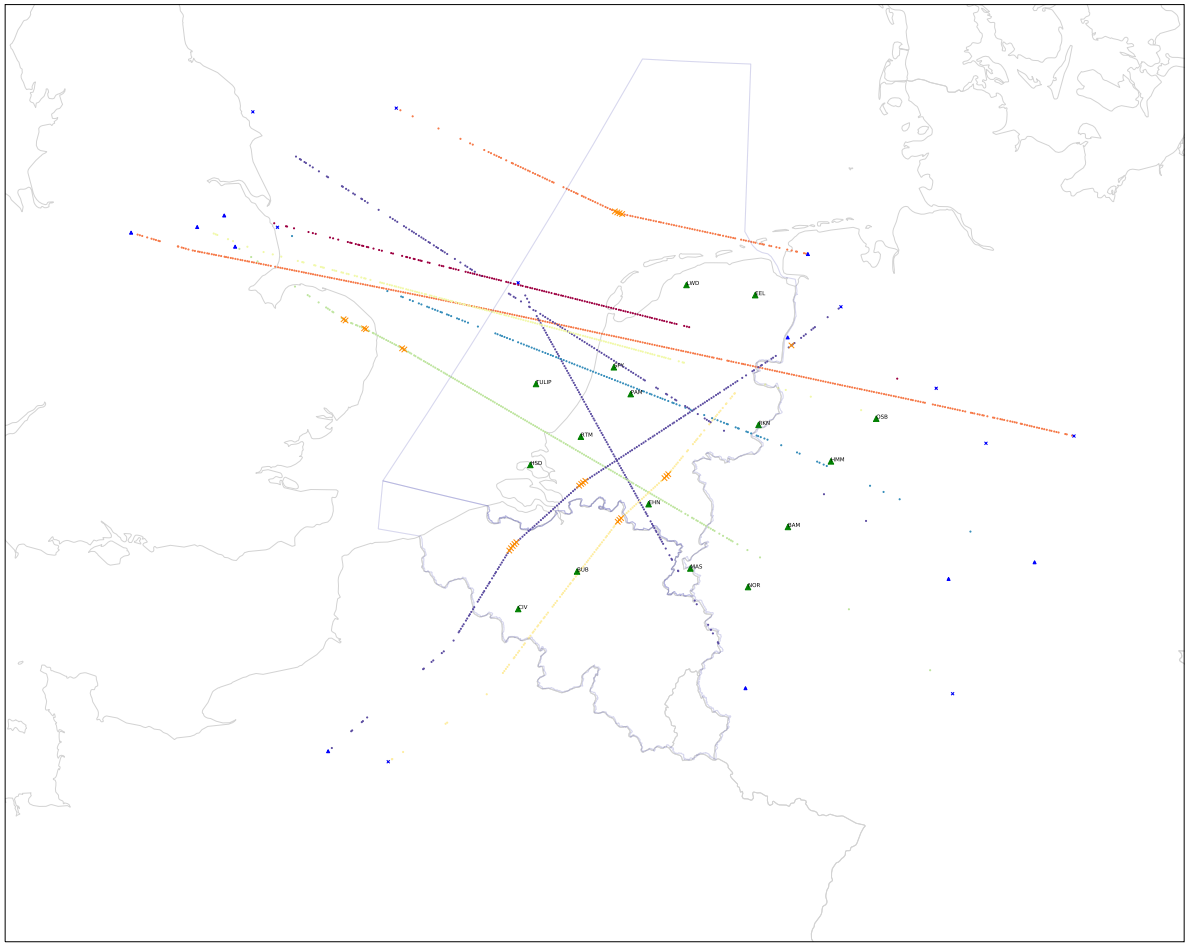


Figure A.3: ADS-B data plotted for 10 different flight. Orange markers indicate a detected turn.

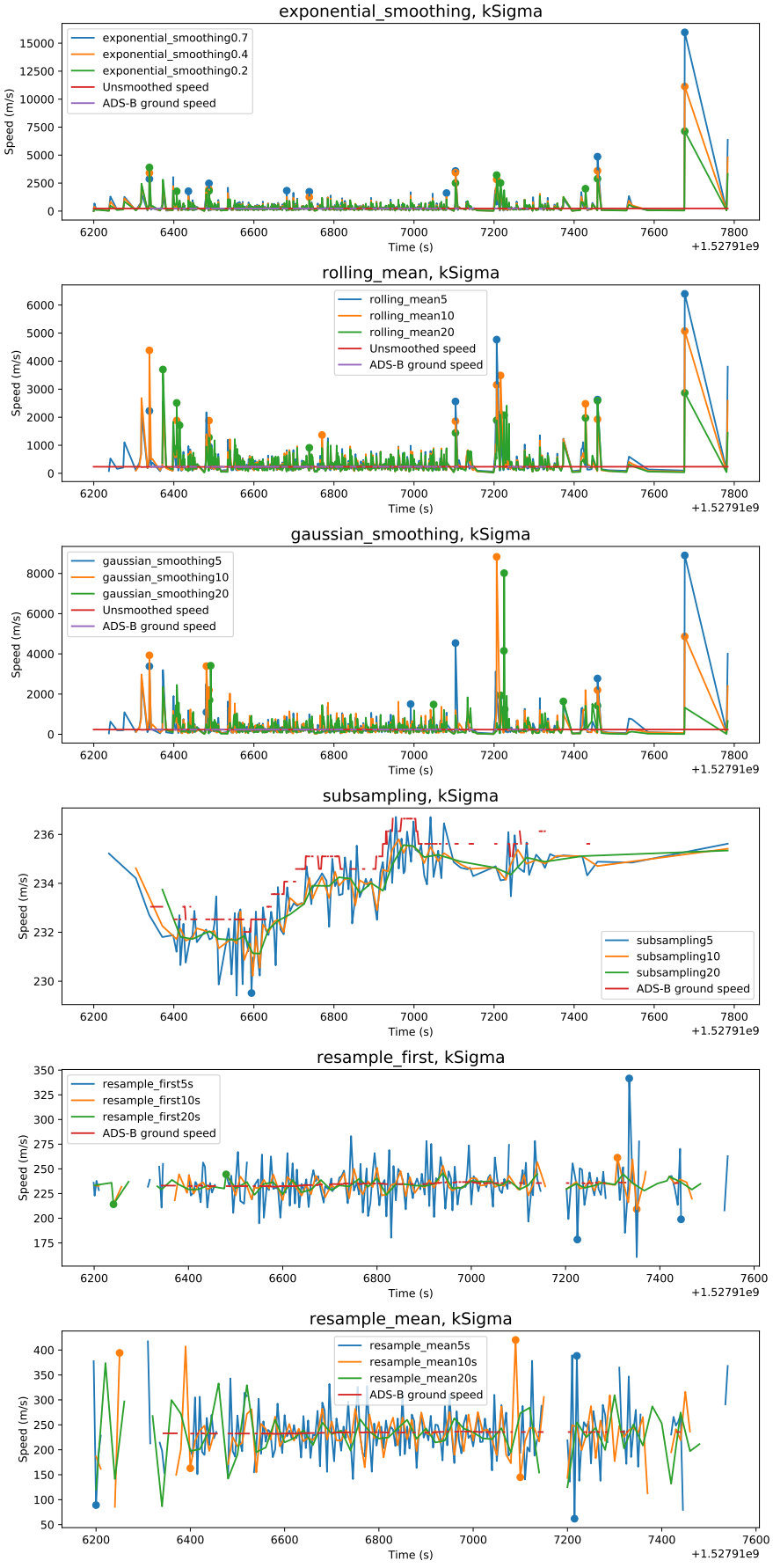


Figure A.4: The speed of flight 218897382 (Eurocontrol id) as calculated from the latitude and longitude data.

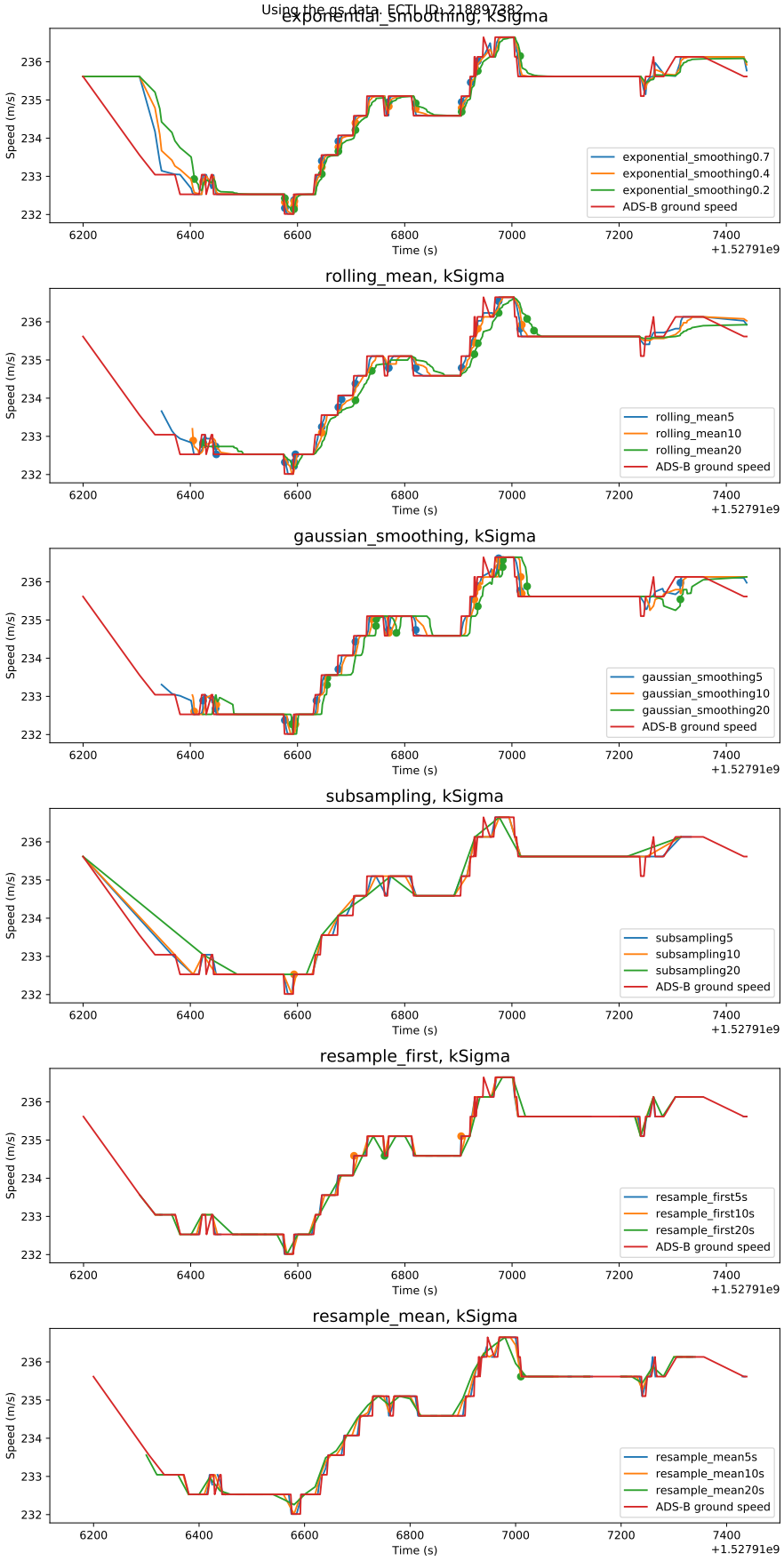


Figure A.5: The speed of flight 218897382 (Eurocontrol id) from the ADS-B data.

B

Clustering results

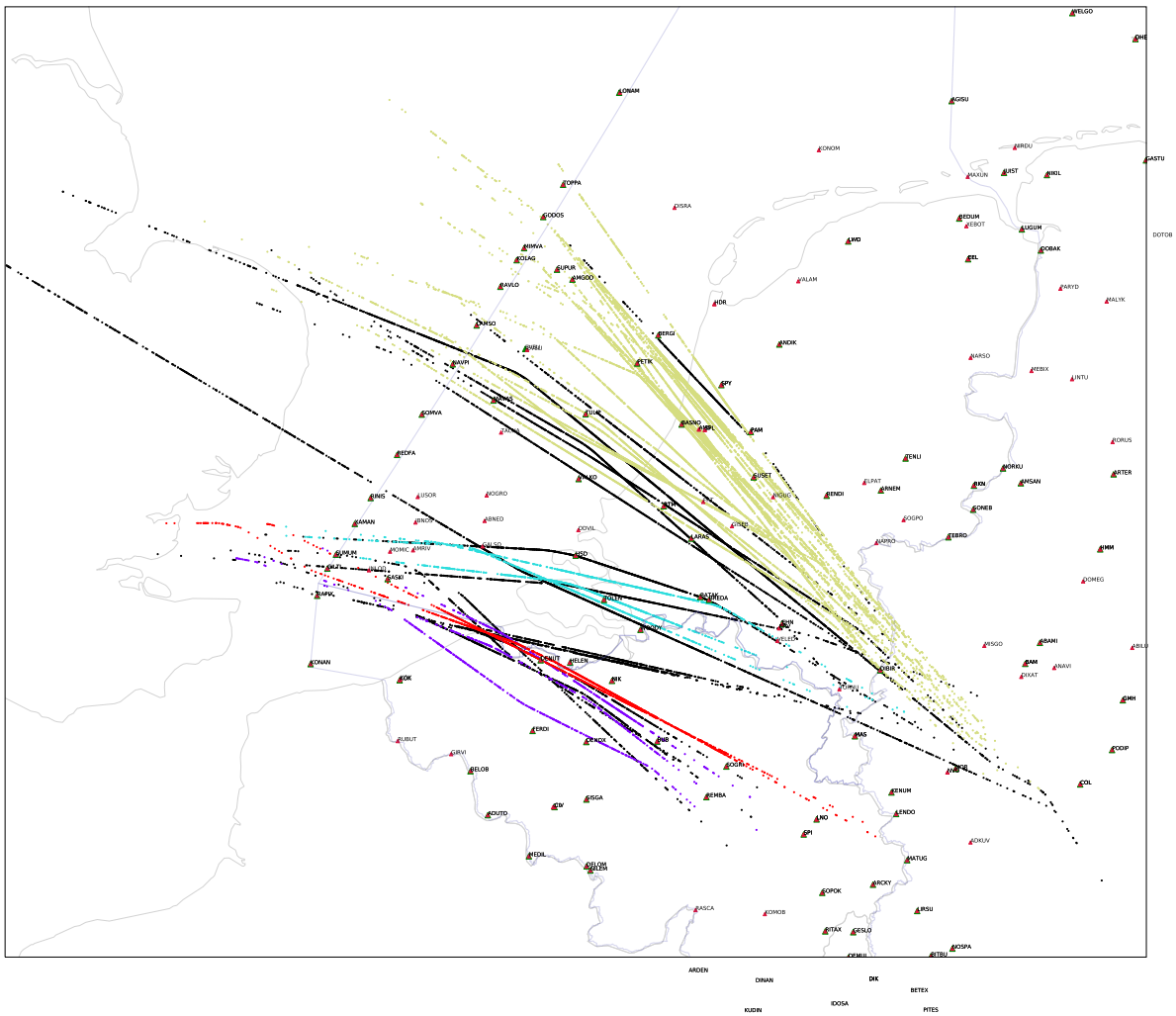


Figure B.1: Clustering result using PCA and DBSCAN. Departures from Frankfurt by Lufthansa for one day.

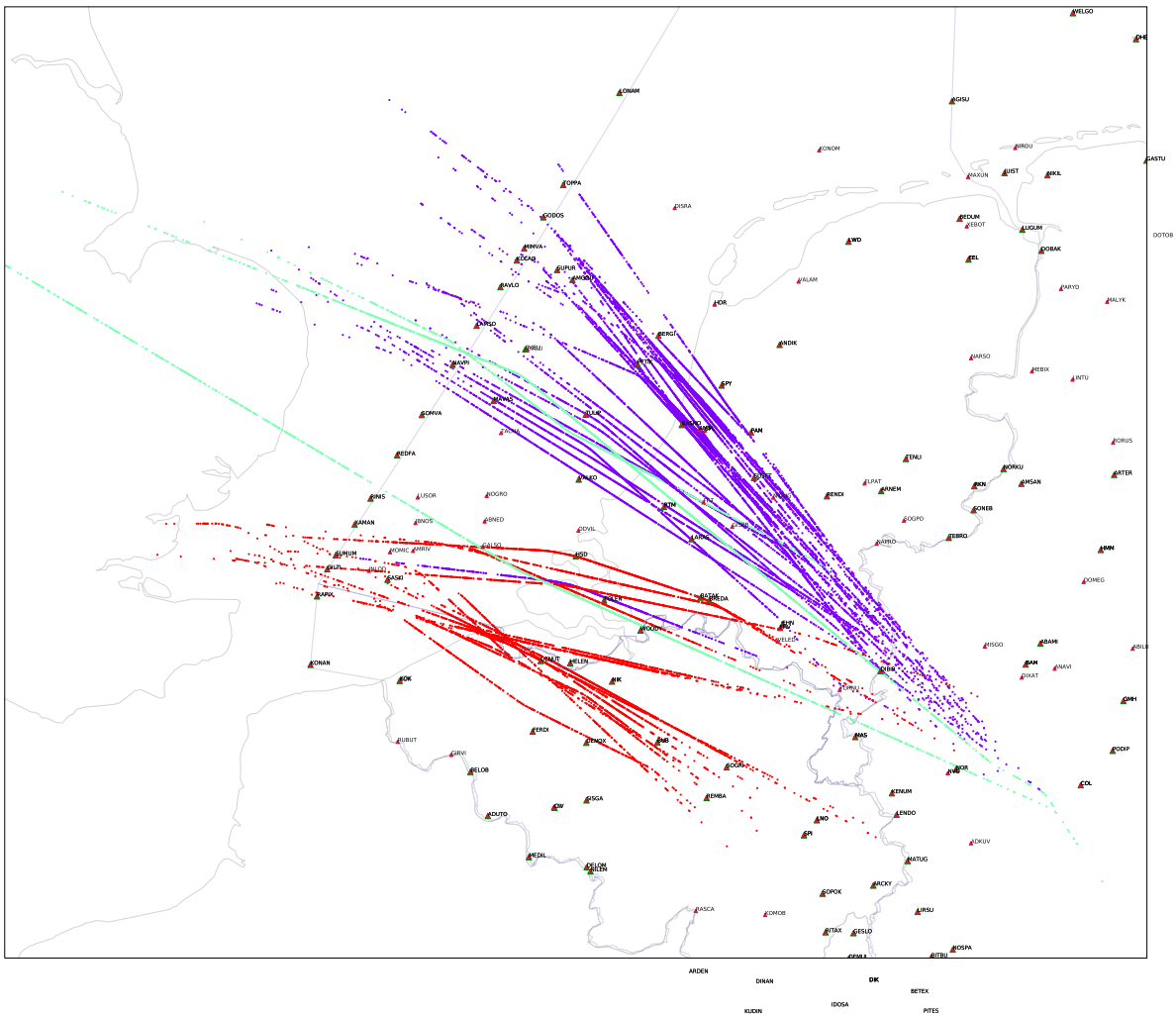


Figure B.2: Clustering result using specral clustering. Departures from Frankfurt by Lufthansa for one day.

C

Project Gantt chart

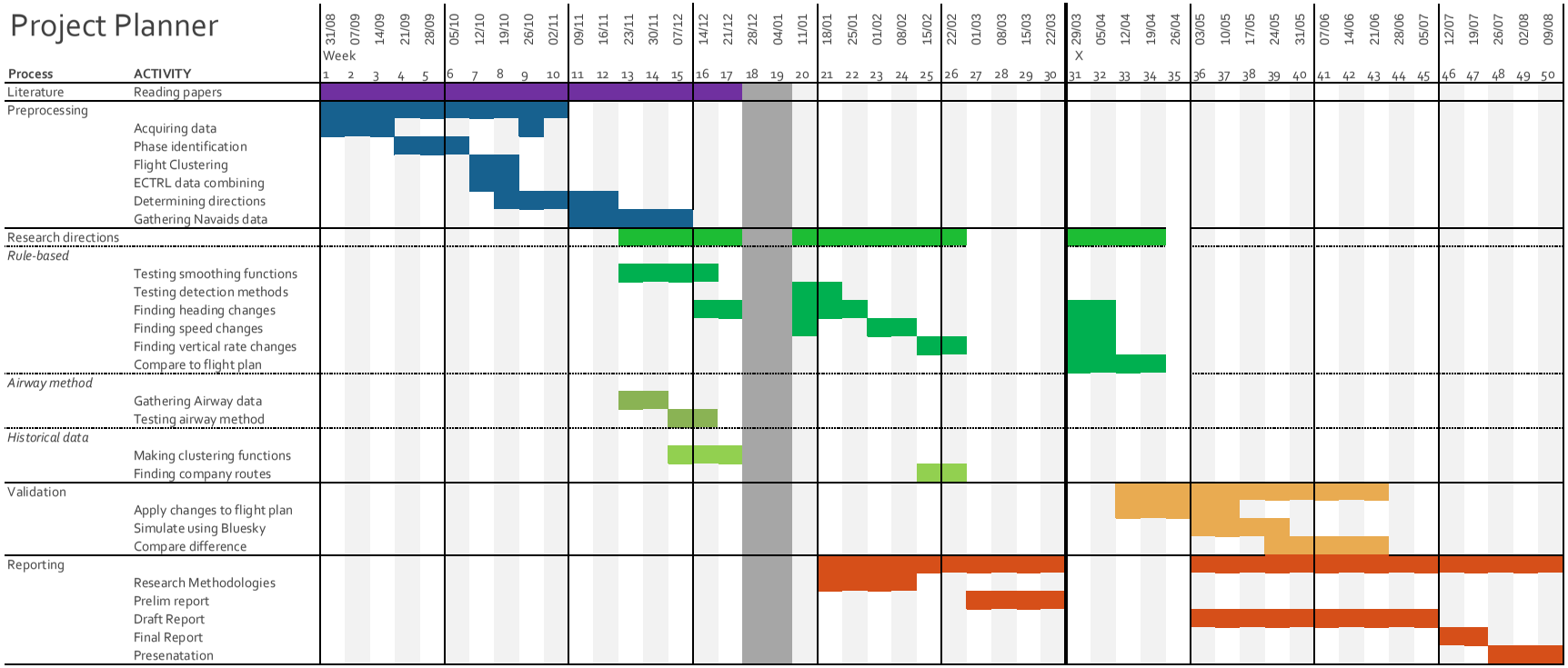


Figure C.1: Gantt chart for the thesis project