Aircraft Trajectory Prediction using ADS-B Data

YANG, X.; Sun, Junzi; Rajan, R.T.

Publication date
2022

Document Version
Final published version

Published in
42nd WIC Symposium on Information Theory and Signal Processing in the Benelux (SITB 2022)

Citation (APA)
YANG, X., Sun, J., & Rajan, R. T. (2022). Aircraft Trajectory Prediction using ADS-B Data. In J. Louveaux, & F. Quitin (Eds.), *42nd WIC Symposium on Information Theory and Signal Processing in the Benelux (SITB 2022)* (pp. 113-122)

Aircraft Trajectory Prediction using ADS-B Data

Xuzhou Yang
Faculty of EEMCS
Delft University of Technology
Delft, The Netherlands
Email: x.yang-20@student.tudelft.nl

Junzi Sun
Faculty of Aerospace Engineering
Delft University of Technology
Delft, The Netherlands
Email: j.sun-1@tudelft.nl

Raj Thilak Rajan
Faculty of EEMCS
Delft University of Technology
Delft, The Netherlands
Email: r.t.rajan@tudelft.nl

***Abstract*—Automatic Dependent Surveillance - Broadcast (ADS-B) is a surveillance technology that is used extensively in Air Traffic Control (ATC) applications. Aircraft equipped with ADS-B transponders actively broadcast navigation information such as position, altitude, and velocity, and thus ATC is able to track aircraft continuously, even in regions not covered by traditional radars. However, raw ADS-B messages are typically contaminated with noise, which is typically mitigated using model-based tracking methods to predict the trajectories. In this work, we propose and evaluate the performance of several filtering strategies for trajectory prediction on an existing open source *TrajAir* aircraft data set and our own data set i.e., collected by Delft university of technology (TUD). In our evaluation, we observe the standard Kalman filter cannot accurately track the aircraft trajectory, especially for sharply maneuvering targets. A fading-memory filter tracks maneuvering targets but introduces delay in estimates, and requires a trade-off between responsiveness and smoothness by target-specific parameter tuning. The Kalman filter with augmented process noise also involves similar trade-off and parameter tuning. Finally, the particle filter performs the best during target maneuvers but admits more noise during steady-state and increases computational cost. In this paper, we present various filtering techniques, and study the performance of these algorithms on the *TrajAir* and *TUD* aircraft data sets.**

***Index Terms*—ADS-B, Kalman filter, Particle filter.**

I. Introduction

Automatic Dependent Surveillance - Broadcast (ADS-B) is an surveillance technology that is used extensively in Air Traffic Control (ATC) applications. Aircraft broadcast ADS-B messages periodically with on-board Mode-S transponders, which include navigational information such as surface/airborne position, airborne velocity, call sign, operational status, etc. ADS-B enables ATC ground stations to track aircraft continuously in regions that are not covered by traditional radars, as its coverage can be greatly extended by ground-based or space-based ADS-B receivers. It is considered to be a key component of the future air transportation system and is mandated both by EUROCONTROL [3] in Europea and FAA [4] in the U.S. since 2020.

Prior to the introduction of ADS-B, ATC applications heavily relied on the primary surveillance radar (PSR) and the secondary surveillance radar (SSR). PSR provides slant distance as well as aircraft's azimuth information with respect to the radar location, while SSR provides aircraft's altitude and identity. However, inherent limitations of PSR and SSR technology hinder further improvement in accuracy and coverage. ADS-B is thus introduced to enhance situational awareness for ATC controllers and pilots.

In this paper, we evaluate several model-based tracking methods on aircraft ADS-B data set. Section II explains the information provided by the ADS-B messages and data pre-processing techniques used to extract the relevant data. In Section III, we build theoretical foundations of our tracking methods with state space models and a Bayesian framework and introduce the standard Kalman filter under constant velocity dynamics (CV-KF). We further explore advanced filtering algorithms in Section IV, i.e., the Kalman filter with augmented noise (AP-KF), fading-memory Kalman filter (FM-KF), and the particle filter (PF) are introduced. In Section V, these methods are applied to predict aircraft trajectories, and a comparison between different methods and more comments are provided. Finally, in Section VI we summarize the results, with insights on future work.

II. ADS-B Decoding and Pre-processing

Nowadays, most of the aircraft are equipped with an ADS-B system. It is thus easy to acquire these ADS-B signals, and thus data, using an appropriate reciever system. Furthermore, an open-source package pyModeS[1] [10] provides us comprehensive functionalities to decode these ADS-B messages. In this section, we briefly look at message parsing, and the pre-processing of decoded data and relevant assumptions.

A. Description of data set

In this paper, we work with two realistic ADS-B data sets. The first data set is an open source data set called the *TrajAir* dataset [2], contributed by the AirLab from the robotics institute at Carnegie Mellon University. The other data set is collected by the faculty of aerospace engineering (AE) at TU Delft [5], which we call the *TUD* data set.

1) TrajAir data set: The *TrajAir* data set contains fully decoded ADS-B messages. The data set is collected at the Pittsburgh-Butler Regional Airport. In this data set, we have

This work is partially funded by the European Leadership Joint Undertaking (ECSEL JU), under grant agreement No 876019, the ADACORSA project - "Airborne Data Collection on Resilient System Architectures."

[1]https://github.com/junzis/pyModeS
[2]https://theairlab.org/trajair/

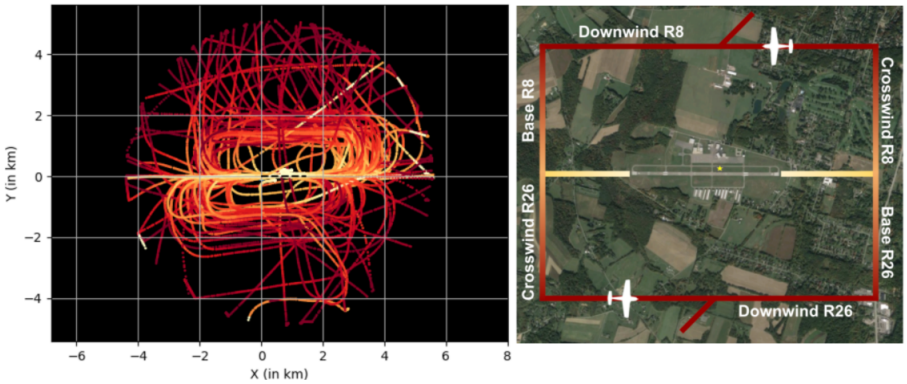


Fig. 1: *TrajAir* dataset: The left figure shows a snippet of processed aircraft trajectories and right figure demonstrates the left traffic pattern and nomenclature for the runways at the airport [2].

the information about an aircraft's status, including timestamps, geographical coordinates, velocity readings, track angles, altitudes, and vertical rates at every valid time instance. That is to say, the alternate transmitting behaviour of ADS-B is not seen. Trajectories of landing or takeoff of a group of aircraft are visualized in Fig.1. We can clearly see the lobes for traffic patterns around this airport. The right part of this figure shows the "Left Traffic" patterns. These patterns are rectangular-shaped with left-handed turns relative to the direction of landing or takeoff. Lighter color of trajectories for lower altitude [2].

2) TUD data set: The *TUD* data set contains demodulated (*not* decoded) ADS-B signals. The data set records about 15 minutes of air traffic near the region of Delft, covering most part of the southern Holland. Every entry contains timestamp, International Civil Aviation Organization (ICAO) address, receiving power, garbling (True or False), cyclic redundancy check (CRC) sign, and the 112-bit message string. ADS-B broadcasts different types of messages alternately. For position and velocity messages, which are of particular interest in our application, the airborne transmitting frequency is 2Hz. However, the *TUD* data set provides trajectories mostly from commercial jets. To include more additional aircraft trajectories, we use the *TrajAir* data set, which also contains trajectories from light general aviation (GA) aircraft.

B. Decoding

In this section, we summarize the details of time of arrival decoding, airborne position and airborne velocity decoding from [1].

1) Time of arrival decoding: ADS-B is not designed to contain any time of transmit information, but both data sets timestamp received signals. So we associate aircraft positions with time of arrival instead. Assumptions are made when we replace time of transmit with the time of arrival. We assume that for a sequence of messages, the time of propagation from source to receiver is very short and approximately the same. That means: 1) the aircraft does not travel a large distance between two consecutive transmissions with reference to the receiver; 2) there is no large difference in propagation time due to multi-path. In this paper, these two assumptions generally hold.

2) Airborne position decoding: A typical airborne position message contains longitude, latitude, and altitude of an aircraft. It is trivial to decode altitude but longitude and latitude are encoded in Compact Position Reporting (CPR) format. We use locally unambiguous position decoding for our own data set. Locally unambiguous position decoding [1] requires a known reference position. It should be close to the decoded position, e.g., within a range of 180 nautical miles (NM). The advantage is that from every piece of encoded messages we can decode a position. Here we choose the faculty of Aerospace Engineering building as the reference point.

3) Airborne velocity decoding: The airborne velocity message reports velocity decomposed in East-West, North-South and vertical directions. In the field of civil aviation, it is common to compute the track angle without considering altitude changes. It is trivial to decode the message itself. But it is worth noting that only ground speed can be used in our application. The ground speed of aircraft is the sum of the true airspeed vector and the wind velocity vector.

C. Data pre-processing and formatting

After decoding, we reorganize the data into tables of records. A record or a row in a table contains an aircraft's two-dimensional positions and velocity, associated with a timestamp. The initial timestamp and positions are set to zeros and other timestamps and positions in this table are calculated with respect to this. A table contains consecutive records for an aircraft. A ready-for-use table of a BOEING 737-4Z9 flying over southern Holland is shown in Table I. The first available position in the data sequence is set as $(0, 0)$. Based on the assumptions in section II-B, we assign messages received within, say, one-second interval with the *same* timestamp. Similar pre-processing is applied to the *TrajAir* dataset as well, but with a finer step in time. This pre-processing technique makes sure that for each time instance both the position and velocity data are available.

1) Two flight scenes: We chose two typical flight scenes from each data set respectively, which we refer to as *Scene*

TABLE I: A snippet of fully decoded ADS-B data

Time(s)	icao	P_x(m)	P_y(m)	V_x(m/s)	V_y(m/s)
0	4CA8AD	0	0	175.8	-143.9
1	4CA8AD	157.6	-129.0	175.8	-143.9
1	4CA8AD	241.1	-197.4	175.8	-143.9
2	4CA8AD	334.2	-273.6	175.8	-143.9
2	4CA8AD	412.9	-338.0	175.8	-143.9
3	4CA8AD	505.3	-413.7	175.8	-143.9
3	4CA8AD	594.7	-486.9	175.8	-143.9
4	4CA8AD	704.0	-576.4	175.8	-143.9
4	4CA8AD	764.6	-626.0	175.8	-143.9
5	4CA8AD	849.8	-695.8	175.8	-143.9

1 and *Scene 2* in this paper. We use X_{data} to denote raw trajectories.

- *Scene 1:* This scene contains a trajectory of an jet liner, which flew in linear motion for some time period, and then made a lazy turn. The aircraft maintains cruising speed and altitude in this period. The trajectory of the aircraft is shown in Fig.2a.
- *Scene 2:* It contains the landing trajectory of a GA aircraft. The aircraft made sharper turns and changed its velocity frequently. Compared to that in Scene 1, this trajectory exhibits more abrupt changes in states. This is shown in Fig.2a.

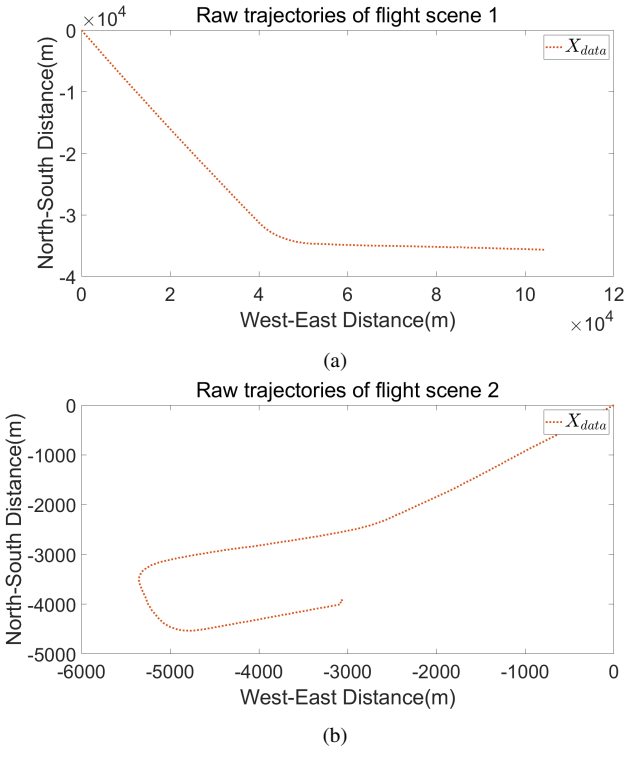


Fig. 2: Two flight scenes. Scene 1 is from the *TUD* data set and Scene 2 is from the *TrajAir* data set.

III. State space model and Kalman Filter

Almost all existing tracking methods heavily rely on the models of the aircraft motion. However, in our task, the precise knowledge of aircraft dynamics is not assumed. Furthermore, it is not computationally efficient to use a very sophisticated model. Thus, we rely on simple dynamic models in this application. It is then an important problem that how we can mitigate the model mismatch caused by this oversimplification.

A. Kalman filter

The task of trajectory prediction can be considered as a state estimation problem. It requires the algorithm to retrieve signal of interest from noisy data and construct a reasonable (regarding target dynamics) trajectory from available data. If we consider a linear Gaussian state space model, then the Kalman filter (KF) is an optimal filter. Here, the process and the measurement equations are given respectively by

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} \quad (1)$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \quad (2)$$

Here, the previous state $\mathbf{x}_{k-1}$ is transformed to the current state $\mathbf{x}_k$ by the process matrix $\mathbf{F}$ and corrupted by process noise $\mathbf{w}_{k-1}$. The second equation, i.e., the measurement equation describes how the system's output $\mathbf{z}_k$ is related to internal states through measurement matrix $\mathbf{H}$. In the set up of the Kalman filter, we assume the noise to be white, zero-mean Gaussian, and independent from each other, which can be represented as

$$p(\mathbf{w}) \sim \mathcal{N}(0, \mathbf{Q}) \quad (3)$$

$$p(\mathbf{v}) \sim \mathcal{N}(0, \mathbf{R}) \quad (4)$$

The Kalman filter works in a recursive manner. At each recursive step, it performs prediction and then correction, and computes a factor called Kalman gain. This factor controls the trade-off between prior knowledge and data.

We first define the initial state and posterior covariance matrix, $\mathbf{x}_0$ and $\mathbf{P}_0$. $\mathbf{x}_0$ is simply set according to the first available ADS-B record in a data sequence. Practically, we fill all diagonal entries of $\mathbf{P}_0$ with positive values. According to [6], whether the initial values are large or small, the filter always converges.

For the prediction phase:

$$\hat{\mathbf{x}}_k^- = \mathbf{F}\hat{\mathbf{x}}_{k-1} \quad (5)$$

$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \quad (6)$$

and the correction phase, we have

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \quad (7)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-) \quad (8)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \quad (9)$$

Here we use $\hat{\mathbf{x}}_k^-$ to denote the predicted state at the k-th step while $\hat{\mathbf{x}}_k$ denotes the corrected state estimate at the k-th step. Similar notations are used for $\mathbf{P}_k^-$ and $\mathbf{P}_k$. The matrix $\mathbf{K}_k$ is the Kalman gain computed at the k-th step.

B. Constant-velocity dynamic model

To ensure the smooth functioning of the Kalman filter, we need to choose proper dynamic models. Commercial airliners usually maintain designated speed, heading, and altitude during en route flying. The movement can be considered as uniform linear motion. Therefore, a constant velocity (CV) model is sufficient in most cases. We consider the CV model in a two-dimensional space, with x and y representing two-dimensional positions and $\dot{x}$, $\dot{y}$ two-dimensional velocities. The state vector is then defined as $\mathbf{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^T$. Given recursive time step δt, the process matrix $\mathbf{F}$ is given by

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Note that in this task we define a set of states whose measurements can be directly extracted from ADS-B data (i.e., positions and velocities), which is equivalent to set $\mathbf{H}$ as an identity matrix.

C. Constant-velocity Kalman filter (CV-KF)

The constant velocity Kalman filter (CV-KF) is a Kalman filter under the assumption of constant-velocity dynamics. The pseudo code for CV-KF is presented in Algorithm 1.

Algorithm 1 The CV-KF filter

Input: $\mathbf{x}_0$, $\mathbf{P}_0$, sequence of data $\mathbf{z}_{1,k}$
Output: Sequence of state estimates $\mathbf{x}_{1,k}$, posterior covariance $\mathbf{P}_{1,k}$ and Kalman gains $\mathbf{K}_{1,k}$
Initialize $\mathbf{x}_0$ and $\mathbf{P}_0$
For $t = 1$ to n **do**
    Project $\mathbf{x}_{t-1}$ to $\mathbf{x}_t^-$
    Project $\mathbf{P}_{t-1}$ to $\mathbf{P}_t^-$
    Compute $\mathbf{K}_t$
    Update $\mathbf{x}_t^-$ to $\mathbf{x}_t$ with $\mathbf{K}_t$ and $\mathbf{z}_t$
    Update $\mathbf{P}_t^-$ to $\mathbf{P}_t$ with $\mathbf{K}_t$
end

The CV-KF is guaranteed to give optimal estimates under the assumption of the linear model with white Gaussian noise. However, real systems do not always fulfill these assumptions. and non-linear dynamics cannot be ignored. Specifically, in our application, when an aircraft is close to the airport, it must follow certain arrival or departure procedures. The procedures may require the aircraft to make a series of turns and pass designated waypoints in order to align with the runway.

We evaluate the performance of CV-KF in the two scenes from the *TrajAir* and *TUD* datasets respectively, which was discussed in Section II-C1.

- *Scene 1:* The results are shown in Fig.3a - 3c, where X_{CV-KF} denotes the filtered trajectory by CV-KF and X_data the raw data points. Here, x, y are position coordinates and $\dot{x}$ and $\dot{y}$ velocities. From Fig.3a and Fig.3b we can observe that the filter diverges on position estimates when the target performs a turn. Fig.3c shows that the filter fails to estimate velocity.
- *Scene 2:* The results are shown in Fig.4a - 4c, where the correction always lags the transition of states. It seems that after the filter enters a steady state, it loses the ability to track changes in the aircraft's states.

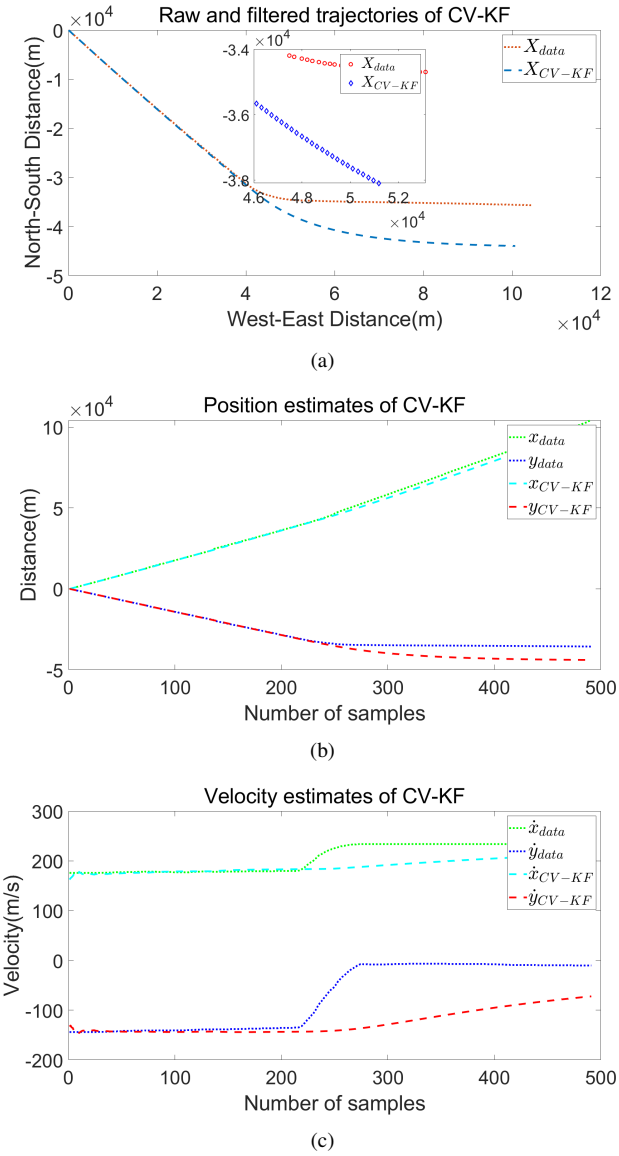


Fig. 3: *Scene 1:* Predicted trajectory, position and velocity estimates using CV-KF

The CV-KF fails mainly due to model mismatch, and thus the assumed oversimplified model fails to capture the real dynamics of maneuvering aircraft. Hence, we have to adapt our algorithm to enable accurate tracking. A good algorithm relies on both the model to capture dynamics and the filter to fuse prior knowledge and data.Traditionally, more advanced dynamical models have been proposed, however there is no silver bullet to this problem. Alternatively, we can retain the simplified CV-model but propose advanced filtering techniques

to enable more accurate tracking.

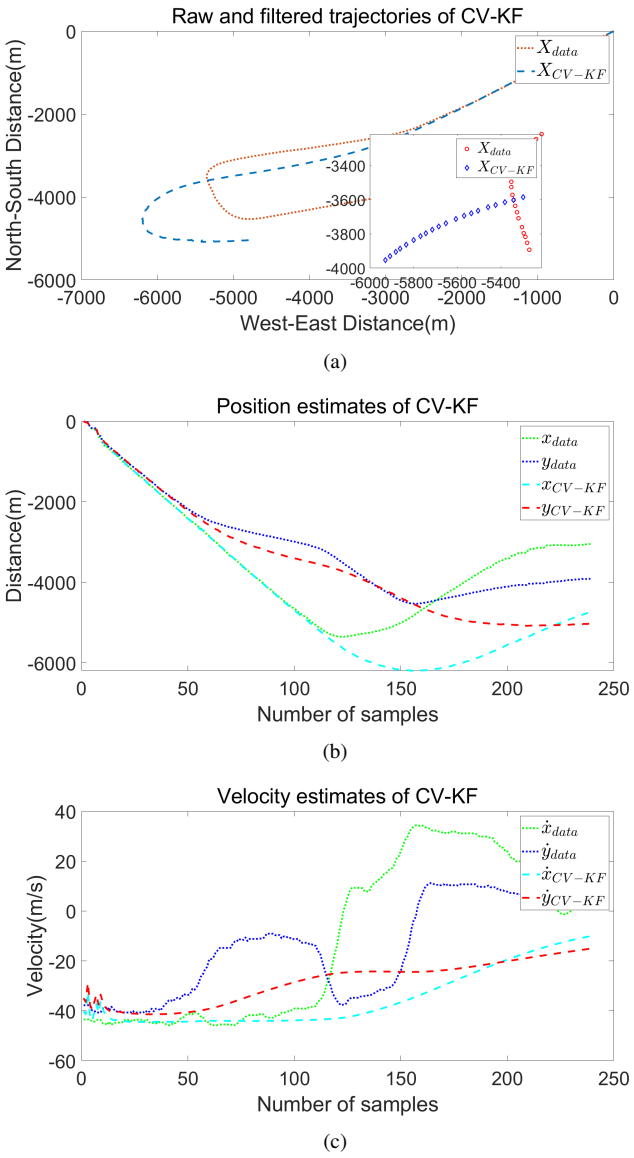


Fig. 4: *Scene 2:* Predicted trajectory, position and velocity estimates using CV-KF

IV. Advanced filtering algorithms

In this section we explore advanced filtering algorithms, to overcome the limitations of the CV-KF discussed in the previous section.

A. Augmented process noise Kalman fitler (AP-KF)

We now introduce the Augmented process noise Kalman Filter (AP-KF). Recall that the posterior covariance matrix is computed by

$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \tag{11}$$

To augmented process noise is equivalent to increase the values in diagonal entries of the $\mathbf{Q}$ matrix. Then for each step $\mathbf{P}_k^-$ will increase, as compared to that of the CV-KF. As mentioned above, to remove the measurement noise as much as possible, sometimes we set $\mathbf{Q}$ as zero. But AP-KF incorporates $\mathbf{Q}$ to compensate for model mismatch [7]. The Kalman gain is now given by $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1}$, and since $\mathbf{H}$ is an identity matrix, and both $\mathbf{P}^-{}_k$ and $\mathbf{R}$ are diagonal matrices, the Kalman gain is reduced to

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_k^-(\mathbf{P}_k^- + \mathbf{R})^{-1} \\ &= \begin{bmatrix} \frac{p_{k1}}{p_{k1}+r_1} & & & \\ & \frac{p_{k2}}{p_{k2}+r_2} & & \\ & & \frac{p_{k3}}{p_{k3}+r_3} & \\ & & & \frac{p_{k4}}{p_{k4}+r_4} \end{bmatrix}\end{aligned} \tag{12}$$

where p_{ki} and r_i refer to diagonal elements of $\mathbf{P}_k^-$ and $\mathbf{R}$, respectively. It is then clear that every diagonal entry of $\mathbf{K}_k$ increases as p_{ki} increases or as every diagonal entry of $\mathbf{Q}$ increases. Thus the AP-KF increases the Kalman gain as compared to the CV-KF, giving more weights to the measurements. However, this benefit comes at the cost of admitting more measurement noise. The extreme case is that the process noise covariance is large enough such that the filter discards the prediction and simply follows the measurement, which is not desired, and hence this method requires the tuning of $\mathbf{Q}$.

B. Fading memory Kalman filter (FM-KF)

The fading memory Kalman filter (FM-KF) is an alternative method to augment the posterior covariance matrix. For older prediction and measurement, we aim to increase the covariance matrices by multiplying a factor greater than one, and let the factor shrink (but always greater than one) for newer predictions and measurements. Thus, the covariance matrices at are revised to be

$$\tilde{\mathbf{Q}}_k = \alpha^{K-2k+2}\mathbf{Q}_k, k \leq K \tag{13}$$

$$\tilde{\mathbf{R}}_k = \alpha^{K-2k}\mathbf{R}_k, k \leq K \tag{14}$$

where K denotes the total number of time steps in the filtering process.

After some mathematical manipulation, the final effect on posterior covariance matrix is almost identical to that of augmenting process noise [6]. The revised posterior covariance matrix is given by

$$\tilde{\mathbf{P}}_k^- = \alpha^2\mathbf{F}\tilde{\mathbf{P}}_{k-1}\mathbf{F}^T + \mathbf{Q}_{k-1} \tag{15}$$

The implementation of a FM-KF relies on the hyper parameter α. A larger α indicates that the "memory" is shorter and the filter is more able to track changes of target's states. In our application, a larger α gives the filter more flexibility to handle maneuvers. However, in practise we have to tune α for every given target, which is a limitation.

C. Particle filter (PF)

In Section III-C, we discussed the model mismatch i.e., discrepancy between the assumed linear model and the actual nonlinear model. Moreover, precise knowledge of the sensor noise model is not assumed. We do not know the statistics of

the measurement noise, nor do we know if it is appropriate to assume the noise to be Gaussian. Furthermore, we observe that the process noise is not straightforward to model due to external factors e.g., atmospheric disturbances. Therefore, we need a filtering method that does not depend on the restrictive assumptions of Gaussian linear state space models, for example the particle Filter (PF) [12].

The particle filter is an instance of sequential importance sampling (SIS), where we are interested in a general state space model of the form

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \mathbf{w}_k) \quad (16)$$

$$\mathbf{z}_k = h_k(\mathbf{x}_k, \mathbf{v}_k) \quad (17)$$

where for the kth time instance, $f_k(\cdot)$ denotes the process model, $h_k(\cdot)$ denotes the measurement model, $\mathbf{w}_k$ indicates the process noise, and $\mathbf{v}_k$ indicates the measurement noise. Note that a Gaussian linear model is not assumed here. Under the assumption that the Markov property holds, we have

$$p(\mathbf{x}_k|\mathbf{x}_{1,k-1}, \mathbf{z}_{1,k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \quad (18)$$

$$p(\mathbf{z}_k|\mathbf{x}_{1,k}, \mathbf{z}_{1,k-1}) = p(\mathbf{z}_k|\mathbf{x}_k) \quad (19)$$

We start with the sequential estimation of $p(\mathbf{x}_{1,k}|\mathbf{z}_{1,k})$ and the estimation of the marginal $p(\mathbf{x}_k|\mathbf{z}_{1,k})$ will be a by-product. Using (18) and (19), we can write

$$\begin{aligned} p(\mathbf{x}_{1,k}|\mathbf{z}_{1,k}) &= p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{1,k-1}|\mathbf{z}_{1,k-1}) \\ &= \frac{p(\mathbf{x}_{1,k}, \mathbf{z}_{1,k})}{\int p(\mathbf{x}_{1,k}, \mathbf{z}_{1,k}) d\mathbf{x}_{1,k}} = \frac{p(\mathbf{x}_{1,k}, \mathbf{z}_{1,k})}{Z_k} \end{aligned} \quad (20)$$

where Z_k is the normalizing constant at the kth instant. The posterior distribution is proportional to the joint distribution in (20), and thus we approximate $p(\mathbf{x}_{1,k}, \mathbf{z}_{1,k})$ via a set of generated particles. We define the weights of particles as

$$\omega_k(\mathbf{x}_{1,k}) = \frac{p(\mathbf{x}_{1,k}, \mathbf{z}_{1,k})}{q_k(\mathbf{x}_{1,k})} \quad (21)$$

where $q_k(\cdot)$ is a proposal distribution. According to sequential importance sampling [12], the recursive relation between the current and past weights are given by

$$\omega_k(\mathbf{x}_{1,k}) = \omega_{k-1}(\mathbf{x}_{1,k-1}) \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})}{q_k(\mathbf{x}_k|\mathbf{x}_{1,k-1}, \mathbf{z}_{1,k})} \quad (22)$$

where we exploit (20), and we choose a proposal distribution such that

$$q_k(\mathbf{x}_k|\mathbf{x}_{1,k-1}, \mathbf{z}_{1,k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k). \quad (23)$$

Finally, the estimation is obtained by

$$\hat{p}(\mathbf{x}_{1,k}|\mathbf{z}_{1,k}) = \sum_{i=1}^{N} W_k^{(i)} \delta(\mathbf{x}_{1,k} - \mathbf{x}_{1,k}^{(i)}) \quad (24)$$

$$\hat{p}(\mathbf{x}_k|\mathbf{z}_{1,k}) = \sum_{i=1}^{N} W_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}) \quad (25)$$

where N is the number of particles and $W_k^{(i)}$ is the normalized weight for the i-th particle at time k.

To combat the problem of degeneracy in practical use, resampling may be used. However, resampling can also limit parallel processing and cause sample impoverishment. Hence, it is only performed when the following metric

$$N_{eff} \approx \frac{1}{\sum_{i=1}^{N} (W_k^{(i)})^2} \quad (26)$$

is smaller than a preselected value N_T, which is typically $N_T = \frac{N}{2}$. We now present the pseudo code of the particle filter with SIS and resampling techniques.

Algorithm 2 The SIS particle filter

Input: N streams of particles from prior pdf p of $\mathbf{x}_0$
Output: N particles conformed to $p(\mathbf{x}_t|\mathbf{z}_{1,t})$

For $i = 1$ to N **do**
  Draw $\mathbf{x}_0^{(i)} \sim p(\mathbf{x})$; initialize N streams.
  Set $W_0^{(i)} = \frac{1}{N}$; All initial weights are equal.
end
For $k = 1$ to n **do**
  For $i = 1$ to N **do**
    Draw $\mathbf{x}_k^{(i)} \sim q(\mathbf{x}|\mathbf{x}_{k-1}^{(i)}, \mathbf{z}_k)$
    $\omega_k^{(i)} = \omega_{k-1}^{(i)} \frac{p(\mathbf{z}_k|\mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{z}_k)}$
  end
  For $i = 1$ to N **do**
    Compute normalized weights $W_k^{(i)}$
  end
  Compute N_{eff}.
  If $N_{eff} \leq N_T$; N_T preselected
    Resample $\{\mathbf{x}_k^{(i)}, W_k^{(i)}\}_{i=1}^N$ to get $\{\bar{\mathbf{x}}_k^{(i)}, \frac{1}{N}\}_{i=1}^N$
    Set $\mathbf{x}_k^{(i)} = \bar{\mathbf{x}}_k^{(i)}$, $\omega_k^{(i)} = \frac{1}{N}$
  end
end

V. Simulation and analysis

In this section we present the trajectory prediction results of the advanced filtering algorithms discussed in Section IV, i.e., augmented noise Kalman filter (AP-KF), the fading-memory filter (FM-KF), and the particle filter (PF). TableII lists some notations we use in this section.

TABLE II: Notations for different filters

Filter	Trajectory	Position coordinates	Velocities
AP-KF	X_{AP-KF}	$x_{AP-KF}\ y_{AP-KF}$	$\dot{x}_{AP-KF}\ \dot{y}_{AP-KF}$
FM-KF	X_{FM-KF}	$x_{FM-KF}\ y_{FM-KF}$	$\dot{x}_{FM-KF}\ \dot{y}_{FM-KF}$
PF	X_{PF}	$x_{PF}\ y_{PF}$	$\dot{x}_{PF}\ \dot{y}_{PF}$

A. AP-KF

As discussed in Section IV-A, we incorporate the $\mathbf{Q}$ matrix with its diagonal entries filled with positive values σ^2. In the experiments, we set the noise power σ to 10. Results of Scene 1 and Scene 2 are shown in Fig.5a - 5c and Fig.6a - 6c respectively. For both the scenes, the AP-KF predicts

the positions, velocities and generates smooth trajectories, reasonably well. On the outset, it seems that the raw and predicted trajectories are identical, however a closer look reveals the filter do not completely follow the measurements. We observe that the aircraft's velocities are much harder to track than positions. Comparing the velocity estimates in both scenes, we find that augmenting process noise is a reasonably good technique to track abrupt changes of states but may not give as satisfying results when the target is in steady state. In Fig.5c we observe that the estimates give many small spikes while the aircraft seems to maintain a constant velocity according the measurements. If we set a smaller noise power by decreasing the stand deviation values of the noise distribution, the magnitudes of these spikes will be smaller but it impairs the filter's tracking capability as well. There is a trade-off between smoothness and agility. Making covariance of process noise larger gives the filter more flexibility to handle maneuvers at the cost of being more vulnerable to disturbances.

B. FM-KF

A similar trade-off exists for the fading memory Kalman filters since the two methods, as explained, are fundamentally identical. More interestingly, FM-KF has a tunable parameter α, which is set between 1 and 1.5 practically. A larger α forces the filter to have a "shorter memory". For Scene 1, α is set to 1.05, and we observe in Fig.7a - 7c that while the filter gives good position estimates, the velocity estimates are notably lagging from the velocity values. In the case of Scene 2, we explore two values of α i.e., $\alpha = 1.2$ and $\alpha = 1.5$. The results of Scene 2 are presented in Fig.8a and Fig.8c, where we observe for a larger α, we have less lag and less smoothness in the prediction. Hence, for the FM-KF the parameter α controls the trade-off. A small α makes the filter to rely more on past measurements, which makes the filter more stable and generates smoother results. At the same time, the filter will be less responsive. A large α, however, forces the filter to quickly adapt to the changes of target's states and admit more noise.

C. PF

The last set of results are produced for the particle filter, which is shown in Fig.9a - 9c and Fig.10a - 10c, for Scene 1 and Scene 2 respectively. An important parameter to tune is the number of particles initialized (N). Unlike the Kalman filters, PF relies on Monte Carlo simulation to sample from probability distributions of our interest. It starts from any arbitrary initial distribution and gradually adjusts the distribution to be more and more similar to real posterior distribution. So with more initialized particles the filter can obtain more samples for adjustment and gives a more precise approximation. See for example, the Fig.10c, where not surprisingly, the prediction with 20000 particles is smoother than that with only 5000 particles. Moreover, due to the extreme flexibility of PF, both trajectories have no evident lag or deviation, which indicates that PF effectively handles the model mismatch problem. PF is also more robust, with better responsiveness and smoothness, particularly in contrast to FM-KF.

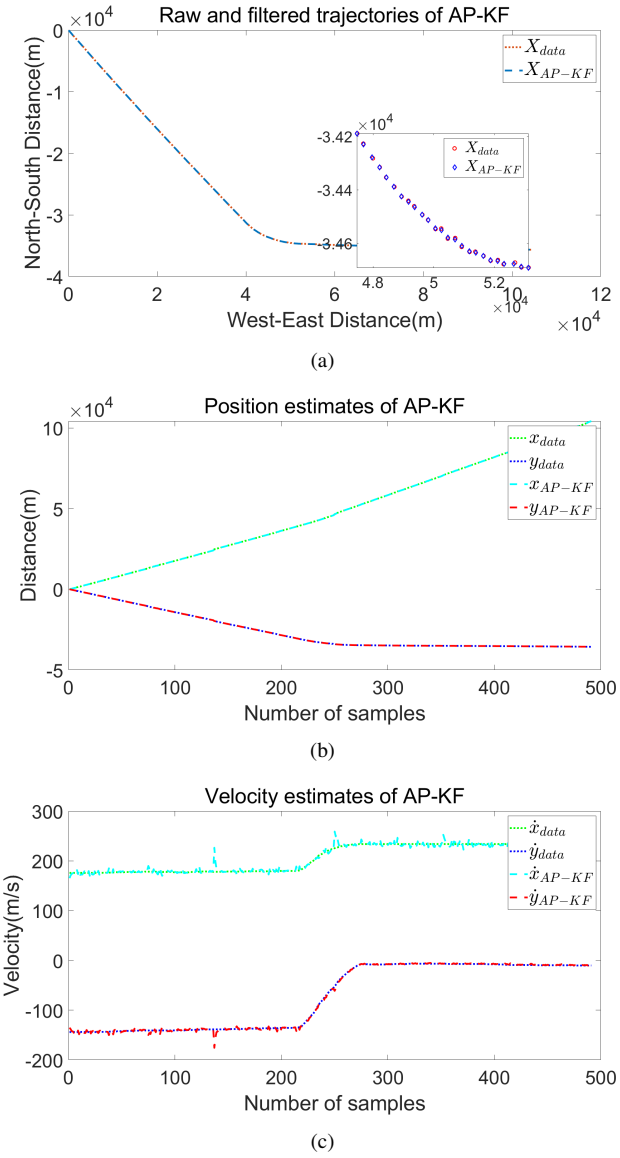
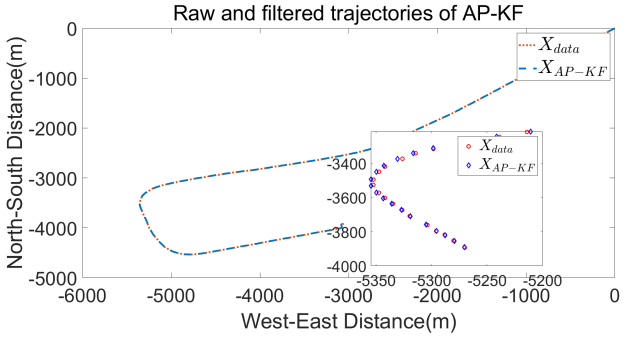


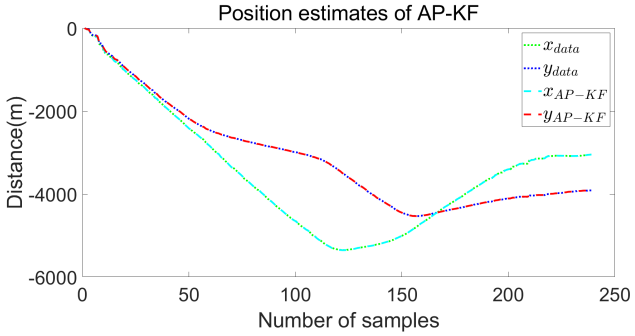
Fig. 5: *Scene 1:* Predicted trajectory, position and velocity estimates using AP-KF

VI. Conclusion and Future Work

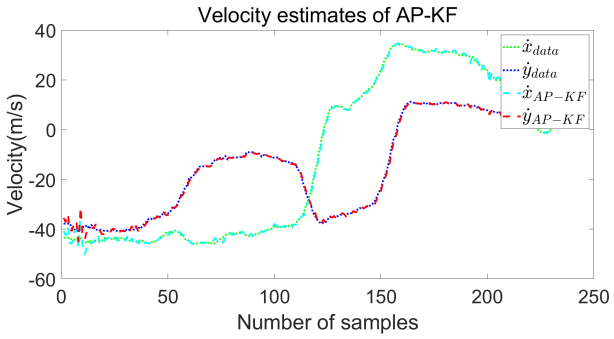
In this paper, we presented relevant knowledge of working with ADS-B system and evaluated several target tracking methods on realistic ADS-B data sets. These methods are based on models of simplified flight dynamics. We discussed the CV-KF algorithm and showed through experiment results that it does not meet our requirements, due to simplified Gaussian linear state space model assumption. To overcome the limitation of CV-KF, we propose three improved methods, namely the AP-KF, FM-KF, and PF. Simulation results show that all three methods offer improvements as compared to the CV-KF on trajectory, position and velocity estimation in the two flight scenes. In particular PF outperforms the other



(a)

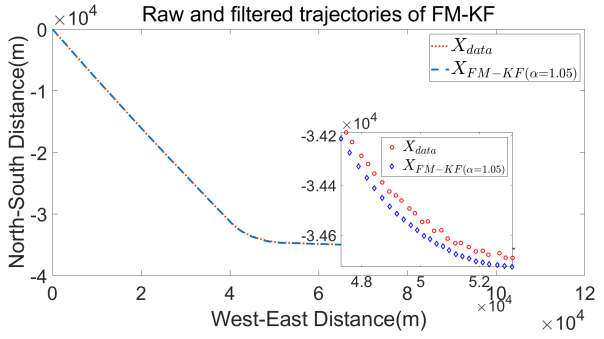


(b)

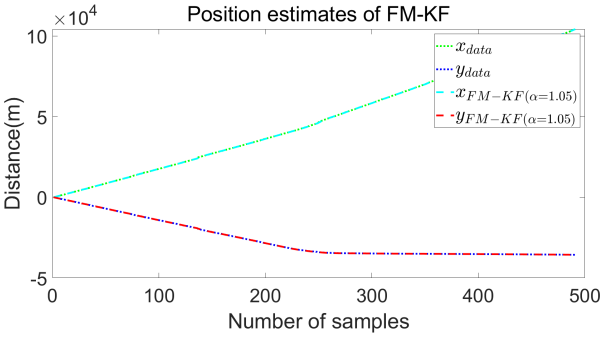


(c)

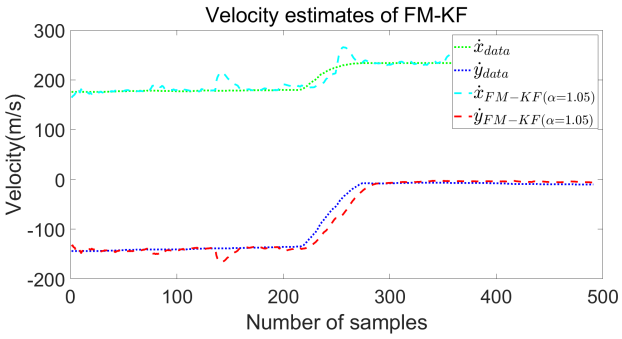
Fig. 6: *Scene 2:* Predicted trajectory, position and velocity estimates using AP-KF



(a)



(b)



(c)

Fig. 7: *Scene 1:* Predicted trajectory, position and velocity estimates using FM-KF

solutions, since it overcomes the underlying assumption of Gaussian linear state space models, of the Kalman filters.

The accuracy of PF for the prediction comes with the cost of large memory and computational load, and hence other non-linear filtering methods and non-parametric models could be explored. In general, the use of only one dynamic model creates a bottleneck for the tracking performance, hence a multiple model approach or data-driven approaches could yield more optimal results [11]. Furthermore, in this paper we explored only a limited part of the data set i.e., 2 scenes, and that from 2 flights, which is a limitation. Additional experiments which use a large number of flights from both the *TrajAir* and *TUD* data sets must be evaluated to investigate the performance of the proposed solutions.

REFERENCES

[1] J. Sun, *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals*, 2nd ed. TU Delft OPEN Publishing, 2021.

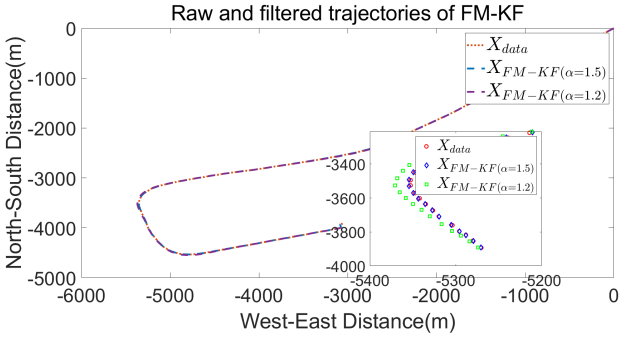
[2] J. Patrikar, B. Moon, J. Oh, and S. Scherer, "Predicting Like A Pilot: Dataset and Method to Predict Socially-Aware Aircraft Trajectories in Non-Towered Terminal Airspace," arXiv:2109.15158 [cs.RO], Sep. 2021.

[3] Regulation (EU) 2020/587 by European Union Aviation Safety Agency. Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32020R0587.

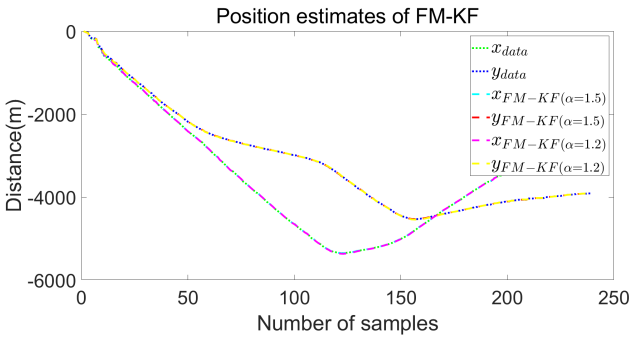
[4] Code of Federal Regulations by Federal Aviation Administration. Retrieved from https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-91/subpart-C/section-91.225.

[5] W. Huygen, "ADS-B Signal Integrity and Security Verification Using a Coherent Software Defined Radio: Mitigation of the threat of maliciously injected signals in ADS-B networks," M.S. thesis, faculty of aerospace engineering, TU Delft, Delft, 2021. Available: http://resolver.tudelft.nl/uuid:1129afb3-304f-4c63-afa3-ea9f2bd73f86.

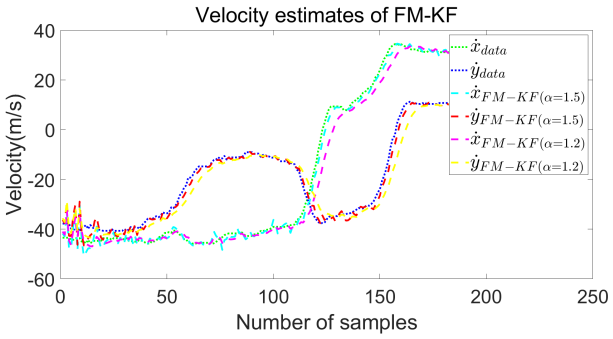
[6] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2006.



(a)



(b)



(c)

Fig. 8: *Scene 2:* Predicted trajectory, position and velocity estimates using FM-KF

[7] P. Zarchan and H. Musoff, *Fundamentals of Kalman Filtering: A Practical Approach*, VA, Reston:American Institute of Aeronautics and Astronautics, pp. 616-617, 2005.

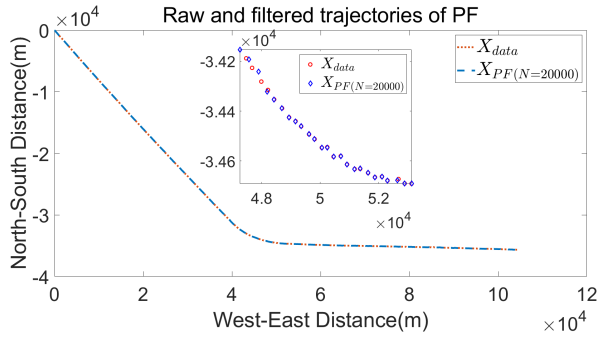
[8] B. Ristic, M. S. Arulampalam, N. Gordon, *Beyond the Kalman Filter, Particle Filters For Tracking Applications*, Artech House, 2004.

[9] A. F. M. Smith and A. E. Gelfand, "Bayesian statistics without tears: A sampling-resampling perspective", Amer. Statist., vol. 46, no. 2, pp. 84-87, 1992.

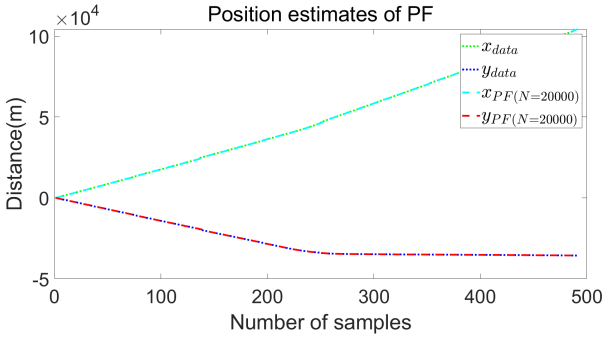
[10] J. Sun, H. Vû, J. Ellerbroek and J. M. Hoekstra, "pyModeS: Decoding Mode-S Surveillance Data for Open Air Transportation Research," in IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 7, pp. 2777-2786, July 2020, doi: 10.1109/TITS.2019.2914770.

[11] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part V. Multiple-model methods," in IEEE Transactions on Aerospace and Electronic Systems, vol. 41, no. 4, pp. 1255-1321, Oct. 2005, doi: 10.1109/TAES.2005.1561886.

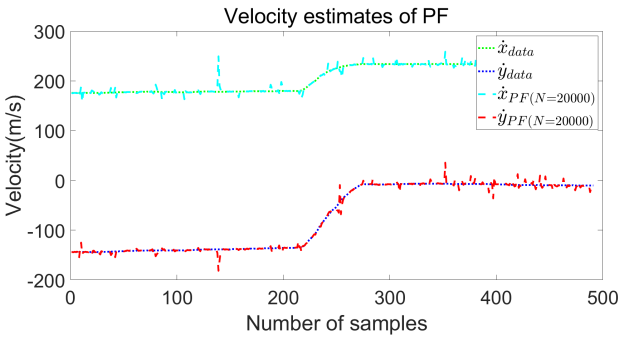
[12] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*. San Diego, CA, USA: Academic, 2015.



(a)

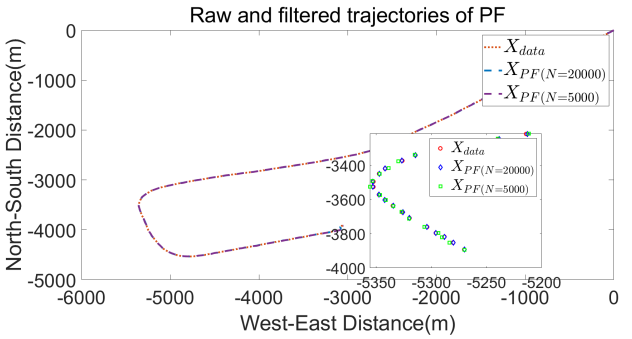


(b)

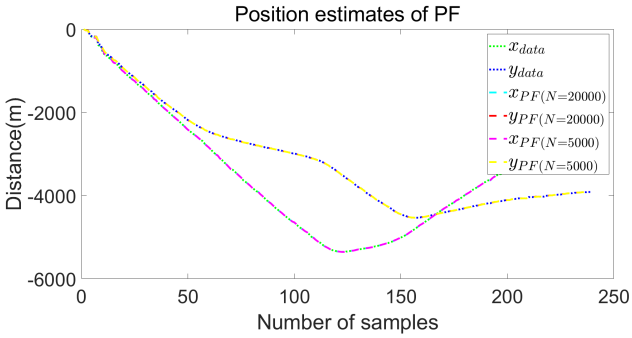


(c)

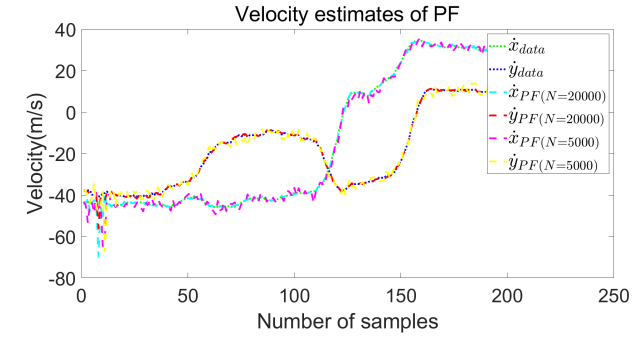
Fig. 9: *Scene 1:* Predicted trajectory, position and velocity estimates using PF



(a)



(b)



(c)

Fig. 10: *Scene 2:* Predicted trajectory, position and velocity estimates using PF