Efficient and Effective 4D Trajectory Data Cleansing

*TAN Xin, SUN Xiaoqian, ZHANG Chunxiao, WANDELT Sebastian**

National Key Laboratory of CNS/ATM, School of Electronic and Information Engineering, Beihang University,
Beijing 100191, P.R. China

(Received ; revised ; accepted)

Abstract: As the rapid development of aviation industry and newly emerging crowd-sourcing projects such as Flightradar24 and FlightAware, large amount of air traffic data, particularly four-dimension (4D) trajectory data, have become available for the public. In order to guarantee the accuracy and reliability of the results, data cleansing is the very first step in analyzing 4D trajectory data, including error identification and mitigation. This study investigates data cleansing techniques for the 4D trajectory data as follows. Back propagation (BP) neural network algorithm is applied to repair errors. Newton interpolation method is used to obtain even-spaced trajectory samples over a uniform distribution of each flight's 4D trajectory data. Furthermore, we propose a method to compress the data while maintaining the intrinsic characteristics of the trajectories. Density-based spatial clustering of applications with noise (DBSCAN) is applied to identify remaining outliers of sample points. Experiments are performed on a data set of one-day 4D trajectory data over Europe. The results show that our proposed techniques achieve more efficient and effective reults than existing approaches. Our work contributes to the first step of data preprocessing and lays foundation to further downstream 4D trajectory analysis.

Key words: 4D trajectories; data cleansing; outlier detection; repair

CLC number: U8 **Document code:** **Article ID:**

0 Introduction

The aviation industry has been developed rapidly in recent years, and the air transportation system is facing with huge challenges for both management and technologies. The emergence and dissemination of data science provide powerful tools to analyse and manage the air transportation system. Air traffic data, especially 4D trajectory data are used for various analysis tasks, such as aircraft conflict detection and resolution[1], airspace congestion management[2], and air navigation route optimization[3]. 4D trajectories refer to a series of points composed of the coordinates of longitude, latitude, altitude and the corresponding timestamp of aircraft in the air. With the emergence of 4D trajectory data, airborne calculations can be used to predict the sequence of the aircraft at the intersection of the busy or congested airspace, and it can facilitate the decision-making process of the air traffic controllers.

As the driving force of scientific and technological innovation, "data" accounts for a rising proportion of assets, and increasingly becomes another major factor of production after "land" and "capital". Data cleansing is the prerequisite for the 4D trajectory data analysis. In order to obtain more accurate and reliable results, outliers and unreliable data must be cleansed beforehand. Otherwise, it will not only prolong the data processing time and efforts, but also mislead the final result. In the field of air transportation, research focusing on data cleansing is still relatively rare[4-6], and the percentage of abnormal data is not very large. However, they are widely distributed[7], almost throughout every trajectory, affecting the reliability of the results negatively. The common abnormal data includes data timestamp error[4,7]; missing data, such as longitude and latitude, identification information and duplicated data[4,5,7]. In addition to these anomalies, there are also some logical errors that are hard to find intuitively,

***Corresponding author:** Wandelt Sebastian, Professor, E-mail: wandelt@buaa.edu.cn

How to Cite this article:

such as data jumping[5,7,8] which indicates too large distance, too long time gap or too fast speed, etc. Therefore, there is an urgent requirement for 4D trajectory data cleansing method.

In this paper, we present a rich set of data cleansing methods to deal with 4D trajectory data. BP neural network is proposed to repair the errors which are detected by average speed between two adjacent points in the trajectories. Newton interpolation is used to filter out inconsistent data and to fix the frequency of the points in 4D trajectories. To reduce the experimental complexity for further analysis tasks while maintaining the characteristics of the trajectories, we apply the unit cube sampling method to cut down the data size. In addition, the clustering method DBSCAN is used to identify outliers of trajectories. We carry out experiments on one-day 4D trajectory dataset in European area. Results show that our proposed techniques can better cleanse the 4D trajectory data.

The remainder of the paper is organized as follows. We provide the literature review on data cleansing techniques in Section 1. The methodology of data cleansing for 4D trajectory data is proposed in Section 2. Experimental results are reported in Section 3. The paper concludes with Section 4.

1 Literature Review

Research on data cleansing mainly focuses on improvement and management of the data quality. Although there are significant achievements, the investment on data quality control and data cleansing needs to increase continually[9]. Basic principles of data cleansing techniques include identifying determinants which affect the data quality, defining the cleansing requirements, and establishing the cleansing model[10-13].

Data cleansing is the very first step in the data preprocessing for trajectory data analysis and applications, however, most papers did not provide sufficient justifications for the data cleansing or data preprocessing before trajectory analysis[5,8]. To check the integrity of ADS-B data, Krozel J, et al.[5] apply a suite of Kalman filters to smooth out noise, identify and suppress erroneous data, coast between data dropouts, and provide the current best state estimates. Experiments are performed on simulated ADS-B data signals and demonstrate that the approach is promising to data integrity check. 4D trajectory data cleansing is applied by Patroumpas K, et al.[8], who develop one-pass heuristics to eliminate inherent noise. The authors provide reliable trajectory representations and present various bounds for trajectory error detection. Since it handles trajectories online and discards the errors directly, it is inapplicable with data offline. To manage ADS-B data collection efficiently and integrate with other flight related data, Martínez-Prieto M A, et al.[14] devise AIRPORTS DL with Data Lake architecture to reconstruct gate-to-gate trajectories and to derive parameters, such as the predictability or the fuel consumption. It discards useless messages or aligns field values to satisfy the AIRPORTS data model, and determines trajectories when different flights use the same callsign as well. However, it requires extra information such as the history trajectory data to detect these data. The traffic library for the Python programming language introduced by Olive X, et al.[15] present how to access different sources of data, leverage processing methods to clean, filter, clip or resample trajectories, and compare trajectory clustering methods on a sample dataset of trajectories above Switzerland. The paper handles missing data, slicing, querying or resampling with Pandas library, but it is not usually sufficient for a high requirement of data cleansing.

BP neural network algorithm is widely used in various fields such as aviation industry[16], manufacturing industry[17], engineering[18], medical industry[19], geology[20] etc.. Lin Sen, et al.[21] establish a sensor error correction model which combines particle swarm optimization (PSO) with BP neural network algorithm to reduce nonlinear characteristics and improve test accuracy of the system. BP neural network has three or more than three layers, including the input layer, hidden layer and the output layer. The upper and lower layers are connected completely. There is no connection between each neuron in the same layer. To construct a BP neuron network, first, set random parameters of each layer. Calculate the output based on input data and the initial parameters. Next, following the direction of reducing the loss between output and the actual target, amend the connection from the output layer weights to the middle layer-by-layer, and finally return to the input layer [22].

In this study, our goal is to clean data for the preparation of further data analysis. We present a framework for cleansing 4D trajectories in the following section.

2 Methodology

As the recorded 4D trajectory data can be erroneous, the common abnormal data includes data timestamp error, missing data, duplicated data and off-couse data[4,5,7,8]. Data cleansing is the very first step before further data analysis. First, typical statistical outlier detection techniques are used to detect the errors. We focus on the flight speed element, and the data with high speed which beyond the maximal flight speed will be fixed by applying BP neural network or dropped out directly if the data is unnecessary. Second, the 4D trajectory data of one flight may be recorded in 8 minutes, 12 minutes or even longer. The sampling frequency is not fixed, in other words, sampling data is missing to some extent. In order to repair this inconsistent data, newton interpolation is applied according to fixed time or fixed distance. Moreover, we drop out similar records which are not important but occupy storage based on unit cube method. In addition to the methods above, we also use clustering method to detect outliers of trajectories.

2.1 Four-Dimensional (4D) trajectory model

The 4D trajectory data elements we get include the information on aircraft number, time, latitude, longitude, altitude, and speed. A 4D trajectory is a sequence of 4D points[23,24].There are three dimensions for space information and one dimension for time. The 4D point P is:

$$P=(Lat,Lon,Alt,T) \qquad (1)$$

Where Lat is the latitude of the point P, and Lon is the longitude of point P. Alt is the altitude of point P. T is the time at point P. Then, a 4D trajectory Tr is defined by the following formulation:

$$Tr=[P_1,P_2,\cdots,P_n] \qquad (2)$$

A 4D trajectory dataset D is a collection of 4D trajectories, which is defined as follows:

$$D=\{Tr_1,Tr_1,\cdots,Tr_m\} \qquad (3)$$

2.2 BP neural network

In this section, we introduce the BP neural network to repair errors. Firstly, we apply the average speed between two adjacent points P_1 and P_2 to detect the errors with the following formulations:

$$C_{21}=\arccos(\cos(Lat_1)\cdot\cos(Lat_2)\cdot\cos(Lon_1-Lon_2) \\ +\sin(Lat_1)\cdot\sin(Lat_2)) \qquad (4)$$

$$d_{12}=\frac{R\cdot C_{12}\cdot\pi}{180} \qquad (5)$$

$$D_{12}=\sqrt{d_{12}^2+(Alt_1-Alt_2)^2} \qquad (6)$$

$$v_{12}=\frac{D_{12}}{T_2-T_1} \qquad (7)$$

Where $R=6371.0$ km. Lat_1, Lat_2 are the latitudes of P_1 and P_2. Lon_1, Lon_2 are the longitudes of P_1 and P_2. Alt_1, Alt_2 are the altitudes of P_1 and P_2. C_{12} is the radian between P_1 and P_2. d_{12} is the great circle distance between P_1 and P_2. D_{12} is the actual distance between P_1 and P_2. v_{12} is the average speed between points P_1 and P_2.

Because the speed of the current commercial plane is less than the speed of sound, if v_{12} is larger than 1200 km/h, we take for P_2 as an error point. Then, we apply BP neural network to repair P_2 based on the correct points. Fig. 1 is the structure of BP neural network.

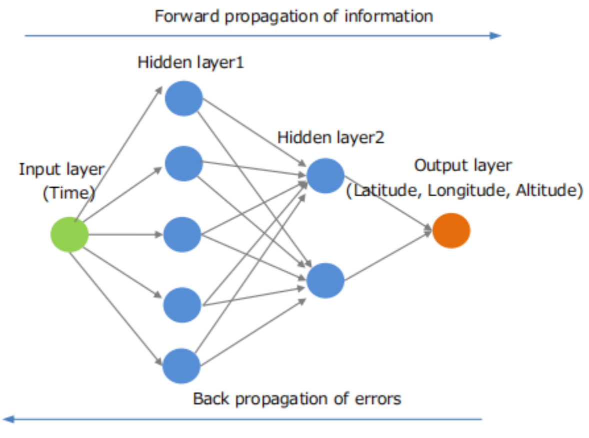

Forward propagation of information
Hidden layer1
Hidden layer2
Input layer
(Time)
Output layer
(Latitude, Longitude, Altitude)
Back propagation of errors

Fig. 1 The neural network structure

The BP neural network includes input layer, hidden layer, and output layer. Given a network, there are N

nodes and *L* layers. The activation function defined as the sigmoid function is:

$$f(x) = \frac{1}{1-e^{-x}} \tag{8}$$

The error of mean square function is used to describe the loss between the true value and the output value:

$$E(y, y\_) = \frac{\sum_{i=1}^{n}(y - y\_)^2}{n} \tag{9}$$

Where *x* is the input value, *y* is the true value, and the *y_* is the output value.

The input value of *i*th neuron at *l*th layer is $net_i^{(l)}$ and the output value of *i*th neuron at *l*th layer is $h_i^{(l)}$, the forward propagation is

$$net_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l)} h_j^{(l-1)} + b_i^{(l)} \tag{10}$$

$$h_i^{(l)} = f(net_i^{(l)}) \tag{11}$$

Where $W_{\mathrm{ij}}^{(l)}$ is layer connection value from the *i*th neuron of *l*th to the *j*th neuron of (*l*+1)th layer, and $b_{\mathrm{i}}^{(l)}$ is the offset of *i*th neuron of *l*th layer, $f(\cdot)$ is the activate function.

To update the weight and offset, the gradient descent functions of the error function are defined as:

$$W_{ji}^{(l)} = W_{ji}^{(l)} + LearnRate \cdot \frac{\partial E(i)}{\partial W_{ji}^{(l)}} \tag{12}$$

$$b_{ji}^{(l)} = b_{ji}^{(l)} + LearnRate \cdot \frac{\partial E(i)}{\partial b_{ji}^{(l)}} \tag{13}$$

The parameter *LearnRate* is updated by a decay function:

$$LearnRate = \frac{LearnRate}{1 + d \cdot step} \tag{14}$$

Where *d* is the decay factor to update the learning rate. Note that, if the learning rate is too large, the model would be over-training; otherwise, if it is too small, the optimization speed would be too slow.

BP neural network with two hidden layers can implement any nonlinear mapping, without limiting the number of hidden nodes. Therefore, we choose four-layer neural network with two hidden layers in our model.

The process of 4D trajectory data training based on the BP neural network is summarized as follows:

1) Take one flight of 4D trajectory data, and detect and discard errors. Set the remained data as training dataset. Set nodes at each layer of the BP neural network.

2) Set the activation function as sigmoid function. Define the value function MSE(*y*,*y_*) representing the square sum of the output error.

3) Set the learning rate. Lay down a feedback regulation.

4) Choose time as input parameter. Set longitude, latitude and altitude as output separately. Feed data to train.

5) Input errors of time detected in step 1) to the trained neural network model. Collect the output which is the repaired data.

6) Select the next BP neural network data, back to step 1). Stop until all the trajectories are repaired.

2.3 Newton Interpolation with sliding window

Newton interpolation is one of the most popular interpolation methods. Data based on Newton interpolation with sliding window can reach high level accuracy, and computational efficiency can be gained as well[25].

Assume x_0, x_1, x_2, x_3 are a set of independent variables that are not equal to each other, and *f*(*x*) is a dependent variable. In this paper, we apply the 4-order Newton interpolation to fix the frequency of points in trajectories:

$$f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_j - x_i} \tag{15}$$

$$f[x_i, x_j, x_k] = \frac{f[x_i, x_k] - f[x_i, x_j]}{x_k - x_i} \tag{16}$$

$$f[x_i, x_j, x_k, x_l] = \frac{f[x_i, x_j, x_l] - f[x_i, x_j, x_k]}{x_l - x_i} \tag{17}$$

$$N_0 = f(x_0) \tag{18}$$

$$N_1(x) = N_0 + f[x_0, x_1] \cdot (x - x_0) \tag{19}$$

$$N_2(x) = N_1(x) + f[x_0, x_1, x_2] \cdot (x - x_0) \cdot (x - x_1) \cdot (x - x_2) \quad (20)$$

$$N_3(x) = N_2(x) + f[x_0, x_1, x_2, x_3] \cdot (x - x_0) \cdot (x - x_1) \cdot (x - x_2) \cdot (x - x_3) \quad (21)$$

Where $f[x_i, x_j]$, $f[x_i, x_j, x_k]$ and $f[x_i, x_j, x_k, x_l]$ is the 2-order difference quotient, the 3-order difference quotient, and the 4-order difference quotient respectively. By following equations (18)-(21) we get the 4-order newton interpolation $N_3(x)$.

2.3 Unit cube method

Some of the 4D trajectory data is redundant. In order to save storage and not destroy the integrity of the data, the process of sampling based on distance is applied. As shown in Fig. 2, a 4D trajectory includes a series of points in 3D spaces throughout the flight. There are some points that are redundant and the characteristics of the trajectory are not changed if they are removed. Therefore, we cut the 3D space with cubes in the same volume. The points are all put into their corresponding cubes. Then, we select one point in each cube to represent points in it.

In Fig. 2, blue and red points are the whole data. The first cube is set at the first point, and then the subsequent cubes are set one by one according to the trend of the data. Finally, all the points are filled into the respective cubes. Select a data point from each cube, such as the red are the points selected. The final data are the remained red points.

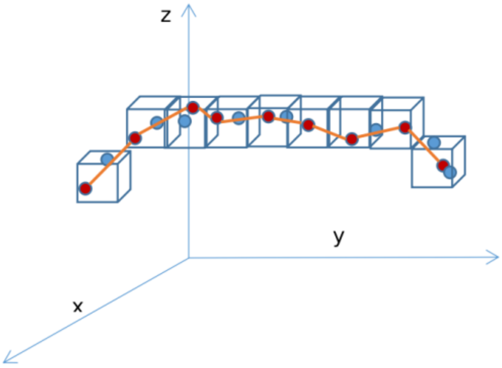


Fig. 2 One demo of unit cube method

In this paper, we set the latitude to 0.5degree, longitude to 0.5degree, and altitude to 500ft as the cubic size. The size of the cube is set according to the requirement of data accuracy. In other words, if more precise data is required, the unit cube is supposed to be set smaller, while the size of the remained data would be larger.

2.4 Trajectory clustering

To filter noises of the trajectories or the outliers of trajectories, we apply the clustering method DBSCAN[26], which views clusters as points in the high density areas. In detail, DBSCAN clusters the points together (in ε Neighborhood with the at most $MinPts$ samples). There are two parameters pes and $MinPts$ for the clusters. The ε -neighborhood contains the points with the distance less than ε for a given point x_i in the dataset D .i.e. $N_\varepsilon(x_j) = \{x_i \in D \mid dis(x, x_j) \le \varepsilon\}$. pes represents the maximum distance between two samples for them which can be considered as in the same neighborhood. If the number in the neighborhood $N_\varepsilon(x_j)$ is more than $MinPts$, then a new cluster is generated. With the iteration of the DBSCAN, the points are added to the clusters until all points in the dataset are labelled. In this section, we just use spatial information of the trajectories. When the trajectory is separated from other combined trajectories or consists of several sporadic points, it will be identified as noises.

The width of a victor way is 8nm (14.8km) according to provisions of the ICAO, and we use longitude and latitude to cluster and the 0.1 degree is about 11km. In our experiments, we set the parameter pes to 0.1. For the parameter $MinPts$, we set it to 20.

3 Results

3.1 Datasets and experimental setup

Despite their high value in aircraft surveillance, positional data streams are not error-free, particularly

ADS-B messages relayed from aircraft. Spurious coordinates indicate impossible positions across a flight. Satellite transmission problems may lead to delayed or missing messages. In addition, satellite transmission problems may lead to delayed or missing messages. There may be also glitches in altitude values[8]. Moreover, when aircraft is above the sea area data is lacking. As we experimentally verified (cf. Section3.6) errors may concern up to 0.9%, so data cleansing is a necessary step before any further processing of aircraft trajectories.

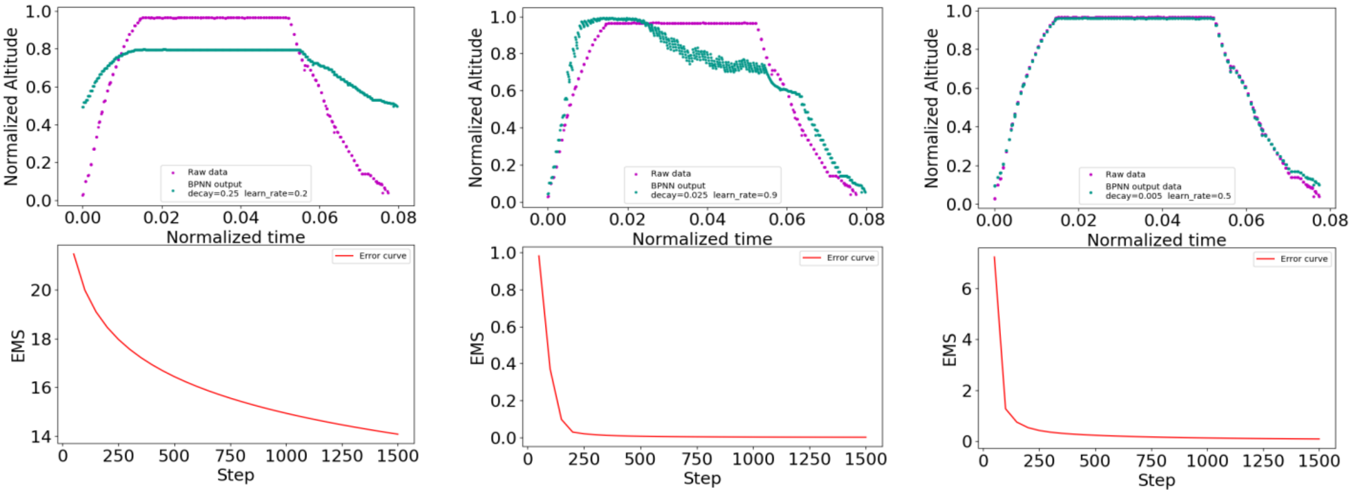
To evaluate the performance of 4D trajectory data cleansing techniques, a 4D trajectory dataset in European area is used as case studies. The dataset includes one-day 4D trajectories on January 1st, 2018. There are 5,905,137 records of 18,286 aircraft in total. We extract 4D trajectories of the aircraft 471F86 for the experimental purpose. The 4D trajectories of the flight DLH3EJ in the whole month of January, 2018 are also included to verify the performance of the clustering algorithm.

We perform the experiments on a laptop equipped with four-core i7-6300U 2.50GHz, and 16 GB DRAM. The methods are all coded with python3.6.7.

3.2 Selection of parameters for BP neural network

In this section, we perform a set of experiments to select proper parameters to build the BP neural network and our training model. The trajectory data of aircraft 471F86 from Wroclaw (WRO) and Dortmund (DTM) are used. There are 575 records, and 16 errors are detected based on calculating the average speed between two adjacent records. We apply BP neural network with the data after cutting down the errors for model training.

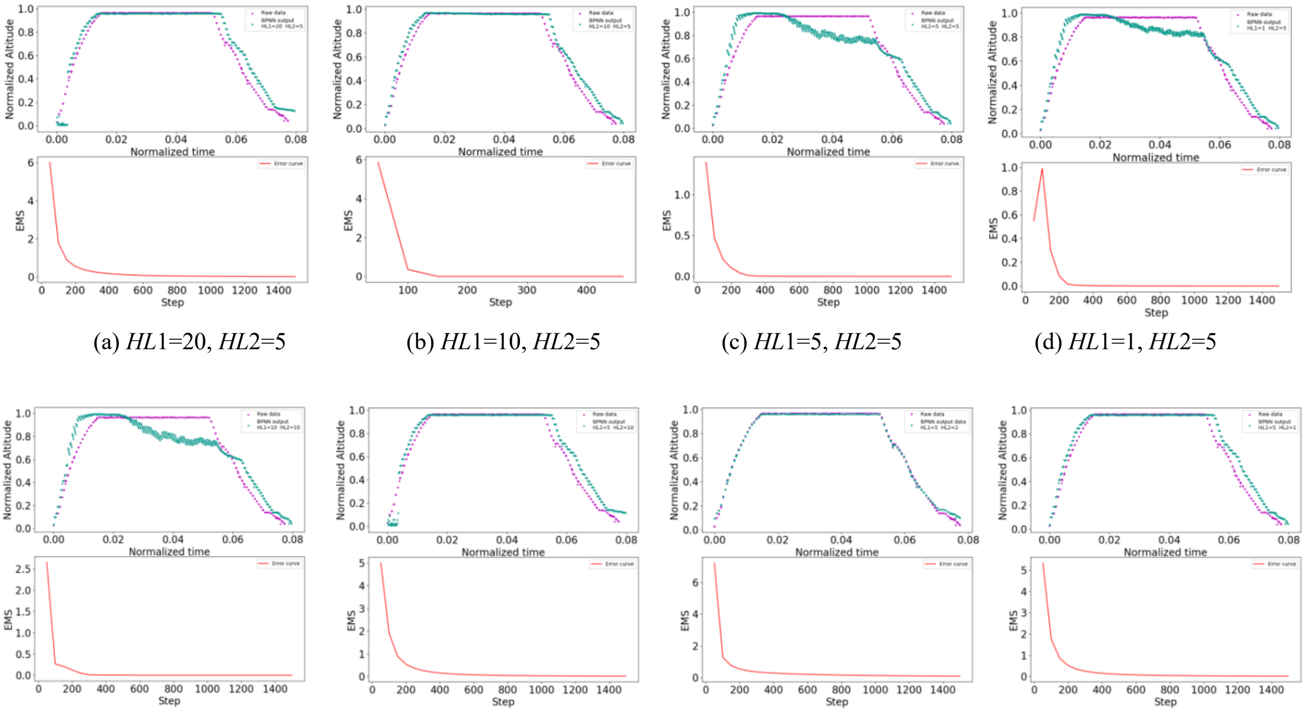
Firstly, we compare the performance of the BP network method with different parameters of learning rate r_l and decay rate r_d . Fig. 3 shows the performance of the BP neural network with different values of learning rate and decay rate. Fig. 3(a) is the result with learning rate set to 0.2 and decay rate set to 0.25, which is under-trained. Fig. 3(b) is the results with learning rate set to 0.9 and decay rate set to 0.025, which is over-trained. Fig. 3(c) is the results with learning rate set to 0.5 and decay rate set to 0.005. Observed from Fig. 3, if the learning rate is set too large and decay is small, the model would be over-trained. Otherwise, learning rate is too small and decay is large, the final model would be under-trained and the training process would be slow. Therefore, we set the learning rate to 0.5 and decay rate to 0.005 in our final training.



(a) $r_l = 0.2$， $r_d = 0.25$　(b) $r_l = 0.9$， $r_d = 0.025$　(c) $r_l = 0.5$， $r_d = 0.005$

Fig. 3 Performance of the BP neural network with different learning rate and decay rate

The number of nodes at each hidden layer influences the training quality and speed. We test eight scenarios with different combinations (*HL*1, *HL*2) of the number of nodes at each hidden layer, which are listed in Table 1. Where, *HL*1 is the number of nodes in the first hidden layer, and *HL*2 is the number of nodes in the second hidden layer. In each experiment, we set the learning rate to 0.5 and decay rate to 0.005. Moreover, the training accuracy is set to 0.0001 and the training process will stop if the results reach the training accuracy or the training steps are up to 3000. The training time is also reported in Table 1. Fig. 4 shows the performance of the eight scenarios. From Table 1, we find that setting *HL*1 to 10 and *HL*2 to 5 costs the shortest time to train the model. However, the performance of the combination (10, 5) shown in Fig. 4(b) is not as good as that of the combination (5, 2) shown in Fig. 4(g). Therefore, in our final experiments, we set 5 nodes at the first hidden layer and 2 nodes at the second hidden layer.



(a) *HL*1=20, *HL*2=5 (b) *HL*1=10, *HL*2=5 (c) *HL*1=5, *HL*2=5 (d) *HL*1=1, *HL*2=5

(e) *HL*1=10, *HL*2=10 (f) *HL*1=5, *HL*2=10 (g) *HL*1=5, *HL*2=2 (h) *HL*1=5, *HL*2=1

Fig. 4 Performance of the number of nodes at hidden layers with different combinations

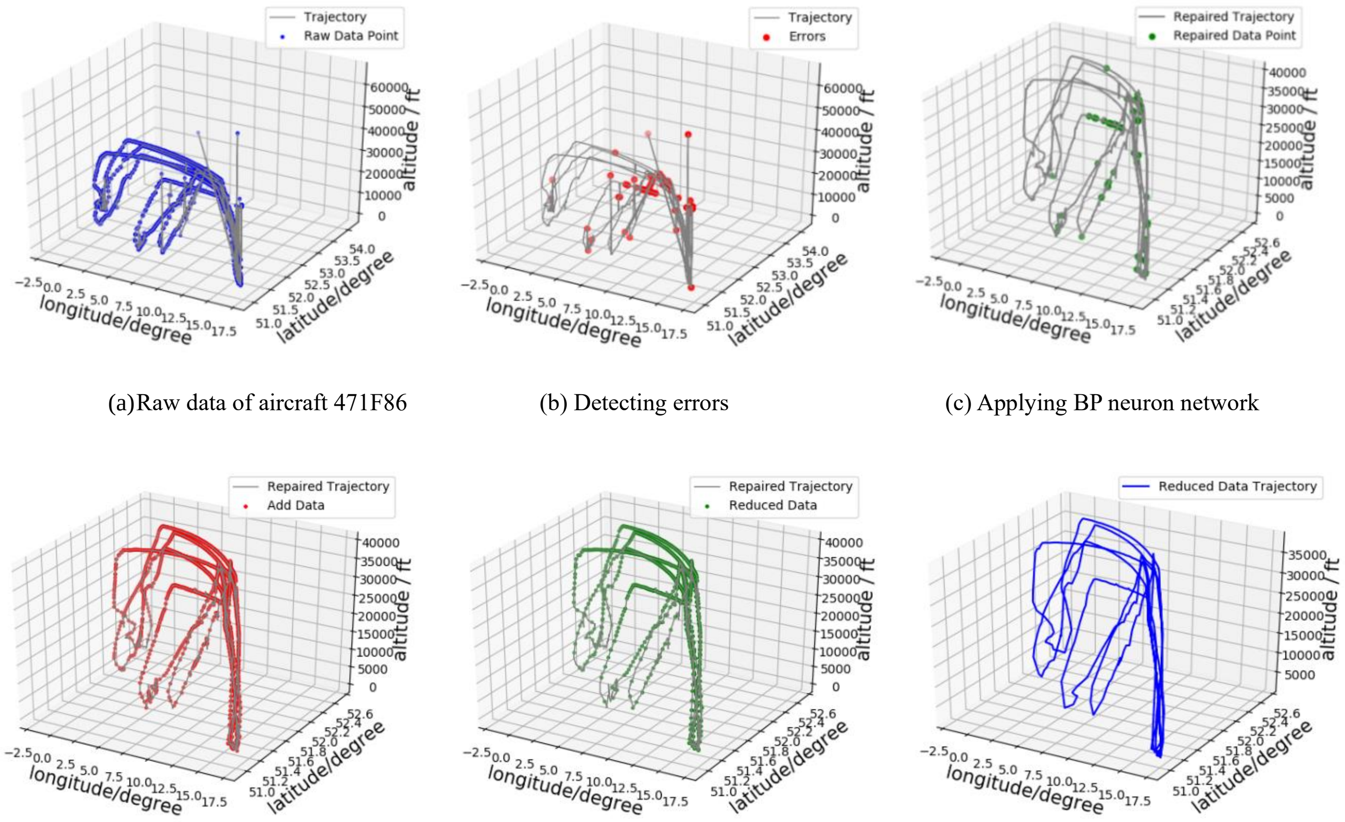
Table 1 Performance of hidden layer1 (L1) and hidden layer2 (L2) with different combinations

*HL*1	*HL*2	Training time (s)
20	5	7.4293
10	5	0.5506
5	5	1.6302
1	5	1.1581
10	10	2.5313
5	10	1.8105
5	2	1.3999
5	1	1.4474

3.3 Results on real 4D trajectories

In this section, we apply our BP neural network model, newton interpolation method and unit cube method on real 4D trajectories. Fig. 5 shows the results of the trajectories of the aircraft 471F86. The 471F86 aircraft finished eight flights between Wroclaw (WRO) and Dortmund (DTM), Wroclaw (WRO) and Eindhoven (EIN), Wroclaw (WRO) and Luton (LTN), Wroclaw and (WRO) Birmingham (BHX) on January 1st, 2018. There are 4,161 records in total.

Fig. 5(a) shows the raw 4D trajectory data. We can see that some obvious wrong points, which drop or rise sharply as well as are far away from the major trajectory. Fig. 5(b) shows the errors. The red points are the errors detected by calculating the average speed between two records. There are 50 red points which means that we detect 50 errors. Fig. 5(c) shows the result after being repaired by BP neural network model. The input layer is time and the output layer is set as longitude, latitude, and altitude separately to repair errors. Fig. 5(d) shows the results of interpolation data. The red points are the added points. There are 4,161 records before applying Newton interpolation method and 5,149 records after applying the method. In Fig. 5(e), the green points are the final reduced points by the unit cube method. Only 900 points are selected, which shows that the unit cube method cuts down the data size vastly but keeps the trajectories. Fig. 5(f) shows the trajectories after reducing records which is the finnal results of 4D trajectories of aircraft 471F86 after applying the set of data cleansing method. From Fig. 5(f), we can see that the unit cube method can eliminate the glitches which cannot be detected by average speed. Comparing Fig. 5(f) with the row trajectory data shown in Fig. 5(a), the off-cours data and glitchs are well processed, which shows the effectiveness on our data cleansing method.



(a)Raw data of aircraft 471F86

(b) Detecting errors

(c) Applying BP neuron network

(d) Applying Newton interpolation

(e) Applying unit cube

(f) Final trajectories

Fig. 5 Data cleansing techniques on 4D trajectories of aircraft 471F86

3.4 Result of comparison

In this section, we illustrate some data cleansing methods and compare Kalman filter with our method. As the existing papers present, most of the data cleansing methods dealing with the trajectory data just drop the error when it is detected. This kind of process is the simplest but can not assure the integrity of the original trajectory data. In addition, Kalman filtering method is used in trajectory data cleansing. This kind of method can smooth glitches but some obvious glitches do still exist, which means it is not effective enough. Although, our method is relatively complex to some extent, by comparing Kalman filtering method with the method we proposed, the results show that our method can remove the errors utmostly which demonstrates effectiveness.

Here we implement Kalman filter to get a comparison with our method. Fig. 6 shows the result after applying Kalman filtering. From Fig. 6 we can see some errors are elimilated but some obvious glitchs still exist. Comparing Fig. 6 with Fig. 5(f), our method is more effective than Kalman filter.

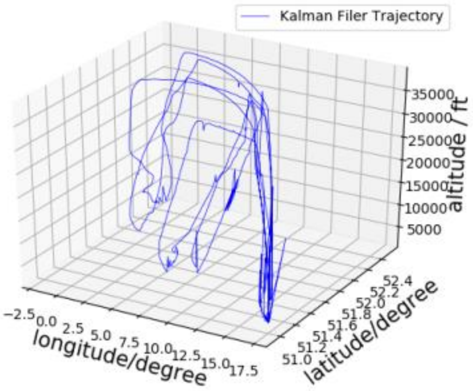


Fig. 6 Kalman filter on 4D trajectories of aircraft 471F86

3.5 Result analysis of clustering

In this section, we report the performance of the DBSCAN algorithm to identify the outliers of trajectories. Fig. 7 shows the result of clustering the trajectories of OD pair London (LHR)-Brussels (BRU) and OD pair London (LHR)-Dusseldorf (DUS) in one day. Two additional flights of Zurich (ZRH)-Brussels (BRU) and Brussels (BRU) -Copenhagen (CPH) are also included in the test data. Observed from Fig. 7, except trajectories from LHR to BRU and LHR to DUS are clustered together, which are shown in blue. The trajectories of ZRH-BRU and BRU-CPH and noise points are clustered into other categories and displayed in different colors.

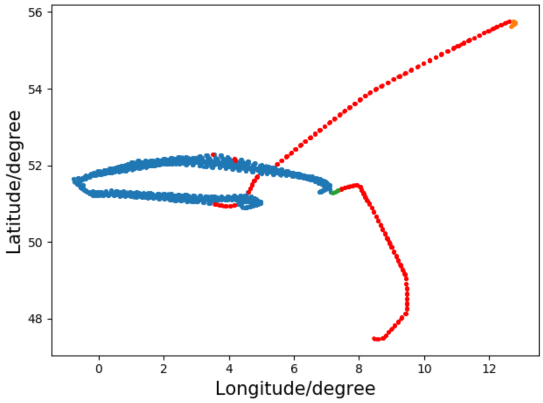


Fig. 7 Clusters of trajectories of LHR- BRU and LHR- DUS

We also apply the DBSCAN algorithm to identify the trajectory noises of DLH3EJ flight, which flies from Oslo Gardermoen (OSL) to Frankfurt Int'l (FRA) in a month. Fig. 8 reports the results of DBSCAN algorithm based on flight DLH3EJ. The red points are identified as noises. Observing from Fig. 8, we can obtain the similar conclusion in Fig. 7. Therefore, the DBSCAN algorithm can be used to detect outliers of trajectories.

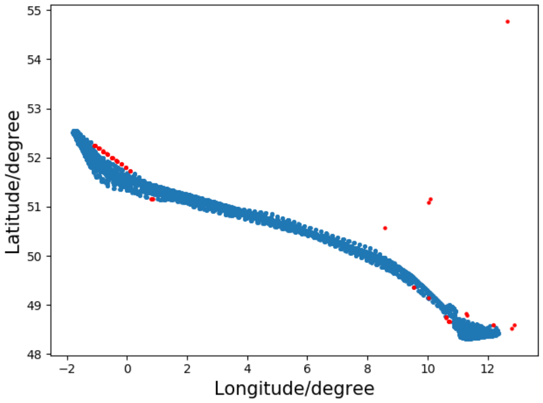


Fig. 8 Clusters of trajectories of flight DLH3EJ

3.6 Data cleansing methods on one-day trajectories

In this section, we apply the data cleansing methods on one-day 4D trajectories in the European area. There are 5,905,137 records of raw data in total. For the data size is relatively large, the three-dimension visualization is not a good option, so the data are shown in two-dimensions with longitude and latitude.

Fig. 9 shows the raw one-day 4D trajectory data. When aircraft fly above the area of sea, most of the data are lacked. If the records of one flight are less than 20, then the flight is dropped out directly.

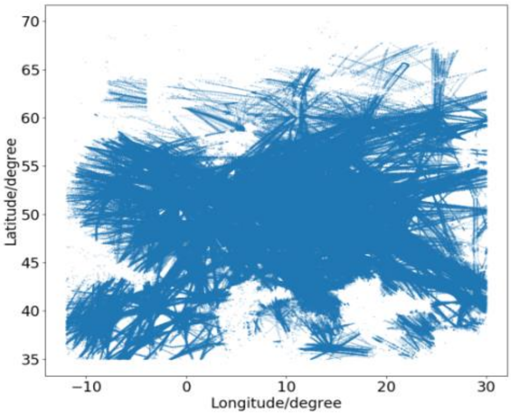
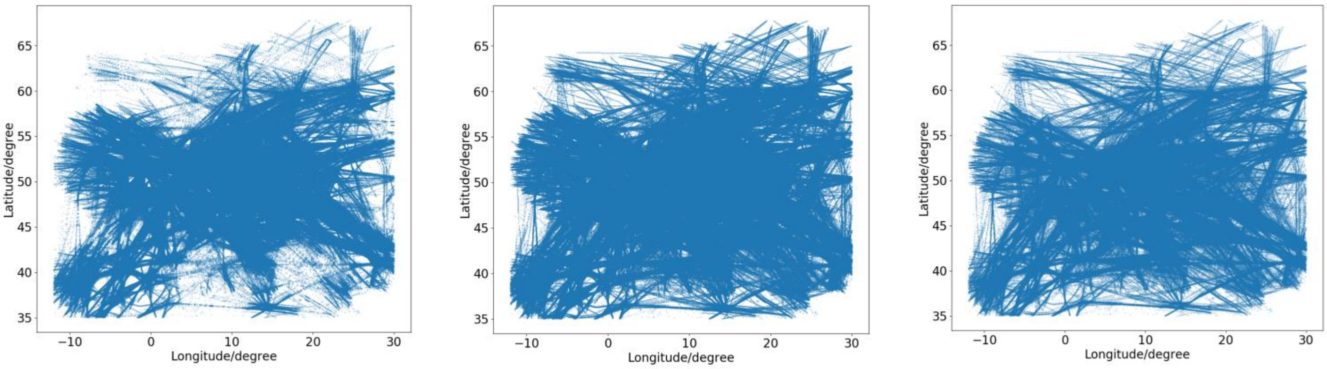


Fig. 9 One-day records in Europe

Fig. 10 shows one-day trajectories after applying the data cleansing methods. Fig. 10(a) shows the data after repairing the errors of each flight based on BP neural network. Fig. 10(b) shows the result of filling data based on Newton interpolation. Fig. 10(c) shows the result of reducing data based on unit cube. From Fig. 10(c) we can notice that the off-course points are well trimmed and the points above the sea area are filled, which shows the effectiveness of the data cleansing method we proposed.

There are 5,905,137 records of raw 4D trajectory data in total. After dropping out the duplicates, 5,012,518 points are left. There are 4,989,510 points after rounding the unreliable data detected by average speed. In our experiment, 6,286 errors are detected and repaired based on the BP neural network method. There are 7,894,063 records existing after applying Newton interpolation and 1,699,110 left after cutting down data based on unit cube.



(a) Applying BP neural network (b) Applying Newton interpolation (c) Applying unit cube

Fig. 10 Data cleansing techniques on one-day trajectories

4 Conclusions

In this paper, we presented a rich set of data cleansing techniques for the 4D trajectory data. The errors were detected by the average speed, and we applied BP neural network to deal with the errors. By using the newton interpolation method, we fixed the frequency of the points in the 4D trajectory data. To reduce computational complexity while maintaining data characteristics, the unit cube sampling method was used to cut down the size of the 4D trajectory data significantly. Comparing with the Kalman filtering method, our data cleansing method obtained a better result. DBSCAN method was applied to identify outliers of trajectories. Experimental results showed the efficiency of our proposed data cleansing techniques.

Data cleansing, especially for the 4D trajectory data is a very new issue. In the future work, we will test our data cleansing methods for the data with a longer period. We will also investigate other methods to address error identification problems.

Acknowledgements

This study is supported by the National Natural Science Foundation of China (Grants No. 61861136005, No. 61851110763, and No. 71731001).

References:

[1] Ribeiro V F, de Almeida Rodrigues H T, de Faria V B, et al. Conflict detection and resolution with local search algorithms for 4D-navigation in ATM[C]// International Conference on Intelligent Systems Design and Applications. Springer, Cham, 2018: 129-139.

[2] Jackson M R C, Gonda J, Mead R, et al. The 4D trajectory data link (4DTRAD) service-Closing the loop for air traffic control[C]// 2009 Integrated Communications, Navigation and Surveillance Conference. IEEE, 2009: 1-10.

[3] Rosenow J, Fricke H, Schultz M. Air traffic simulation with 4d multi-criteria optimized trajectories[C]// 2017 Winter Simulation Conference (WSC). IEEE, 2017: 2589-2600.

[4] Martínez-Prieto M A, Bregon A, García-Miranda I, et al. Integrating flight-related information into a (big) data lake[C]//2017

IEEE/AIAA 36th Digital Avionics Systems Conference (DASC). IEEE, 2017: 1-10.

[5] Andrisani D, Ayoubi M, Hoshizaki T. Aircraft ADS-B Data Integrity Check[C]// AIAA 4th Aviation, Technology, and Operations Conf. 2004.

[6] Desell T, Clachar S, Higgins J, et al. Evolving neural network weights for time-series prediction of general aviation flight data[C]// International Conference on Parallel Problem Solving from Nature. Springer, Cham, 2014: 771-781.

[7] Ali B S, Schuster W, Ochieng W, et al. Analysis of anomalies in ADS-B and its GPS data[J]. GPS Solutions, 2016, 20(3): 429-438.

[8] Patroumpas K, Pelekis N, Theodoridis Y. On-the-fly mobility event detection over aircraft trajectories[C]// Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2018: 259-268.

[9] Bohannon P, Fan W, Geerts F, et al. Conditional functional dependencies for data cleaning[C]// 2007 IEEE 23rd international conference on data engineering. IEEE, 2007: 746-755.

[10] Koudas N, Saha A, Srivastava D, et al. Metric functional dependencies[C]// 2009 IEEE 25th International Conference on Data Engineering. IEEE, 2009: 1275-1278.

[11] Grzymala-Busse J W, Goodwin L K, Grzymala-Busse W J, et al. Handling missing attribute values in preterm birth data sets[C]// International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing. Springer, Berlin, Heidelberg, 2005: 342-351.

[12] Batista GEAPA, Monard MC. An analysis of four missing data treatment methods for supervised learning[J]. Applied Artificial Intelligence, 2003, 17(5-6): 519-533.

[13] Shan Y, Deng G. Kernel PCA regression for missing data estimation in DNA microarray analysis[C]. In IEEE International Symposium on Circuits and Systems, 2009: 1477-1480.

[14] Garcıa I, Martınez-Prieto M A, Bregón A, et al. Towards a scalable architecture for flight data management[C]// 6th International Conference on Data Science, Technology and Applications. 2017.

[15] Olive X, Basora L. A python toolbox for processing air traffic data: a use case with trajectory clustering[C]// Proceedings of the 7th OpenSky Workshop, 2019, 67: 73-84.

[16] Ni X, Wang H, Che C. Risk index prediction of civil aviation based on deep neural network[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2019, 36(02): 313-319.

[17] Li J, Yao X, Wang X, et al. Multiscale local features learning based on BP neural network for rolling bearing intelligent fault diagnosis[J]. Measurement, 2020, 153: 107419.

[18] Li Y, Li J, Huang J, et al. Fitting analysis and research of measured data of SAW micro-pressure sensor based on BP neural network[J]. Measurement, 2020, 155: 107533.

[19] Shi Y, Li Y, Cai M, et al. A lung sound category recognition method based on wavelet decomposition and BP neural network[J]. International Journal of Biological Sciences, 2019, 15(1): 195.

[20] Huang X, Jin H, Zhang Y. Risk assessment of earthquake network public opinion based on global search BP neural network[J]. PloS One, 2019, 14(3).

[21] Lin S, Wang G, Chen Y, et al. Warehouse environment parameter monitoring system and sensor error correction model based on PSO-BP[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2017 (3): 13.

[22] Zhao H, Zeng X, He Z. Low-complexity nonlinear adaptive filter based on a pipelined linear recurrent neural network[J]. IEEE Transactions on Neural Networks, 2011, 22(9): 1494-1507.

[23] Wandelt S, Sun X. Efficient compression of 4D-trajectory data in air traffic management[J]. IEEE Transactions on Intelligent Transportation Systems, 2014, 16(2): 844-853.

[24] Wandelt S, Sun X, Hartmut F. ADS-BI: compressed indexing of ADS-B data[J]. IEEE Transactions on Intelligent Transportation Systems,

2018, 19(12): 3795-3806.

[25] Wang Q, Guan Y, Wang A, et al. Comparison of GPS satellite orbit three-dimension coordinate interpolation algorithms[J]. Progress in Geophysics, 2014, 29(02): 573-579. (in Chinese)

[26] Wang L, Peng B. Track clustering based on LOFC time window segmentation algorithm[J]. Journal of Nanjing University of Aeronautics & Astronautics, 2018, 50(5): 661-665. (in Chinese)

Xin Tan is a Master student at Beihang University. Her major interests are big data and machine learning.

Xiaoqian Sun is an Associate Professor with the School of Electronic and Information Engineering at Beihang University. She obtained her Ph.D. in Aerospace Engineering from Hamburg University of Technology in Germany in 2012. Her research interests mainly include air transportation networks and multi-modal transportation.

Chunxiao Zhang is a Ph.D. student at Beihang University. Her major interests are multi-modal transportation network design and analysis.

Sebastian Wandelt works as a Professor at Beihang University. He received a Ph.D. degree in computer science from Hamburg University of Technology in Germany in 2011. His research interests are intelligent transportation systems and scalable data management.