Integrating pyModeS and OpenSky Historical Database

Sun, Junzi; Hoekstra, Jacco

DOI
10.29007/mmsb

Publication date
2019

Document Version
Final published version

Published in
Proceedings of the 7th OpenSky Workshop 2019

Integrating pyModeS and OpenSky Historical Database

Conference Paper · November 2019

DOI: 10.29007/mmsb

CITATIONS
0

READS
56

2 authors:

Junzi Sun
Delft University of Technology
22 PUBLICATIONS **147** CITATIONS
SEE PROFILE

Jacco Hoekstra
Delft University of Technology
96 PUBLICATIONS **641** CITATIONS
SEE PROFILE

EPiC Series in Computing

Volume 67, 2019, Pages 63–72

Proceedings of the 7th OpenSky Workshop 2019

Integrating pyModeS and OpenSky Historical Database

Junzi Sun and Jacco M. Hoekstra

CNS/ATM, Control and Simulation Research Group
Aerospace Engineering Faculty
Delft University of Technology, the Netherlands

Abstract

A large quantity of Mode S data is being gathered by the OpenSky receiver network every day. Information regarding common flight states, such as position, ground speed, and the vertical rate is broadcast by ADS-B and has already been decoded and made available for researchers via the OpenSky historical database. However, there is still a large amount of Mode S communication data that has not yet been fully explored. Specifically, the information contained in Enhanced Mode S Surveillance downlink messages can be utilized to better support ATM research. The challenge of decoding such information lies in the implicit inference process for Mode S Comm-B messages. This paper presents a new open library, `pymodes-opensky`, which connects the existing open-source `pyModeS` decoder to the raw Mode S messages from the OpenSky historical database through the Impala shell. It also presents a convenient workflow that can be used to obtain additional information regarding airspeeds, flight intentions, and meteorological conditions of a given flight from the OpenSky database. An analysis based on a global dataset from OpenSky is conducted, and the associated Mode S interrogation statistics in different regions are shown.

1 Introduction

Aircraft surveillance in the early days relied on primary radar, which can only determine the slant distance and azimuth angle of aircraft. Commonly, civil primary radars are not designed to provide the elevation angle of the object. Hence, air traffic controllers had to rely on other methods to determine the altitude (and identity) of aircraft. The secondary surveillance radar was thus developed to acquire additional information from aircraft. The early version of aircraft transponders complied with Mode A and Mode C communication protocols. These protocols allow the identity and the pressure altitude of an aircraft to be downlinked through transponder interrogations. Later on, the Mode S selective interrogation protocol was developed to enable more types of information to be transmitted upon request of secondary surveillance radar.

Nowadays, the Mode S transponder has become mandatory for airliners in most controlled airspace. Several Mode S capabilities are also mandatory in some parts of the world. For instance, Europe requires all aircraft with a takeoff mass over 5700 kg and a maximum speed of over 250 kt to have the Mode S Enhanced Surveillance (EHS) and Mode S extended squitter capabilities enabled. The Mode S extended squitter is the base technology for the Automatic Dependent Surveillance-Broadcast (ADS-B) service, which provides accurate position and velocity measurements of aircraft based on the global navigation satellite system.

Information broadcast by ADS-B can be easily received and decoded with a simple ground receiver setup. Several crowd-source receiver networks exist providing aggregated flight information with extended coverage. In Europe, two of these networks are FlightRadar24 and OpenSky. While FlightRadar24 is focused on commercial service, OpenSky [8] is a research-oriented network, which provides easy access to historical flight data for ATM researchers and other players in the aviation industry.

Many studies have explored the decoded ADS-B data from the OpenSky network. There is also an existing tool, `traffic`, that enables easy access to the derived ADS-B data [7]. Still, a large amount of recorded raw Mode S data has not been efficiently utilized. Unlike ADS-B, where all messages contain identifiers regarding the message types, messages related to Mode S service do not always contain such information. Thus, decoding becomes a challenging task for third-party observers.

The inference of Mode S messages types has been investigated by our recent study [11], which produced an open-source library (`pyModeS`[1]) that makes the decoding of these messages convenient for third-party observers. In addition to decoding ADS-B messages, `pyModeS` can identify and decode many common categories of Mode S messages. The ability to obtain more valuable data from Mode S communications allows several of our studies to be conducted with higher accuracy [9, 10]. These capabilities make `pyModeS` a good candidate to decode a large number of raw Mode S messages in the OpenSky historical database. Once information contained in Mode S signals has been extracted, the aircraft airspeeds, target altitudes, and meteorological condition information can serve as a valuable addition to open ADS-B data when conducting ATM studies.

This paper focuses on a practical approach for researchers to combine `pyModeS` library and OpenSky raw Mode S data, which can, in turn, provide more flight state information for ATM studies. An open-source library, `pymodes-opensky`,[2] is made public along with this paper, which is primarily focused on extracting information from EHS messages, including selected vertical intention reports, track and turn reports, and heading and speed reports. It also provides the possibility to extract weather information from meteorological routine air reports and meteorological hazard reports whenever they are available. With a lower-level interface, the library can also be used to query both ADS-B and Mode S raw messages directly from OpenSky historical database through the Impala shell.

2 Mode S information and inference

The original design of aircraft transponders supports two primary modes for surveillance (Mode A and Mode C). These two protocols allow aircraft to report their identities (squawk codes) and barometric altitudes to air traffic controllers upon the interrogation of secondary surveillance radar (SSR). The information contained in these communications is very limited, and there are significant drawbacks when applying them in high-density air traffic situations.

Compared to Mode A/C, Mode S is a newer system that was designed in the 1980s [1] to enable SSR to selectively interrogate more information from aircraft. Mode S provides different types of information distinguished by the uplink (or downlink) format code. Currently, 11 out of 24 Downlink Formats (DF) are defined and implemented. Among these 11 DF codes, DF 17 (civil), 18 (civil) and 19 (military) are defined as Mode S extended squitter, and the corresponding messages are broadcast automatically without the need for SSR interrogation. For example,

[1] Available at https://github.com/junzis/pymodes.
[2] Available at https://github.com/junzis/pymodes-opensky.

the well known Automatic Dependent Surveillance-Broadcast (ADS-B) is an application that builds upon the Mode S extended squitter.

Besides ADS-B, two main surveillance services are formed using messages of DF 20 and 21, by groups of different message types that are known as Comm-B Data Selector (BDS) codes. These services are Mode S Elementary Surveillance (ELS) and Mode S Enhanced Surveillance (EHS) [3]. They allow different groups of aircraft information to be selectively interrogated. ELS is designed for air traffic controllers to obtain basic information such as callsigns and communication capabilities of aircraft, while EHS provides air traffic controllers more information on aircraft flight states such as intent, airspeeds, and turn performance.

Based on the ICAO Annex 10 [4], Figure 1 summarizes these different concepts of Mode S downlink communication, as well as their relationships among the definitions of different types of messages. Apart from the aforementioned ADS-B, ELS and EHS services, additional Mode S messages, reporting meteorological information are also shown in Figure 1 and will be discussed briefly in this paper.

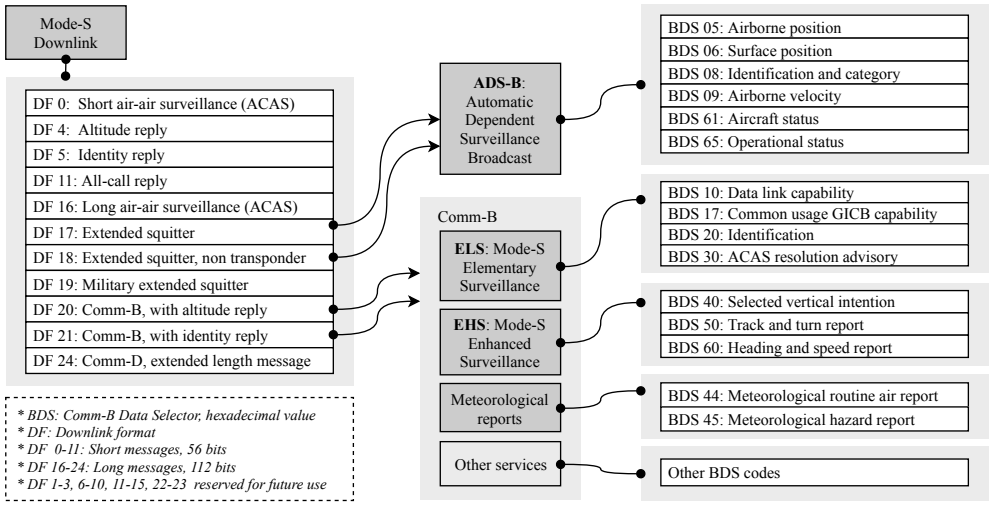

Figure 1: Relationship between Mode S services and downlink data

All known Mode S message structures are defined by the communication protocols in ICAO standards [5, 6]. Once the message type can be identified, the corresponding information can be easily decoded. For the ADS-B message, a Type Code is included in each message. Each Type Code corresponds to a unique BDS code, which can be further used to identify the structure of the message for decoding. For Comm-B communications, the message types are not explicitly indicated in the message. Since each downlink message corresponds to a specific uplink request with a specific BDS code, the air traffic controllers always know the expected structure and the format of a downlink message. However, as a third party observer, this poses a major challenge for decoding and making use of the information included in Comm-B messages.

One key capability of `pyModeS` is the ability to infer common types of Comm-B messages. For instance, all the BDS codes from Figure 1 can be inferred using `pyModeS`. It uses a combination of heuristic and probabilistic methods to infer the possible BDS codes of a message. As a summary, Table 1 shows the heuristic logic that is designed as the primary BDS inference method.

Furthermore, since BDS 50 and 60 have a similar structure, many messages can potentially

Table 1: BDS code identification rules

BDS	Bits	Parameter	Identification rules
10	1 - 8	BDS	Bits equal to 00010000
	10 - 14	Reserved	Bits must all be zeros
17	7	BDS 20 enabled [a]	Bit equals to 1
	29 - 56	Reserved	Bits must all be zeros
20	1 - 8	BDS	Bits equal to 00100000
	9 - 56	Call sign	Only contain 0-9, A-Z, or space
30	1 - 8	BDS	Bits equal to 00110000
	29 - 30	Threat type	Bits must not equal to 11
	16 - 22	ACAS III	Less than 48
40	1 : 2-13 [b]	MCP/FCU selected altitude	Status consistent [c]
	14 : 15-26	FMS selected altitude	Status consistent
	27 : 28-39	Barometric pressure setting	Status consistent
	40 - 47	Reserved	Bits must all be zeros
	52 - 53	Reserved	Bits must all be zeros
50	1 : 2-11	Roll angle	Status consistent, range: [-60, 60]
	12 : 13-23	True track angle	Status consistent
	24 : 25-34	Ground speed	Status consistent, range [0, 600]
	35 : 36-45	Track angle rate	Status consistent
	45 : 46-56	True airspeed	Status consistent, range [0, 500]
60	1 : 2-12	Magnetic heading	Status consistent
	13 : 14-23	Indicated airspeed	Status consistent, range [0, 500]
	24 : 25-34	Mach number	Status consistent, range: [0, 1]
	35 : 36-45	Barometric vertical rate	Status consistent, range [-6000, 6000]
	46 : 47-56	Inertial vertical rate	Status consistent, range [-6000, 6000]

[a] BDS 20 is the code that has to be enabled for all transponders to provide the minimum Mode S capabilities.
[b] This format b1:b2-b3 indicates the status bit at b1 with value bits from b2 to b3.
[c] Consistency indicates that when a status bit is zero, all value bits must also be zeros.

be identified with both BDS 50 and 60. The probabilistic method in `pyModeS` allows further examination of this type of message by taking into consideration the ground speed information from ADS-B and wind (if known). The details of this inference step can be found in [11].

3 Working with OpenSky Mode S data

The OpenSky historical database collects different types of Mode S messages through its network of receivers. Among these, ADS-B messages are decoded, and only the decoded data is stored. The majority of the remaining messages, including DF 4/5 and DF 20/21 messages, are stored with the raw format. Together with the raw messages, only corresponding altitude or identity codes are decoded. Other information included in DF 20/21 messages is not directly available in this database.

Given any raw message, the ICAO address can be found with a recovery process involving the parity data [2]. It is also possible to make use of the potential ICAO address, BDS code, and parity bits to validate the correctness of the messages [11]. In the OpenSky, the ICAO address has already been computed and stored along with the raw data.[3] Having the corresponding

[3] In some cases, the stored raw messages are modified, and the parity is replaced by computed ICAO address. Hence, it is not recommended to use the pyModeS ICAO inference method on OpenSky raw messages.

Table 2: Parameter decoded automatically by `pymodes-opensky` library

(a) EHS parameters

BDS	Parameters	variable
40	MCP/FCU selected altitude	`selalt40mcp`
	FMS selected altitude	`selalt40fms`
	Barometric pressure setting	`p40baro`
50	Roll angle	`roll50`
	True track angle	`trk50`
	Ground speed	`rtrk50`
	Track angle rate	`gs50`
	True airspeed	`tas50`
60	Magnetic heading	`hdg60`
	Indicated airspeed	`ias60`
	Mach number	`mach60`
	Barometric altitude rate	`vr60baro`
	Inertial vertical velocity	`vr60ins`

(b) Meteorological parameters

BDS	Parameters	variable
44	Wind speed	`wind44spd`
	Wind direction	`wind44dir`
	Static air temperature	`temp44`
	Average static pressure	`p44`
	Turbulence	`turb44`
	Humidity	`hum44`
45	Turbulence	`turb45`
	Wind shear	`ws45`
	Microburst	`mb45`
	Icing	`ic45`
	Wake vortex	`wv45`
	Static air temperature	`temp45`
	Average static pressure	`p45`
	Radio height	`rh45`

ICAO addresses saved alongside the messages enables the fast extraction of flight information using the `pymodes-opensky` library.

Among all raw messages, the most interesting part of the information for many ATM studies are included in EHS messages, which contain the intent, turn rates, and various types of airspeeds. This type of information is also most desired by air traffic controllers. Hence, most commonly interrogated by the SSR in many regions. Meteorological reports are not often interrogated. However, whenever they are available, they can provide valuable information such as wind, air pressure, and air temperature. Both EHS and meteorological messages from OpenSky can be conveniently processed by `pymodes-opensky`. Table 2 is a list of parameters from EHS and meteorological reports directly supported by the `pymodes-opensky` library.

The library provides a set of interfaces connecting the Mode S decoder with the OpenSky historical data. Specifically, it allows users to automatically extract and discover information contained in EHS and meteorological reports if they are available. The library has two levels of interfaces to the OpenSky historical database, both through the Impala shell. The lower level interface returns data directly (with some extra columns) from the database. The query supports the extraction of both decoded ADS-B information (decoded by OpenSky) and raw Mode S messages. The following example code shows how to establish a low-level query using `pymodes-opensky`. The ICAO address and the geographical boundaries can be passed as optional parameters to the query. By altering the query type from `adsb` to `raw`, one can obtain raw Mode S messages instead of decoded ADS-B data from the database.

```
from pymodes_opensky import OpenskyImpalaWrapper

opensky = OpenskyImpalaWrapper()

df = opensky.query(
    type="adsb",                    # or "raw"
    end="2019-10-01 09:30:00"       # UTC time
    icao24=["4844C6"],              # optional
    bound=[30, -20, 65, 20],        # optional
)
```

Based on the previous low-level interface, the higher-level interfaces generate queries using the combination of time and ICAO filters. After raw messages from the historical database are obtained, they are decoded automatically using the `EHSHelper` and `MeteoHelper` based on the query results of the `OpenskyImpalaWrapper`. All (or a subset) of the available parameters from the previous Table 2 are provided directly to the users. Multiple ICAO addresses can be used in the same query to increase efficiency. By default, all BDS 40, 50, and 60 parameters are decoded. However, the desired list of BDS codes can be specified. In the following example, the process of obtained flight states contained in EHS messages is shown:

```
from pymodes_opensky import EHSHelper

ehs = EHSHelper()

# optional: change from default BDS 40/50/60
ehs.require_bds(["BDS50", "BDS60"])

df = ehs.get(
    icao24=["4844C6"],
    start="2019-10-01 07:30:00" # UTC time
    end="2019-10-01 09:30:00"   # UTC time
)
```

Similarly, meteorological information can also be obtained and decoded using the meteorological interface. By default, only BDS 44 messages, the meteorological routine reports, are obtained. However, one can enable the decoding of BDS 45 messages, the meteorological hazard reports, by providing an optional parameter to the query. An example is shown as follows:

```
from pymodes_opensky import MeteoHelper

meteo = MeteoHelper()
df = meteo.get(
    start="2018-07-19 15:00:00",
    end="2018-07-19 15:10:00",
    icao24=["49d304", "4007f9"],
    include45=True, # optional
)
```

4 Experiment and results

4.1 Demonstration with a single flight

In this section, an example flight is used to demonstrate the decoding capabilities of the `pymodes-opensky` library. Figure 2 shows the ground track that is constructed based on the decoded ADS-B information from OpenSky historical database through the Impala shell. A flight (KL1793) from Amsterdam to Munich on October 1st, 2019 is selected. We choose this flight due to the good receiver coverage in the areas it flew over. Thus, the trajectory data through the entire flight can be obtained.

At the same time information from EHS is decoded and shown along with relevant ADS-B flight states in Figures 3 and 4. In Figure 3, the barometric altitude profile from ADS-B and selected altitude profile from the selected vertical intent report (BDS 40) of EHS are plotted. The outliers in the trajectory data have been filtered out using a moving median filter with a window size of 11 samples. In this figure, the selected altitude of this trajectory is shown in red alongside the actual altitude flown which is shown in black.

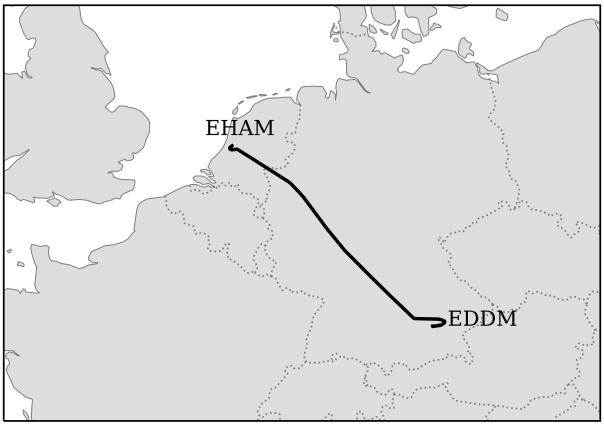

Figure 2: Ground track of the demonstration flight (KL1793, October 1st, 2019)

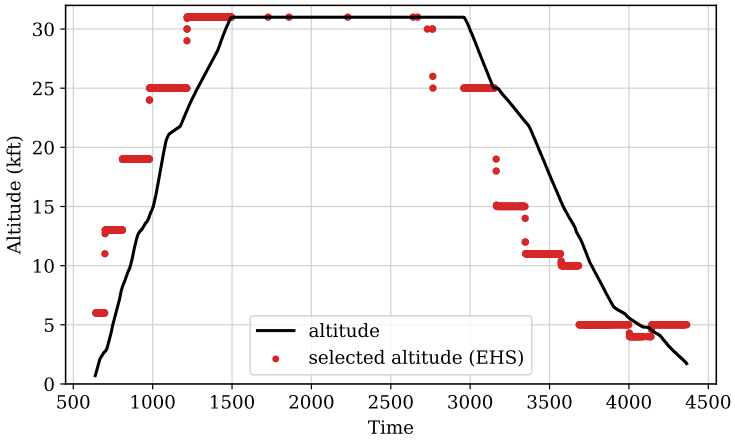

Figure 3: The ground track of the demonstration flight

In Figure 4, different types of speeds from EHS track and turn reports (BDS 50), as well as speed and heading reports (BDS 60), are plotted alongside the ground speed from the ADS-B reports. The data is filtered using the same median filter as mentioned earlier. The two ground speed profiles (in black and red) overlap with each other. All four types of speed in EHS (ground speed, true airspeed, indicated airspeed, and Mach number) can be accurately decoded using the `pymodes-opensky` library.

4.2 Comm-B statistics from a large-scale dataset

In this part of the experiment, we want to investigate the frequency of different Comm-B report types in different world regions using the raw Mode S data from the OpenSky historical database. The goal is to exploit the BDS inference capability of `pyModeS` to analyze the usage of ELS and EHS services at a larger scale.

First, four groups of data are obtained on October 1st, 2019 at 00:00, 06:00, 12:00 and

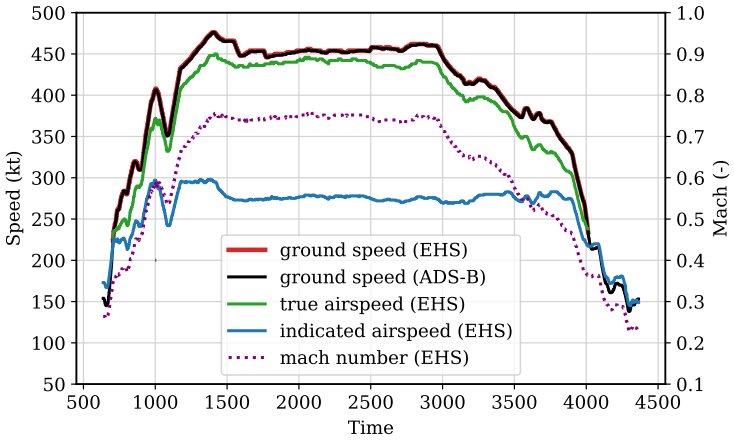

Figure 4: The ground track of the demonstration flight

18:00 hours. Each group contains two minutes of decoded ADS-B and raw Mode S data from the entire OpenSky network. Then, the locations of the raw messages are combined with the position from the ADS-B data based on the ICAO addresses. After that, four snapshots are merged into a single dataset, where the BDS codes of all messages are inferred using `pyModeS`. Finally, the statistics of the BDS distribution in different geographical regions are shown in Table 3. The message numbers only refer to the messages collected by OpenSky network. These values depend on the actual receiver coverage in different regions. For large countries with low coverage, the messages number can be much lower than smaller regions with full coverage.

In this table, only regions with more than 2000 messages are included, which filters out regions that do not actively initiate Mode S interrogations, do not have a sufficient number of flights, or do not have enough receiver coverage. Messages identified as `Empty` do not contain any information (all zero bits) except the altitude or identity code. Messages identified as `N/A` either have multiple possible BDS codes after the inference or do not have many matching BDS codes from ELS or EHS. There exists a major difference across the continents. In Europe, the most common interrogated messages are EHS messages. However, in North America or Asia, the interrogations are different, in general, and more ELS messages are interrogated. Based on the percentages of messages with unknown types, it is likely that a greater number of other types of messages are being interrogated in these regions.

5 Conclusion

In this paper, we discussed an approach for exploring a large amount of undecoded information contained in the OpenSky historical database. Based on the existing open-source tool `pyModeS`, a new Python library `pymodes-opensky` is developed to automatically retrieve data from the OpenSky using the Impala shell and to decode these raw Mode S messages.

The primary message type investigated in this paper is the Mode S Comm-B message, which is originated by secondary surveillance radar interrogation. They contain additional valuable flight information to the commonly used ADS-B messages. Upon successful inference and decoding using `pyModeS`, we were able to obtain information such as target altitude, true airspeed,

Table 3: Global statistics of BDS interrogations (decimal numbers represent percentages)

Country	Messages	BDS10	BDS17	BDS20	BDS30	BDS40	BDS50	BDS60	Empty	N/A
Germany	481062	2.8	3.3	5.9		27.5	19.4	37.1	0.7	3.4
United Kingdom	427782	2.5	1.9	10.2	1.4	25.6	24.0	30.6	0.3	3.5
France	408353	1.4	1.2	4.3		31.2	17.1	40.2	2.3	2.2
Italy	202207	2.9	2.1	5.0	0.1	31.9	10.8	42.3	2.9	2.0
Netherlands	149335	1.3	1.3	3.9		24.4	20.4	43.8	0.4	4.4
Austria	115974	4.3	4.6	6.4		22.5	23.9	33.7	3.0	1.7
Poland	112555	2.3	4.1	4.1		25.0	25.7	31.0	2.2	5.6
Belgium	108786	1.2	0.9	3.4		25.6	21.9	43.7	0.3	2.8
Sweden	105799	1.2	1.5	12.8	4.0	26.0	12.7	37.1	0.4	4.2
Spain	102655	4.6	1.8	4.8		37.7	13.2	34.2	2.1	1.7
Australia	83523	0.5	0.7	1.8		29.0	30.9	30.6	0.1	6.5
Czechia	83062	2.8	3.8	4.3		24.6	26.6	32.7	3.0	2.3
Hungary	67141	2.3	2.1	4.5		25.2	27.1	33.7	3.0	2.2
Ireland	54151	15.2	0.9	3.4	11.4	19.7	20.5	25.5	0.4	3.1
Switzerland	54075	1.8	1.6	4.3		30.0	13.1	44.5	0.9	3.9
Turkey	53634	2.1	1.0	20.9		25.2	24.7	24.8	0.1	1.4
Russian Federation	52213	8.2	2.9	13.2		23.1	21.7	24.5	0.5	5.8
Denmark	49824	2.1	2.0	10.5		24.8	23.0	34.1	0.2	3.3
Croatia	46704	2.6	2.0	5.9		28.5	20.8	35.7	2.6	2.0
Slovakia	45991	2.1	2.0	4.7		23.0	28.1	35.0	3.2	1.9
Norway	39069	0.8	0.5	17.6		24.4	21.0	30.9	0.1	4.7
Bulgaria	32018	2.3	1.2	11.5		26.9	27.3	29.4	0.2	1.3
Romania	30783	2.2	1.4	9.8		26.6	26.9	30.5	0.6	2.0
India	28036	2.3	1.6	3.7	0.8	26.2	28.7	28.1	0.5	8.3
Portugal	27960	4.0	1.6	15.3		29.2	17.6	30.7		1.7
United Arab Emirates	24470	8.6	0.4	6.4	7.9	24.3	22.3	22.2	0.9	7.0
Serbia	23135	2.2	1.6	4.5		30.6	24.7	33.6	2.1	0.7
Latvia	22538	1.6	4.3	12.3	2.0	23.9	13.1	33.2	2.3	7.3
Belarus	21956	3.5	4.8	7.0		29.4	20.5	26.3	3.2	5.2
Bosnia and Herzegovina	21107	2.1	1.7	5.6		29.7	22.2	35.7	3.0	0.1
Korea	20393	6.7	4.6	25.0	2.0	21.7	15.7	19.7		4.6
Malaysia	20286	11.3	1.5	13.8	5.7	22.8	19.3	19.9	0.1	5.6
Lithuania	20050	2.0	4.2	10.6		24.4	18.5	31.4	2.9	6.0
Qatar	18863	5.2	0.6	1.7	3.9	26.8	25.3	24.5	2.7	9.3
Slovenia	18482	4.5	4.6	8.2		21.3	22.8	34.9	3.5	0.3
Greece	18047	3.0	2.0	15.5		27.1	24.1	24.7	1.1	2.5
United States	17943	20.5	8.6	14.6		8.4	13.2	12.5	11.8	10.5
Estonia	17475	5.6	3.5	11.1	2.6	22.4	10.3	35.2	1.9	7.4
Finland	15094	9.8	2.4	10.8	3.3	25.9	8.2	29.8	1.6	8.2
China	14190	3.3	2.8	4.7		40.6	28.9	16.1	0.1	3.4
New Zealand	14038	6.2	4.4	15.0	5.9	11.1	23.7	19.6	5.7	8.3
Thailand	11823	1.7	1.2	3.2	0.1	31.5	25.3	32.0	0.4	4.7
Iran	10900	6.1	0.4	2.5	4.7	27.7	27.5	27.5	0.3	3.3
Luxembourg	10364	1.9	1.9	5.8		26.5	15.5	45.6	0.8	2.0
Indonesia	10145	15.8	1.6	3.9	10.5	23.9	19.3	18.1	0.7	6.2
Japan	10129	18.7	15.7	2.0	0.2	20.6	17.9	18.7		6.2
Ukraine	8455	2.0	1.1	8.4		29.3	24.2	25.6	0.9	8.6
Albania	7720	3.5	2.6	6.9		30.6	20.1	33.4	2.2	0.7
Cyprus	5852	1.9	1.1	9.3		27.7	29.1	30.5	0.2	0.3
Åland Islands	5395	1.9	2.1	11.1	6.8	25.2	4.7	39.7	0.9	7.7
Isle of Man	4814	10.1	1.2	9.4	6.6	18.3	21.8	28.7	0.3	3.7
Philippines	4467	0.8	0.5	0.5		31.9	30.1	31.2	0.2	4.8
Peru	3902	3.0	1.7	11.3		26.8	27.6	28.0		1.6
Israel	3682	2.7	1.1	24.4		22.8	23.4	23.1	0.5	2.0
Guernsey	3423	0.5	1.6	2.0		32.8	28.4	34.3	0.2	0.2
Taiwan	3119	8.0	16.8	15.7	1.5	20.1	14.1	14.8	0.2	8.8
Korea	2853	2.4	7.1	13.7	2.0	23.4	25.4	23.6	0.1	2.4
Kazakhstan	2652	0.4	0.3	75.3		7.4	7.4	7.6	0.1	1.5
Canada	2611	8.8	5.5	35.7		11.9	7.9	20.9	0.4	9.0
Saudi Arabia	2273	1.6	0.5	1.8		31.9	28.6	28.4	1.1	6.1

indicated airspeed, Mach number, magnetic heading, roll angle, and turn rate. Meteorological information in Mode S can also be obtained using a similar interface, whenever these reports are available.

One of the primary limitations discovered during the study of this paper is the time needed to obtain query results through the Impala shell. It can take a long time to process large queries through the Impala shell interface. Thus, the ICAO address filter is mandatory in the higher-level interface, which acts as a measure to reduce the size of query results. This end user-orientated approach is also not efficient for the community as a whole, as the same raw data may need to be downloaded and decoded by different users. A potential future recommendation is to integrate `pyModeS` on the receiver side so that Mode S messages can be decoded and transmitted to OpenSky directly.

Currently, one of the optimal use cases for the `pymodes-opensky` library is to obtain additional information for specific trajectories, as shown in section 4.1 of this paper. Furthermore, in section 4.2, the inference capability of `pyModeS` library is verified with the experiment using the global data set. A future improvement to these libraries will be focused on increasing decoding speed and providing more customized interfaces for Mode S decoding.

References

[1] John L Baker, VA Orlando, WB Link, and WG Collins. Mode s system design and architecture. *Proceedings of the IEEE*, 77(11):1684–1694, 1989.

[2] Jeffrey L Gertz. Fundamentals of mode s parity coding. Technical report, Massachusetts Institute of Technology, Lincoln Laboratory, 1984.

[3] Robert D Grappel, Garrett S Harris, Mark J Kozar, and Randall T Wiken. Elementary surveillance (els) and enhanced surveillance (ehs) validation via mode s secondary radar surveillance. *Project Report ATC-337, Lincoln Lab., MIT*, 2008.

[4] ICAO. *Annex 10 to the Convention on International Civil Aviation, Aeronautical Telecommunications*. International Civil Aviation Organization, 2002.

[5] ICAO. *Technical Provisions for Mode S Services and Extended Squitter*. International Civil Aviation Organization, 2008.

[6] ICAO. *Technical Provisions for Mode S Services and Extended Squitter, 2nd Edition*. International Civil Aviation Organization, 2012.

[7] Xavier Olive. traffic, a toolbox for processing and analysing air traffic data. *Journal of Open Source Software*, 4(39):1518, 7 2019.

[8] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. Bringing up opensky: A large-scale ads-b sensor network for research. In *Proceedings of the 13th international symposium on Information processing in sensor networks*, pages 83–94. IEEE Press, 2014.

[9] Junzi Sun, Henk AP Blom, Joost Ellerbroek, and Jacco M Hoekstra. Particle filter for aircraft mass estimation and uncertainty modeling. *Transportation Research Part C: Emerging Technologies*, 105:145–162, 2019.

[10] Junzi Sun, Jacco M Hoekstra, and Joost Ellerbroek. Aircraft drag polar estimation based on a stochastic hierarchical model. In *8th International Conference on Research in Air Transportation*, 2018.

[11] Junzi Sun, Huy Vû, Joost Ellerbroek, and Jacco M Hoekstra. pymodes: Decoding mode-s surveillance data for open air transportation research. *IEEE Transactions on Intelligent Transportation Systems*, 2019.