SkyPos: Real-world evaluation of self-positioning with aircraft signals for IoT devices

Yago Lizarribar*†, Domenico Giustiniano*, Gérôme Bovet‡, Vincent Lenders‡
*IMDEA Networks, Spain †Universidad Carlos III de Madrid, Spain †armasuisse, Switzerland
*{name.lastname}@imdea.org ‡{name.lastname}@armasuisse.ch

***Abstract*—Positioning based on aircraft signals has been proposed as an alternative to satellite-based positioning systems (e.g. GPS). However, so far, no deployment of this technique exists, and the real-world performance remains unclear. This paper contributes at better understanding the performance tradeoffs under realistic conditions. We implement SkyPos, a localization system for GPS-denied areas or location integrity that opportunistically uses the large availability of aircraft signals to self-localize receivers. We analyze SkyPos with data collected from hundreds of sensors and thousands of aircraft around Europe. Our results show that we can achieve median accuracy down to 10 m in seconds, enabling almost real-time positioning or location verification using aircraft signals at scale.**

***Index Terms*—IoT, SDR, Localization, Avionics.**

I. Introduction

The literature on outdoor and indoor localization is vast. While indoor localization research makes use of different signals in the terrestrial radio frequency spectrum such as WiFi, UWB, Bluetooth, or Zigbee, outdoor localization is widely dominated by Global Navigation Satellite System (GNSS) based positioning such as Global Positioning System (GPS), Galileo, GLONASS, and the like, to the point of becoming a commodity in all sorts of hardware and environment.

In recent years, aircraft signals have been proposed as an alternative means of positioning. In [1], they looked into opportunistically using aircraft signals (ADS-B) to perform indoor localization. In [2], this method has been refined to require less time for data collection. Positioning with aircraft signals has several advantages over classical satellite-based systems. First, since aircraft signals are much stronger than satellite signals, they penetrate much better in buildings and work in obstructed or indoor environments. Second, getting a first position is much faster (under 3 seconds of data collection and a few minutes to obtain the fix in [2]). Finally, GPS based systems are highly vulnerable to spoofing attacks [3], [4] while positioning based on aircraft signals requires a more complex, distributed, and multi-device attacker setup [5].

In GPS spoofing scenarios, an additional source of location for verification purposes would increase the *location integrity* and quantify the trust of the provided position. Integrity is essential for systems where secure positioning is critical, such as vehicular to everything (V2X) communications, and it is currently under standardization within the third Generation Partnership Project (3GPP) in Release 17 and beyond for the design of 5G Advanced and toward 6G systems [6].

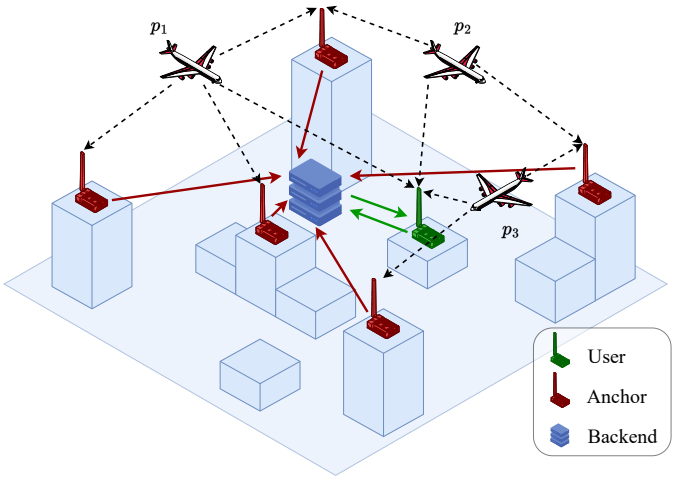


Fig. 1: General overview of SkyPos. Several aircraft (p_1, p_2, p_3) pass through an area with anchors and the user that wants to self-localize. Each anchor (red) gathers the messages from each aircraft, timestamps them, and sends it to a central backend. The user (green) also collects messages and queries the backend. It receives the timestamps from the other anchors and self-localizes using the anchors' and its own timestamps.

3GPP has also envisioned several positioning applications and defines several positioning methods for 5G networks [7] that do not rely directly on GNSS information. There is also active research in improving accuracy and reducing latencies, as shown in [8].

Since [1], [2] laid the theoretical foundations for positioning based on aircraft signals a few years ago, we are not aware of any real-world deployed system making use of this localization technique so far. The actual performance one may get in the wild is thus yet not clear.

In this work, we report from SkyPos (Figure 1), a first large-scale implementation that leverages thousands of crowd-sourced ADS-B reference stations from the OpenSky Network [9] to position IoT devices across Central Europe. We systematically evaluate the performance of this real-world deployment comprising low-cost software-defined radios and higher-end ADS-B reference stations synchronized through GPS. We analyze the tradeoffs in measurement error, synchronization offset, clock drift, and sensor placement errors.

The contributions of this paper are the following:

- We introduce an architecture and an implementation for self-localization using aircraft signals that leverages the existing ADS-B crowdsourced sensors from the OpenSky Network;
- We provide an extensive evaluation of the architecture's performance and the accuracy of the localization with real-world data;
- We present an approach to solving the multivariate self-localization problem given the measurement error, synchronization offset, clock drift, and sensor placement errors we find in our data, as well as insights into which aircraft to select to increase accuracy.

To the best of our knowledge, there has not been any attempt to design a real system for embedded sensors to self-localize using aircraft signals at a large scale, and the limited prior work in this field has addressed the problem only by means of simulations or in small setups with a few sensors [1], [10], [11]. In contrast, in our work, we perform our theoretical and experimental analysis with a large set of embedded sensors across Europe and provide a prototype architecture.

The paper is divided as follows: In Section II, we describe the main challenges faced when designing the proposed system, and Section III provides a quick primer on Automatic Dependent Surveillance-Broadcast (ADS-B) and how it could be exploited for self-localization. Sections IV-V provide the theoretical foundations and our proposed architecture, respectively. In Section VI, we evaluate our approach with data obtained from real sensors from OpenSky Network. We discuss the most relevant research regarding the opportunistic use of aircraft signals (Section VII) and provide some final remarks in Section VIII.

II. Challenges

Self-positioning using aircraft signals poses significant challenges, as the specification was not designed with this use case in mind. We describe the main challenges in the following paragraphs.

a) System Design: We show an overview of our system in Figure 1. For our prototype architecture, several design choices have to be made. Unlike GPS satellites, aircraft do not send absolute timing information in their messages. Therefore, to localize the device, we need several receivers to gather those messages. By considering the differences in arrival times among these sensors, it is possible to estimate the location of the desired sensor. We leverage existing air traffic surveillance networks like OpenSky [9], which already store aircraft positions in their backend received by ADS-B signals.

Each receiver only sends a list of ADS-B meta-data $(\{m_{p_1}, t^{r_1}_{p_1,A_1}\}, \{m_{p_3}, t^{r_3}_{p_3,A_1}\}, \ldots)$, where the tuple $(m_{p_i}, t^{r_k}_{p_i,A_j})$ represents the message m sent by aircraft p_i and received by anchor A_j at time r_k. That meta-data can be composed of the fields present in Figure 2. There are different ADS-B types, including position, velocity, or status. We only collect those, including position discarding the rest, thus reducing computation and storage requirements.

These messages are sent to the backend, reducing the data amount compared to sending the raw In-Phase and Quadrature (IQ) signals. Besides, as we have access to a large set of data, we propose to rely on in-memory storage, which allows us to reduce query times. The user also would have to send a list of the aircraft messages collected in a period of time so that the backend can match the messages to those collected from reference sensors.

Fig. 2: Structure of an ADS-B message frame. The numbers in parentheses represent the number of bits of each field in the message.

b) Synchronization and Timing constraints: In time-based localization, good timing accuracy is a must. For users, we do not impose large timing constraints on the system clock (users in our study were using Network Time Protocol (NTP), and is sufficient) nor on the timing of samples. Tools like $\mathrm{dump1090}$ [12] provide microsecond-level accuracy, which we show is sufficient.

For anchors/reference receivers on the other side, there are harder restrictions. We focus only on a subset of OpenSky's network sensors that have GPS-disciplined oscillators that can provide $30\,ns$ accuracy. Using only a subset of all the sensors in the network also has the benefit of reducing the storage and bandwidth cost of the whole infrastructure.

c) Clock imperfections: Receivers (specifically their RF frontends) are subject to imperfections in their internal oscillators which can decrease the localization accuracy. To overcome this, we propose a mathematical framework that relies on a few reference sensors with disciplined oscillators across the network. These *reference sensors* allow us to reduce the load when computing the position of users with lower-cost sensors.

d) Performance Validation: Previous literature has relied heavily on simulations or controllable setups with few sensors. To have a more extensive analysis, we rely on LocaRDS [13], a dataset containing aircraft and sensor information from OpenSky [14], a well-known air traffic surveillance, for several hours and days. This dataset contains data from up to 700 Internet of Things (IoT) sensors spread all over Europe and with varying coverage.

III. ADS-B for positioning information

Traditional systems for air traffic surveillance include Primary Surveillance Radar (PSR) which are radars that detect the presence of aircraft by measuring radio wave reflections, and Secondary Surveillance Radar (SSR) which can request more information about their flight state. This is possible because aircraft include radar transponders that reply with the required information each time a ground station interrogates them. ADS-B can be regarded as an enhancement of SSR systems, which is also used to send flight status information, but in this case, without the need for interrogation (hence the term Automatic) [15]. ADS-B is steadily being adopted by many aircraft. Starting January 2020, all aircraft in the US were mandated to include ADS-B transponders, and Europe

```
*8d3452c85851967ae91c9ae20503;
    - CRC: e20503 (ok)
    - DF 17: ADS-B message.
      - Capability     : 5
      - ICAO Address   : 3452c8
      - Extended Squitter  Type: 11
      - Extended Squitter  Sub : 0
        - F flag   : odd
        - T flag   : non-UTC
        - Altitude : 15225 feet
        - Latitude : 81268 (not dec)
        - Longitude: 72858 (not dec)
```

Fig. 3: Example of decoded ADS-B message containing aircraft position.

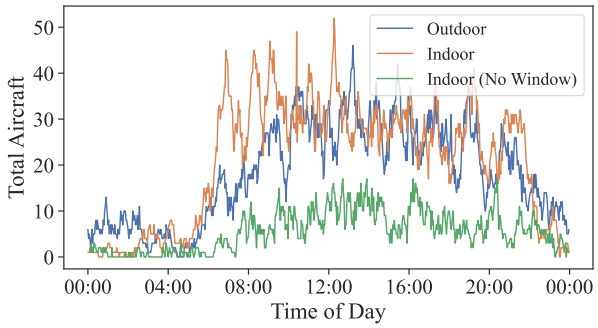


Fig. 4: Number of distinct aircraft received by 3 sensors during the day, two indoor and one outdoor. Data was aggregated in 2-minute time windows

will follow this in 2023, thus making it a widely employed technology in aviation.

The downlink channel of ADS-B operates on the 1090 MHz band, and messages are encoded with Pulse Position Modulation (PPM). Each ADS-B frame is 112 bits long [15], and the frame structure is shown in Figure 2. The International Civil Aviation Organization (ICAO) address is a 24-bit number that is used to uniquely identify an aircraft and the ADS-B message can be one of 9 types containing information regarding its position, velocity, and status. Using common open source software like *dump1090*[1] we can capture, decode these messages and timestamp these messages (with relatively high precision even [16]), as it is shown in Figure 3.

In recent years, several platforms have sprouted that, based on these ADS-B transmissions, provide real-time plane tracking and various related services. Examples of such platforms are flightaware[2] or flightradar24[3]. Within all these, a notable example is the Opensky-Network [9], [14], which is composed of a crowdsourced sensor network monitoring air traffic, also providing data for researchers in this field.

Recent literature shows that, apart from air traffic surveillance, it is possible to exploit aircraft messages for different purposes like synchronization [16], [10], positioning [1], [2] or privacy assessment [11]. The key insight is that by having multiple aircraft broadcasting status messages periodically, they effectively become an ad-hoc 'satellite network'. Moreover, aircraft travel at substantially lower altitudes than satellites, and messages are transmitted with relatively high Signal-to-Noise Ratio (SNR). Thus these signals can easily be received and decoded with low-cost off-the-shelf Software-Defined Radio (SDR) and antennas. In Figure 4, we show the number of distinct aircraft IDs for 3 sensors, 2 located indoors (one in an upper floor office with windows and the other in a lower floor with no windows) and 1 outdoors, received in a 24-hour window with a standard dipole antenna. Except for times at night (many airports close at night, thus reducing the air traffic volume), there are between 30-40 distinct aircraft IDs detected per second for the outdoor and the upper office sensor and up to 15 for the sensor in the office with no windows.

We compare this to GPS reception. For that, we use the gnss-sdr software [17] and an active GPS antenna. For the indoor locations, we could not obtain a fix at any point in time, whereas for the outdoor location, we could obtain a position at any point during the day. This further proves that we can exploit ADS-B information for self-positioning, but there could be hours at night where some accuracy might be lost due to fewer aircraft being present.

As mentioned in Section I, aircraft do not provide the time the message was transmitted, so a sensor alone could not compute its position from multiple messages. However, with a network like OpenSky, where multiple sensors are receiving the same broadcasts, we exploit the difference in arrival times to provide positioning for other nodes. As mentioned earlier, we build on top of the work in [1] and extend it to cover more realistic scenarios with a greater variety of sensors and spatial distributions.

IV. Methodology

There is a considerable amount of literature regarding the problem of aircraft multilateration (MLAT), where the objective is precise tracking of aircraft [18], [19], [20]. In contrast, in this paper, we consider the problem of positioning the user using aircraft signals, thus converting it into a *Reverse MLAT* problem. This problem was first formulated in [1].

The core idea of this method is for multiple receivers to decode aircraft messages which contain the aircraft's position, timestamp them with their local clocks, and send this information to a backend infrastructure for location processing. We can distinguish two types of receivers:

- **Target receiver** or **User**, whose position is unknown and to be estimated. The requirements are not as strict as for the reference receivers; in fact, virtually any equipment capable of decoding aircraft messages would qualify as a target receiver.
- **Reference receivers** or **Anchors**, which are accurately localized and have the necessary equipment to correct their internal clock errors (offsets and drifts). These sensors are already part of an existing air traffic surveillance network (i.e., OpenSky [9]), which also reports aircraft locations to the backend.

[1]https://github.com/antirez/dump1090
[2]https://es.flightaware.com/
[3]https://www.flightradar24.com/

A. User Localization

Let us define the 3D distance between a static user $\mathbf{u} = [x_u, y_u, z_u]$ and an aircraft position $\mathbf{p_i^r} = [x_{p_i}^r, y_{p_i}^r, z_{p_i}^r]$ at any time r as:

$$d_{u,p_i}^r = ||\mathbf{u} - \mathbf{p_i^r}||_2 \tag{1}$$

Every time an aircraft transmits an ADS-B message, the time at which the user receives it (in its local time) can be written as:

$$\hat{t}_{u,p_i}^r = t_{p_i}^r + \frac{d_{u,p_i}^r}{c} + \theta_u + \delta_u \cdot (\Delta t_{u,0}^r) + \varepsilon_u \tag{2}$$

Being $\hat{t}_{u,p_i}^r$ the time at which the user received the message, $t_{p_i}^r$ the absolute time at which the message is transmitted by the aircraft, θ_u and δ_u are the receiver's clock offset and drift respectively, and ε_u additive white gaussian noise corresponding to measurement error. $\Delta t_{u,0}^r$ is the elapsed time from an absolute initial reference t_0 to the current sensor time t_u^r.

Assuming that the aircraft's position is known (ADS-B signals continuously broadcast this information), we have a total of 6 unknowns in our system: the 3D coordinates of the receiver, offset and drift of the receiver's local oscillator, and the transmission time of the message. Since the transmission time is unknown, multilateration systems take another receiver (v) and subtract Equation (2) for both receivers:

$$\tau_{u,v,p_i}^r = \frac{d_{u,p_i}^r}{c} - \frac{d_{v,p_i}^r}{c} + \theta_{u,v} + \delta_u \cdot \Delta t_{u,0}^r - \delta_v \cdot \Delta t_{v,0}^r + \varepsilon_{u,v} \tag{3}$$

To solve for all the unknowns present in Equation (3), we would at least need 10 different measurements. To reduce the amount of data, the approach in this paper uses **reference receivers**, whose GPS coordinates are known and which possess stable oscillators so that their clock drift is negligible. Thus, Equation (3) becomes (instead of v we will represent reference receiver j as A_j):

$$\tau_{u,A_j,p_i}^r = \frac{d_{u,p_i}^r}{c} - \frac{d_{A_j,p_i}^r}{c} + \theta_{u,A_j} + \delta_u \cdot \Delta t_{u,0}^r + \varepsilon_{u,A_j} \tag{4}$$

We can collect all measurements for the target receiver u. The cost function is formed by the sum of the p-norm of all the residuals (where p can be any order of the norm, throughout this text, we will use the 2-norm). We then obtain the values for $S(\mathbf{u}, \theta_u, \delta_u)$ as:

$$\begin{aligned} S(\mathbf{u}, \theta_u, \delta_u) &= \frac{1}{2} \sum_{\forall A_j, p_i, r} (\tau_{u,A_j,p_i}^r - \hat{\tau}_{u,A_j,p_i}^r)^2 \\ &= \frac{1}{2} \sum_{\forall A_j, p_i, r} \eta_{u,A_j,p_i,r}^2 \end{aligned} \tag{5}$$

Where τ_{u,A_j,p_i}^r is the right hand side of Equation 4. The objective is to minimize the cost function S. To achieve this goal, we compute the gradients for each of the terms (3D coordinates x_u,y_u,z_u, offset θ_u and drift μ_u) as:

$$\frac{\partial S}{\partial x_u} = \sum_{\forall A_j, p_i, r} \eta_{u,A_j,p_i,r} \cdot \frac{(x_u - x_{p_i}^r)}{c \cdot d_{u,p_i}^r} \tag{6}$$

$$\frac{\partial S}{\partial y_u} = \sum_{\forall A_j, p_i, r} \eta_{u,A_j,p_i,r} \cdot \frac{(y_u - y_{p_i}^r)}{c \cdot d_{u,p_i}^r} \tag{7}$$

$$\frac{\partial S}{\partial z_u} = \sum_{\forall A_j, p_i, r} \eta_{u,A_j,p_i,r} \cdot \frac{(z_u - z_{p_i}^r)}{c \cdot d_{u,p_i}^r} \tag{8}$$

$$\frac{\partial S}{\partial \theta_u} = \sum_{\forall A_j, p_i, r} \eta_{u,A_j,p_i,r} \tag{9}$$

$$\frac{\partial S}{\partial \mu_u} = \sum_{\forall A_j, p_i, r} \eta_{u,A_j,p_i,r} \cdot \Delta t_{u,0}^r \tag{10}$$

With these equations, it is possible to obtain the position, offset, and drift of the target with optimization methods widely employed in the literature like Levenberg-Marquardt (LM) [21], BFGS [22], or Gradient-Descent and its variants [23] among others. In practice, we observe little difference between these methods for this particular problem.

B. Reference Receiver Offset estimation

One last step to reduce the amount of data needed is to precompute the reference receiver's clock offset, leaving only the parameters belonging to the user as unknowns. To estimate the offset for the reference receivers, we start from a similar equation as Equation (4), noting that the drift for both receivers is negligible, that is, $\delta_u \approx 0;\ \forall u$. We can then transform the equation into the following:

$$\tau_{A_j,A_k,p_i}^r = \frac{d_{A_j,p_i}^r}{c} - \frac{d_{A_j,p_i}^r}{c} + (\theta_{A_i} - \theta_{A_k} + \varepsilon_{A_i,A_j}) \tag{11}$$

Or in its matrix form:

$$\boldsymbol{\tau} = \mathbf{d} + \mathbf{F} \cdot \boldsymbol{\theta} + \boldsymbol{\varepsilon} \tag{12}$$

Where $\hat{\boldsymbol{\tau}}$ and $\mathbf{d}$ are $M \times 1$ vectors (where M is the number of aircraft messages) containing the measured and the geometrical Time Difference of Arrival (TDOA):

$$\hat{\boldsymbol{\tau}} = \begin{bmatrix} \cdots & \hat{\tau}_{A_j,A_k,p_i}^r & \cdots \end{bmatrix}^T \quad \forall i = 1 \dots M \tag{13}$$

$$\mathbf{d} = \begin{bmatrix} \dots & \frac{d_{A_j,p_i}^r}{c} - \frac{d_{A_k,p_i}^r}{c} & \dots \end{bmatrix}^T \quad \forall i = 1 \dots M \tag{14}$$

Matrix $\mathbf{F}$ is an $M \times N$ matrix with N being the number of sensors with 1 and -1 at the columns of sensors u and v respectively:

$$\mathbf{F} = \begin{bmatrix} 0 & \cdots & 1 & \cdots & -1 & \cdots & 0 \\ 0 & 1 & \cdots & -1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & \cdots & & -1 & 0 \end{bmatrix} \tag{15}$$

Finally, θ and ϵ are vectors of size $N \times 1$ containing the unknown offsets and noise, respectively.

Generally, this system of equations yields an over-determined system that can be solved via Least Squares, returning the offsets for the reference receivers in Equation (4):

$$\hat{\boldsymbol{\theta}} = (\mathbf{F^T F})^{-1} \mathbf{F}^T (\hat{\boldsymbol{\tau}} - \mathbf{d}) \tag{16}$$

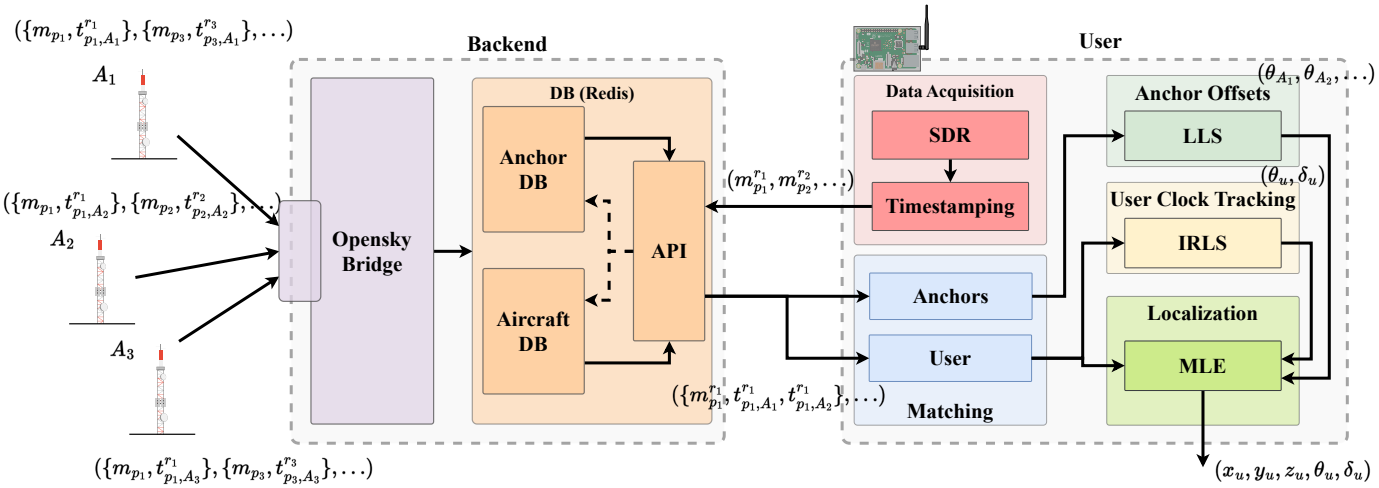


Fig. 5: SkyPos proposed architecture overview.

C. Dilution of Precision (DOP)

Dilution of Precision (DOP) has been a popular metric to evaluate the precision of localization systems for both Time of Arrival (TOA) [24] and TDOA [25] approaches (which include GPS systems) for the last decades. In recent years, DOP has been proposed in several localization systems as a method to estimate optimal receiver configuration [26]. In essence, DOP and its variants quantify the expected error amplification between the estimated and the actual position of the object that will be localized. To evaluate the DOP for our scenario, we take Equation 4 and calculate the Jacobian (for the sake of simplicity, we removed the drift):

$$\frac{\partial \tau^r_{u,A_j,p_i}}{\partial(\mathbf{u}, \theta_u)} = \begin{bmatrix} \frac{x_u - x^r_{p_i}}{d^r_{u,p_i}} & \frac{y_u - y^r_{p_i}}{d^r_{u,p_i}} & \frac{z_u - z^r_{p_i}}{d^r_{u,p_i}} & 1 \end{bmatrix} \quad (17)$$

We can construct the matrix $\mathbf{H}$ with several aircraft that contain the Jacobian for all measurements. We notice that this equation is actually that of the DOP for TOA [24], thus if we calculate the matrix Q by:

$$\mathbf{Q} = (\mathbf{H}^T \mathbf{H})^{-1} \quad (18)$$

This matrix will contain the DOP values per each of the directions and offset (this method can be extended to include drifts):

$$\mathbf{Q} = \begin{bmatrix} \sigma^2_x & \sigma_{xy} & \sigma_{xz} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma^2_y & \sigma_{yz} & \sigma_{y\theta} \\ \sigma_{xz} & \sigma_{yz} & \sigma^2_z & \sigma_{z\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_{z\theta} & \sigma^2_\theta \end{bmatrix} \quad (19)$$

With this matrix, we can estimate how precise our position is based on the selected aircraft per direction in the 3D space. It is also interesting to note that the receiver's position does not depend directly on the chosen anchors, which was experimentally observed in [1]. However, having more anchors makes it possible to obtain a more diverse set of aircraft (i.e., aircraft surrounding the sensor and not in a specific region), which can reduce the uncertainty in the position estimation.

V. SkyPos: Self-Positioning for IoT devices

We propose to use the collected aircraft information and to provide users with an on-demand self-positioning service. The architecture is illustrated in Figure 5, and it consists of two main components: the **Backend**, which collects data from the anchors and stores their messages, and the **User**, which collects its own messages and queries the Backend for the collected messages and then attempts to perform its self-positioning.

A. Backend

1) Database: The database contains two indices, from (i) the sensors and (ii) the aircraft, respectively. Their role is as follows:

- Within the sensors index, we store information on the sensor location and serial number.
- In the aircraft index, we find the message hash, the corresponding aircraft's ICAO address (a unique identifier per aircraft), the time it is processed at the server, the location of the aircraft at the transmit time, the sensors that receive the message and the local timestamp at which each sensor processes the message.

This database is continuously updated with OpenSky Network's data, with older data being regularly erased so that the storage requirements are reduced. To reduce the query times, we run in-memory database instances in Redis. The sensors that are present in the database are not all of the ones present in the OpenSky Network. As mentioned in Section II, we select a subset of "trusted" anchors that also incorporate GPS Disciplined Oscillator (GSPDO) for higher precision timestamping. The precision of these devices is $30\,ns$, which is enough for several localization applications.

2) API: The API gets the user request and processes the data as described in the remainder of this section.

After unpacking all the messages, it constructs a query for the aircraft index using the ICAO address. When results arrive, distinct sensor identifiers are selected, and another query is made to the sensors index. Once the results of both queries

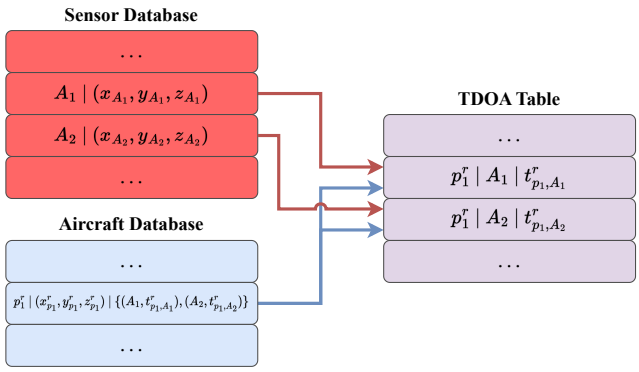


Fig. 6: TDOA table pre-processing from aircraft and sensor databases.

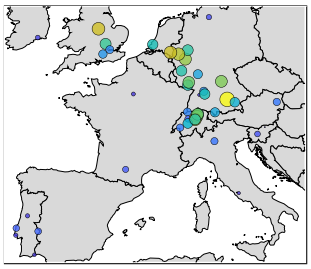
are gathered, data is prepared in the following manner: A table is constructed where each row represents the time difference of arrival measurement between the user and each of the anchors that have received the message. Each row contains the aircraft's ID, the reference receiver's serial number, and the measured TDOA in seconds. If a message m_i is received by N_{m_i} reference sensors, then N_{m_i} rows are added to the table. Thus, if the total number of valid messages is M, this table will have $\sum_{m_i=1}^{m_M} N_{m_i}$ rows (notation as seen in Figure 1). The main reason for constructing such a table is to reduce the time in the optimization step since otherwise it would need to be constructed every iteration of the optimizer, slowing down the computation considerably. The whole process can be visualized in Figure 6.

B. User

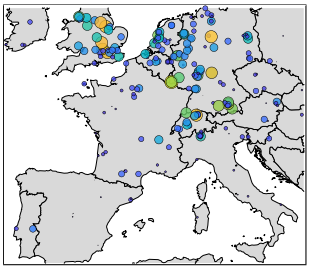
1) Data collection: The user or receiver is the service's client that wants to self-localize. The main requirements for the user are to possess a suitable receiver tuned at the 1090 MHz band and software able to decode ADS-B messages and timestamp the gathered messages. This can be easily achieved today with an inexpensive RTL-SDR receiver, some open source software like *dump1090* [12], and an embedded machine supporting such hardware and software (most widely available operating systems fulfill such requirements). The tool reports data in the format shown in Figure 3 which is sent together with the timestamp at which it was received by the sensor. The precision achievable with a common board like the Raspberry Pi is $1\,\mu s$, which would be too high to achieve meaningful accuracy. However, by using higher-precision anchors, it is possible to overcome this limitation.

Serial	**Latitute**	**Longitude**	**Altitude**	**Type**	**Verified**
Integer	Float	Float	Float	String	Bool

TABLE I: Sensor dataset structure



(a) Radarcape sensors



(b) dump1090 sensors

Fig. 7: Sensor locations across Europe. The circle size and color represent the number of messages received per sensor.

Fulfilling the aforementioned requirements, the user first collects and decodes aircraft messages, including the aircraft's ICAO address and its latitude, longitude, and altitude (see Figure 3). After collecting messages for a certain time, the user serializes and sends the data to the service's endpoint through the API.

2) Localization: The user receives the data of the anchors from the API as a stream of $(m_1, t_1^{A_1}, m_1, t_1^{A_2}, \ldots)$, where m_i is the ith received message and $t_i^{A_j}$ is the time at with anchor j received that message (see Figure 5). It then splits the process into the steps mentioned in Section IV. Data from anchors are used to estimate their offsets via linear estimators. In parallel, the user performs an initial estimation of its clock offset and drift. Depending on the type and quality of the receiver, measurements might be noisy, so we use the Iteratively Reweighted Least Squares (IRLS) technique [27] to obtain better initial estimations. After preprocessing, the user's position r is obtained via any of the methods described in Section IV.

VI. Evaluation

A. Dataset

The dataset for our evaluation is described in LocaRDS [13], a dataset from the Opensky Network researchers that provides information on thousands of real aircraft collected by hundreds of sensors operated by volunteers across Central Europe.

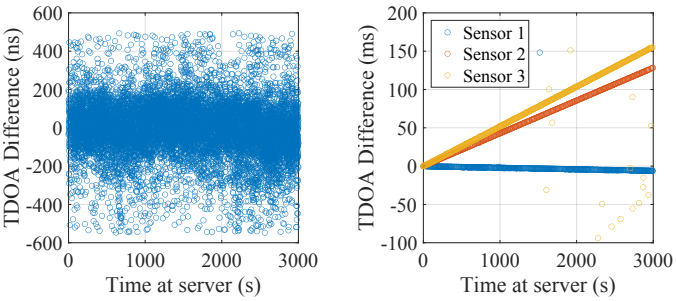
The data structure for sensor and aircraft databases is shown in Tables I and II respectively. The *Verified* field in Table I refers to whether the sensor position is verified by its operators. For our evaluation purposes, we will only use as reference sensors those that have been verified.

B. Evaluation of the dataset

To perform the evaluation, we first split the dataset between 2 types of sensors: *Radarcape* and *dump1090* sensors (their distribution across Europe is shown in Figure 7). These sensors have different properties and are thus evaluated separately to understand the impact of the sensor type better.

Message ID	**Time at Server**	**Aircraft ID**	**Latitude**	**Longitude**	**Baro. Altitude**
Integer	Float	Integer	Float	Float	Float
Geo. Altitude	**Num. Measurements**	**Sensors**	**Timestamps**	**RSS**	
Float	Integer	Array {Int}	Array {Float}	Array {Float}	

TABLE II: Aircraft dataset structure



(a) Anchor clock offset (Radarcape sensors).

(b) User clock offset (dump1090 sensors).

Fig. 8: Reference receivers' clock offsets over time and compared to each other. 8a shows the difference in geometric/measured TDOA for 2 radarcape receivers. 8b shows the offset over time for 3 sensors running the dump1090 software.

1) Radarcape sensors: The first part of our evaluation is done with Radarcape sensors [28]. These sensors are specially designed to receive ADS-B messages. They incorporate GPS receivers and GSPDO, meaning their clocks remain stable within tens of nanoseconds. As mentioned earlier, we used only sensors that had verified positions, so for this set of experiments, we had 37 sensors shown in Figure 7a. The density of such sensors is especially high in Switzerland and Germany, and it is also where these sensors receive more messages, as can be observed (a larger circle means a larger number of messages are received).

The fact that these sensors have GSPDO is more notable when computing the difference between the expected TDOA between 2 reference sensors and the measured TDOA as it is shown in Figure 8a, where over the course of one hour, those differences remain constant except for variations due to measurement noise. Using Least Squares to compute the mean offsets (using one of the sensors as the absolute time reference), we can observe that those offsets remain within the ± 200 ns.

To perform the evaluation, we take one of the Radarcape sensors and randomize its position. We then compute the offsets for the remaining sensors used as a reference and select the 5 sensors with the highest common message count with the target receiver. The number of anchors can vary depending on the user's position with respect to them. In general, and as mentioned in Section IV, the number of anchors does not directly affect the performance, and we did not see much improvement from adding more anchors. This matches the observations shown in [1]. We then compute the position of the receiver with that data.

Figure 9 shows how considering offsets in the model already has a notable effect on the accuracy. For the 2D scenario (latitude and longitude), the median error when considering the offset is 21 m, whereas it reaches 24 m when not considered. Moreover, in the former case, all the sensors are localized within a 41 m error and a 65 m error in the latter. We select 21 sensors with at least 10 other anchors and vary the number to observe the effect on accuracy. The results match what we predict in Section IV: The number of anchors does not significantly improve accuracy (even with 1 anchor, we can still maintain a median accuracy of 15 m).

It is also worth commenting that errors for the 3D case (latitude, longitude, and altitude) are higher because of the receivers and aircraft geometry. The DOP is much higher in the vertical axis than in the horizontal axes, and thus errors propagate faster. This is better visualized in Figure 10. Because distances on the ground plane are larger than vertical distances (100 km separating receivers and aircraft maximum altitude is around 10 km), small variations in the delay estimation generate larger uncertainties in the vertical axis (Δz) than in the horizontal axis (Δx). Despite this, the tendency is similar, and considering offset improves its counterpart notably in this case.

For our next set of experiments, we look into how the number of messages impact the location accuracy. In the experiments shown in Figure 11a, we randomly select a fraction of the received messages (to simulate lower message rates). It is noticeable how even using 1% of the number of messages can lead to a similar accuracy rather than using the entire message dataset.

We show the 25th and 75th percentiles (black lines) and median (red dashed line) for the amount of time and localization accuracy, respectively, in Figure 12. In these figures, it can also be seen that 1% of the data is a 'sweet spot' where the amount of time taken to process the location cannot be greatly reduced and where the location accuracy does not get impacted very much.

In the time between Figure 11a and Figure 11b, we also modify several optimization parameters, which result in the median error being reduced to 13 m, and we are also able to localize 90% of the sensors within 25 m.

In Figure 11b, we perform a similar experiment, but instead of randomly selecting a fraction of the messages, we select contiguous messages to observe the effect of short captures on accuracy. In this case, depending on the number of messages, we can observe that accuracy varies more noticeably. One of the reasons why accuracy drops at a larger rate could be that on shorter, contiguous streams of messages, fewer aircraft are seen, and the locations received are close to each other, thus worsening the DOP.

2) dump1090 sensors: When analyzing sensors running the dump1090 software only, from Figure 8 we can observe that the offsets and drifts are much higher, in the order of tens of seconds, and looking at drifts, we can deduce 2 categories: receivers with drift of less than 2 Parts Per Million (PPM), which matches the specification for the version 3 of the popular RTL-SDR dongles [29]; and receivers with drifts in the order of tens of PPM which are older versions of the same SDR type.

Figure 13 and Figure 14 show the results when using the whole dataset. For both the 2D and 3D cases, the positioning error is significantly higher than when using Radarcape sensors, with median errors of 450 m and 2000 m, respectively. The fact that offsets and drifts are higher could be one of the reasons to explain this accuracy decrease, but there are other factors. These sensors lack GPS, thus relying on the user to manually input their location, which might be inaccurate for a

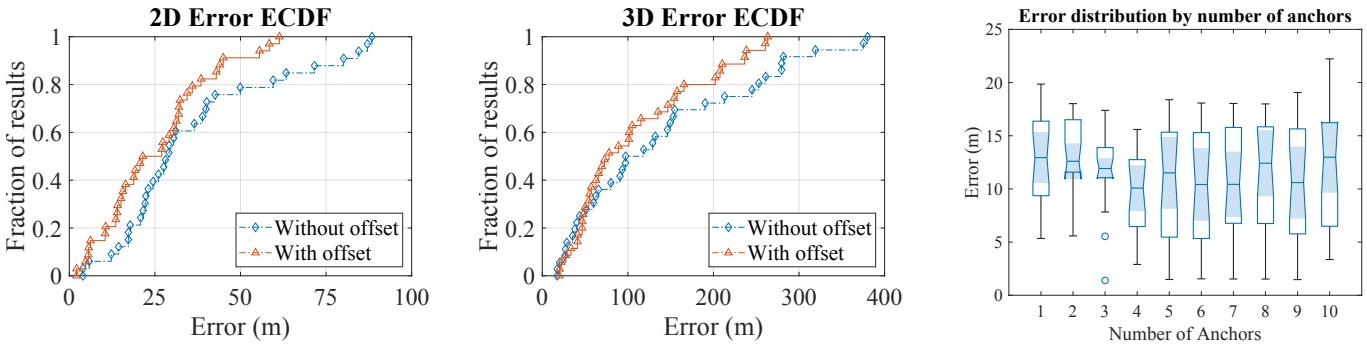


Fig. 9: On top: The accuracy changes when clock offset is considered and when not. Below: How error is distributed when varying the number of anchors used for self-localization (for the 21 sensor subset).

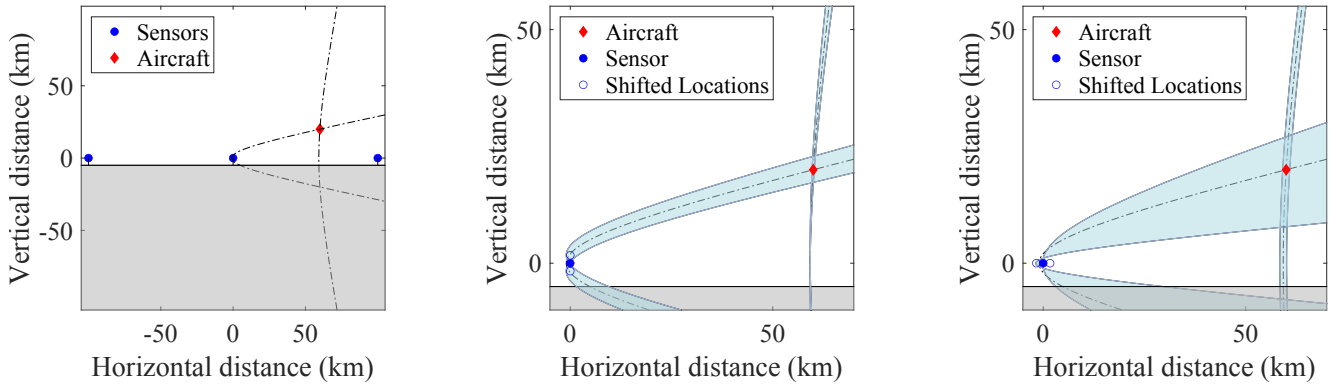
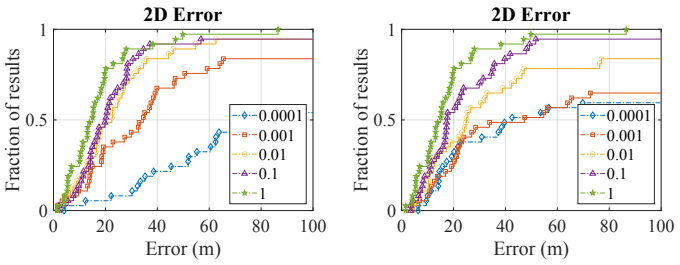
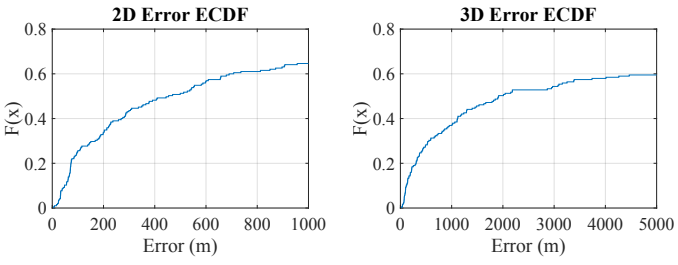


Fig. 10: 2D visualization on why errors in altitude are generally larger than errors in the horizontal plane for the problem of self-localization with aircraft. The figure in the center represents a characteristic scenario for 3 sensors (in blue) and an aircraft (in red) flying at 10km altitude. In the rightmost figure, we show how an error of 1km in the vertical plane leads to a smaller error ϵ than when errors are on the horizontal plane. Thus, from the numerical point of view, the optimizing routine will converge faster to the correct horizontal position than the vertical one.



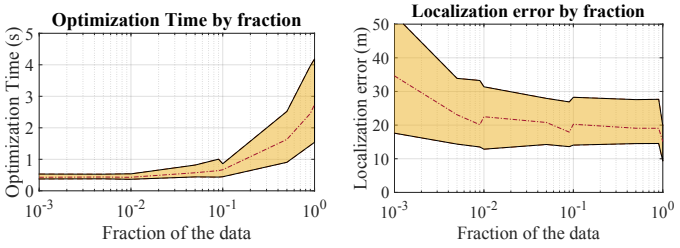
(a) Varying Message rate (b) Varying Time window

Fig. 11: Error varying message rate and time windows.



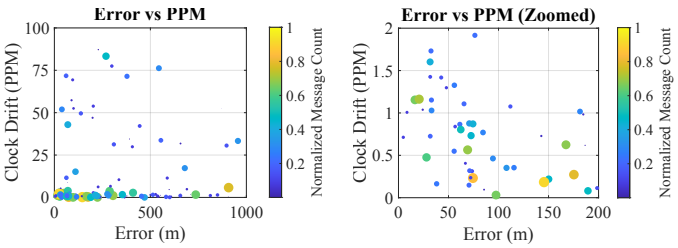
(a) 2D error (b) 3D error

Fig. 13: Results for dump1090 sensors.



(a) Time vs messages (b) Accuracy vs messages

Fig. 12: Time and accuracy varying the amount of data.



(a) All Recievers (b) Zoomed

Fig. 14: Error with respect to drift and message count.

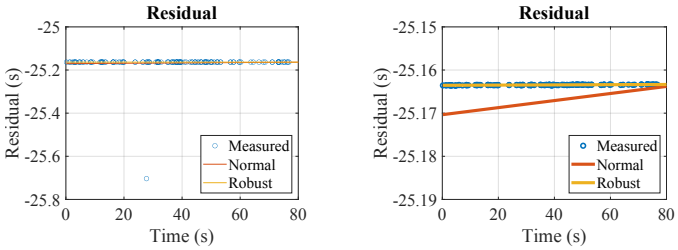


Fig. 15: Normal vs Robust Estimation of the residuals.

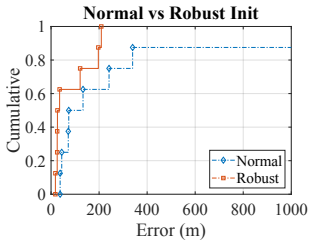


Fig. 16: ECDF of the error using normal vs robust initialization

number of reasons. The ground truth could therefore be biased. Also, many of these receivers are located in interior spaces, which may also add errors due to sensors not seeing as many aircraft or due to multipath effects.

We asked 8 users to verify their positions and their Radio Frequency (RF) frontend. The returned predictions are shown in Table III. One important aspect is that the offset and drift are orders of magnitude bigger than the position parameters. Thus, choosing a good starting point is key to obtain an accurate position of the sensor. The naive approach, looking at Figure 8b, would be to perform a linear regression and use the intercepts as the starting points for the algorithm.

However, methods based on naive linear regression can decrease the performance in the presence of outliers. To avoid this issue, we decide to use IRLS [27], which consists in adding weights to all measurements and iteratively updating the weights and intercept until the change is less than a specified tolerance. The difference between both approaches can be seen in Figure 15, where for that particular sensor of the dataset, the presence of an outlier tilts the intercept estimation, whereas, for the robust case, it can discard it from the offset/drift estimation.

Both in Table III and Figure 16, we can see the difference in results between using normal and robust initialization techniques. For the former case, the median error is around 100 m, with a few sensors above that value. In the case of robust initialization, 5 out of 8 sensors can be accurately localized within 30 m.

Type	Location	$\mathbf{E_{nor}}$ (m)	$\mathbf{E_{rob}}$ (m)	Drift (PPM)
RTL-SDR	Indoor	339.60	18.07	1.9985
RTL-SDR	Indoor	133.08	121.10	38.7784
FlightAware	Outdoor	71.65	25.59	0.5646
RTL-SDR	Outdoor	241.42	97.49	63.1380
RTL-SDR	Indoor	73.72	196.81	42.9413
RTL-SDR	Outdoor	44.48	25.63	1.3269
RTL-SDR	Indoor	37.83	26.58	51.9510
RTL-SDR	Indoor	1140.12	35.64	63.1393

TABLE III: Predicted positions for verified sensors. $\mathbf{E_{nor}}$ refers to positions obtained using regular least squares to predict the offset, whereas $\mathbf{E_{rob}}$ refers to the robust initialization method via IRLS

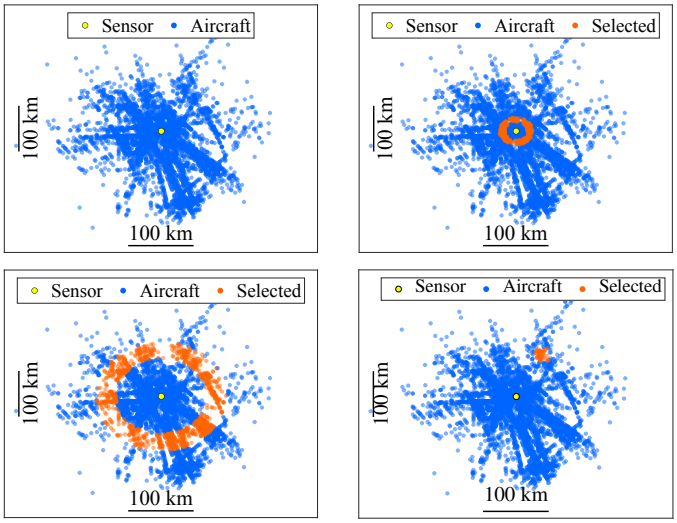


Fig. 17: Capture of a sensor and the measured aircraft during an hour. Upper right corresponds to selecting aircraft close to the sensor, lower-left to aircraft far from it, and lower right to a bad selection, where all aircraft are clustered in a narrow space.

Another interesting fact is that it is possible to capture the device type with the drift parameter. For example, 3rd user's receiver, as per the documentation [29], has an oscillator offset of 0.5 PPM, which is indeed what we can measure with our approach. The same case happens for the RTL-SDR receivers.

As for the offset, we have not explored the synchronization possibilities since our dataset contains no ground truth for the actual time. However, it is readily implemented as well. Therefore, simultaneous localization and synchronization could be a future line of research.

C. Intuition on Aircraft Selection

One important factor in assessing the quality of a localization system is to estimate how uncertain the position is. That uncertainty is primarily driven by which and how many aircraft are chosen to perform the localization, analogous to GPS satellites. In Figure 17, we show the captured aircraft by a sensor in an hour window and 3 different scenarios: aircraft messages captured within 10-20 km, within 30-40 km, and clustered around a small region 40 km away from the sensor. The results vary notably in each scenario.

In the former, the vertical DOP is the lowest, but the uncertainty in the planar directions increases. On the other hand, for aircraft within 30-40 km, we observe a reduction in the planar DOP and an increase in the vertical. The positioning errors reflect the expected uncertainties in each of the directions. In the last case, where messages are clustered in a small region, we measure DOP values > 100 in all directions, and the optimizers fail to find an acceptable solution. This also is

expected since close aircraft produce ill-conditioned problems for the optimizer.

The problem of selecting the best aircraft that minimize the DOP is non-trivial, especially with large amounts of data where combinations grow exponentially. It is out of the scope of this paper, but it would be an interesting future line of research. Besides, geometric DOP minimization does not always work best; other approaches might yield better localization results, as shown in [26].

Even without an algorithm that yields the absolute minimum DOP, it is possible to establish heuristics to have a good enough set of aircraft to perform the localization task. The best DOP values are obtained when aircraft are separated apart, which can be guaranteed by trying to select messages surrounding the sensor. It might appear as a 'chicken-and-egg' problem because to select aircraft that surround the sensor, its position should be known. A straightforward solution is to take the average position of the observed aircraft within a time window and use it as the starting point. We observed that this approach yields errors in the lower 10s of km, which is a good starting point. It could be possible to iteratively re-select sensors and perform the localization again should more accurately be needed.

These heuristics also reduce the messages needed for localization without significantly losing accuracy. For example, we can draw a circle around the sensor, divide it into equal sectors and select a few aircraft belonging to each sector. This way, the spatial diversity is preserved, but the message amount is reduced, reducing the time to compute a correct position.

For example, in Figure 18, we show the results of applying sectorization (16 and 32 divisions) and their effect on the overall accuracy. We can observe a notable improvement not only in the number of messages but also in the accuracy. However, improving geometrical distribution provides better results for devices with lower PPM values. For older versions of the RTL-SDR, the improvement is not as significant, thus indicating that localization performance is also affected by the radio frontend quality.

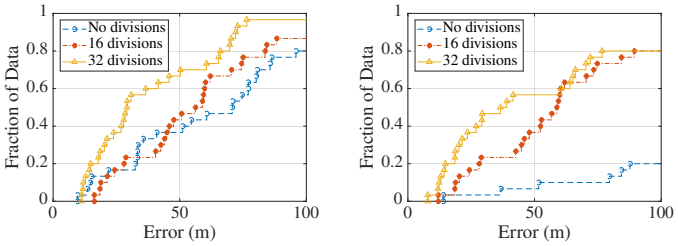


Fig. 18: Results when sectorizing and picking aircraft per sector. For both Robust (left) and Non-robust (right) initialization.

D. Architecture performance

We set up a server and a Raspberry Pi 4 as a User to develop and test the architecture. The server has 16 GB of memory, an 8-core 2.2 GHz Intel Xeon CPU, and 100 GB of disk space. The API and the Localization Engine are implemented in Python. To develop the API, we use Flask-Restful [30], and the localization module for the Raspberry Pi relies on SciPy's optimization modules [31]. For the database, we choose Redis [32], a popular in-memory database in combination with Redisearch [33] that allows creating indices and querying data with more complex search parameters. Data for sensors and aircraft is introduced in the database, occupying 5.71 GB of memory.

To evaluate the performance of the different stages, we collect the messages from each sensor into a JSON object and send them through the API. Each aircraft message is encoded in around 62 bytes. Thus, 1000 messages can be encoded in about 60 kB, which is relatively small by today's commercial internet providers. We evaluate the time per stage with the number of messages ranging from 10 to 10,000, and the main results are summarized in Table IV.

The time taken to query the database and obtain the matching messages is relatively small, and even for high amounts of messages, it only takes around 1 second. On the other hand, when computing the TDOA pairs for the user with reference sensors, the time increases notably, taking up to 30 seconds to process 10,000 messages.

Joining these results with the ones shown in Figure 12, for 1000 messages which are around 40 seconds of data collection,

Time to query Database (in seconds)							
Amount of Messages	μ	σ	**1%**	**25%**	**50%**	**75%**	**99%**
10	0.00658	0.00045	0.00531	0.00652	0.00659	0.00666	0.00808
50	0.01072	0.00115	0.0058	0.01082	0.01092	0.01103	0.01328
100	0.0153	0.00087	0.01031	0.0152	0.01529	0.01547	0.01774
500	0.05451	0.00338	0.04364	0.05379	0.05413	0.05483	0.06959
1,000	0.10819	0.00858	0.06626	0.10741	0.10835	0.11008	0.15233
5,000	0.54248	0.02363	0.48013	0.53748	0.54053	0.54483	0.61043
10,000	1.12815	0.04312	0.982	1.12037	1.12617	1.14578	1.20786
Time to compute TDOA values (in seconds)							
Amount of Messages	μ	σ	**1%**	**25%**	**50%**	**75%**	**99%**
10	0.03044	0.00492	0.02879	0.02904	0.02926	0.02954	0.05453
50	0.12524	0.00198	0.12383	0.12429	0.1245	0.1251	0.13488
100	0.24099	0.00193	0.23761	0.24031	0.24067	0.24116	0.25089
500	1.2122	0.02664	1.1984	1.2033	1.2062	1.2079	1.3417
1,000	2.5457	0.04442	2.5093	2.5169	2.5245	2.552	2.7252
5,000	13.9	0.12358	13.712	13.801	13.9	13.986	14.261
10,000	31.121	0.33737	30.669	30.941	31.063	31.195	32.603

TABLE IV: Time to execute different components of the proposed system.

it would take on average less than 4 seconds to compute the position of the user (not counting network latency) which is an acceptable delay for static or slowly-moving devices. After an initial estimation, with fewer messages, it would be possible to recompute the position, leading to lower estimation times.

VII. Related Work & Discussion

The literature regarding outdoor and indoor localization is vast. While indoor localization research uses different signals in the radio frequency spectrum, outdoor localization is widely dominated by GPS and the like. With techniques like Precise Point Positioning, it has been possible to achieve sub-decimeter level accuracies over the last 20 years [34].

However, since the position is computed at the receiver, GPS based systems are vulnerable to spoofing or jamming attacks [3], [4]. This area of research is currently very active, and even though there are some commercial solutions to avoid these issues, it remains an open challenge. Works to enable localization in GPS-constrained areas have been proposed in recent years via UAV swarms [35], [36], or integrating multiple information sources [37] for dense vehicular networks. However, the former requires specific and coordinated ad-hoc deployments or more complex changes in both infrastructure and users, as in the latter. This work explores leveraging existing signals and sensor networks and providing a location solution with little overhead for resource-contrained devices like a Raspberry Pi.

In recent years, the use of aircraft signals has drawn a great deal of attention to solve a variety of problems not only related to aircraft positioning. In [1], they looked into opportunistically using aircraft signals to perform indoor localization. However, their analysis only extended to a few sensors and did not consider scalability or further development of the proposed platform. In our research, we extend this analysis to hundreds of sensors and different topologies and provide results on the overall performance of the proposed architecture.

In [16], aircraft signals and their waveforms were exploited to obtain time of arrival measurements with less than a nanosecond time precisions. ADS-B signals have also been used to achieve sensor network synchronization with accuracies similar to the NTP, as it is stated in [10]. Recently, the research from [11] has looked into the privacy of sensor networks for avionics. In this last piece of research, the methodology used is similar to what is proposed, but most analysis is done in the simulation domain, and only one case with real sensors is presented.

Apart from user positioning, SkyPos can also be used in location verification for sensor networks. ADS-B signals are sent in 'clear text,' some papers have attempted to fabricate fake signals to confuse surveillance systems [5], [38]. Nevertheless, these attacks generally require multiple transmitters, with accurate synchronization and calculated delays in the transmission times to be effective. From the scope of this paper, such an attack would have little impact on the localization performance. If an attacker tried to inject fake ADS-B messages to the user, those would get discarded in the backend, so to make it feasible, it would need to target multiple reference receivers. This would become infeasible with more aircraft messages collected and reference sensors used.

The proposed solution in this paper has several advantages, including notable accuracy in outdoor or high synchronization scenarios, requires little data to obtain an initial position, and can be implemented in low-cost sensor networks with a cheap SDR receiver like the RTL-SDR and readily available open source software. Work in [1], [2] has attempted indoor localization similarly using aircraft signals to this work. The former relies on sending fully decoded messages to a central backend and performs aircraft tracking simultaneously. In our work, we show that it is feasible to send less data and it is possible to achieve slightly better accuracies even for sensors with very high offsets and drifts. The latter relies on operating on the raw message, notably improving accuracy but increasing computation and storage requirements. Hence, running on low resource Internet-of-Things (IoT) devices is not feasible.

Our work aims to minimize the data exchange between centralized entities and allows the user to compute its position instead of offloading it to a single backend.

However, SkyPos is not free of limitations. In the following lines, we mention the following most important ones:

- Low altitude accuracy. As mentioned in previous sections, because of the geometry of the underlying positioning problem, optimization algorithms tend to lose numerical precision on the vertical coordinates. Other sources of altitude might be necessary to achieve higher altitude precision.
- Not suitable for fast-moving users. Due to the necessity of accumulating data, sending it to the backend, and later processing it, it cannot support highly dynamic users, such as drones. However, for slow-moving scenarios, it can be an additional source of position verification instead of relying on users' input.
- Variability due to time of day. Usually, we observed fewer aircraft at night so that it could affect the system's overall accuracy at night time.

Despite these limitations, we have shown that it is possible to build a localization infrastructure with aircraft signals and obtain notable results. Future works will look into better aircraft selection methodologies that reduce the data needed and, thus, the time to compute the user's position.

VIII. Conclusion

In this paper, we have addressed the problem of localizing sensors using aircraft signals instead of the widely employed GPS. We build on previous research, develop a prototype architecture that can be embedded in existing aircraft monitoring solutions, and test it against hundreds of thousands of data points. The results obtained comparing GPS based with our proposed solution showed that we can get reasonable positioning accuracy in almost real-time using lower-cost hardware. Although it is not a complete substitute for all GNSS based positioning services, we demonstrated that it could be used as a positioning method for embedded sensors in IoT networks in GPS-denied areas for example, or as position verification mechanism for GPS based receivers.

ACKNOWLEDGEMENT

The research conducted by IMDEA Networks was sponsored in part by armasuisse under the Cyber and Information Research Program, in part by the project MAP-6G, reference TSI-063000-2021-63, granted by the Ministry of Economic Affairs and Digital Transformation, and the European Union-NextGenerationEU through the UNICO-5G R&D program of the Spanish Recovery, Transformation and Resilience Plan, and in part by the FPU19/03102 scholarship from the Spanish Ministry of Universities (MIU).

REFERENCES

[1] M. Eichelberger, K. Luchsinger, S. Tanner, and R. Wattenhofer, "Indoor Localization with Aircraft Signals," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, (New York, NY, USA), pp. 1–14, Association for Computing Machinery, Nov. 2017.

[2] A. Canals, P. Josephy, S. Tanner, and R. Wattenhofer, "Robust indoor localization with ADS-B," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, (New Orleans Louisiana), pp. 505–516, ACM, Oct. 2021.

[3] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, (New York, NY, USA), pp. 75–86, Association for Computing Machinery, Oct. 2011.

[4] J. Noh, Y. Kwon, Y. Son, H. Shin, D. Kim, J. Choi, and Y. Kim, "Tractor Beam: Safe-hijacking of Consumer Drones with Adaptive GPS Spoofing," *ACM Transactions on Privacy and Security*, vol. 22, pp. 12:1–12:26, Apr. 2019.

[5] D. Moser, P. Leu, V. Lenders, A. Ranganathan, F. Ricciato, and S. Capkun, "Investigation of multi-device location spoofing attacks on air traffic control and possible countermeasures," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, MobiCom '16, (New York, NY, USA), pp. 375–386, Association for Computing Machinery, Oct. 2016.

[6] S. Bartoletti, H. Wymeersch, T. Mach, O. Brunnegård, D. Giustiniano, P. Hammarberg, M. F. Keskin, J. O. Lacruz, S. M. Razavi, J. Rönnblom, F. Tufvesson, J. Widmer, and N. B. Melazzi, "Positioning and Sensing for Vehicular Safety Applications in 5G and Beyond," *IEEE Communications Magazine*, vol. 59, pp. 15–21, Nov. 2021.

[7] R. Keating, M. Säily, J. Hulkkonen, and J. Karjalainen, "Overview of Positioning in 5G New Radio," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, pp. 320–324, Aug. 2019.

[8] Q. Liu, R. Liu, Y. Zhang, Y. Yuan, Z. Wang, H. Yang, L. Ye, M. Guizani, and J. S. Thompson, "Management of Positioning Functions in Cellular Networks for Time-Sensitive Transportation Applications," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2023.

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

[10] S. Zhu, X. Zheng, L. Liu, and H. Ma, "AirSync: Time Synchronization for Large-Scale IoT Networks Using Aircraft Signals," *IEEE Transactions on Mobile Computing*, vol. 22, pp. 69–83, Jan. 2023.

[11] S. Sciancalepore, S. Alhazbi, and R. Di Pietro, "Receivers location privacy in avionic crowdsourced networks: Issues and countermeasures," *Journal of Network and Computer Applications*, vol. 174, p. 102892, Jan. 2021.

[12] S. Sanfilippo, "Dump1090 README," May 2023.

[13] M. Schäfer, M. Strohmeier, M. Leonardi, and V. Lenders, "LocaRDS: A Localization Reference Data Set," *Sensors*, vol. 21, p. 5516, Jan. 2021.

[14] M. Schafer, M. Strohmeier, M. Smith, M. Fuchs, R. Pinheiro, V. Lenders, and I. Martinovic, "OpenSky report 2016: Facts and figures on SSR mode S and ADS-B usage," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–9, Sept. 2016.

[15] J. Sun, *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals*. TU Delft OPEN, 2021.

[16] R. Calvo-Palomino, F. Ricciato, B. Repas, D. Giustiniano, and V. Lenders, "Nanosecond-Precision Time-of-Arrival Estimation for Aircraft Signals with Low-Cost SDR Receivers," in *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 272–277, Apr. 2018.

[17] C. Fernández-Prades, J. Arribas, M. Majoral, G. Araujo, A. Arnold, C. Avilés, M. Branzanti, Á. Cebrián-Juan, A. Cecilia-Luque, L. Esteve, F. Fabra, D. Fehr, P. Gupta, G. LaMountain, M. Lenhart, J. Melton, D. Miralles, M. Molina, R. Muñoz, C. O'Driscoll, D. Pubill, A. Ramos, J. Schindehette, L. Tonetto, and S. van der Linden, "GNSS-SDR," Apr. 2022.

[18] R. Kaune, C. Steffes, S. Rau, W. Konle, and J. Pagel, "Wide area multilateration using ADS-B transponder signals," in *2012 15th International Conference on Information Fusion*, pp. 727–734, July 2012.

[19] G. Galati, M. Leonardi, I. A. Mantilla-Gaviria, and M. Tosti, "Lower bounds of accuracy for enhanced mode-S distributed sensor networks," *IET Radar, Sonar & Navigation*, vol. 6, pp. 190–201, Mar. 2012.

[20] I. A. Mantilla-Gaviria, M. Leonardi, G. Galati, and J. V. Balbastre-Tejedor, "Localization algorithms for multilateration (MLAT) systems in airport surface surveillance," *Signal, Image and Video Processing*, vol. 9, pp. 1549–1558, Oct. 2015.

[21] H. P. Gavin, "The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems," *Department of Civil and Environmental Engineering, Duke University*, 2023.

[22] C. A. Floudas and P. M. Pardalos, *Encyclopedia of Optimization*. Springer Science & Business Media, 2008.

[23] S. Ruder, "An overview of gradient descent optimization algorithms," June 2017.

[24] R. Santerre, A. Geiger, and S. Banville, "Geometry of GPS dilution of precision: Revisited," *GPS Solutions*, vol. 21, pp. 1747–1763, Oct. 2017.

[25] J. Bard and F. Ham, "Time difference of arrival dilution of precision and applications," *IEEE Transactions on Signal Processing*, vol. 47, pp. 521–523, Feb. 1999.

[26] M. Wang, Z. Chen, Z. Zhou, J. Fu, and H. Qiu, "Analysis of the Applicability of Dilution of Precision in the Base Station Configuration Optimization of Ultrawideband Indoor TDOA Positioning System," *IEEE Access*, vol. 8, pp. 225076–225087, 2020.

[27] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics - Theory and Methods*, vol. 6, pp. 813–827, Jan. 1977.

[28] "ADS-B Receiver MLAT, Radarcape, ASTERIX, Dataports JSON, Binary." http://radarcape.com/.

[29] "A Datasheet for our RTL-SDR.com V3 Dongles." https://www.rtl-sdr.com/a-datasheet-for-our-rtl-sdr-com-v3-dongles/, June 2017.

[30] "Flask-RESTful — Flask-RESTful 0.3.8 documentation." https://flask-restful.readthedocs.io/en/latest/.

[31] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, Mar. 2020.

[32] "Redis." https://redis.io/.

[33] "RediSearch." https://redis.io/docs/stack/search/.

[34] J. Kouba and P. Héroux, "Precise Point Positioning Using IGS Orbit and Clock Products," *GPS Solutions*, vol. 5, pp. 12–28, Oct. 2001.

[35] H. Sallouha, M. M. Azari, and S. Pollin, "Energy-Constrained UAV Trajectory Design for Ground Node Localization," June 2018.

[36] Q. Liu, R. Liu, Z. Wang, and J. S. Thompson, "UAV Swarm-Enabled Localization in Isolated Region: A Rigidity-Constrained Deployment Perspective," *IEEE Wireless Communications Letters*, vol. 10, pp. 2032–2036, Sept. 2021.

[37] Q. Liu, R. Liu, Z. Wang, L. Han, and J. S. Thompson, "A V2X-Integrated Positioning Methodology in Ultradense Networks," *IEEE Internet of Things Journal*, vol. 8, pp. 17014–17028, Dec. 2021.

[38] F. Shang, B. Wang, F. Yan, and T. Li, "Multidevice False Data Injection Attack Models of ADS-B Multilateration Systems," *Security and Communication Networks*, vol. 2019, p. e8936784, Mar. 2019.

Yago Lizarribar is a PhD student at the IMDEA Networks Institute in Madrid. He obtained a BSc and MSc in Engineering in 2016 and 2018 from University of Navarra and an MSc in Telematics Engineering in 2020 from University Carlos III de Madrid. He previously worked at the MIT Media Lab on autonomous lightweight vehicles and robotics. His current work focuses on crowdsourced spectrum-monitoring, software-defined radio, signal processing and machine-learning.

Domenico Giustiniano is a Research Associate Professor (tenured) at IMDEA Networks, Madrid, Spain. He holds a PhD in Engineering from the University of Rome Tor Vergata (2008). Before joining IMDEA, he was a Senior Researcher and Lecturer at ETH Zurich. He also worked for a total of four years as Post-Doctoral Researcher in industrial research labs (Disney Research Zurich and Telefonica Research Barcelona). He has authored over 100 international papers, he is Leader of the OpenVLC Project and Co-Founder of the non-profit Electrosense Association. His current research interests cover Battery-Free IoT, Large-scale Spectrum Analytics, and 5G+ Localization Systems.

Gérôme Bovet is the head of data science at the Cyber Defence Campus, where he leads a research team and a portfolio of about 30 projects. His work focuses on machine/deep learning approaches applied to cyber-defence use cases, with emphasis on anomaly detection, adversarial and collaborative learning. He received his Ph.D. in networks and systems from Telecom Paris, France, in 2015 and an Executive MBA from the University of Fribourg, Switzerland in 2021.

Vincent Lenders is the founding Director of the CyberDefence Campus and the head of the Cyber Security and Data Science Department at armasuisse Science and Technology. He is also the co-founder and chairman of the executive boards of the OpenSky Network and Electrosense associations. He holds a Ph.D (2006) and a Master's (2001) degree in electrical engineering and information technology both from ETH Zürich. He was also postdoctoral research faculty in 2007 at Princeton University in the USA.