KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Technology

First cycle, 15 credits

Processing world scale air traffic data to find Near Mid-Air Collisions

LEOPOLD HERMANSSON

Stockholm, Sweden 2023

Processing world scale air traffic data to find Near Mid-Air Collisions

Leopold Hermansson, leopoldh@kth.se

Tuesday 23rd May, 2023

Supervised by:

Corentin Perret-Gentil
Daedalean AG
cpg@daedalean.ai

Tim Taubner
Daedalean AG
tt@daedalean.ai

Prof. Anders Forsgren
KTH OPTSYS
andersf@kth.se

Abstract

In order to increase the safety of all air travel, technologies that continue to augment the pilot's ability to avoid collisions and stay clear of danger are needed. But, before these can be certified and deployed, their performance and potential failure cases have to be understood. This requires evaluating a model of the system on simulated encounters, consisting of different trajectories that should replicate the real world.

This is commonly done using a statistical encounter model, which produces large amounts of data but relies on the accuracy of the statistical model, thus limited in its ability to produce realistic data. The goal with this project is to create an encounter dataset of real trajectories that would provide an alternative to encounter models.

This is done using an ADS-B dataset from The OpenSky Network (provided by Daedalean AI), consisting of 226 billion air traffic data points from 2019. First, a solution to efficiently query and reconstruct trajectories from the dataset is designed and implemented. Using it, a NMAC (Near Mid-Air Collision) dataset is created to demonstrate the viability of ADS-B as a source for creating an encounter dataset, and to prove the capabilities of the designed solution.

Keywords: Air traffic safety; Detect and Avoid; ADS-B

Acknowledgements: I would like to thank my supervisors at Daedalean, Corentin Perret-Gentil and Tim Taubner, for the resources needed to make this project possible and their guidance throughout the process. Also, thanking my supervisor from the KTH Optimization and Systems Theory department, Anders Forsgren, for all the feedback and support.

Contents

Definitions

General Acronyms

SAA
See And Avoid, Aircraft mode of operation where a pilot sees and avoid intruders

VTD
Visual Traffic Detection, system that detects traffic using a camera

DAA
Detect And Avoid, Aircraft mode of operation where a system detects and avoid intruders

ADS-B
Automatic Dependent Surveillance-Broadcast, aircraft surveillance technology based on the aircraft packet broadcast

ICAO
International Civil Aviation Organization

ICAO 24-bit ID
Unique aircraft identifier used for ADS-B broadcast, e.g. 0x407E61

Safety Metrics

HMD
Horizontal Miss Distance

VMD
Vertical Miss Distance

Position (r)
Relative position

Time until collision (τ)
$r^2/(r\dot{r})$ or distance divided by closing speed

Modified distance (DMOD)

Modified time until collision (τ_{mod})
An alternative of τ and defined as $(r^2 - \text{DMOD}^2)/(r\dot{r})$

NMAC

Near Mid-Air Collision, is an encounter with high risk, alternatively defined as a vertical separation of less than 100 ft and horizontal separation of less than 500 ft (in this paper NMAC is used to refer to the latter if not otherwise stated)

LoWC

Loss of Well Clear, a situation where the pilot no longer has safe separation to surrounding traffic and action to mitigate the risk could be required, alternatively the FAA has proposed a definition for UAVs, HMD of less than 4000 ft and VMD of less than 450 ft and τ_{mod} of less than 35 s with a DMOD = 4000 ft (in this paper LoWC is used to refer to the latter if not otherwise stated)

Chapter 1

Introduction

When flying in VFR (Visual Flight Rules), the pilot is responsible for remaining well clear of surrounding air traffic. This type of flight is referred to as SAA (See and Avoid), where the pilot looks out for traffic and avoids it if necessary. However, due to several humans factors this is not entirely reliable [Hob91].

Combined with the fact that air traffic is expected to continue growing, the number of potential collision will also, thus new solutions have to be developed in order to meet ever-increasing safety requirements. DAA (Detect and Avoid) systems is one such solution to augment the pilot's ability to detect other aircraft, and in the future enable autonomous flight.

The detection part of a DAA system is responsible for gathering information about surrounding aircraft. This information is then passed to the avoidance system, responsible for keeping safe separation to other aircraft.

There are multiple technologies that can be implemented for detection, one of which from Daedalean AG, who are currently developing and preparing for certification of an AI/ML-based VTD (Visual Traffic Detection) system. But, regardless of the detection system, simulation on a large amount of varied encounters allows evaluation of the system and proving that it is safe.

Figure 1.1: Daedalean AI Visual Traffic detection system.

In order to provide accurate metrics and results, the simulated encounters must reflect the environment that system is used in. Currently, this is done using encounter modelling, firstly a statistical model is fitted to a large dataset of aircraft trajectories. The model can then be used to create trajectories with statistically similar behavior as the real data used in training. Now, a set of trajectories can be combined to create encounters.

For example, as part of evaluating the upgrade of TCAS II (Traffic Collision Avoidance System)

to version 7.1, the version currently mandated by ICAO [EAS15], an encounter model was used to generate encounters [EGK09]. Specifically, a Bayesian Network was trained using real aircraft trajectories from Radar.

Even though statistical models have been the main source for encounters in air traffic safety research, it is impossible for them to perfectly replicate the behavior of real air traffic. This is due to the fact that there is only a limited amount of data, thus not all priors can be taken into account, and it is not always possible to model every conditional relationship.

In the case of the encounter model used to evaluate TCAS II, created by The MIT Lincoln Lab [Koc+08], a number of parameters had to be tested to find the optimal structure given the amount of data available. A model with too few parameters can only hold so much information and thus under fitted, while too many parameters on the other hand lead to an inaccurate and possible overfitted model.

If real encounters are used instead, it is possible to mitigate these issues. Especially, it is expected that metrics from the simulation are closer to what is expected when deployed. Another benefit of having access to a dataset of real encounters, is that the scope can be reduced to specific a region or time, providing greater insight in the expected performance in specific traffic distributions.

1.1 Air traffic data sources

This project will look at an alternative source, namely ADS-B (Automatic Dependent Surveillance-Broadcast), a public source of air traffic data that potentially allows more control and transparency in research. Since its creation in 1996 by the MIT Lincoln Laboratory for Air Traffic Control, it has since then become a key technology used by aircraft and ground stations for air traffic surveillance.

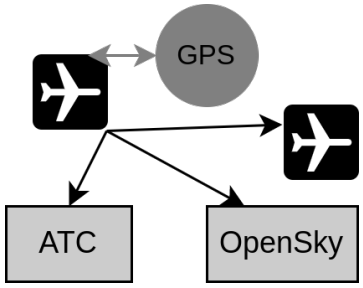

Figure 1.2: ADS-B.

The design of ADS-B allows aircraft-to-aircraft communication of position and operational status; see Figure 1.2. This means that even without ATC, pilots get increased situational awareness. This has been shown to reduce the accident rate for aircraft in GA (General Aviation) by 40–60% [HK19].

ADS-B can be split into two subsystems, referred to as *ADS-B In* and *ADS-B Out*, that can be used independently; Table 1.1.

ADS-B Type	Function	Uses
Out	Broadcast position and operational status	Aircraft and certain airport ground vehicle
In	Receive position and operational status of nearby sender	Aircraft, ATC and other users (OpenSky, FlightRadar24, ...)

Table 1.1: ADS-B In and ADS-B Out.

ADS-B Out

This uses on-board GNSS (Global Navigation Satellite System) and an ADS-B transmitter to broadcast position, along with some operational status. Importantly this broadcast is done continuously at 1 Hz as long as the system is active, regardless of presence of any receivers in range. As the frequencies used by ADS-B require permission to broadcast on, the use of ADS-B Out is limited to aircraft and airport ground vehicles.

ADS-B In

Unlike ADS-B Out, ADS-B In does not require permission to use, as it only captures and parses ADS-B packets.

In-flight ADS-B In is used to enhance situational awareness by displaying nearby aircraft to the pilot, assisting the pilot in SAA, for example as using display as in Figure 1.3. On the ground a system that only utilizes ADS-B In can be used for air traffic surveillance and capture, example of such are ATC (Air Traffic Control) and live aircraft tracking websites such as OpenSky and FlightRadar24.

Figure 1.3: Avidyne ADS-B advisory system [Avidyne].

Protocol

ADS-B broadcasts one packet for each field of data. These packets consist of the ICAO 24-bit address identifiers (unique to every aircraft), and a 56-bit message with the first 5 bits being the type code of the message. Using Table 1.2, the type code of the message can be identified and then decoded.

Type Code	Data frame content	Unit/Datatype
1–4	Aircraft identification	Callsign
5–8	Surface position	Degrees latitude and longitude when on-ground
9–18	Airborne position (w/Baro Altitude)	Degrees latitude and longitude and feet over mean sea level
19	Airborne velocities	Vertical rate in feet per minute, heading in degrees and speed in knots
20–22	Airborne position (w/GNSS Height)	Degrees latitude and longitude and meter over mean sea level
23–27	Reserved	
28	Aircraft status	Emergency/priority status and ACAS RA broadcast
29	Target state and status information	Upcoming turn and altitude change
31	Operational status	Status, e.g. enroute or landing

Table 1.2: ADS-B Type Code and content, [ADSB].

One special feature of ADS-B is that positional fields utilize an encoding know as CPR (Compact Position Reporting), with the position sent in alternating frames of reference. Using two packets received close to each other, it is possible to achieve a precision of 5 m with only 35 bits instead of 45 bits [Dut+17]. This is useful, as the amount of data that can be sent on a specific frequency is limited, thus the larger the packets are, the higher the risk of interference between multiple broadcasts.

ADS-B security issues

According to [WZD20], one major problem of ADS-B is that the protocol implementation makes it vulnerable to many attacks, which often require only limited knowledge and equipment. The simplest of such is *spoofing*, which allows an attacker to transmit any data it wants.

This makes ADS-B inappropriate as the only data source in safety-critical systems because there exists no built-in protection against such attacks, rendering the data received from ADS-B untrustworthy by itself. This is also why a DAA system for autonomous flight that only relies on ADS-B is not feasible.

It should be added that a spoofing attack is only possible for targeted scenarios, where the position of single aircraft can be faked. Moreover, historic ADS-B data is commonly used as a source in aviation investigations, indicating that the data can generally be regarded as trustworthy.

Other air traffic data sources

Until recently, the main sources for air traffic data have been the radar technologies PSR and SSR. These work by having one or more ground station that calculate the position of the aircraft.

PSR (Primary Surveillance Radar)

This works by emitting a radio wave and then recording the incoming radio waves while rotating the antenna. The distance can then be calculated using the time difference the speed of the radio wave. Speed and features like size can also be derived from incoming radio wave.

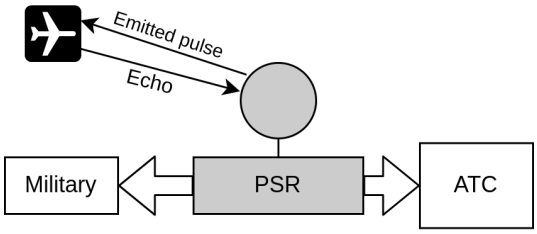

Figure 1.4: Primary Surveillance Radar.

The main advantage of PSR is that it does not require cooperation from the aircraft, thus making it suitable for military use. However, due to the cost and ground infrastructure needed, it is not always possible to install. Moreover, obstruction such as weather (cloud, precipitation, ...), buildings and mountains can create noise, making detection more difficult.

SSR (Secondary Surveillance Radar)

This works similarly to PSR but instead of recording the echo created when the emitted radio wave bounces of the aircraft, it interrogates the required transponder on-board the aircraft, which then transmits time, an identifier and the altitude. Using the time difference from when the transponder replied to when it was received the distance can be calculated the same as for PSR.

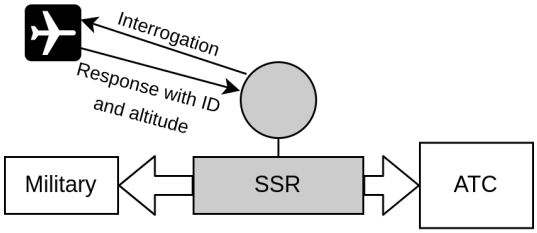

Figure 1.5: Secondary Surveillance Radar.

Even though SSR does not have the ability to detect non-cooperative aircraft, it is in many cases more useful than PSR. Because of the design, it can use cheaper and less powerful equipment, which has allowed its installation in all commercial aircraft.

When first introduced, SSR was installed on the ground and used by ATC, the ability for it to interrogate aircraft meant that it was possible to automatically identify the detected aircraft, which is helpful when giving instructions.

One DAA system that is currently used and even mandated on all commercial aircraft in the U.S. and European airspace is TCAS (Traffic Collision Avoidance System). The detection component is done using on-board mode-s transponders, a type of SSR, that interrogate all aircraft within a certain radius. In the case that the avoidance system predicts a collision, the pilot is alerted and instructed on a preventive maneuvers that has been decided on with the other TCAS equipped aircraft.

Method	Ground equipment	On-board equipment
PSR	Yes	No
SSR	Depending on application	Yes
ADS-B	No	Yes

Table 1.3: Equipment requirements

Source comparison

ADS-B differs from PSR, as the later can detect aircraft and calculate the position without any equipment or cooperation required by the tracked object. On the other hand ADS-B is considered more accurate because it utilizes GPS, and ADS-B data is updated every second meanwhile the radar data from the US military updates every 4.5 or 12 second depending on range [Koc+10].

It is also more robust in areas where the line of sight can be interrupted, it is possibly to "Fly under the Radar" because data from lower altitude contain high amount of noise, and it is not possible to detect an aircraft if is not in direct line of sight. This has made ADS-B the only viable data source for low altitude air traffic [WUW19].

The main issue with using historic ADS-B is that not all aircraft, especially in General Aviation, have the necessary ADS-B Out equipment, thus there is air traffic missing. According to a report published by the FAA [Adm19], only 51% of GA aircraft carry ADS-B Out as of October 1, 2019. Note that the FAA [FAAADSB] and EASA [EASAADSB] introduced requirements in 2020 to mandate the use of ADS-B transponder in many airspaces, and further mandates are expected.

Chapter 2

Dataset: Global 2019 air traffic data

2.1 The OpenSky Network

The ADS-B dataset this project will analyze to find encounters and traffic distribution, was acquired by Daedalean AG from The OpenSky Network. The OpenSky Network is "*A Large-scale ADS-B Sensor Network for Research*" [Sch+14]. It has been collecting trillions of air traffic messages for research since 2013. These messages include ADS-B data that has been gathered using a network of ground stations. By using low-cost ADS-B ground station and volunteers, the network now has more than 5000 receivers, making it the largest of its kind [OpenSkyFacts].

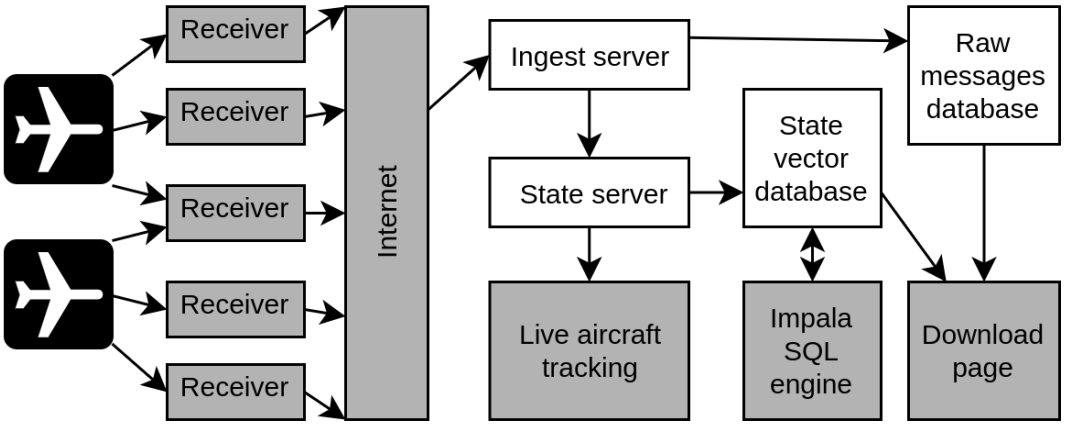

Figure 2.1: Model of The OpenSky Network infrastructure.

Following the model in Figure 2.1, the receivers bought and hosted by volunteers first relay the ADS-B messages over the internet to a central ingest server. The messages are stored along with information about where the receiving ground station was and the signal strength of the received package. This data can later be accessed as raw data which can be used for example to evaluate Multilateration algorithm [Sch+14] (a method of deriving origin of a signal based on the measured signal strength from the receivers, it can be used to increase the confidence ADS-B data).

The messages are then passed to a server that keeps a state vector (defined in Section 2.2) for every aircraft. Using the new messages, these state vector are kept up to date, and if no new information has been received for more than fifteen minutes it is deleted.

Two things then happen to the state vectors: they are pushed to user through an API, and every second a copy of the state vector is stored. The historical state vectors can either be accessed for research using an Impala shell or retrieved directly.

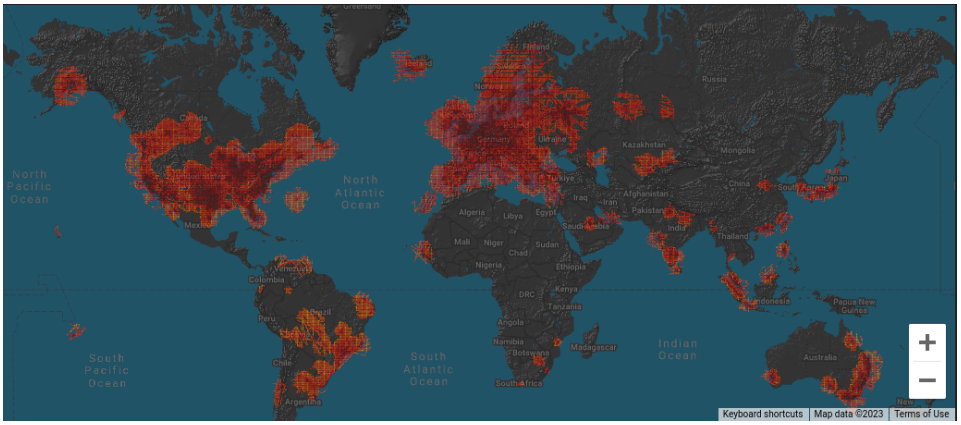

Figure 2.2: OpenSky coverage in 2019, shows captured data, which is the intersection of ADS-B equipped traffic and receiver coverage, sourced from [OpenSkyFacts].

Figure 2.3: Visualization of commercial aircraft from the European Space Agency.

By utilizing volunteers to maintain and house receivers, it is possible for The OpenSky Network to cover almost any area. Figure 2.2 shows a heat map of where data has been received in 2019, and there are mainly two things to note. The regions with the best coverage are the US and Europe. There is some coverage of South America and Australia but almost no coverage of Asia and Africa.

By comparing with Figure 2.3 that shows the global air traffic, it is possible to estimate what areas The OpenSky Networks does not cover, and it is possible to see that over land, The OpenSky Network coverage seems to mostly match the total traffic. However, outside the

US and Europe, only the regions surrounding major cities seem to have coverage. Here the population density is higher, internet and electricity are also more likely available, both of which are restriction for where ground stations can be installed.

2.2 Dataset content

The 2019 dataset that was made available for this project by Daedalean AG, consists of approximately 8700 Parquet files, a file format developed by Apache for storing data [ApacheParquet]. Every file holds all data from an hour. With an average file size of 500 MB, this already represents 4.3 TB of data. As the Parquet files were compressed at a rate of around 8, the actual uncompressed size is about 35 TB. For reference, the table size limit in one of the most popular databases, PostgreSQL, is 32 TB [PSQLLIM].

Field	Description	Example
time	POSIX time of generation	1480760792
icao24	Unique ICAO 24-bit address	a0d724
lat	Latitude	37.89 deg
lon	Longitude	−88.93 deg
velocity	Ground speed	190.85
heading	Heading in degrees from north	265.8 deg
vertrate	Vertical rate	0
callsign	Callsign if broadcasted	UPS858
onground	Aircraft grounded	false
alert	ATC flag	false
spi	Special Purpose Identification	false
squawk	ATC assigned identifier	7775
baroaltitude	Barometric altitude	9144
geoaltitude	GNSS altitude	9342.12
lastposupdat	Time when position was last updated	1480760704.359
lastcontact	Time when contact was last made	1480760709.997
hour	POSIX hour	1480759200

Table 2.1: OpenSky state vector description [OpenSkyData]

The data in each file was structured as a list of state vectors for each aircraft. State vectors are an abstraction of the raw ADS-B packet that holds relevant values as seen Table 2.1. These are generated every second up until 300 seconds after the last message. Note that the data in the state vectors are the last known values that were received. Thus, it is possible to use a custom stable interpolation and filtering method to estimate position.

2.3 Review of previous work

This section will discuss our interpretation of previous systems to handle the OpenSky dataset based on published work and for The OpenSky Network, information available on their website [OpenSkyData].

The OpenSky Network Hadoop cluster

Due to the large amount of data stored by OpenSky, a multi-node cluster is used. This allows computation to be done in parallel and for data to be distributed across multiple instances, increasing performance and redundancy.

The framework used for the cluster is Apache Hadoop [ApacheHadoop], which is designed for processing large amounts of data across a cluster. It serves two purposes:

1. It creates a distributed file system that supports redundancy and efficient storage of data
2. A distributed computing engine, allowing work to be distributed between the nodes and utilizing the distributed file system efficiently

In order to make the cluster usable, Apache Impala [ApacheImpala], a SQL query engine that utilizes the underlying Hadoop cluster to run queries is installed. It uses a method known as MapReduce that works by splitting the query work, so that every node in the cluster processes parts of the data and then redistributes the results from each node.

This design based on a Hadoop cluster, allows for extremely flexible usage of the underlying data while also giving redundancy and possibility to scale the amount of allocated compute as needed. Because the OpenSky Network expects multiple simultaneous users to work with the data, the overhead of running the cluster is offset due the average utilization being high, increasing the efficiency of the overall system.

But, the limitation is that it does not utilize the structure of the data and queries beyond the time component. Instead, the aim is much more on flexibility and robustness over efficiency. As of the current implementation, the only optimization used is to utilize the fact that the data is split by the hour; thus if a time range is supplied, Apache Impala is able to limit the work to only the necessary files.

MIT Lincoln Labs

The MIT Lincoln Laboratory for Air Traffic Control, is a research laboratory that has been conducting research with the FAA (Federal Aviation Administration) within the field of air traffic safety for the past 50 years. Noteworthy research by the MIT Lincoln Labs includes creating ADS-B (Section 1.1) in 1996, and in 2007 creating an airspace encounter model that has been used to evaluate improvements to TCAS (Chapter 1) and development of future avoidance systems [MITLL].

Their approach to storing and processing an ADS-B traffic dataset is discussed in [Wei+20]. They used a partial dataset that the OpenSky Network makes available to anyone, containing data for only Mondays. Specifically, the dataset used included 85 Mondays across 3 years, and they chose to only store US traffic during ingestion.

One of the uses described was to train statistical aircraft models for multiple different aircraft classes and sizes. This only requires reading all the data for a specific aircraft type, thus they were interested in organizing to optimize reading from a specified aircraft type.

The air traffic data is stored on a distributed file system attached to their cluster, allowing the data to be accessed by multiple compute instances and allowing for storage redundancy. In the file system, the observations are then stored in CSV (Comma Separated Values) files, and organized with the help of a registry to retrieve aircraft type, make and model, based on these fields a four layer folder structure is used. Later data can be retrieved used the format:

```
/{year}/{aircraft type}/{range of seats}/{range of ID}/{date and ID}.csv

Example: /2020/Rotorcraft/Seats_001_010/A00C12_A00D20/2020-03-16_05_A00CDE.csv
```

This design allows fetching of all tracks matching the keys above efficiently (this is what the computations of aircraft trajectory models use). However, to recreate traffic in a region, this method would require going through every file. Therefore, this yields little to no improvement over using the state vector files, as in both cases every file has to be read.

Furthermore, the dataset processed only includes 85 days in contrast to a whole year in our dataset. The authors also choose to limit the storage to continental US data only which is about 40% of the total data (assuming that the distribution is similar to what was observed in our dataset). Thus, we have close to ten times more data. Also, considering the fact that we want to rely on less compute power then the 512 core cluster used by the MIT Lincoln Laboratory, the methods described in this paper would not be feasible for our project.

One takeaway that we got from their solution, is that when storing one aircraft in each file the authors found that the files were too small (less than 1 MB in size). This resulted in an IOPS and network bottleneck during high load. This is due to a large portion of the time needed to read a file is spent on the initial access.

Their solution was to replace the lowest folder with ZIP archives, the result is that all data for a range of aircraft only requires fetching one file. Note that this worked for their use case, as the application is expected to use most of the data in the archive it reads.

2.4 Our solution

As none of the previous solution have the features we require, part of this project included the design, implementation and testing of a new solution to make the dataset accessible for analysis and encounter simulation. Specifically, it should fulfill the requirements in Table 2.2 efficiently, and because none of the previously discussed solutions indexed the data based on position, they can only do Requirements 2.2 somewhat efficiently.

#	Purpose	Description
1.1	Analysis	Retrieve statically correct sample of the traffic in a specified region and time range
1.2	Analysis	Compute macro metrics on various parameters
2.1	Encounter simulation	Retrieve all traffic in a specified region and time range
2.2	Encounter simulation	Retrieve traffic for an aircraft in a specified time range

Table 2.2: Requirements for storage solution.

Database

The first component of the solution is a database, allowing to execute queries efficiently. In order to support fast geographic queries, it has to implement indexing and support queries based on location. This is done by storing the data in a data structure such as a KD-tree, that allows for multidimensional range queries in $O(\sqrt{\text{total points}} + \text{retrieved points})$ time.

Two popular databases that have geographic indices are MongoDB and PostgreSQL with the PostGIS extension. As we did not require the flexibility provided by using a document-based database like MongoDB, we opted for PostGIS as it generally is faster.

The PostGIS extension also adds multiple functions that make working with geographic data easier, such as converting between different coordinate formats and spatial reference systems, as well as calculated distances. For our geographic queries, the `ST_Within` function is utilized, allowing retrieval of all the observations within a certain radius of a point. This is efficient as it utilizes the spatial index.

Sampling data

After ingesting just one of the 8700 files into the database, it was already 10 GB in size. Extrapolating from this number it would have been around 87 TB after all data was ingested. Thus, it would not have been possible store all the data in a database, especially as when it grows larger, one issue is that the data no longer fits in memory, thus the memory has to be flushed. Making the queries orders of magnitude slower and possibly resulting in OOM (Out Of Memory) crashes.

This meant that only parts of the entire dataset could be stored in the database. Utilizing the fact that aircraft have a max speed: sampling observations at some interval, the ones that are not sampled are expected to be close by.

For example, if we have sampled the data such that the maximum distance an aircraft can travel between each sampling is d km. Then we can be sure that a query with a d km margin would retrieve at least one observation from each aircraft, as if the data was not sampled.

The sampling rate was chosen as one sample per 10 minutes. Thus, assuming that we are interested in aircraft flying under 10′000 ft, as this is typically the ceiling for VFR (Visual Flight Rule, where DAA systems are supposed to be used), aircraft are only allowed to travel at an airspeed of 250 knots [14 CFR 91.117], and a maximum wind speed of 50 knots. The maximum ground speed is 300 knots, thus traveling at most 92 km per ten minutes. From our sampling parameters and expected traffic, adding a margin of 100 km and extending the time range by ten minutes, results in no missed observations.

The resulting database has $1.30 \cdot 10^8$ entries and is only 22 GB in total, about 4000 times smaller than a database containing the complete dataset. After adding indices on time, identifier and position, it is about 50 GB.

At this size, it is possible to use consumer hardware to run queries on the whole dataset in a couple of minutes. From our testing, the choice of storage type was the most important for performance, the difference between having an SSD versus an HDD was up to 100 times difference when the database index was not cached data and 10 times faster when cached.

Object Category Table

In addition to the time and position available in the dataset, we also want to be able to query on aircraft category. As this information is not available in the OpenSky dataset, a derived metadata table has been added to the database.

- Fixed-Wing: Smaller planes mainly used in GA (General-Aviation)
- Airliner: Larger planes mainly used for Commercial-Aviation
- Rotorcraft
- Unknown

This table was created by Daedalean AG, by combining multiple existing aircraft ICAO ID database and then using an algorithm that takes the manufacture and model name to predict an aircraft category. Thus mapping ICAO 24-bit ID to an object category.

By joining the ICAO 24-bit ID column in the traffic table and metadata table, traffic from only specified aircraft category can be retrieved.

Data layer

Looking back at the requirements in Table 2.2. 1.1 and 1.2 are fulfilled using only the database, while 2.1 and 2.2 are only partially fulfilled. This requires the solution to be able to retrieve entire flight trajectories to the precision of the original dataset, which is not present in the database due to the sampling done in Section 2.4.

In order to allow for the retrieval of entire trajectories, an extra storage layer is added. For this a key-value store, a service that allows storing and retrieving data using a key, was used.

Objects that contain the state vectors in between each sampling are stored compressed in the object store, and the key is added to the corresponding entry in the traffic database, see Figure 2.4. Thus, when querying the database, a set of keys are also returned that can be used to reconstruct of the entire trajectory.

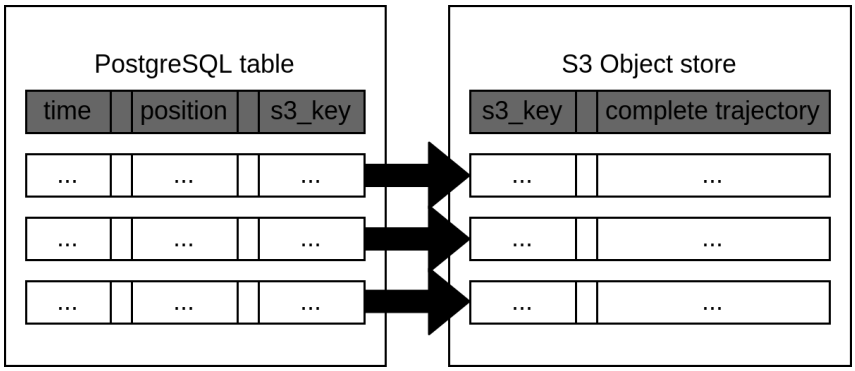

Figure 2.4: PostgreSQL S3 relation.

For the key-value store, we used an S3 compatible one, an API that allows interaction over HTTP [S3DOCS]. It was chosen because the implementation is agnostic to the service that stores the objects, thus it can be easily be changed without modifying the user application. HTTP was chosen because it is generally an efficient and well-supported protocol to retrieve data. In our implementation, we used Ceph, a distributed and redundant storage application that can be accessed through an S3 API [CEPHDOCS].

While the solution now meets our requirements in Table 2.2, we ran into similar issues as described in the paper from the MIT Lincoln Laboratory [Wei+20]. When storing the trajectories 10 minutes at a time, the created objects were very small (10 KB), and thus not utilizing the storage server well because of HTTP connection overhead. This also resulted in server errors and dropped connections because of the amount connections needed.

To solve this, the amount of data in each stored object had to be increased so that fewer objects need to be fetched, there is a trade-off here as the larger the object are the less precision we have when retrieving. In other words, it is necessary to find a way to group the data such that: the objects are large enough, while minimizing the expected amount of unnecessary data. When benchmarking the object store, we found that the overhead for 10 MB objects was insignificant because, as it became bandwidth limited.

We chose to split the data in each file of the dataset by region. As the main purpose of this system is to reconstruct all traffic, it can be assumed that nearby aircraft are very likely also included in the result of a query. The splitting was done using a simple algorithm (implementation in Figure 2.5):

1. Sort the data by latitude
2. Split the data into equal groups
3. Sort the data in each group by longitude
4. Split the data in each group into equal groups

```
fn split_data(data) {
    // Calculate necessary number of longitude and latitude splits
    n_lat, n_long = calculate_splits(data);

    // Sort data by latitude and split into groups
    data.sort_on_latitude();
    lat_groups = data.split_into(n_lat);

    results = vec![];
    // For each group, sort by longitude and split into groups
    for lat_group in lat_groups {
        lat_group.sort_on_longitude();
        groups = lat_group.split_into(n_long);
        results.extend(groups);
    }
    results
}
```

Figure 2.5: Splitting algorithm.

In Figure 2.6, the result is that the data from each file is split into geographical groups that are approximately 10 MB in size. This also means that multiple database entries will have the same key, thus in order to benefit from the grouping the duplicate keys have to simply be ignored when downloading.

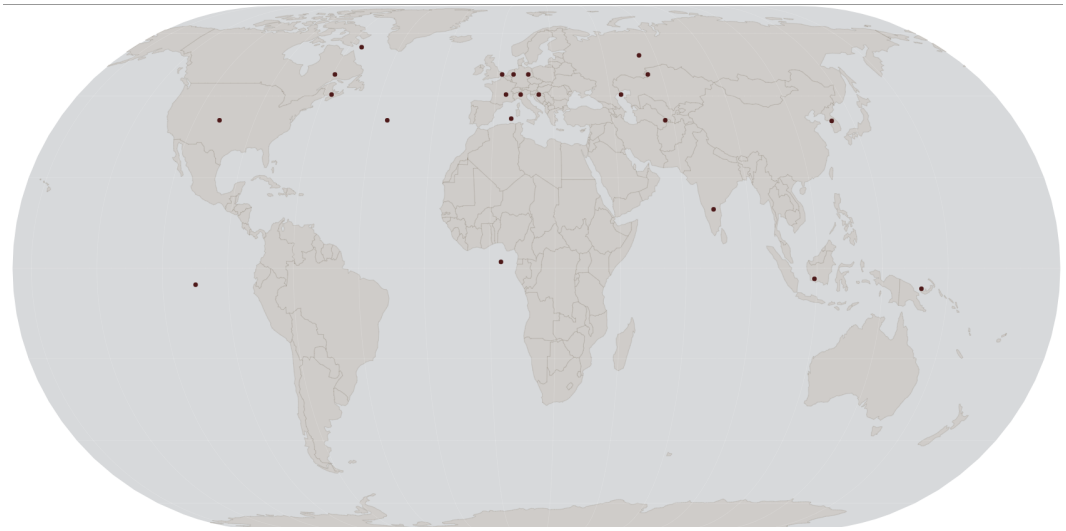

Figure 2.6: Center of each group after splitting the data by longitude and latitude.

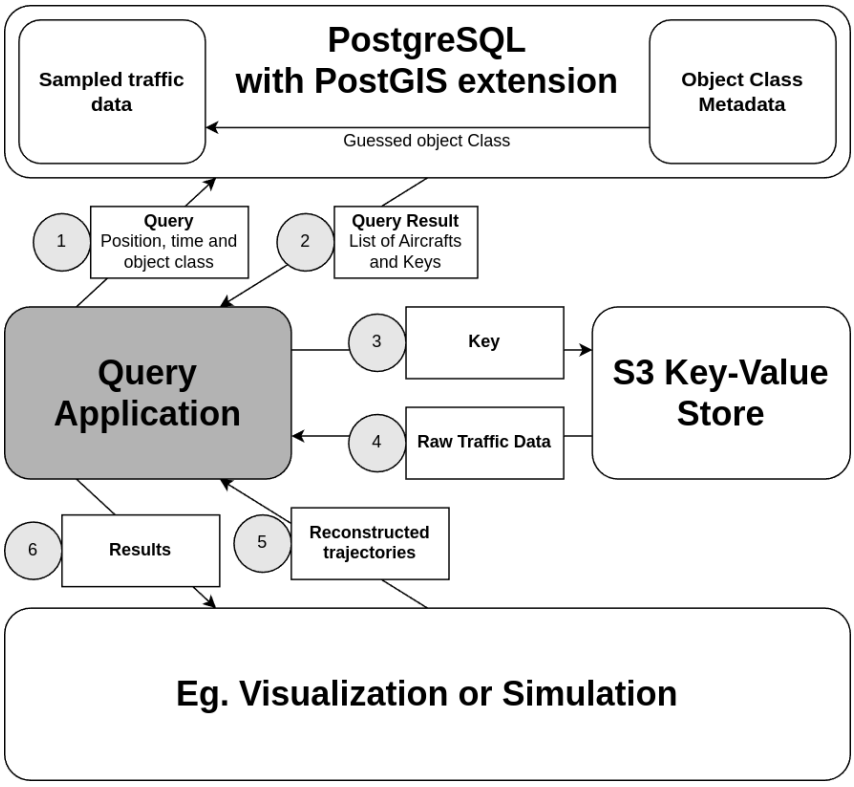

Figure 2.7: Mockup of workflow.

The diagram in Figure 2.7 shows how a complete system looks like. The user starts by interacting with the Postgres database to retrieve aircraft and keys. For statistical analysis such as distribution of traffic, this might be enough. If complete trajectories are required, the user fetches it from the S3 Key-Value Store using the keys it had previously received from the database. Lastly, the data belonging to the time and region can be reconstructed. This can be used for visualization or simulation.

Chapter 3

Analysis: Density and type distribution of traffic

ADS-B usage

In the 2019 dataset, there were 158, 848 unique ICAO 24-bit ID that have broadcasted position, of those 103, 957 were registered in the United-States based on the ICAO 24-bit ID range it uses, which is `0xa00000-0xafffff` [ICAOannex10]. In total, there were approximately 200, 000 aircraft registered in the US in 2019 [Statista], thus it can be concluded that at least 50% of the planes in the US carry an ADS-B transponder able to broadcast position. It should be noted that this is a conservative figure as there are multiple factors that would result in an aircraft not being present in the dataset. For example, there might be aircraft that have not flown during the year or at least not flown in any area covered by the OpenSky network.

Comparing this number with the reported figure for ADS-B usage 60% usage, it seems reasonable to assume that the aircraft coverage in the dataset is somewhere between 50–60%. Thus, any findings from this dataset have to take into consideration that there are aircraft missing.

ADS-B usage impact

Analyzing the impact on close encounters we can quantify the impact ADS-B adoption has on usage. Define x_i as the number of close encounters by aircraft i, p as the probability of an aircraft broadcasting ADS-B. Thus, the expected number of close encounters in the dataset for aircraft i is $x_i p$ and as the probability of being in the dataset is p, the expected number of close encounters per aircraft is $p^2 x_i$. Thus, the expected total number of close encounters is $\sum_{i=1}^{n} p^2 x_i = p^2 \sum_{i=1}^{n} x_i$, or that the expected the number of close encounters in the dataset to be proportional p^2. This means that only 25–36% off the close encounters will be present in the dataset assuming that the probability of broadcasting ADS-B is 50–60%.

Remark. In the analysis above, a complete geographic coverage is assumed, thus it is only valid in region were that is true, and the probability off broadcasting ADS-B is assumed to be the same for every plane, thus independent of the expected number off encounters. However, as mentioned previously a plane that utilizes ADS-B has better situational awareness and therefore a lower probability of being in an accident.

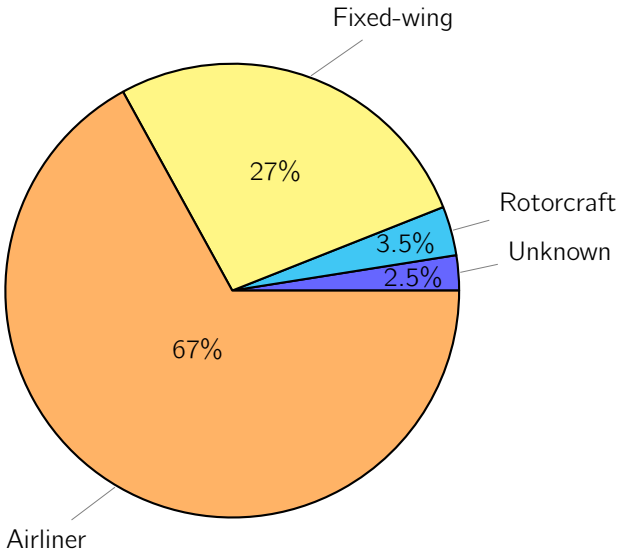

Figure 3.1: Distribution on flight data by type

For this dataset to work well, it is important that is has sufficient geographical coverage and traffic distribution. Figure 3.1 shows the class distribution in the dataset using the guessed classes. As can be expected, the largest class is airliner as the adoption of ADS-B is highest, and it accounts for a major part of air traffic. The next class is fixed wing, which accounts for 27% of the data and lastly, 3.5% of the data is from rotorcraft.

Chapter 4

Safety state definition

4.1 Near Mid-Air Collision

Due to strict regulation and the generally high level of safety in air traffic, collision do not happen frequently enough compared to for example road traffic. Thus, it is necessary to define a less strict event, where the risk of a collision is above acceptable amount.

An agreed upon definition does not exist, but the most common definition is the TCAS standard [ACAS]: "Two aircraft simultaneously coming within 100 feet vertically and 500 feet (0.08 NM) horizontally." .

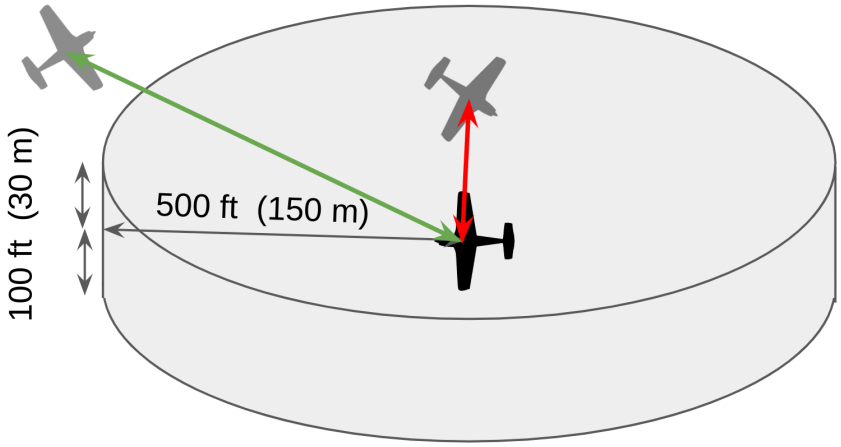

Figure 4.1: NMAC definition.

For reference, in VFR (Visual Flight Rule) two aircraft travelling towards each other at the maximum allowed speed of 250 knots will have a relative speed of 850 ft/s, thus colliding within 0.6 seconds of a NMAC.

The FAA has an alternative definition [FAANMAC1]:

"A NMAC is defined as an incident associated with the operation of an aircraft in which a possibility of collision occurs as a result of proximity of less than 500 feet to another aircraft, or a report is received from a pilot or a flight crewmember stating that a collision hazard existed between two or more aircraft."

4.2 Loss of Well Clear

Even though *Well Clear* is used multiple times in *Rules of the Air* [ICAOR], a rule book created by ICAO in order to standardize air traffic, it is never defined [MJ18], and neither is it by EASA and the FAA. Up until recently, this has not been a problem because a human pilot is able to understand it [URClearED]: *"In aviation, "Well Clear" refers to a state in which a pilot considers the situation of the aircraft to be safe in relation to the surrounding traffic."*.

However, with the development of UAV (Unmanned Aerial Vehicle), this is no longer enough because this definition is difficult to implement in such autonomous system. Instead, a metric based definition has to be developed. The requirement of a defined *Well Clear* zone: must be large enough to mitigate collision risks, and small enough to minimize operational impacts [WBE17].

One attempt to establish an analytical definition has been made by the MIT Lincoln Lab [WEF11]. They propose, that given a target conditional NMAC probability defined as Equation (4.1), a definition is created by simulating encounters.

$$P(\text{NMAC} \mid \text{LoWC}) = \text{the probability of a NMAC given a LoWC} \tag{4.1}$$

For example, if a definition in the form of an area around the aircraft is assumed, then encounters are simulated while keeping track of the probability of a NMAC for every point around the aircraft. The result is an approximation of the probability of the conditional NMAC probability, see Figure 4.2.

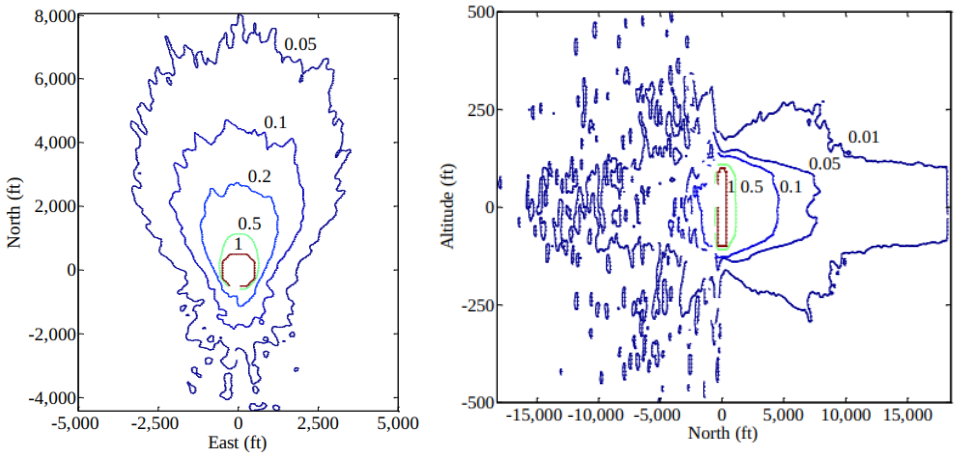

Figure 4.2: Contours of conditional NMAC risk in the horizontal and vertical plane [WEF11].

Limitations of the method and shape

The definition proposed by the authors [WEF11], assuming a conditional probability $P(\text{NMAC} \mid \text{LoWC}) = 0.05$, would be an "acorn" shape 8000 ft ahead of the aircraft, 3000 ft laterally, 3000 ft behind and 300 ft altitude. But importantly, this is assuming random pairs of sampled trajectories, thus this definition might not yield the same probability when used in the real world. It would be expected that many of these encounters could be mitigated by the pilot in the other aircraft and as stated in the paper "Encounters are modeled that represent aircraft randomly "blundering" into NMAC.".

The problem with their approach is that the intruding aircraft is never expected to mitigate in simulation, but in the real world it would be expected that the likelihood of any aircraft mitigating is higher in the event that they are head-to-head. Thus, this simulation overstates the NMAC probability when the intruding aircraft has the ownship in it field of view, or flying towards each other.

Also, there is a limitation of the simple fixed shape definition proposed above, as there is no consideration for the relative speed of the aircraft, which can be assumed to impact the risk of a NMAC at different distances. Thus, newer proposed definitions use τ_{mod} (Modified Time to Collision), a variation of τ that takes into account the non-linear risk when an aircraft is approaching and is defined as:

$$\tau_{\text{mod}} = \frac{r^2 - \text{DMOD}}{r\dot{r}}$$

At large distance τ_{mod} behaves similarly to τ, but as the separation r approaches DMOD, τ_{mod} will quickly go to zero, see Figure 4.3. This has been shown to better represent the risk of a NMAC. The FAA states that a pilot has on average a total reaction time of 12.5, and on average 6.5 seconds from the moment the pilot is aware of a collision course [FAAADV]. This means that a mitigation has to be initiated well before the NMAC zone in order to stay clear.

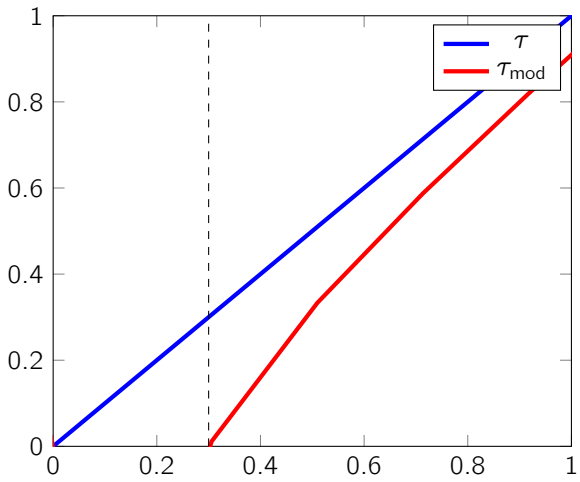

Figure 4.3: τ vs τ_{mod}, DMOD = 0.3

By modifying τ, some properties are no longer true: it no longer represents an event in the same way τ estimates when the collision will occur, and it is no longer linear [WBE17].

The definition for *Well Clear* that is used in this project as well, and currently the most used popular for UAVs [Wu+18], is defined as:

- HMD (Horizontal Miss Distance) of less than 4000 ft
- VMD (Vertical Miss Distance) of less than 450 ft
- τ_{mod} less than 35 s with mod = 4000 ft

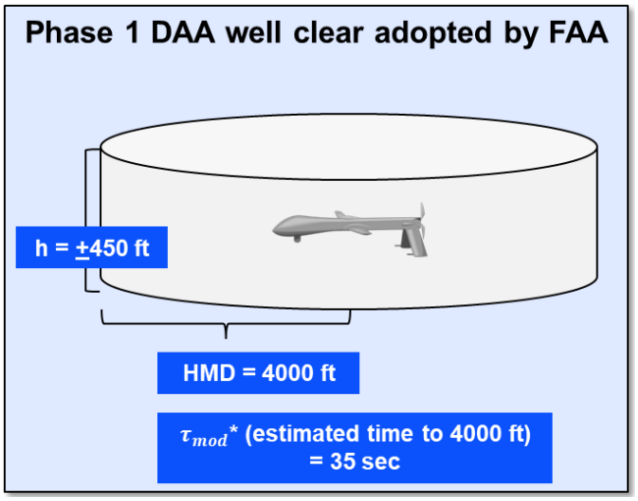

Figure 4.4: LoWC definition [DEFWC].

A simple interpretation is: in order to stay well clear, a horizontal separation of more than 4000 and vertical separation of more than 450 ft should be kept for the next $\approx$ 35 s.

Chapter 5

Finding Near Mid-Air Collisions

As discussed in the introduction, one of the goals for this project is to create a dataset of real encounters for evaluation of DAA systems. Specifically, it is interesting to find encounters that resulted in a NMAC (Near Mid-Air Collision) and LoWC (Loss of Well Clear), because a DAA (Detect And Avoid) system would have been expected to mitigate these events.

5.1 Report database

To find NMAC encounter in the dataset, we first looked at public incident databases: AVHerald [AVH], FAA [FAANMAC] and SUST [SUST]. These are typically incidents that are investigated in order to understand what caused the NMAC.

The processing of each source was done in three steps:

1. In order to retrieve all the reports, a script for each site was written. This is done by first scraping each website and saving the source locally. This was done using the Python library *selenium* [SELENIUM], it works by creating a Chrome instance that can be interacted with through code.
2. All the source files are parsed and processed into a generic format including incident time, location and aircraft identifiers.
3. The traffic database (create in Chapter 2) is queried with the parameters from the parsed incidents. If a pair of trajectories are found then it is visualized using a custom tool we created, that replays the trajectories, see Figure 5.1, allowing manual confirmation that it matches the report.

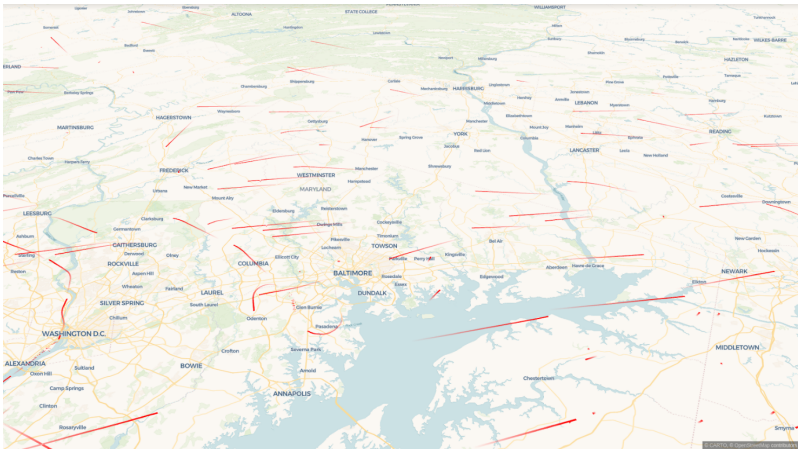

Figure 5.1: Trajectory visualization.

The encounters reported in 2019:

- AVHerald did not have any midair incidents reported.
- FAA NMAC system had 49 reported NMAC where both aircraft were reported to carry transponder and an identifier for both aircraft existed, but only 2 encounters were found in the traffic data. We expect this is due to the low quality reports.
- SUST had 13 reported incidents, of which only 1 was a NMAC and was also in the traffic data.

Most of the databases we looked at contained very few NMAC: because actual collision are quite uncommon and the procedures for reporting NMAC are not always followed. One explanation is that it relies on the pilots to voluntarily report incidents. Problematic because it is often difficult for pilots to correctly perceived for example separation, and pilots often lack motivation to submit a report [FAANMAC2]. Therefore, we concluded that using NMAC databases is not a viable solution to create a dataset of encounters.

5.2 Air traffic dataset

Instead of relying on an external source for NMAC such as NMAC database, we used the dataset introduced in Chapter 2 to find the NMAC. This was done by replaying all the air traffic from the dataset, and for each aircraft checking if any nearby aircraft is within the range of a NMAC. We decided look for NMAC between Fixed-Wing aircraft, removing:

- Airliners because the DAA systems we are expecting to evaluate are not design for them.
- Helicopters were removed because they have the ability to stay stationary making many of the definition of NMAC and LoWC inapplicable.

The work detailed in Chapter 2, allows retrieving a list of state vectors for each aircraft. The coordinates in the state vectors are converted to the ECEF (Earth-centered, Earth-fixed) coordinate system, a Cartesian coordinate system centered and fixed on the Earth. The trajectories are inserted into interpolation splines, that allows position to be sampled at arbitrary timestamps.

The method for finding NMAC is then the following:

1. With the trajectory splines, the simulation now steps through the time range at a desired rate, which we set to one second as it is the update frequency used by ADS-B (see Section 1.1).
2. At each timestep, the position of every aircraft is sampled and inserted in a data structure (we called `GridStorage`, implementation in Section 5.2) that allows retrieving within a radius efficiently.
3. For each aircraft, retrieve nearby aircraft and check whatever a NMAC has occurred.

Due to the amount of data, the simulation was split into 4 hours batches with a 10 minute overlap. By storing the result per batch, this also allows the simulation to be restarted if necessary. The total runtime for the simulation given a one gigabit connection to the object store is approximately 9 hours, but due to other simultaneous workloads it took approximately 40 hours, running on one core and 4 GB of RAM.

GridStorage

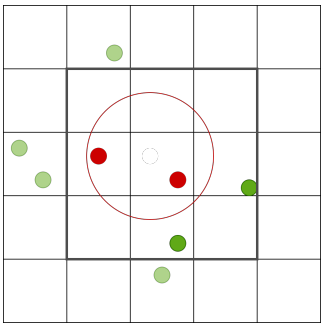

Figure 5.2: GridStorage example.

Because we are only interested in points at a predefined radius, this can be achieved by storing the points in grid with the same squares size as the predefined radius. As such, only the adjacent squares have to be checked for NMACs, example in Figure 5.2. Those can be fetched in constant time by storing the grid in a hash map. Thus, using this data structure the time complexity of each step is $O(|\text{aircraft}| \cdot |\text{neighboring aircraft}|)$, where

$$|\text{neighboring aircraft}| \approx 1 + \frac{(3r)^3}{4\pi r^3/3} \cdot E[\text{NMAC per aircraft}] \approx 1 + 6 \cdot 0 \approx 1 \tag{5.1}$$

With knowledge of the density, Equation (5.1) can be approximated to 1. Thus, the expected runtime of each timestep is $O(\text{number of aircraft})$.

Chapter 6

Analysis: Near Mid-Air Collisions dataset

Using the method above, 9481 NMAC occurrences of fixed-wing aircraft were found in the air traffic data. In order to remove most of the false detection, we tried to filter the data for what we expected as on-ground detection, and coordinated flight:

- Grounded: encounters where any of the aircraft were moving slower than 25 m/s, roughly the stall speed of a typical fixed-wing.
- Coordinated flight: if multiple NMACs occurred or the planes were in a NMAC collision range for more than 20 s.

After filtering for the possible false detection, 2610 encounters were left.

6.1 Example encounter

An example of NMAC collision that was not reported, but was present in the dataset.

Aircraft 1	Tail number: N387CS (ICAO 24-bit ID: `0xA4758E`)
Aircraft 2	Tail number: N737TX (ICAO 24-bit ID: `0xA9E6CB`)
Time	1561486913 / June 25, 2019 18:21:53 GMT
Latitude	37°38'34 N
Longitude	121°42'21 W

Figure 6.1: Trajectory at $t - 30$ s, $t - 5$ s and $t - 0$ s to the closest point.

	Altitude 1	Altitude 2	Separation
30 s	1097 m	1082 m	3147 m
5 s	1082 m	1097 m	500 m
0 s	1090 m	1105 m	118 m

6.2 Conclusions from the dataset

Using the NMAC dataset we were able to extract information about aircraft at the time that LoWC was detected.

Field Of View

One limitation of some systems is that they have a maximum operational field of view, in the case of a Visual Traffic Detection system this is limited by the cameras, similarly for human pilots. Figure 6.2 shows the distribution of the incoming angle of the intruder aircraft relative to the heading. From this plot, it seems like the majority of NMAC would have been avoidable if an air traffic detection with a FOV (Field Of View) of more than 90 deg, where 70% of the encounters came from.

Looking at the remaining 30% of encounters it seems like there is an improvement in having a wider FOV, but it is important to note that this data is only from NMAC. Thus, there exists a bias towards intruders that were not detected, and that is correlated with the position of the intruding aircraft. Therefore, we would assume this plot to have proportionally fewer aircraft outside the FOV of the pilot, if it had every encounter.

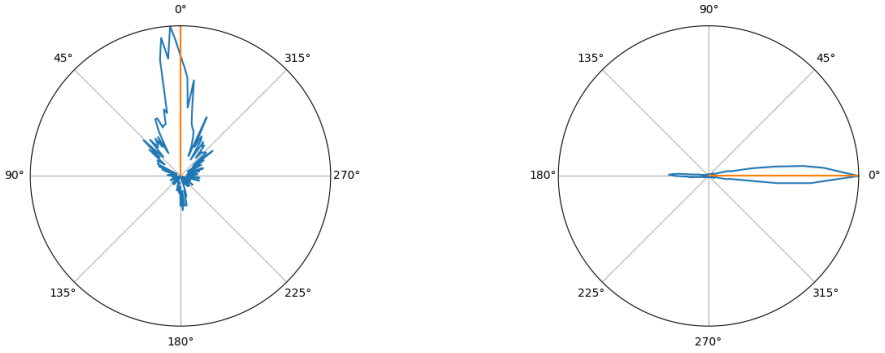

Figure 6.2: Distribution of encounter horizontal and vertical angle in respect to heading, derived from NMAC dataset at the time LoWC was detected with the intruding aircraft.

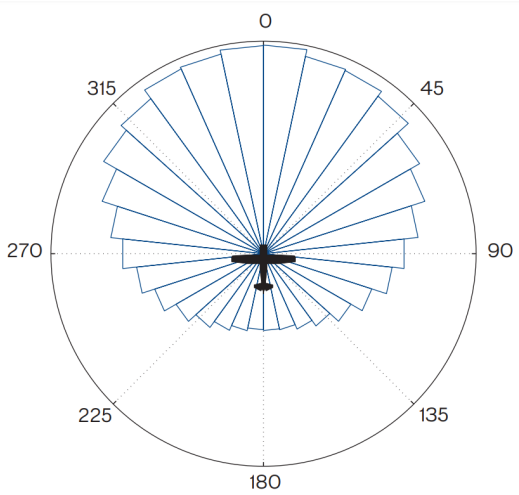

Figure 6.3: The uncorrelated encounter model generated this bearing distribution of one million simulated encounters between two aircraft when both are operating under visual flying rule (VFR) [Koc+08].

By comparing Figure 6.3 and Figure 6.2, the distribution difference between randomly generated encounters and real encounters that resulted in NMAC can be determined.

For example, the acceptable FOV would vary a lot depending on which dataset is used, the NMAC dataset shows that a large FOV is less important than what the random encounters indicate.

Distance and speed

From Figure 6.4 and Figure 6.4, it seems like LoWC happened later than we would expect as the distance was very close to DMOD in many cases. This means that τ_{mod} is possibly not very reliable, one problem with noticed is that the predicted closest point, which is necessary to calculate the miss distance, is not robust if the plane is in a maneuver.

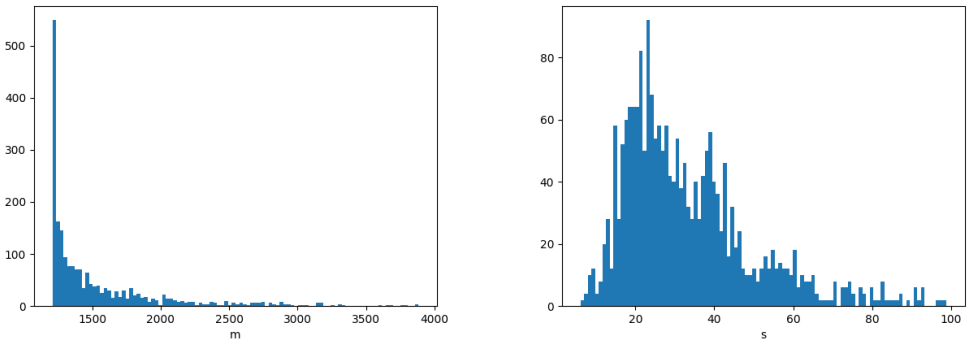

Figure 6.4: Distribution of encounter the separation and τ, derived from NMAC dataset at the time LoWC was detected with the intruding aircraft.

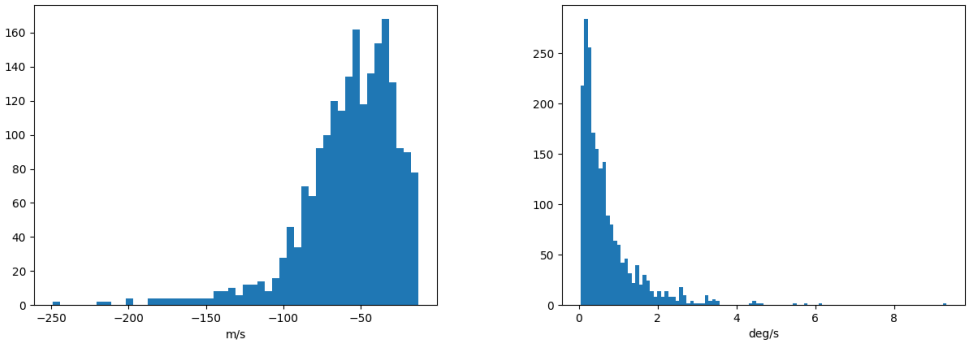

Figure 6.5: Distribution of encounter closing speed and angular velocity in respect to heading, derived from NMAC dataset at the time LoWC was detected with the intruding aircraft.

Chapter 7

Conclusion and future work

7.1 Conclusion

By comparing ADS-B (Automatic Dependent Surveillance–Broadcast) to alternative air traffic sources, we showed its advantages and drawbacks for our use. Importantly, ADS-B is more broadly available than Radar data that has previously been used. This is mainly because ADS-B can be captured cheaply and reliably, whereas Radar is almost exclusively available for military purposes.

We have shown that it is possible to efficiently work with historic ADS-B data at a large scale. This was achieved by designing and implementing a solution that allows a 35 TB dataset to be queried for both large scale distribution analysis, and fine-grained trajectory reconstruction, to potentially augment or replace the usage of encounter models for simulation.

With the help of the trajectory reconstruction, a dataset of NMACs detected independently of reports was created. By not relying on reports by pilot and ATC (Air Traffic Control), this dataset is an order of magnitude larger than those publicly available.

The NMAC (Near Mid-Air Collision) dataset allowed us to gain insight in cases where pilots probably fail detect in time, thus unable to avoid an intruding aircraft. It can also be used to compared with the requirements of DAA (Detect And Avoid) systems to understand how they relate to real world safety, such as number of NMAC that would be avoidable by a system following them.

7.2 Future work

The solution created in Chapter 2 can be used to confirm previous research results, and allow new insights to be gained about air traffic safety, specifically:

- Validating claims that were based on encounter models. For example, calculating conditional probability of NMAC given a LoWC for different definitions of *Well Clear*, explained in Section 4.2.
- Calculating and comparing the performance metrics of DAA systems in different regions, day of week and time of day. Because, some metrics depend on traffic and aircraft class distribution.
- Exploring a newer dataset created after 2020, where we expect higher adoption of ADS-B, due to the mandates that were introduced during that year. This would increase the amount of encounters we expected in the dataset considerably.

References

[14 CFR 91.117] *Convention on International Civil Aviation 14 CFR 91.117*. https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-91/subpart-B/subject-group-ECFRe4c59b5f5506932/section-91.117. Accessed: 2023-05-20.

[ACAS] EUROCONTROL. *Decision criteria for regulatory measures on TCAS II version 7.1*. https://www.eurocontrol.int/sites/default/files/2022-03/eurocontrol-safety-acas-guide-4-1.pdf. Accessed: 2023-04-12.

[Adm19] Federal Aviation Administration. *NextGen Equipage: ADS-B Out Equipage Rates Are Increasing, but FAA Must Address Airspace Access Issues*. Tech. rep. Federal Aviation Administration, 2019.

[ADSB] *ADS-B Reference*. https://mode-s.org/decode/content/ads-b/1-basics.html. Accessed: 2023-04-04.

[And89] J.W. Andrews. "Modeling of Air-to-Air Visual Acquisition". In: *The Lincoln Laboratory Journal* 2.3 (1989).

[ApacheHadoop] *Apache Hadoop*. https://hadoop.apache.org/. Accessed: 2023-05-17.

[ApacheImpala] *Apache Impala*. https://impala.apache.org/. Accessed: 2023-05-17.

[ApacheParquet] *Apache Parquet*. https://parquet.apache.org/. Accessed: 2023-05-17.

[AVH] *The Aviation Herald*. https://avherald.com/. Accessed: 2023-05-02.

[Avidyne] *Avidyne Mixes Active-Surveillance and ADS-B Traffic*. https://www.ainonline.com/aviation-news/ain-news-live/sun-n-fun/2013-04-11/avidyne-mixes-active-surveillance-and-ads-b-traffic. Accessed: 2023-05-18.

[AVweb] *FAA: GPS Outage Won't Count As ADS-B Violations*. https://www.avweb.com/aviation-news/faa-gps-outage-wont-count-as-ads-b-violations/. Accessed: 2023-04-24.

[BEG09] Thomas B. Billingsley, Leo P. Espindle, and John Daniel Griffith. "TCAS Multiple Threat Encounter Analysis". In: *MIT Lincoln Laboratory Report ATC-359* (2009).

[CEPHDOCS] *Ceph object gateway s3 api*. https://docs.ceph.com/en/latest/radosgw/s3/. Accessed: 2023-05-02.

[DEFWC] MIT Lincoln Laboratory. *Low C-SWaP Detect and Avoid: Defining Well Clear*. https://ntrs.nasa.gov/api/citations/20200000623/downloads/20200000623.pdf. Accessed: 2023-05-03.

[Dut+17] Aaron Dutle, Mariano Moscato, Laura Titolo, and César Muñoz. "A Formal Analysis of the Compact Position Reporting Algorithm". In: Dec. 2017, pp. 19–34.

[EAS15] EASA. *TCAS Version 7.1 requirement in EU airspace*. Tech. rep. EASA, 2015.

[EASAADSB] *Amendment to the Airspace Requirements on ADS-B and Mode S*. https://www.easa.europa.eu/en/newsroom-and-events/news/amendment-airspace-requirements-ads-b-and-mode-s. Accessed: 2023-04-04.

[EGK09] L. P. Espindle, J. D. Griffith, and J. K. Kuchar. *Safety Analysis of Upgrading to TCAS Version 7.1 Using the 2008 U.S. Correlated Encounter Model*. Project Report ATC-349. Massachusetts Institute of Technology, Lincoln Laboratory, 2009.

[FAAADSB] *FAA Airspace ADS-B requirement*. https://www.faa.gov/air_traffic/technology/equipadsb/research/airspace. Accessed: 2023-05-01.

[FAAADV] Federal Aviation Administration. *Advisory Circular: Pilots' Role in Collision Avoidance*. https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_90-48D.pdf. Accessed: 2023-05-02.

[FAANMAC] Federal Aviation Administration. *NMACS Search Form*. https://www.asias.faa.gov/apex/f?p=100:33:::NO:::. Accessed: 2023-05-02.

[FAANMAC1] Federal Aviation Administration. *NMACS System Information*. https://www.asias.faa.gov/apex/f?p=100:35:::NO::P35_REGION_VAR:2. Accessed: 2023-04-12.

[FAANMAC2] Federal Aviation Administration. *FAA Near Midair Collision System (NMACS)*. https://www.asias.faa.gov/apex/f?p=100:35:::NO::P35_REGION_VAR:1. Accessed: 2023-05-02.

[HK19] Daniel Howell and Jennifer King. "Measured Impact of ADS-B In Applications on General Aviation and Air Taxi Accident Rates". In: *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. 2019, pp. 1–9.

[Hob91] Alan Hobbs. *Limitations of the See-and-Avoid Principle*. Apr. 1991.

[ICAOannex10] *ICAO Annex 10: Aeronautical Telecommunications*.

[ICAOR] International Civil Aviation Organization. *Rules of the Air*. https://www.icao.int/Meetings/anconf12/DocumentArchive/an02_cons[1].pdf. Accessed: 2023-05-02.

[ICAORANGE] International Civil Aviation Organization. *Registration of aircraft addresses with mode s transponders*. https://www.icao.int/Meetings/AMC/MA/NACC_DCA03_2008/naccdca3wp05.pdf. Accessed: 2023-05-03.

[Koc+08] Mykel J. Kochendedrfer, Leo P. Espindle, James K. Kuchar, and J. Daniel Griffith. "A Comprehensive Aircraft Encounter Model of the National Airspace System". en. In: *Lincoln Laboratory Journal* 17.2 (2008), pp. 41–53.

[Koc+10] Mykel Kochenderfer, Matthew Edwards, Leo Espindle, James Kuchar, and J. Griffith. "Airspace Encounter Models for Estimating Collision Risk". In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 33 (Mar. 2010), pp. 487–499.

[MITLL] The MIT Lincoln Laboratory. *Celebrating 50 Years of Research and Innovation*. https://www.ll.mit.edu/sites/default/files/publication/doc/2022-03/MITLL_50th_ATC_Book_Web_Rev2.pdf. Accessed: 2023-04-12.

[MJ18] Guido Manfredi and Yannick Jestin. "Are You Clear About "Well Clear"?" In: *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2018, pp. 599–605.

[OpenSkyData] *Opensky documentation for internal server*. https://opensky-network.org/data/impala. Accessed: 2023-04-04.

[OpenSkyFacts] *Opensky Foundation Facts Page*. https://opensky-network.org/network/facts. Accessed: 2023-04-04.

[PSQLLIM] *PostgreSQL limits*. https://www.postgresql.org/docs/current/limits.html. Accessed: 2023-05-01.

[Ryk21] Kyle Bradley Ryker. "Development of a detect-and-avoid sensor solution for the integration of a group 3 large unmanned aircraft system into the national airspace system". 2021.

[S3DOCS] Amazon Web Services. *Amazon S3 REST API Introduction*. https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html. Accessed: 2023-05-02.

[Sch+14] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm. "Bringing up OpenSky: A large-scale ADS-B sensor network for research". In: *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. 2014, pp. 83–94.

[SELENIUM] *Selenium with Python*. https://selenium-python.readthedocs.io/. Accessed: 2023-05-05.

[Statista] *Number of aircraft in the United States from 1990 to 2022*. https://www.statista.com/statistics/183513/number-of-aircraft-in-the-united-states-since-1990/. Accessed: 2023-05-17.

[SUST] The Swiss Transportation Safety Investigation Board. *Reports on aviation events*. https://www.sust.admin.ch/en/reports/reports-on-aviation-events/reports-on-aviation-events. Accessed: 2023-05-02.

[URClearED] C.I.R.A. CENTRO ITALIANO RICERCHE AEROSPAZIALI SCPA (Coordinator). *URClearED (A Unified Integrated Remain Well Clear Concept in Airspace D-G Class)*. https://www.dlr.de/fl/en/desktopdefault.aspx/tabid-1149/1737_read-66178/. Accessed: 2023-05-03.

[WBE17] Gilbert Wu, Vibhor Bageshwar, and Eric Euteneuer. "An Alternative Time Metric to Modified Tau for Unmanned Aircraft System Detect And Avoid". In: June 2017.

[WEF11] Roland Weibel, Matthew Edwards, and Caroline Fernandes. "Establishing a Risk-Based Separation Standard for Unmanned Aircraft Self Separation". In: *Europe Air Traffic Management Research & Development Seminar* (July 2011), pp. 14–17.

[Wei+20] Andrew Weinert, Ngaire Underhill, Bilal Gill, and Ashley Wicks. "Processing of Crowdsourced Observations of Aircraft in a High Performance Computing Environment". In: *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. 2020, pp. 1–6.

[Wu+18] Gilbert Wu et al. "Well Clear Trade Study for Unmanned Aircraft System Detect And Avoid with Non-Cooperative Aircraft". In: June 2018.

[WUW19] Andrew Weinert, Ngaire Underhill, and Ashley Wicks. "Developing a Low Altitude Manned Encounter Model Using ADS-B Observations". In: *2019 IEEE Aerospace Conference*. 2019, pp. 1–8.

[WZD20] J. Wang, Y. Zou, and J. Ding. "ADS-B spoofing attack detection method based on LSTM". In: *EURASIP Journal on Wireless Communications and Networking* 160 (2020).