Journal of Open Aviation Science (2024), Vol.**2**
doi:10.59490/joas.2024.7882

JOAS

PROCEEDINGS | *The 12th OpenSky Symposium*

Filtering Techniques for ADS-B Trajectory Preprocessing

Xavier Olive [*,1,2] Jan Krummen [3] Benoit Figuet [3,4] and Richard Alligier [2]

[1]ONERA DTIS, Université de Toulouse, France
[2]Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, Toulouse, France
[3]Centre for Aviation, Zurich University of Applied Sciences, Winterthur, Switzerland
[4]SkAI Data Services, Zurich, Switzerland
*Corresponding author: xavier.olive@onera.fr

(Received: 25 Oct 2024; Revised: 18 Mar 2025; Accepted: 20 Mar 2025; Published: 21 Mar 2025)
(Editor: Junzi Sun; Reviewers: Antonio Franco and Matthias Schäfer)

Abstract
This paper addresses the issue of noisy, uncertain and quantized data in crowdsourced ADS-B and Mode S data and explores propositions of implementations of preprocessing techniques to address them. After a description of ADS-B data focused on sources of noise and uncertainty, we present in detail a selection of filters that have been implemented in the traffic library, and widely used in the constitution of open datasets used in further research. We also illustrate the results of the filtering with trajectory data collected by The OpenSky Network and by inexpensive RTL-SDR receivers.

Keywords: ADS-B; filtering; anomaly detection; outlier detection

1. Introduction

The increasing coverage of crowdsourced open aircraft trajectory data has brought Automatic Dependent Surveillance–Broadcast (ADS-B) data to the forefront as a primary source of open data for aviation research. This data is extensively used in various research applications, including the extraction of operational milestones [1, 2], the optimization of operations [3], or various safety [4, 5], security, or environmental [6, 7, 8] concerns.

Standard avionics systems often use data fusion techniques to combine information from multiple sensors, such as *Global Navigation Satellite System* (GNSS, very accurate but subject to signal multipath, atmospheric conditions), *Inertial Navigation Systems* (mostly accelerometers and gyroscopes), barometric altimeters and Pitot systems, to provide a more accurate and robust estimate of the aircraft's state. This fused data is then used to generate the ADS-B messages broadcast to ground stations and other aircraft.

From a data analyst perspective, all sources of trajectory data, such as radar data, quick access recorder (QAR) data, and ADS-B, are subject to noise (from measurement or transmission), uncertainty (which can be estimated) and quantization. ADS-B, in particular, is known for its error corrections, including *cyclic redundancy checks* (CRC), and for broadcasting uncertainty information together with the positional and velocity estimates.

Moreover, aggregating several sources of information for similar signals (such as barometric and geometric altitude), and collecting it from a network of crowdsourced receivers of various quality (due to the quality of the antenna, the location of the receivers and the implementation of the demodulation process) also introduces potential errors in the data. One or more receivers can introduce corrupted messages to the network, and the timestamping of messages (at the receiver's location) can be a source of more errors in the data.

Errors, uncertainty, and noise present a significant challenge for further research in aviation data processing: the need for filtered and clean data. Without proper filtering, erroneous data can lead to inaccurate conclusions and unreliable research outcomes, esp. when looking at interactions between aircraft (e.g., aircraft deconfliction, collision risk modelling) or estimations of further quantities (e.g., fuel flow and other pollutants).

To address these challenges, two main categories of filters are typically employed. The first category focuses on the detection and exclusion of corrupted or irrelevant data, utilizing mostly outlier detection methods [9]. The second category aims to mitigate quantization effects and leverage the measurements in uncertainty, often employing Kalman-based filters [10, 11] to fill in data gaps after resampling and smoothing the data.

The remaining of the paper is structured as follows. In Section 2, we present the types of information collected by The OpenSky Network, including ADS-B and other relevant Mode S messages, and highlight the potential sources of noisy information. In Section 3, we detail various methods to filter data, particularly positional information, noisy signals, and quantized information. Section 4 refers to the associated implementations within the `traffic` library [12].

2. Sources of data and errors

Data demodulated from the 1090 MHz frequency and decoded arrives in various downlink formats, which are detailed comprehensively in [13]. Among these formats, the most relevant for typical applications are the following:

- **DF 4**: Surveillance Altitude Reply;
- **DF 5**: Surveillance Identity Reply;
- **DF 17**: Extended Squitter (ADS-B);
- **DF 20**: Comm-B Altitude Reply;
- **DF 21**: Comm-B Identity Reply.

Of these, only ADS-B (DF 17) includes a *cyclic redundancy check* (CRC) to verify the integrity of the binary message. In the other downlink formats, the CRC instead returns the ICAO 24-bit address of the aircraft's transponder. Even with CRC checked message, it is always difficult to guarantee that a received message has not been corrupted. Messages collected by many receivers in a crowdsourced network have a higher chance of being valid.

2.1 Incorrect time information

Mode S messages do not include any timestamp information. Instead, timestamps are typically appended by the receiver based on the time of message reception, rather than the time of transmission from the aircraft. This approach usually presents no major issues when the data comes from a single receiver. However, challenges arise when using data from a crowdsourced service.

Some receivers are equipped with GPS clocks, which accurately timestamp messages, making them suitable for multilateration purposes. However, other feeders may incorrectly timestamp their messages due to clock drift, relying on inaccurate system clocks or cheap RTL-SDR receivers which are prone to affine drift. Additionally, network latency can introduce further errors in the reception time. As a result, properly timestamping received messages poses a significant challenge.

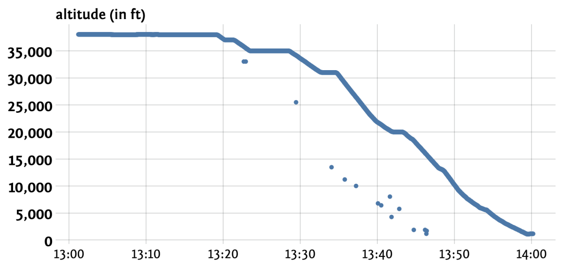


Figure 1. Altitude measurements in this part of the flight suggest that one of the receivers has a clock synchronisation issue.

Figure 1 shows some artefacts which are very likely to be due to timestamp issues.

2.2 Incorrect altitude measurements

Altitude information is included in several downlink formats (specifically DF 4, DF 17, and DF 20), which can lead to inconsistencies when this data is aggregated—such as when it is merged into a state vector table, as is commonly done by many providers, including the OpenSky Network. The OpenSky Network supplies both raw data and (in some cases) partially decoded tables for all downlink formats. In addition, it offers higher-level abstractions, most notably the `StateVectorsData4` table, which consolidates information from various sources. However, the altitude data, from DF 4 and DF 20 in particular, is prone to erratic data points. (see Figure 2)

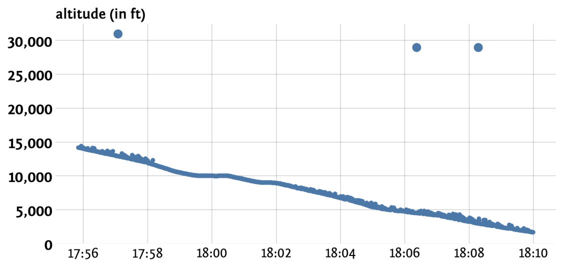


Figure 2. Some obviously erroneous altitude data points in this flight should be invalidated.

2.3 Quantization artefacts

Physical quantities measured by aircraft are processed by onboard systems, which fuse and filter data from various sources. These quantities are then quantized to fit within a limited number of bits before being transmitted via Mode S messages. For example, depending on the type of message, altitude can be encoded using 11 bits (in the form of *Gillham code*, with increments of 100 or 500 feet) or 13 bits (with increments of 25 feet). This quantization process can introduce threshold effects, making altitude measurements appear unnecessarily noisy. (see Figure 3)

2.4 Incorrect measurements in BDS 5,0 and BDS 6,0

Another source of error arises from the decoding of DF20 and DF21 messages. Both of these formats include a BDS (*Comm-B Data Selector*) payload. Some of these BDS codes are also present in

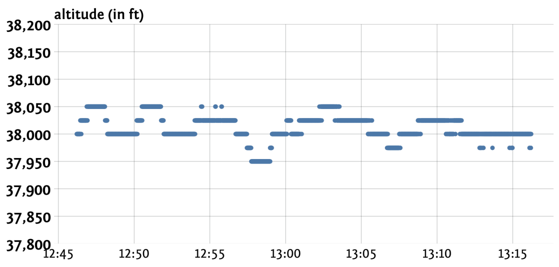


Figure 3. The altitude signal is subject to measurement noise and quantization issues. This flight is most likely flying a constant altitude of 38,000 ft (FL 380).

ADS-B messages, with a flag in the header (the `typecode`) indicating the type of information being decoded. ADS-B messages are broadcast regardless of who will decode it, so this flag is essential for interpreting the data. DF 20 and DF 21 messages, however, are sent in response to a request from a *Secondary Surveillance Radar* (SSR), which knows how to decode the data it requested, even though this information is not explicitly stated in the message. Without access to the uplink message, we can only hypothesize about the content, as outlined in the paper accompanying the pyModeS library [14]. In particular, distinguishing between BDS 5,0 (*Track and Turn Report*) and BDS 6,0 (*Heading and Speed Report*) is challenging without extensive context from previously received messages for the same aircraft. Misinterpreted BDS codes can lead to further errors in the trajectory data. (see Figure 4)

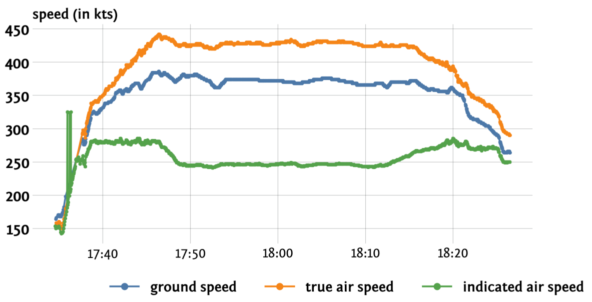


Figure 4. In the speed profile of this flight, some messages are incorrectly decoded as BDS 6,0, leading to incorrect indicated airspeed (IAS) values during the climb. The true air speed (TAS) is available in BDS 5,0 messages.

2.5 Inherent noise in some measurements

Some information provided by ADS-B messages is directly measured, while other physical quantities are computed based on those measurements. For example, the vertical rate (usually expressed in feet per minute) cannot be directly measured. In BDS 6,0 messages, two types of vertical rate are provided: the *barometric altitude rate* and the *inertial vertical velocity*. Only the second one is also present in ADS-B messages (specifically, BDS 0,9 messages)

The *barometric altitude rate* is derived from the air data system, likely by taking the derivative of the altitude signal, which is measured from static pressure. This approach is prone to noise, and the differentiation process tends to amplify the noise in the signal. On the other hand, the *inertial vertical velocity* is computed by combining barometric information with vertical acceleration (the

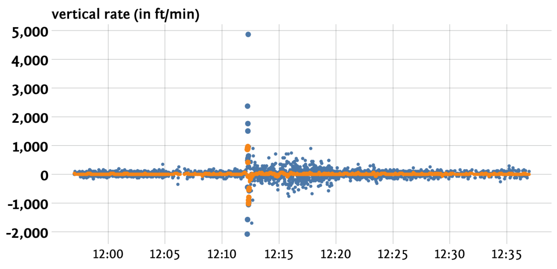


Figure 5. The barometric altitude in rate (blue) is in general more noisy than the inertial vertical velocity (orange). Stronger noise is often a sign of experiencing turbulence. The clear anomalies near 12:12 UTC was an air pocket experienced by the aircraft (Source: experience of the author onboard the aircraft)

second derivative of altitude) from the inertial system, resulting in a filtered and smoother version of the vertical rate. (see Figure 5)

2.6 Errors in position messages

For most aircraft, airborne position is determined using a combination of GNSS signals and inertial data[1]. However, when the aircraft is on the ground, GNSS signals are susceptible to interference or multipath, leading to faulty or lack of measurements. Additionally, certain inertial data used for corrections in flight cannot be applied when the aircraft is on the ground, which can further degrade the accuracy of ADS-B positional data. (see Figure 6a)

Since aircraft position estimates rely heavily on GPS, they are inherently vulnerable to GPS *Radio Frequency Interferences* (RFI), such as *jamming* or *spoofing* [15]. During GPS jamming, the aircraft loses access to accurate GPS-based positioning and must instead rely on less precise systems, such as ground-based navigation aids or inertial navigation. In the case of GPS spoofing, where falsified GPS signals are generated, the aircraft may calculate an incorrect position, which is then broadcast in the ADS-B messages. (see Figure 6b)

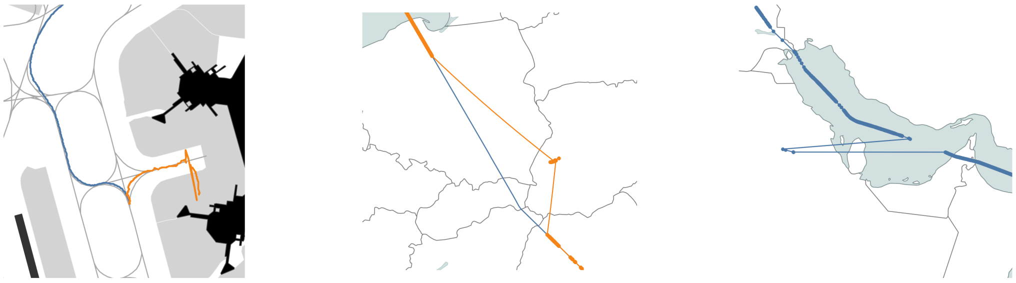
One of the most common causes of incorrect positions in ADS-B state vectors is the method of encoding positional data. Latitude and longitude are encoded using 17 bits in the *Compact Position Reporting* (CPR) format, which efficiently represents positions with high resolution while using fewer bits. CPR balances global positional ambiguity with local accuracy. Two types of position messages—identified by odd and even frame bits—are broadcast alternately. Positions can be decoded using either a single message along with a previously known position, or by combining both odd and even messages.

If the positional information becomes outdated, the decoding process can produce an incorrect position. (see Figure 6c) To prevent this, message timestamps are typically checked before CPR decoding, but issues can still arise in some edge cases. Furthermore, the timestamping challenges mentioned in Section 2.1 can increase the likelihood of decoding faulty positions, which must be detected and filtered out.

2.7 Noise added by the post-processing

Some of the resulting data may include irrelevant data points due to the way it has been aggregated. By offering higher-level abstractions to data analysts, data providers can inadvertently introduce

[1]The combination highly depends on the aircraft type and its specific avionics systems. Airbus aircraft typically use data fusion, whereas Boeing aircraft rely on it less for ADS-B OUT.



(a) Localization imprecision when aircraft is on the ground, esp. during pushback (orange)

(b) Flight THY9BP on Sep. 17, 2024, spoofed to a location near Lviv, Ukraine; trajectory in blue is from FlightRadar24 (MLAT)

(c) CPR decoding errors due to unlucky timing of received position messages (above the Arabic gulf)

Figure 6. Errors in position messages due to poor GPS precision, multipath, spoofing and CPR decoding errors.

artefacts, making it more challenging to apply consistent filtering across different data sources.

For instance, the OpenSky Network provides a `StateVectorsData4` table which contains values which are forward propagated. Since the propagation logic is relatively straightforward, the `traffic` library can easily offer functions to reverse-engineer and invalidate these propagated values.

Other providers like FlightRadar24 generate state vectors that could to be derived from Kalman filters. This additional processing layer makes it more difficult to develop robust filtering strategies, as the implementation details of the first-layer filters are unknown. As a result, these post-processed trajectories can sometimes fail in edge cases, complicating the task of further filtering and analysis.

3. Description of filters

In this section, we present various categories of filters that perform well on raw ADS-B trajectories or state vectors sourced from the OpenSky Network database. Specifically, we focus on filters based on rolling windows, derivative filters, clustering filters, and a particular implementation of a consistency filter. We close the section with a presentation of Kalman filters.

3.1 Filters based on rolling windows

A *sliding window* or *rolling window* filter works by moving a fixed-size window (the *kernel*) across the data and performing operations on the values within the window at each step. Two widely used methods for reducing noise in trajectory data are the moving mean and moving median filters. These filters calculate either the mean or the median of the data points within the window and use that value to replace the central point. Both approaches are effective in smoothing noise and reducing smaller peaks, as demonstrated in Figure 7.

The moving median, in particular, is highly effective when large outliers are present, as it is robust against extreme values. In contrast, the moving mean can become unsuitable in such situations because it is sensitive to outliers, leading to poor filtering performance. For both filters, the window width is the key tuning parameter: wider windows produce stronger smoothing effects.

Another useful approach (implemented by default in the `traffic` library) applies a median filter that computes thresholds based on a sliding window, identifying and replacing values that exceed acceptable thresholds with `NaN` values. This method is particularly helpful for handling extreme noise without relying on averaged values.

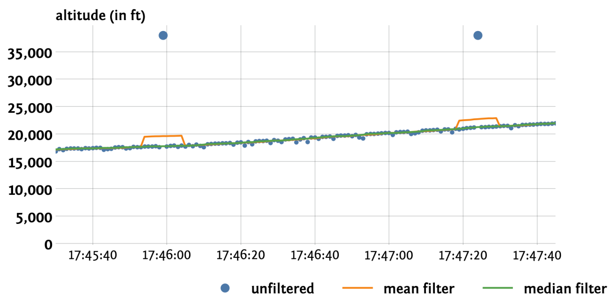


Figure 7. Moving mean and moving median with a kernel size of 11 applied to altitude data of a flight showing noise and two outliers of large magnitude.

Additionally, these filters can be chained together for even better performance, allowing them to address different types of noise and inconsistencies more effectively. Combining multiple filters often results in cleaner, more reliable data processing.

3.2 Derivative filter

While single outliers are a common type of erroneous data points, another challenge arises when multiple consecutive erroneous data points occur. The moving median filter can only effectively remove errors if the number of erroneous points is less than half the kernel size of the filter.

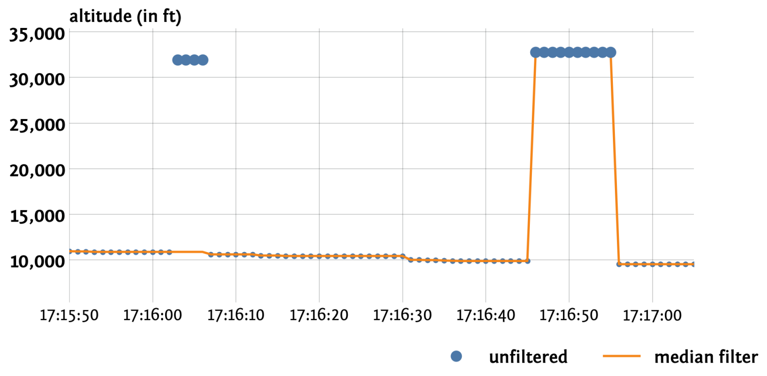


Figure 8. Two examples of multiple consecutive erroneous data points, where one group is filtered by the moving median while the other is not.

In Figure 8, the first group of outliers, consisting of four data points, is successfully filtered out by a moving median with a window size of 11, as the median remains within the range of the true data. However, in the second group, which contains ten consecutive erroneous points, the moving median aligns with the erroneous values and fails to filter them out.

Although increasing the window width could resolve this issue, it also carries the risk of oversmoothing the data, potentially distorting the correct values. Thus, a careful balance between window size and filtering effectiveness is essential to avoid compromising the integrity of the true data.

A specialized filter has been developed to address cases involving consecutive erroneous data points. This filter utilizes timestamp information and parameter values to compute the absolute values of the first and second derivatives of a given parameter. In the context of altitude data, these correspond to absolute vertical velocity and absolute vertical acceleration, respectively.

Users can define threshold values for these derivatives, as well as a kernel size, to control the filtering process. Any data point where the first or second derivative exceeds the specified thresholds is marked as an outlier. Additionally, if two identified outliers are within a distance smaller than the defined window width, all data points between them are also flagged as erroneous and removed.

As shown in Figure 9, this method effectively handles multiple consecutive outliers. Because the first and second derivatives represent interpretable physical quantities, threshold values can be set to reflect the limits beyond which valid observations are not possible. This ensures the removal of all erroneous data points beyond these limits.

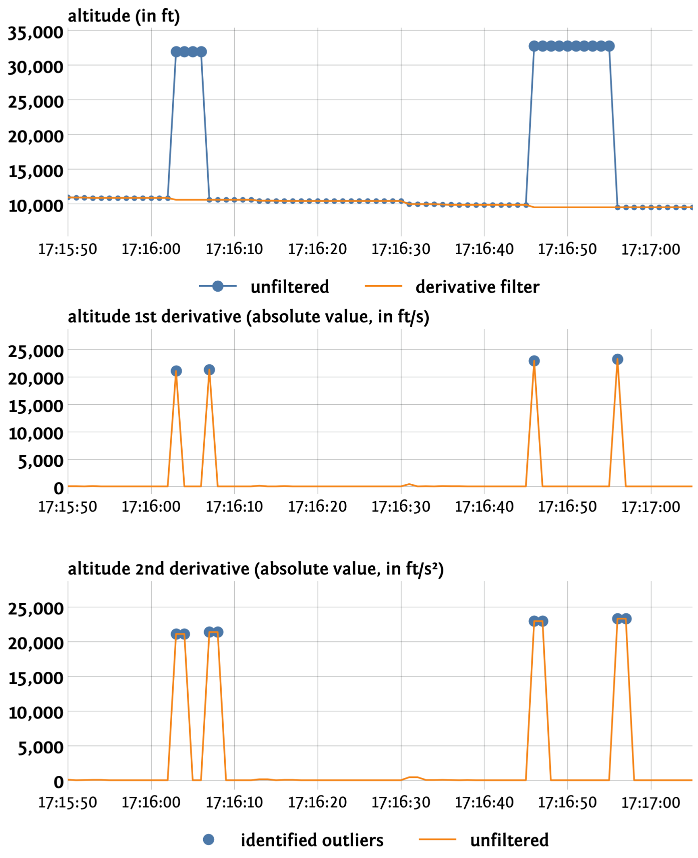


Figure 9. Illustration of the derivative filter concept, which calculates the first and second derivatives of a parameter and removes data points where these derivatives exceed a defined threshold. It also removes any data points between two closely spaced outliers.

3.3 Cluster filter

One situation not addressed by the previously discussed filters is when a series of consecutive erroneous data points occurs at the beginning or end of the data, as shown in Figure 10. These cases are not handled by the derivative filter because they start or end with erroneous values, resulting in only one peak in the derivatives. As a result, the entire sequence of bad data points remains unfiltered.

To address this issue, a third type of filter has been developed using a clustering approach. This filter allows the user to define thresholds for both time and parameter values, as well as a minimum cluster size. The algorithm calculates the time and parameter differences (deltas), and wherever these exceed the specified thresholds, a cut is made, separating the data into clusters.

Data points that belong to clusters smaller than the defined minimum size are then removed from the dataset. In the example shown in Figure 10, with a time threshold of one minute and a parameter threshold of 1000, two clusters are formed. The first cluster contains the 15 erroneous data points. Therefore, by setting the minimum cluster size above 15, these erroneous points would be successfully filtered out.

When properly tuned, this clustering filter can effectively remove consecutive outliers and also handle individual outliers of limited magnitude.

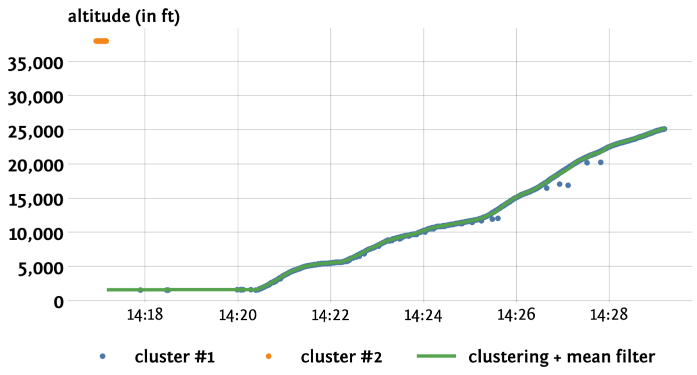


Figure 10. Example of altitude data containing an initial series of erroneous data points, identified by the cluster filter as a distinct cluster. With an appropriately set minimum cluster size, the filter will remove these erroneous points.

3.4 Consistency filter

The filters described in previous subsections focus on detecting erroneous data by examining one variable at a time. However, this approach does not account for inconsistencies between two related variables, such as a parameter and its derivative. This principle is illustrated in Figure 11, where, around 12:20, the altitude data suggests a sharp descent followed by a smoother climb. This portion of the trajectory appears erroneous when compared to the vertical rate and is therefore ruled out.

To check for consistency between two variables (typically a value and its derivative), we calculate the gap between a value v_j at time t_j and the extrapolated value from v_i at time t_i. The gap is defined as the absolute difference:

$$\left| v_j - v_i - \frac{1}{2}(t_j - t_i)\left(\frac{dv}{dt}_i + \frac{dv}{dt}_j\right)\right|$$

If this gap exceeds a threshold multiplied by $\left|t_j - t_i\right|$, then the values at t_i and t_j are considered inconsistent. When this happens, one or both values must be discarded, as keeping both would lead to inconsistencies. Deciding which value to discard is challenging. A greedy approach can be used, starting from the first value in time and assuming it is correct. As we move forward in time, we keep or discard subsequent values based on the computed gap. However, if the initial value is incorrect, this approach may discard many legitimate values.

A more optimal solution involves representing the consistency between values as a graph. Each node corresponds to a value at time t_i, and there is a directed edge between nodes i and j if $t_i < t_j$ and the values at these times are consistent. Once the consistency graph is constructed, every path within the graph represents a sequence of consistent values. The final step is to retain the values that are part of the longest path in the graph, ensuring that the fewest possible values are discarded while preventing any inconsistency.

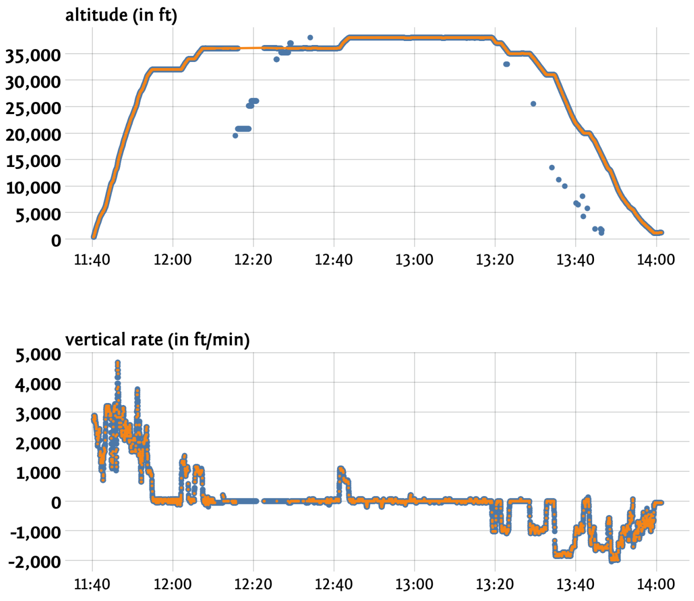


Figure 11. At around timestamp 12:20, an inconsistency is observed between the altitude variation and the vertical rate. As a result, the consistency filter discarded the inconsistent data points.

3.5 Kalman filters

Kalman filters help to predict the most likely state of a system by using noisy measurements to improve and update predictions. In the following, we consider a model for the lateral path of an aircraft, as the vertical path is much simpler on two dimensions. The model implemented in the `traffic` library deals with the three dimensions in one pass, although they could be handled in sequence without loss of generality.

A two-dimensional model for a Kalman filter ADS-B messages provide a sequence of four measurements of the aircraft's state at each second (with potentially invalid `NaN` values): latitude, longitude, ground speed (in kts) and track angle (in degrees). In order to simplify the model and equations, we convert these measurements to a four-dimensional state vector $X = [x, y, \dot{x}, \dot{y}]$ sequence, in the International System of Units (SI) using a conformal projection allowing the Euclidean distance to remain locally valid.

The dynamic of the model remains simple, with no information about the second derivatives. In the following, we use X to denote the state vector and P its associated covariance matrix. We index the state vectors in time with k, considering that indices k and $k + 1$ are separated in time by Δt (which we like to set to one second in our experiments).

The state-transition matrix A is defined as

$$X^{+} = \begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix}}_{X} + w. \tag{1}$$

The process noise w can be expressed based on random variables a_x and a_y (a for acceleration) as:

$$w = \begin{pmatrix} a_x \dfrac{\Delta t^2}{2} \\ a_y \dfrac{\Delta t^2}{2} \\ a_x \Delta t \\ a_y \Delta t \end{pmatrix} \tag{2}$$

We make two assumptions about the random variables: we consider that a_x and a_y are independent, and that the expectation values for both a_x^2 and a_y^2 are equal to σ^2. Then we can express Q, the process noise covariance matrix, as follows:

$$Q = \mathbb{E}\left(w \cdot w^t\right) = \sigma^2 \begin{pmatrix} \dfrac{\Delta t^4}{4} & 0 & \dfrac{\Delta t^3}{2} & 0 \\ 0 & \dfrac{\Delta t^4}{4} & 0 & \dfrac{\Delta t^3}{2} \\ \dfrac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \dfrac{\Delta t^3}{2} & 0 & \Delta t^2 \end{pmatrix} \tag{3}$$

Denoting $\widehat{X}$ the state estimate and $\widehat{P}$ the covariance of the state estimation error at time k, the predicted state and its covariance matrix at the next timestamp are given by:

$$\widehat{X}^{+} = A \cdot \widehat{X} \tag{4}$$

$$\widehat{P}^{+} = A \cdot \widehat{P} \cdot A^T + Q \tag{5}$$

Then the expression of the innovation ν, which is the difference between the measurement values and their predicted values:

$$\nu = \begin{pmatrix} x^m - \widehat{x}^{+} \\ y^m - \widehat{y}^{+} \\ \dot{x}^m - \widehat{\dot{x}}^{+} \\ \dot{y}^m - \widehat{\dot{y}}^{+} \end{pmatrix} \tag{6}$$

The covariance matrix R associated to the measurements can be calibrated based on the known uncertainties from the ADS-B specifications:

$$R = \begin{pmatrix} \sigma^2_{x^m} & 0 & 0 & 0 \\ 0 & \sigma^2_{y^m} & 0 & 0 \\ 0 & 0 & \sigma^2_{\dot{x}^m} & 0 \\ 0 & 0 & 0 & \sigma^2_{\dot{y}^m} \end{pmatrix} \tag{7}$$

Finally, we extend our observation model matrix H which boils down to an identity matrix:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

The remaining equations of the Kalman filter unfold as follows, with S the covariance of the innovation ν and K the Kalman gain:

$$S = H \cdot P \cdot H^T + R \tag{9}$$
$$K = P \cdot H^T \cdot S^{-1} \tag{10}$$
$$\widehat{X} = \widehat{X}^{+} + K \cdot \nu \tag{11}$$
$$\widehat{P} = (\mathbb{I} - K \cdot H) \cdot \widehat{P}^{+} \cdot (\mathbb{I} - K \cdot H)^T + K \cdot R \cdot K^T \tag{12}$$

where we use the Joseph's formula for updating the covariance to avoid any loss of positivity.

Uncertainty information accompanies the positional and velocity data in ADS-B messages. For example, the *Navigation Uncertainty Category for Position* (NUCp), as detailed in [13], is often encoded in the typecode field. It provides key metrics such as the *Horizontal Protection Limit* (HPL), a *containment radius for horizontal position error* (denoted as Rc/μ), and a *containment radius for vertical position error* (denoted as Rc/v).

Recent versions of ADS-B messages offer more precise uncertainty data. The *Navigation Accuracy Category for Position* (NACp) is widely used, defining the *Estimated Position Uncertainty* (EPU), which ranges from below 3 meters (NACp = 11) to over 10 nautical miles, such as when the aircraft is jammed (NACp = 0). Typically, aircraft do not transmit positional information when NACp = 0.

This uncertainty information can be utilized to refine the definition of the covariance matrix R.

Outlier values require special attention, as they should not be treated as valid measurements. The quantity $\nu^\top S^{-1} \nu$ represents the *squared Mahalanobis distance*, which measures the discrepancy between the predicted and actual measurements in the Kalman filter. This distance accounts for the fact that different components of the measurement may have varying degrees of variance and correlation. A large Mahalanobis distance indicates that the measurement is likely an outlier or that there is a significant inconsistency between the prediction and the actual measurement. In such cases, the *normalized innovation* $S^{-1}\nu$ is useful for identifying the specific component with the abnormal value, helping to detect the outlier, and suggesting proceeding with the prediction step without any update on the measurement (see Figure 12)

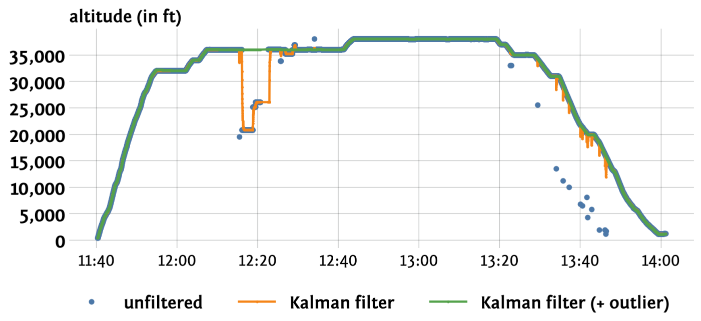


Figure 12. A basic Kalman Filter is affected by outlier measurements, which should be identified and filtered out using the Mahalanobis distance

Extended Kalman Filters (EKF) allow for the modelling of systems that are not limited to linear dynamics. The EKF operates by linearizing the nonlinear system around the current state estimate, utilizing the Jacobian of the state transition function. While EKF can effectively handle trajectories defined by latitude, longitude, speed, and track angles, it does not significantly enhance the filtering quality compared to a linear model.

Kalman smoothing In conventional Kalman filtering, state predictions and corrections rely solely on past measurement data, which can limit the accuracy of estimations, especially in trajectory post-processing. One effective method to improve the estimation precision is to implement the *Rauch-Tung-Striebel* (RTS) smoother, which utilizes both forward and backward passes: the forward pass generates initial estimates based on available measurements, while the backward pass refines these estimates by incorporating results from the forward pass.

We can simplify this process by running two distinct Kalman filters: one for the forward pass, which yields estimations $\widehat{X}_1$ and $\widehat{P}_1$, and another for the backward pass, producing $\widehat{X}_2$ and $\widehat{P}_2$. We then combine these estimates using a weighted average based on the geometric mean of the estimated covariances, leading to an optimal smoothed position estimation $\widehat{X}_s$ defined by the combined inverse covariance matrix $\widehat{P}_s^{-1} = \widehat{P}_1^{-1} + \widehat{P}_2^{-1}$ and the optimal mixing formula, which ensures that our final estimations leverage both past and future data.

4. Implementations in the traffic library

The `traffic` library in Python is an open-source toolset designed for processing and analysing air traffic data. Available at https://github.com/xoolive/traffic and introduced in [12], it offers functionalities for handling large datasets of air traffic trajectories. Built to interface with common data sources like The OpenSky Network, the library enables users to analyse historical aircraft positions, trajectories, and flight patterns. Its applications range from visualization and data cleaning to statistical and machine learning tasks on flight data.

One of the library's key features is its ability to preprocess datasets effectively, particularly in filtering aircraft trajectories and addressing the various patterns discussed in Section 2 using methods presented in Section 3. The primary class provided by the library is the `Flight` class, which encapsulates a `DataFrame` and offers additional methods tailored for trajectory analysis. Here, we introduce the `Flight.filter()` method.

The method works with default arguments for a fast and efficient filtering working in the most commonly encountered trajectories. The `filter="default"` option applies a rolling-window filter introduced in Section 3.1. The `filter="aggressive"` option composes multiple filtering approaches, including median (Section 3.1), derivative (Section 3.2) and clustering (Section 3.3) filters. More advanced filters can be passed as instances of `BaseFilter` classes as long as they implement a method with the following signature:

```
apply(data: DataFrame) -> DataFrame
```

The `traffic.algorithms.filters` module offers a variety of built-in filters, including the consistency filter (Section 3.4) and various implementations of Kalman filters (Section 3.5, and the version adapted to taxiing aircraft [16]). Filters can be composed with the `|` operator (from the `__or__` *dunder method*), which behaves slightly differently than two chained calls to the `.apply()` method.

Indeed, after applying the selected filter (or composition of filters), the `apply()` method addresses any resulting missing `NaN` values using a strategy defined by the user. The default strategy utilizes a backward fill followed by a forward fill to accommodate missing data points. Users can also opt to leave `NaN` values unmodified or apply alternative strategies, such as linear interpolation.

5. Conclusion

To wrap up, we presented a comprehensive overview of methodologies for processing and filtering ADS-B aircraft trajectory data. We began by exploring the different patterns of noisy and erroneous data before introducing filtering techniques, including rolling-window, derivative, clustering, and consistency filters. These simple methods are computationally efficient and effectively mitigate the effects of outlier data points and noisy signals. Kalman filters also provide a solid approach to data filtering, although a full Python implementation of these filters may lead to slower preprocessing times. Accelerated compiled implementations are planned for future iterations of the `traffic` library to enhance performance.

It is important to note that data collected by some data providers is often delivered as a result of preliminary preprocessing, which can mask underlying issues. In contrast, the OpenSky Network stores raw, unfiltered data: our preprocessing techniques are designed to be applied on such raw data. Future work will focus on developing faster, better fine-tuned and more efficient implementations of these methodologies, in order to address as many of the corner cases left behind.

Author contributions

Conceptualization (X.O), Methodology (*all*), Software (*all*), Validation (X.O), Formal analysis (*all*), Investigation (*all*), Data Curation (X.O, B.F), Writing – Original Draft (X.O, J.K, R.A), Writing – Review & Editing (*all*), Visualization (X.O), Project administration (X.O), Funding acquisition (X.O)

Funding statement

The authors are grateful to the EC for supporting the present work, performed within the NEEDED project, funded by the European Union's Horizon Europe research and innovation programme under grant agreement no. 101095754 (NEEDED). This publication solely reflects the authors' view and neither the European Union, nor the funding Agency can be held responsible for the information it contains.

Open data statement

All the data is available as sample datasets in the `traffic` library, and delivered together with releases after 2.12.

Reproducibility statement

All the figures for this paper are available in the documentation for the `traffic` library, which is automatically regenerated with the latest version of the repository for every commit on the master branch, and at least once a week.

References

[1] Michael Schultz, Judith Rosenow, and Xavier Olive. "Data-Driven Airport Management Enabled by Operational Milestones Derived from ADS-B Messages". In: *Journal of Air Transport Management* 99 (Mar. 2022). DOI: 10.1016/j.jairtraman.2021.102164.

[2] Kim Gaume, Richard Alligier, Nicolas Durand, David Gianazza, and Xavier Olive. "Extracting Lateral Deconfliction Actions from Historical ADS-B Data with Median Regression". In: *Proceedings of the 11th International Conference for Research in Air Transportation*. Singapore, July 2024.

[3] Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive, and Emmanuel Rachelson. "The Aircraft Runway Scheduling Problem: A Survey". In: *Computers & Operations Research* (Apr. 2021). DOI: 10.1016/j.cor.2021.105336.

[4] Wenxin Zhang, Alexia Payan, and Dimitri N. Mavris. "Air Traffic Flow Identification and Recognition in Terminal Airspace through Machine Learning Approaches". In: *Proc. of the AIAA SCITECH 2024 Forum*. Orlando, FL, Jan. 2024. DOI: 10.2514/6.2024-0536.

[5] Timothé Krauth, Jérôme Morio, Xavier Olive, and Benoit Figuet. "Advanced Collision Risk Estimation in Terminal Manoeuvring Areas Using a Disentangled Variational Autoencoder for Uncertainty Quantification". In: *Engineering Applications of Artificial Intelligence* 133 (Mar. 2024), p. 108137. DOI: 10.1016/j.engappai.2024.108137.

[6] Marco Pretto, Lorenzo Dorbolò, Pietro Giannattasio, and Alessandro Zanon. "Aircraft Operation Reconstruction and Airport Noise Prediction from High-Resolution Flight Tracking Data". In: *Transportation Research Part D: Transport and Environment* 135 (Oct. 2024), p. 104397. DOI: 10.1016/j.trd.2024.104397.

[7] Ramon Dalmau and Gilles Gawinowski. "The Effectiveness of Supervised Clustering for Characterising Flight Diversions Due to Weather". In: *Expert Systems with Applications* 237 (Mar. 2024), p. 121652. DOI: 10.1016/j.eswa.2023.121652.

[8] Junzi Sun, Xavier Olive, Esther Roosenbrand, Céline Parzani, and Martin Strohmeier. "OpenSky Report 2024: Analysis of Global Flight Contrail Formation and Mitigation Potential". In: *Proceedings of the 43th IEEE/AIAA Digital Avionics Systems Conference (DASC)*. San Diego, CA, Oct. 2024.

[9] Adric Eckstein. "Automated Flight Track Taxonomy for Measuring Benefits from Performance Based Navigation". In: *Proc. of the 2009 Integrated Communications, Navigation and Surveillance Conference*. Crystal City, VA: IEEE, May 2009, pp. 1–12. DOI: 10.1109/ICNSURV.2009.5172835.

[10] Harshad Khadilkar and Hamsa Balakrishnan. "A Multi-Modal Unscented Kalman Filter for Inference of Aircraft Position and Taxi Mode from Surface Surveillance Data". In: *Proc. of the 11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*. Virginia Beach, VA, Sept. 2011. DOI: 10.2514/6.2011-7051.

[11] Homeyra Khaledian, Raúl Sáez, Jordi Vilà-Valls, and Xavier Prats. "Interacting Multiple Model Filtering for Aircraft Guidance Modes Identification from Surveillance Data". In: *Journal of Guidance, Control, and Dynamics* (June 2023), pp. 1–16. DOI: 10.2514/1.G007139.

[12] Xavier Olive. "traffic, a Toolbox for Processing and Analysing Air Traffic Data". In: *Journal of Open Source Software* 4.39 (July 2019). DOI: 10.21105/joss.01518.

[13] Junzi Sun. *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals*. 2021. ISBN: 978-94-6366-402-8. URL: https://mode-s.org/decode/.

[14] Junzi Sun, Huy Vu, Joost Ellerbroek, and Jacco M. Hoekstra. "pyModeS: Decoding Mode-S Surveillance Data for Open Air Transportation Research". In: *IEEE Transactions on Intelligent Transportation Systems* 21.7 (July 2020), pp. 2777–2786. DOI: 10.1109/TITS.2019.2914770. URL: https://ieeexplore.ieee.org/document/8718517/.

[15] Michael Felux, Benoit Figuet, Manuel Waltert, Patric Fol, Martin Strohmeier, and Xavier Olive. "Analysis of GNSS disruptions in European Airspace". In: *Proceedings of the 2023 International Technical Meeting of The Institute of Navigation*. 2023, pp. 315–326.

[16] Xavier Olive, Manuel Waltert, Ryota Mori, and Philippe Mouyon. "Filtering Aircraft Surface Trajectories Using Information on the Taxiway Structure of Airports". In: *Proceedings of the 11th International Conference for Research in Air Transportation*. Singapore, July 2024.