ADS-B Plane Tracking of Unscheduled Flights

Gamma Andhika Octafarras
Vrije Universteit Amsterdam

Nizar el Mahjoubi
Vrije Universteit Amsterdam

Leyu Liu
Vrije Universteit Amsterdam

ABSTRACT

As the amount of planes flying keeps on growing, more of these air crafts are adapting the ADS-B technology for communication. With a large number of air crafts having using the technology, it has become possible to track them quite accurately. The organization OpenSky has many sensors scattered around the world to receive the ADS-B messages, and as a has collected large amounts of the messages. With the improvement of Big Data technologies, it is possible to analyze and process these large amounts of data. The purpose of this paper is to identify flights that are unscheduled and find out their purposes. This paper goes through the methods we have used to identify unscheduled flights and further identifying their types. As the resulting data is still quite large, we also discuss how it is possible top reduce the number of coordinate point and still get a reasonable flight path result. As an outcome, a web-based application based on the data we have processed will visualize the routes of the air crafts.

1. INTRODUCTION

Air crafts globally send out messaging with some sort of identification containing different information about for example their positioning or altitude. These messages are sent out using Automatic dependent surveillance - broadcast (ADS-B) messaging. ADS-B describes a type of emitter which automatically and periodically broadcasts some information to any available sensors. The sensors receiving ADS-B messaging are located on land in air craft control ground stations or at other sensor positions, as well as on other air crafts. Identification is provided using a code provided by the International Civil Aviation Organization (Icao) and is expected to be included in ADS-B messaging. Furthermore, each aircraft should also include its call sign, which refers to a unique code which should be linkeable to any aircraft and is used to identify different scheduled flights. If an aircraft does not provide a call sign which can be linked to the icao database, it can be considered as unscheduled.

The sheer amount of messaging going on at any moment makes the tracking and analysis of air crafts extremely tedious, but companies such as Flightradar24 have used ADS-B messaging to provide live flight tracking. For this project, we have been provided with 620GB of ADS-B messages representing four days ranging from September 16th until September 20st 2016. ADS-B messages contain different types of data and the aim for us was to identify the purposes of those air crafts which do not provide with accurate call sign identification, and interactively visualize these in order to distinguish flight paths.

First, we modified the data in order to get it as small as possible for the path reduction algorithms. Then, we identified different flights of a same aircraft by applying clustering of positioning. We further reduced the size of the data by grouping positions when possible. As the data was set to be analyzed, we first linked it to other available data sets by linking the Icao code where possible, and provide some analysis about the results. Finally, we visualize our findings in an interactive visualization.

2. RELATED WORK

2.1 HDBSCAN algorithm

The position of an aircraft can be represented by altitude, longitude and latitude. The ADS-B message consists of different positions of different aircrafts. In order to separate each flight for each aircraft from all the positions, we can use density-based clustering algorithms to group positions. Besides, when an aircraft is flying, the signal might be blocked sometimes and it is possible that flight messages do not contain all the information of the positions of all the flights, which leads to various density of position values between different flights. So we need to use an algorithm that can cluster flights based on various density of the clusters.

Leland McInnes et al. proposed an efficient algorithm called Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) that can perform clustering for varying density clusters. Similar to DBSCAN, which is a popular density-based algorithm that was proposed by Martin Ester et al.[2], HDBSCAN uses single linkage clustering to transform positions into clusters. However, HDBSCAN achieves better performance by eliminating the epsilon parameter that was used in DBSCAN and adding the min_cluster_size parameter to determine when to form new clusters. It also has various metrics that can be chosen for different kinds of data[4]. The project applies HDB-

SCAN for flights clustering and it performs well by using the "haversine" metric.

2.2 Ramer-Douglas-Peucker Algorithm

The number of positions can be large for each flight where many of them could be redundant and should be removed. In this project, we use an effective line simplification algorithm called Ramer-Douglas-Peucker Algorithm to largely reduce the number of flight positions, which helps speed up the data processing process and makes the visualization of flight routes easier.

D. Douglas et al. proposed the original Ramer-Douglas-Peucker Algorithm. It applies Euclidean distance to measure the distance between different points and reduce the number of points when preserving the shape of the polyline[1]. Though the original algorithm already has good performance, different approaches has been tried in other papers to speed up this algorithm. For example, Hershberger et al. proposed the enhanced Douglas-Peucker algorithm, also called the path hull algorithm, which splits vertices that are found on its convex hull and has the bound time complexity of O(n$\log_n$), while the original algorithm has the bound O(n^2)[3].

3. RESEARCH QUESTIONS

The goal of this project is to identify the unscheduled flights from the large-scale dataset of ADS-B messages, investigate their flight patterns and find out their purposes. In order to analyze the flight messages, we need to use different data processing techniques for large-scale dataset and find additional data source. We came up with several research questions as follows.

- How to decode ADS-B messages?
- How to find out the unscheduled flights and their purposes?
- How to reduce the size of the dataset?
- Is there any fault and error in the ADS-B messages? And how to validate the data?
- How to get the positions of the flights and remove the redundant positions?
- How to visualize the flight routes?

4. DATA

4.1 OpenSky Sensor

The raw data that we work on in this paper comes from many sensors that enthusiasts have put up all over the world and collected by the OpenSky[1] organization, for our data specifically, most of the sensor data comes from Europe with a minority of the data being sensors in Brazil and North America within the time period of 6 days (18th to the 24th September of 2016). The raw dataset contains the respective data: *sensorType, sensorLatitude,, sensorLongitude, sensorAltitude, timeAtServer, timeAtSensor, timestamp, rawMessage, sensorSerialNumber, RSSIPacket, RSSIPreamble, SNR* and confidence.

[1]https://opensky-network.org/aircraft-database

4.2 ICAO 3-Letter Callsign

The ICAO 3-Letter Callsign is a dataset that we have retrieved from the organization International Civil Aviation Organization (ICAO)[2]. This dataset mainly contains the callsigns of commercial airlines that has been registered to the IATA Operational Safety Audit (IOSA), a company which audits commercial airlines. The dataset also contains information that is up to date at the point of this report (October 2019) . We have found that the dataset contains most but not all scheduled airlines. This dataset contains the respective data: operatorCode, operatorName, countryName, countryCode and update.

4.3 FlightRadar24 Icao24

The FlightRadar24 Icao24 is a dataset that contains informations about different aircrafts based on their icao24 code. The dataset is retrieved from FlightRadar24, which should be credible but the drawback being that the dataset has stopped updating since 2017 as FlightRadar24 has changed their terms of service. That should not be a problem in the case of this project as the raw dataset we are using is in the period of 2016. The dataset contains the respective data: icao, regid, mdl, type and operator.

5. PROJECT SETUP

5.1 Technologies

- **Spark**
  Spark[3] is a cluster computing framework which supports data parallelism and is suited to process a large amount of data. Since for this project we have access to instances from AWS[4] , a cloud computing platform, a large amount of computing power is provided and combined with spark, the couple is suited for our use case of processing the large amount (620GB) of data.
- ***adsb-java* Decoder**
  adsb-java[5] is the decoder that we are using for the decoding the raw ADS-B messages, we have chosen technology as it has proven to be reliable as there many previous projects that have used it. The library itself is a java library and since the platform we use, Databricks[6], does not provide java notebooks, we opted for the Scala Programming language instead to implement the library and most of the data processing.
- **Leaflet Javascript**
  We first attempted to create the visualization using d3.js, however we quickly found that Leaflet would be a better fit. Leaflet is created specifically for creating interactive maps using Javascript, which fits perfectly to our intended aim. We thus converted the processed data into Geojson which we used to draw the different elements - flight starting and ending points, as well as their paths - on the map. Geojson is a format used primarily in Javascript for encoding geographic data structures.

[2]https://www.icao.int/safety/istars/pages/api-data-service.aspx/
[3]https://spark.apache.org
[4]https://aws.amazon.com/ec2/spot/
[5]https://github.com/openskynetwork/java-adsb
[6]https://databricks.com/

- **Jupyter notebook**
  We used Jupyter Notebook to run the data processing and included some of the libraries below. A Jupyter notebook is an open-source web application which allows to run code in cell format and visualize the output.

- **Python libraries**
  We used many Python libraries for data processing. Pandas dataframe is the main data structure for reading/writing files and the algorithms. The sci-kit learn library and the HDBSCAN library are used to cluster flights after initial grouping, while the rdp library was installed for the Ramer-Douglas-Peucker algorithm to reduce redundant positions for the flight paths.

5.2 Raw Data Structure

To start of the project, we first looked at the raw dataset that we have retrieved. The dataset is a collection of ADS-B sensor data that is stored in the *avro* format. The size of the whole dataset is around 630 GB and contains 8814 *avro*[7] files. The data contains the columns that have been mentioned in Section 4.1. Though there are 13 columns, we find that only 4 of them are necessary in the case of our project. The columns that are necessary and why are as follows

- sensorLatitude and sensorLongitude
  The *sensorLatitude* and *sensorLongitude* is used in the decoding process to extract more accurate positions of messages.

- timeAtServer
  This is used as timestamp as *timeAtServer* is actually the time that the aircraft broadcasts the message.

- rawMessage
  The *rawMessage* is the main data we will be processing. It is the encoded ADS-B message, which is 112 bits long that breaks down into 5 parts[8]. J. Sun et al. explained the method to extract the message types from the previously mentioned 5 parts.[5]

5.3 Understanding Extracted Data

Data that we need for this project such as the position data is encoded in the *rawMessage* part that has been explained in Section 5.1. Before we start with the processing, we first have to see what the data that can be extracted from the encoded ADS-B messages are, what they represent and how frequent are they. As there is a possibility of two sensors retrieving the same data from an aircrafts single message, duplicate *rawMessage* occurrences has been removed. After the removal of redundant data, the types of messages that has been extracted in order of count are represented in Table 1

[7]https://avro.apache.org/
[8]https://mode-s.org/decode/adsb/introduction.html

Message Type	Count
ALL_CALL_REPLY	5,532,653,183
SHORT_ACAS	3,088,661,593
ALTITUDE_REPLY	1,907,570,801
COMM_B_ALTITUDE_REPLY	1,887,938,529
ADSB_AIRBORN_POSITION_V0	1,116,701,055
ADSB_VELOCITY	1,113,789,503
COMM_B_IDENTIFY_REPLY	748,906,429
IDENTIFY_REPLY	742,612,105
LONG_ACAS	233,559,962
ADSB_IDENTIFICATION	12,4591,457
EXTENDED_SQUITTER	90,345,638
MILITARY_EXTENDED_SQUITTER	43,470,477
MODES_REPLY	31476142
ADSB_AIRBORN_STATUS_V2	21,910,542
ADSB_AIRBORN_STATUS_V1	18,149,823
ADSB_EMERGENCY	10179029
ADSB_SURFACE_POSITION_V0	4,425,532
ADSB_AIRSPEED	3578791
ADSB_SURFACE_STATUS_V2	81,083
COMM_D_ELM	43743
ADSB_SURFACE_STATUS_V1	36,738
ADSB_TCAS	10,653
ADSB_STATUS_V0	10,151

Table 1: Extracted Message Types

5.4 Pipeline

The project is divided into three parts, including data processing, linking additional data source and visualization. The data processing part includes messages decoding and reconstruction, unscheduled flights filtering, flight clustering and redundant positions removing. The additional data source is connected with the original dataset to find out flight patterns and purposes. The final result of the previous steps is presented in the visualization. The pipeline of this project is shown as follows.

- **Extracting message types**
  To get an understanding of the data that we will be processing, we first have to decode the ads-b messages from the raw sensor data. After which we will be able to decide how to proceed

- **Reduce ADS-B messages**
  Since the size of the datasets are substantial, reducing the data while keeping the important ones for this project is necessary as to reduce the amount of time it will take to decode the positions, callsigns and to run the clustering algorithms

- **Filter unscheduled flights**
  To separate unscheduled flights from scheduled flights, we first extract the callsigns of each aircraft and then compare it to the ICAO 3-Letter Callsign dataset mentioned in Section 4.2.

- **Link additional dataset for detecting purposes**
  To further filter out remaining scheduled commercial airlines, we filter each aircrafts unique *icao24* code to another dataset from FlightRadar24 mentioned in Section 4.2.

- **Clustering**

  In order to separate each flight, we use the time gap between two positions to first split the data and then use HDBSCAN to cluster the positions into different flight groups.

- **Reduce positions**

  Ramer-Douglas-Peucker algorithm is applied for each flight to remove the redundant positions to reduce the file size while keeping the original shape of the flight routes for visualization.

- **Visualization**

  The flight paths are visualized using the Leaflet framework in Javascript in order to provide an interactive web visualization. The data is first transformed to geojson format, after which its loaded into the web document using filters based on earlier identified purposes such that the user can interactively explore different air crafts.

5.5 Initial Filtering

With a total of 16,727,632,033 successfully extracted messages, as represented in Table 1, the ones that we will be mainly using are the following

1. *ADSB_AIRBORN_POSITION_V0*
2. *ADSB_SURFACE_POSITION_V0*
3. *ADSB_IDENTIFICATION*

The airborn position is a message that indicates what the current latitude, longitude and altitude of an aircraft whilst it is flying . The surface position indicates the latitude and longitude of an aircraft whilst it is on the ground (0 feet Altitude). The identification indicates the callsign of a the aircraft.

Before we do any processing, we converted the data to a parquet[9] format for better performance. Since the case is that we only need three of the message types, we first filtered out unneeded messages. The result of only keeping the three types of messages is that we end up with 1,245,718,044 messages. The stated amount of messages only accounts for around 7% of the whole dataset. After the initial filtering we have brought down the amount of raw data from 630 GB to around 19 GB.

5.6 Separating Unscheduled Flights

5.6.1 *Defining Scheduled and Unscheduled*

As the aim of this project is to visualize unscheduled flights, we will have to separate the flights between scheduled and unscheduled. To do this we first have to define what we identify as an unscheduled flight. For this project, we have decided to categorize scheduled flights as the following:

- Commercial scheduled airline (eg. KLM, American Airlines, Cathay Pacific etc.)
- Scheduled chartered flights (eg. British Airways, TUIfly, etc.)
- Not a cargo airline (eg. DHL, FedEx, UPS)

Flights other than the two stated above will be stated as unscheduled in the context of this project.

[9]https://parquet.apache.org/

5.6.2 *Extracting Callsigns*

In order to identify scheduled flights, we first have to extract the callsigns of each aircraft from the ADS-B messages. We do this by getting all unique icao24 codes of each aircraft and using the decoder to extract callsigns from the *ADSB_IDENTIFICATION* messages. From the decoded callsigns, we are able to identify commercial airlines as they have a common callsign format which consists of a 3 alphabets followed by 3 to 4 digits[10]. An example of extracted callsigns are represented in Table 2.

Raw Message	Icao24	**Callsign**
8d02b263205153b192e830f49245Y	02b263	TUN1 0
8d06001920341574d608205a450a	060019	MAU45
8c06a069204544b0db9820568fa1	06a069	QTR069
8d0a002220101231c30da0f126f9	0a0022	DAH1006
8d0a008a20101231c38d60b2fe57	0a008a	DAH1085

Table 2: Extracted Callsign Sample

5.6.3 *Separation Methods*

After we are clear on the distinction between scheduled and unscheduled flights. We are then able to find ways to identify the scheduled flights. The first step that we have taken is to retrieve a dataset of flights that is registered in the ICAO database and is registered in the IOSA. We have found it that most commercial scheduled airlines are registered in this dataset. An example of scheduled aircraft according to the callsigns shown in Table 2 would be the callsign *QTR069* with *icao24* code of *06a069*. By comparing it the dataset mentioned previously, we confirm that callsign *QTR069* is a scheduled flight belonging to Qatar Airways.

After we separate based on the ICAO 3-Letter Callsign dataset, we then filter out remaining scheduled flights by using the FlightRadar24 Icao24 dataset. This is necessary as some scheduled aircrafts are not registered in the ICAO dataset. An example from our observation would be Egyptair. For this part of filtering we do not use the callsign, but instead use the *icao24* code, which can be inferred form all message types.

After the applying the two separation methods stated above, we were able to bring the size of the dataset from around 19GB to around 7.8GB.

5.7 Identify Purposes of Different Aircrafts

For the visualization aspect of the this project, we would not only want to show unscheduled flights, but also show the different purposes of each aircraft. Our approach to getting purposes was to first define a set amount of purposes and what they would represent. For this project, we created 8 categories of purposes that we would like to retrieve form the flights, which are as follows.

1. Military
2. Rescue
3. Academy and Clubs
4. Space and Aerospace
5. Private Owners

[10]http://www.airportfreak.yolasite.com/call-signs.php

6. Unscheduled Charter (Commercial)
7. Cargo (Including scheduled)
8. Unknown

The dataset from FlightRadar24 contains operator names for most aircraft, with this we were able to separate most flights based on their purpose, by using methods such as using keywords to separate operator name and using lists of aircraft for a certain purpose and comparing operator names. After separating the dataset from FlightRadar24 into different purposes, we then compared it to our filtered unscheduled flights. The result of the comparison and filtering is represented in Table 3 and Table 4.

Purpose	**Position points**
Military	813,364
Rescue	20,964
Academy and Clubs	259,480
Space and Aerospace	3089
Private Use	3,278,957
Unscheduled Charter	2,420,005
Cargo (Including scheduled)	11,571,894
Unknown	13,465,207

Table 3: Flight purposes position points

Purpose	**No. of Aircrafts**
Military	40
Rescue	2
Academy and Clubs	13
Space and Aerospace	1
Private Use	870
Unscheduled Charter	60
Cargo (Including scheduled)	687
Unknown	2,523

Table 4: Flight purposes aircrafts

After the separation of flights based on their purposes, the data has been brought down from 7.8 GB to around 980 MB.

5.8 Decode and Validate Positions

As the purpose of this project is a visualization on a world map, the most important data we would need to retrieve are the position points, the latitude and longitude of the flights. To to this, the decoder is used on the *ADSB_AIRBORN_POSITION_V0* and *ADSB_SURFACE_POSITION_V0* messages, which results in the latitude, longitude and altitude. The result though has some noise, to reduce some of the noise we did a simple check of distance traveled and time to get an estimated speed. If we detect that from one point to the next the aircraft travels faster than 1000 knots, we remove the next point. The data after all the filtering processes is not that large, but since we are able to use spark, the data should be able to be decoded in a way that can be parallelized well. The steps that had been taken are as follows:

1. Get all unique *icao24* code of all messages
2. Sort all messages by timestamp and *icao24*
3. Group *ADSB_AIRBORN_POSITION_V0* and *ADSB_SURFACE_POSITION_V0* messages into an array with their respective timestamp for each unique *icao24* codes
4. Flatmap over grouped array and run decoder and speed checker for each message, since the decoder is stateful and requires previous position to get an accurate position.

The result is a table of each position points with timestamp and its corresponding *icao24* code.

5.9 Cluster flights

For an aircraft, positions that belong to one flight are densely connected while there is usually a big time gap between two different flights. So we make the assumption that if two positions appear in two timestamps where the gap of the time is big, such as 2 hours, we can consider that the two positions belong to different flights. However, for the positions with small time gaps, it is difficult to determine whether the two positions belong to two different flights, so we use HDBSCAN algorithm to cluster flights with smaller time gaps based on the density of positions. For aircraft 010024, as shown in Figure 1, there are many gaps between different values of altitude which is not a good representation of the flight pattern, while the longitude distribution shows a clear pattern of continuous positions in Figure 2. Therefore, we use the values of longitude to group the data by time gap and then cluster the flights using the HDBSCAN algorithm with the values of longitude and latitude as input.

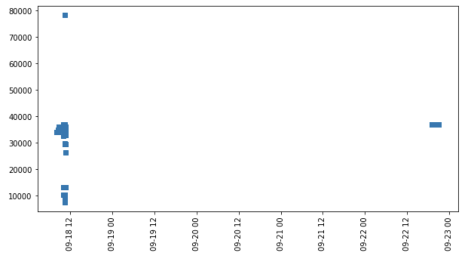


Figure 1: Altitude distribution of aircraft 010024

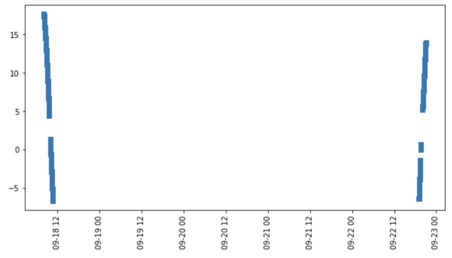
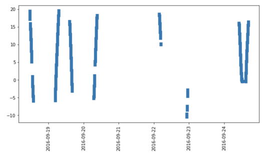


Figure 2: Longitude distribution of aircraft 010024

With a list of positions that belong to different aircrafts, we split the data for each aircraft and apply initial grouping to group the flights with big time gaps. As unscheduled flights, such as an aircraft for military use, usually fly in low altitude where the signal might be blocked, while a commercial flight usually fly in high altitude with more positions that can be processed. Therefore, the positions for unscheduled flights are more sparse and discontinuous with lots of missing values, which makes it more difficult to determine whether two points belong to the same flight. We adjust the parameters in HDBSCAN to cluster the flights with the data we have. Though the accuracy of the result is affected by the missing values, it shows good performance in general. The clustering result for the commercial aircraft 010153 used in Egyptair is shown in Figure 3 with 7 flights, while the result for the aircraft 0200ac used in Royal Moroccan Air Force is shown in Figure 4 with 2 flights.


(a) Longitude distribution before clustering

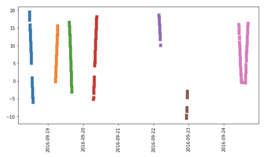
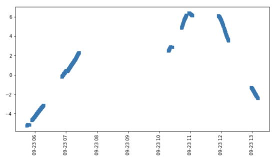

(b) Longitude distribution after clustering

Figure 3: Clustering result for aircraft 010153


(a) Longitude distribution before clustering

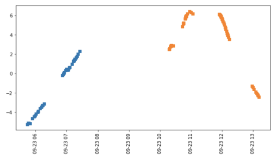

(b) Longitude distribution after clustering

Figure 4: Clustering result for aircraft 0200ac

5.10 Remove redundant positions

To reduce the file size and visualize flights routes, we need to remove the redundant positions in each flight. We apply Ramer-Douglas-Peucker line simplification algorithm, which efficiently reduces a large portion of data and keeps the shape of the flight routes. As shown in table 1, we run the algorithm on a 23MB csv sample file with 16 aircrafts, which originally has 245,458 rows of flight position messages, and the size of the output file has been reduced to 0.149MB with 3709 rows.(1 table)

	File size	Number of rows
Before reduction	23MB	245,458
After reduction	149KB	3709

Table 5: Reduction result

5.11 Analyzing Data

At this point, the data has been formatted to be ready for some preliminary data analysis before visualizing. A total of 785 450 positions were found to correspond to unscheduled air crafts. Of these roughly half (374 635: 47) belong to cargo air crafts. 30% are unknown air crafts (which did not present a matching icao code) and 13% are private air crafts. Military air crafts represent only 5% and charter air crafts represent 4%. Finally, training, rescue and space air crafts respectively account for 0.5%, 0.1% and 0.02%.

We first looked at the most common operators based on purpose to identify those operators which are not included the scheduled flights. For the cargo air crafts, the results are presented in the table below:

Operator	Occurence count
FedEx	191261
UPS	26051
DHL	22678
KLM Cargo	13122
Cargo Air	9982

Table 6: Cargo air crafts based on operator

As can be seen the most common operator is FedEx which accounts for roughly 75% of the top 5 operators for cargo air crafts. The same was done for the military air crafts and the training air crafts. It was unnecessary to do this for the private owners since they do not have an operator, as well as for the rescue purpose since it only counted a single operator, namely German Air Rescue.

Operator	Occurence count
Swiss Air Force	14981
Royal Air Force	7778
Italian Air Force	7081
Airbus Military	1217
French Air Force	1124

Table 7: Military air crafts based on operator

It's surprising to see that the Swiss Air Force has the largest part of the occurrence count in the military air craft position counts. This could lead to believe that this air force was most active during aforementioned period, leading to believe that there was a specific training period during those four days for the Swiss Air Force.

Operator	Occurence count
Oxford Aviation Academy	3575
European Aircraft Private Club	619
Aero Club Bergamo	173
Motorflug-Club Salzgitter eV	91
Aero Club de Valence	86

Table 8: Training air crafts based on operator

The Oxford Aviation Academy seems to be most active during this period. It accounts for roughly 80% of all observations of the top 5 operators as seen in Table 8.

We then were interested in finding out what type of air crafts flew most extensively over the four days. We do this analysis for the private and military air crafts. In fact, the types of air crafts for cargo showed that the Boeing brand dominated, with one Airbus type in the top 5. Below the table for the private air crafts:

Type	Occurrence count
Cirrus SR22	3217
Cirrus SR22T	2941
Piper PA28-161	1828
Beech B200 Super King Air	1668
Bell 429 GlobalRanger	1629

Table 9: Private air craft types

The most common air craft types, the Cirrus SR22 and the Cirrus SR22T, are both hobby types. This also goes for the Piper which is a short-distance small aircraft. The Beech B200 Super King Air is a luxurious jet and the Bell Globalranger a helicopter. We then did the same analysis for the military air crafts. The results are presented in the table below:

Type	Occurrence count
Pilatus PC-21	8649
Boeing 767-2EY	5457
Airbus KC2 Voyager	4738
Pilatus PC21	4241
British Aerospace Avro RJ100	2605

Table 10: Military air craft types

We find that the most common type of air craft for military purposes is actually a training air craft (the Pilatus PC-21). The other three are all long-range military transport air crafts.

The type of air craft and the operator gives us a better view of the overall purposes of the unscheduled flights. It's also interesting to see how the different purposes behave in terms of altitude of flight. These are presented in below figure.

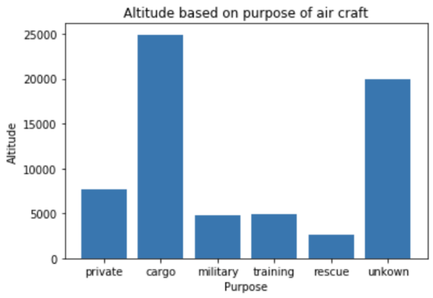


Figure 5: Average altitude per purpose of aircraft

As we can see, the cargo air crafts fly the highest, followed by the unknown category. This could lead to believe that the unknown air crafts refer to air crafts of the same type. The rescue air crafts fly the highest which does make sense since they typically do not travel far and have to act very quickly. The private air crafts then follow, which makes sense since it would be distributed against short-range flights and longer range flights. Finally, military and training air crafts fly approximately at the same height, strengthening the earlier finding that the most common air craft type for the military purpose is a training air craft.

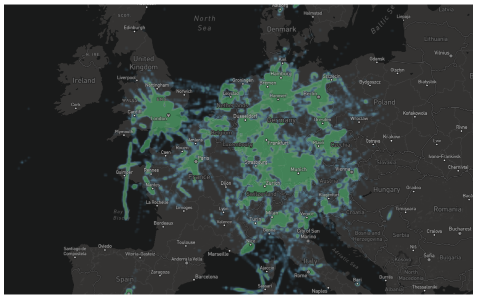


Figure 6: Distribution of privately owned aircraft positions

5.12 Visualization

The final result is a web visualization with all unscheduled air crafts. The visualization is built using Leaflet which is a library used specifically for visualizations of maps. The user gets presented with a world map of all air crafts of a set purpose which the user can filter on by using the drop down menu.

The dots on the map represent starting and ending points of different flights belonging to different air crafts. By clicking on a dot, the user receives a pop up with the following information about the observation: aircraft type, icao code, aircraft pictures (link to follow), altitude, flight number. At the same time, paths of the different flights of an aircraft are presented and intermediary observations are also visualized.

By clicking on a different starting or ending point, the user can investigate a different aircraft with its flight patterns. Below an example of the visualization.

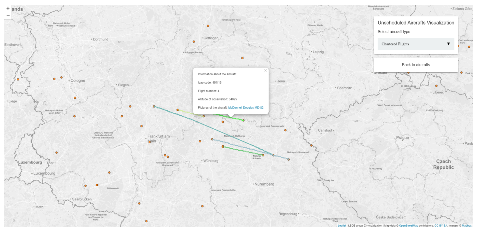


Figure 7: Visualization Example

6. EXPERIMENTS

6.1 Using ICAO API to Identify Scheduled Aircrafts

At the start of the project, we found out that the ICAO organization provided an API to check if a callsign corresponds to a certain airline. This method worked well when querying small amounts of aircrafts. As we started to query the whole dataset, it was apparent that per API call was limited to a payload under around a few hundred kilobytes. As for free use of the api, we only get 100 api calls per month. With the amount of aircrafts we have, that would be impossible. As an alternative, we reached out to the ICAO and managed to retrieve the full dataset. Which we then used for the filtering instead of the ICAO API service.

6.2 Cargo flights Filtered Out

After we have finished all identification of purposes, we have found out that various cargo airlines were missing from the result. We traced back to where it started to get filtered out and found out that most cargo airlines are included in the IOSA ICAO 3-Letter Callsign dataset. After experimenting with the filtering steps of the ICAO dataset, we find that to get all cargo flights, including the scheduled ones, a method would to skip the filtering by ICAO dataset. For the cargo airline part we decided to directly filter with the FlightRadar24 dataset.

7. CONCLUSION

In conclusion to this project, a web-based visualization of unscheduled flights has been produced. The visualization represents the starting point of each aircraft and their respective flight paths and aircraft information when clicked. It is also possible to filter based on the purpose of the aircraft. All of the research questions mentioned in Section 3 have also been answered with this project and the process has been explained for each step. Thus the main goals of this project has been successful.

Unfortunately most of the sensors are located in Europe, thus our analysis are mainly limited to the respective coverage of the sensors. Adding to that that most private planes fly in a low altitude, which prevents the sensor to pick up data without a clear line of view, the data of unscheduled flights are spotty and sometime inconsistent. It is likely that some flights might have been wrongly clustered or filtered out due to the small amount of position data. Nevertheless, we believe that we have received interesting results from the analysis and hope that with more sensors being placed and more aircrafts adopting the ADS-B technology, data will be much more complete for future analysis.

7.1 Future Work

Most of the positions for different types of flights can be clustered and reduced efficiently by the algorithms. However, the clustering results for unscheduled flights are spotty and may not be accurate. As there are many missing values for unscheduled flights, it affects the performance of the algorithms and may cause one flight being grouped into two flights. Besides, the clustering algorithms take a long time to run on large-scale dataset. To achieve better performance, we need to find more effective algorithms that can better fit the input features of the data for unscheduled flights with smaller time complexity.

Another process which can be done is to identify aircraft vehicle type (eg. helicopter, jets, etc) by using the velocity of the aircrafts and altitude, this would currently still be challenging for unscheduled flights as the position points of unscheduled flights are not as complete and hard to cluster appropriately

8. ACKNOWLEDGEMENTS

We would like to firstly thank Peter Boncz for teaching us the LSDE course and for feedbacks of our project. We would like to secondly thank AWS for providing a grant to run clusters in their platform, giving us the opportunity to truly experience the ecosystem of LSDE. Lastly we would like to thank Bamidele Ajayi from the ICAO data analysis team for providing a paid dataset of the aircraft callsigns at no cost whatsoever.

9. WORK DISTRIBUTION

Student	Workload
Leyu Liu	**Algorithms:** Clustering algorithms Path reduction algorithms **Data processing:** Preprocess the data according to aircraft types and reduce file size **Write report**
Gamma	**Data Retrieval:** Retrieving ICAO dataset **Data Processing:** Extracting data from ADS-B Initial filtering of data Separation of unscheduled flights Identifying purposes Decoding position messages **Visualization:** Filtering aircraft by purpose **Write report**
Nizar El Mahjoubi	**Data Retrieval:** Retrieving external data sources and linking to data set **Data analysis:** Preliminary data analysis on the preprocessed data **Visualization:** Converted to Geojson and created a visualization in Leaflet **Write report**

Table 11: Student work distribution

10. REFERENCES

[1] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line of its caricature. 1973.

[2] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 1996.

[3] J. Hershberger and J. Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. 1992.

[4] L. McInnes, J. Healy, and S. Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar 2017.

[5] E. J. H. J. Sun J, Vû H. Pymodes: Decoding mode-s surveillance data for open air transportation research. ieee transactions on intelligent transportation systems. 2019.