TRAJECTORY PATTERN IDENTIFICATION AND CLASSIFICATION FOR ARRIVALS IN VECTORED AIRSPACE

by

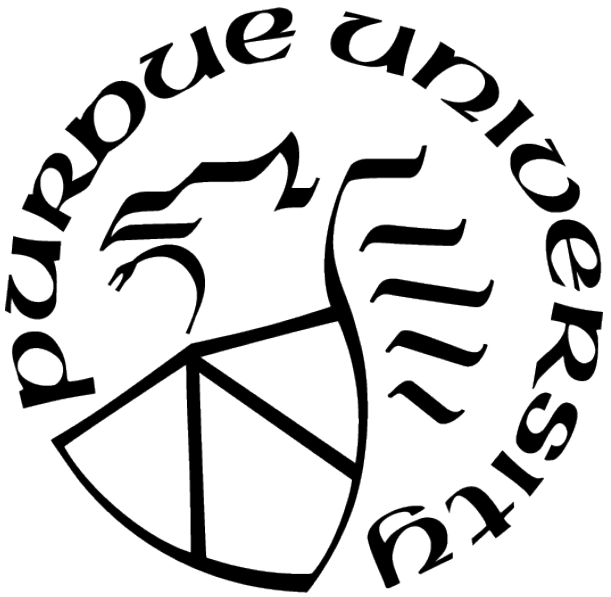
Chuhao Deng

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science



School of Aeronautics and Astronautics

West Lafayette, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Inseok Hwang, Chair

School of Aeronautics and Astronautics

Dr. Dengfeng Sun

School of Aeronautics and Astronautics

Dr. Shaoshuai Mou

School of Aeronautics and Astronautics

Approved by:

Dr. Gregory Blaisdell

ACKNOWLEDGMENTS

This thesis could never be finished without the help of the people around me. I want to thank Professor Inseok Hwang, my advisor, for giving me the opportunity to participate in the big-data project and providing me with valuable and indispensable guidances and advice to this thesis and the development of ideas of trajectory pattern identification and classification, and Doctor Kwangyeon Kim, my mentor, for teaching me what does it take to be a good researcher and always being there when I needed helps. I also want to thank my family for the unconditional love and supports, and my dog, Mozzy, who brings me tremendous amounts of happiness and unforgettable memories everyday.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

As the demand and complexity of air traffic increase, it becomes crucial to maintain the safety and efficiency of the operations in airspaces, which, however, could lead to an increased workload for Air Traffic Controllers (ATCs) and delays in their decision-making processes. Although terminal airspaces are highly structured with the flight procedures such as standard terminal arrival routes and standard instrument departures, the aircraft are frequently instructed to deviate from such procedures by ATCs to accommodate given traffic situations, e.g., maintaining the separation from neighboring aircraft or taking shortcuts to meet scheduling requirements. Such deviation, called vectoring, could even increase the delays and workload of ATCs. This thesis focuses on developing a framework for trajectory pattern identification and classification that can provide ATCs, in vectored airspace, with real-time information of which possible vectoring pattern a new incoming aircraft could take so that such delays and workload could be reduced. This thesis consists of two parts, trajectory pattern identification and trajectory pattern classification.

In the first part, a framework for trajectory pattern identification is proposed based on agglomerative hierarchical clustering, with dynamic time warping and squared Euclidean distance as the dissimilarity measure between trajectories. Binary trees with fixes that are provided in the aeronautical information publication data are proposed in order to categorize the trajectory patterns. In the second part, multiple recurrent neural network based binary classification models are trained and utilized at the nodes of the binary trees to compute the possible fixes an incoming aircraft could take. The trajectory pattern identification framework and the classification models are illustrated with the automatic dependent surveillance-broadcast data that were recorded between January and December 2019 in Incheon international airport, South Korea .

1. INTRODUCTION

1.1 Background and Motivations

Air traffic management encompasses all systems, such as Air Traffic Controllers (ATCs), that assist aircraft to depart from an airport and land at a destination airport [1]. The primary concern of ATCs, who, by using radar, computers or visual references, monitor and direct the movement of aircraft, is $safety$ and they also must direct aircraft $efficiently$ to minimize delays [2]. As the demand and complexity of air traffic increases, how to develop assistant tools for ATCs to guarantee such $safety$ and $efficiency$ becomes crucial and is being studied extensively.

In general, there are two kinds of methods for developing assistant tools for ATCs: physics-based method and data-driven method. Physics-based methods focus on studying the dynamics of aircraft or the airspace system while data-driven methods focus on studying historical flight data. With the advancement of data collecting and processing technologies, data-driven methods have become more and more popular and have been employed for many applications. For separation assurance, Hawley et al. [3] proposed a reinforcement learning based algorithm for predicting and mitigating potential loss of separation events. For anomaly detection, Janakiraman and Nielsen [4] proposed an algorithm for anomaly detection in aviation data by implementing extreme learning machines. Raj et al. [5] proposed an anomaly detection algorithm using temporal logic learning, which generates temporal logic formulas that are easy to be interpreted by human and can be used in real-time monitoring for safety, and they further applied the algorithm to precursor detection for terminal airspace operations [6]. Matthews et al. [7] used scalable data mining algorithm that searches anomalies on large-scale datasets to discover anomalous aviation safety events. Similarly, Li et al. [8] analyzed the flight data and detect abnormal operations using clustering techniques. Other than anomaly detection, data-driven methods can also be used to predict the state information of aircraft, by using probabilistic trajectory models such as Gaussian Mixture Model (GMM) [9, 10] or machine learning models [11], and identify traffic flow patterns [12].

All studies mentioned above rely on clustering for the purpose of data preprocessing, as flights in different patterns have very distinct behaviors and need to be studied individually.

Some used Density-Based Spatial Clustering of Application with Noise (DBSCAN) [3, 5, 6, 8, 9, 11, 12] or k-means [4, 7, 10] and some used domain knowledge like the runway information. To obtain better clustering results, many studies propose new and novel clustering methods. To improve the existing k-means algorithm, Sinaga and Yang [13] proposed an unsupervised k-means clustering algorithm that can automatically find an optimal number of clusters without giving any initialization and parameter selection. Shi et al. [14] proposed an adaptive clustering algorithm based on k-nearest neighbors and density to achieve the similar goal. For clustering of flight trajectories, Peng et al. [15] proposed a trajectory clustering algorithm based on feature representation and selection. However, selecting features from high-dimensional data is time consuming, so Elankavi et al. [16] proposed the Fast Clustering Algorithm that can select features more efficiently by using the minimum spanning tree to remove irrelevant features from the data before clustering. Enriquez [17] presented spectral clustering that, unlike other clustering algorithms, considers not only spatial information, but also temporal values. Liu et al. [18] proposed an improved trajectory clustering method based on fuzzy DBSCAN that implements soft constraints in the conventional DBSCAN method. By using autoencoders, Olive et al. [19] proposed a novel clustering algorithm that extracts the hidden features of trajectories with autoencoders and then cluster trajectories based on their representations in the low-dimensional latent space.

1.2 Objectives and Contributions

Clustering is a crucial step in data preprocessing that can affect the performance of many data-driven methods. Many clustering algorithms have been proposed and applied to flight trajectory clustering where the trajectories in the entire terminal airspace of an airport or the airspace very close to runways are collectively processed at once. However, to the best of our knowledge, none of them focused on identifying various trajectory patterns within one procedure in Standard Arrival Route (STAR) or Standard Instrument Departure (SID) where aircraft are frequently instructed to deviate from the flight procedures by ATCs to accommodate given traffic situations, e.g., maintaining the separation from neighboring aircraft or taking shortcuts to meet scheduling requirements. Such deviation is called vectoring. In

such vectored airspace, the clustering methods mentioned in Section 1.1 could not be directly applied for trajectory clustering because the trajectory patterns in such airspace are deeply embedded in the data with subtle differences in between. To better develop assistant tools for ATCs to use in vectored airspace, getting well-separated clusters is required. From the Automatic Dependent Surveillance-Broadcast data that we have, we observe multiple trajectory patterns that cannot be separated very well by using the clustering algorithms proposed by others. In order to provide ATCs with trajectory pattern information in vectored airspace, we propose an algorithm for trajectory pattern identification, based on agglomerative hierarchical clustering, and trajectory pattern classification, based on a supervised learning technique.

The contributions of this thesis are a new clustering framework that can be applied to the identification of deeply embedded trajectory patterns, in vectored airspace, from historical air traffic surveillance data and a Recurrent Neural Network (RNN) based binary classification architecture that can classify which trajectory pattern an incoming aircraft is most likely going to take in real-time. With such pattern information predicted, ATCs' workload and delays in their decision making process can be reduced.

1.3 Organization

This thesis is organized as follows: Chapter 2 presents the proposed trajectory pattern identification framework and the demonstration data, followed by the identified trajectory patterns and the fixes-based binary tree that serves two purposes: categorizing the trajectory patterns by fixes and being used by the classification models. In Chapter 3, the architecture of the RNN based binary classification model is presented and explained, and the performance of the classification models is evaluated with the demonstration data. In the end, Chapter 4, is a summary and the future work of this thesis.

2. TRAJECTORY PATTERN IDENTIFICATION

This chapter is organized as follows: in Section 2.1, the proposed clustering framework is presented and explained. In Section 2.2, we describe the demonstration data and the preprocessing process. In Section 2.3, we apply the framework to the data and identify multiple trajectory patterns. In Section 2.4, we categorize the patterns based on fixes, provided in the Aeronautical Information Publication (AIP) data, and use fixes-based binary tree for integration.

2.1 Trajectory Pattern Identification Framework

As mentioned in Chapter 1, trajectory clustering is an essential date preprocessing step for data-driven methods. After failing to achieve clear trajectory patterns using popular methods like DBSCAN and k-means due to the noisy property of our data, we proposed a framework based on agglomerative hierarchical clustering and obtained well-separated clusters of trajectories successfully. This section introduces the proposed framework, followed with explanation of some important components in the framework (Sections 2.1.2 and 2.1.3).

2.1.1 Proposed Framework

The proposed framework (Figure 2.1) first utilizes Dynamic Time Warping (DTW) for dissimilarity measure and building the dissimilarity matrix, explained in Section 2.1.2. Then, a dendrogram is constructed using the Ward's linkage method (Section 2.1.3) and the number of patterns is chosen to form preliminary trajectory patterns. After the preliminary trajectory patterns are generated, we visually inspect each pattern and combine similar patterns together to form the final trajectory patterns. Notice that there are some patterns that only contain less than 1% of the total number of trajectories and those patterns are ignored as noise.

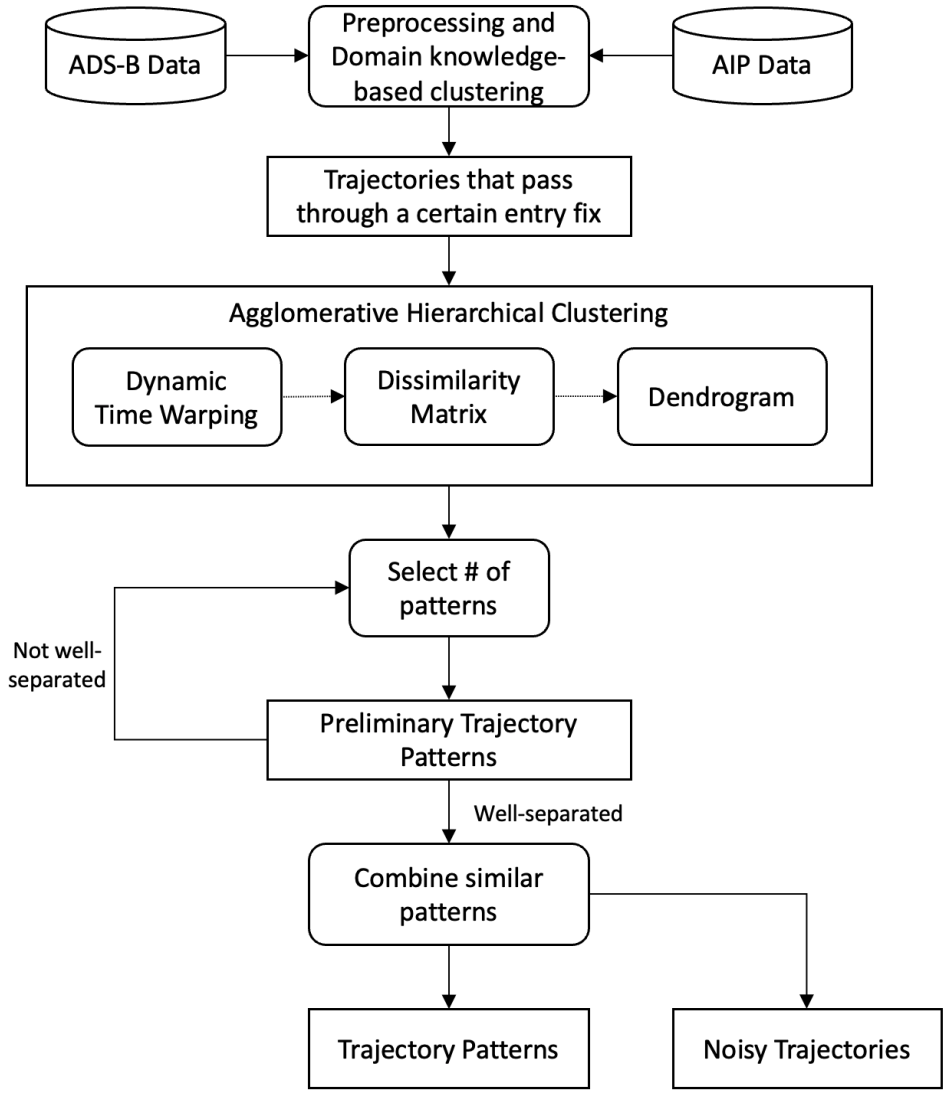
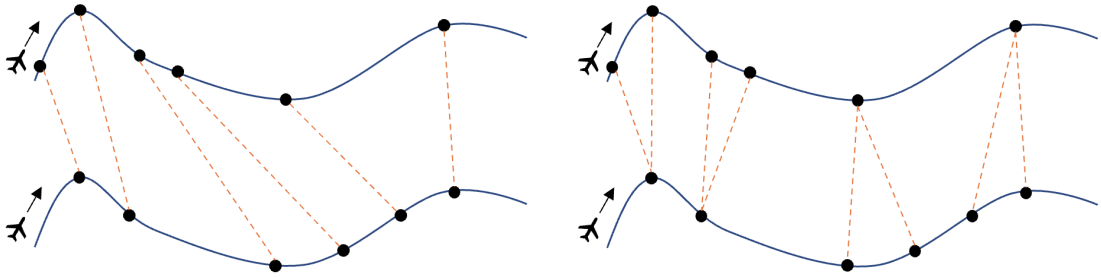


Figure 2.1. Trajectory Pattern Identification Framework

2.1.2 Dissimilarity Measure

Clustering means grouping similar trajectories together, which requires a measure for their similarities. We define the dissimilarity measure as a numerical measure of how different two trajectories are. There are many methods for computing the numerical measure such as the Euclidean distance. However, our data were recorded in real-time at different rates, meaning that there are time discrepancies between the trajectories. Since the distance between each pair of points is computed in order, the time discrepancies could make dissimilarity measure very large despite that the two trajectories are actually very similar (Figure 2.2a). In order to address this issue, we use DTW [20] to find the optimal alignment between two trajectories (Figure 2.2b) before computing the squared Euclidean distance as the dissimilarity measure. The optimal alignment between two trajectories is found by first calculating the distance between the $first$ point in one trajectory and all points in the other trajectory to find the point that delivers the minimum distance, of which is aligned to the $first$ point in the first trajectory. The same computing process repeats to the *second* and all the rest of the points in the first trajectory. When the process is complete, each point is aligned with points that are the closest and the time discrepancies are minimized.



(a) Before Alignment (b) After Alignment

Figure 2.2. Alignment of Two Identical Trajectories Using DTW

With the optimal dissimilarity between trajectories calculated, we can build a dissimilarity matrix. Let the set of all trajectories be $TR = \{tr_1, tr_2, tr_3, ..., tr_n\}$ and the dissimilarity measure between two trajectories calculated using DTW is $DTW(tr_i, tr_j)$, where i can equal to j. Therefore, the dissimilarity matrix is defined as followed.

$$\begin{bmatrix} DTW(tr_1, tr_1) & \dots & DTW(tr_1, tr_n) \\ \vdots & \ddots & \vdots \\ DTW(tr_n, tr_1) & \dots & DTW(tr_n, tr_n) \end{bmatrix}$$

2.1.3 Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering is a bottom-up clustering algorithm that first treats each trajectory as a single cluster, and then uses a linkage method to compute the dissimilarity between clusters. The pair of clusters with the minimal dissimilarity are linked. The linking process is repeated until only one large cluster that contains all the trajectories is remained and a dendrogram is formed, an example presented in Figure 2.3. A unique advantage of using agglomerative hierarchical clustering is that the dissimilarity measure can be computed using a method that works the best with the data, and the precision of the clusters is fully adjustable. For example, in Figure 2.3, if we want four clusters, the dendrogram will be cut in a way such that the three dots on the left will be linked together as a cluster, the two dots in the middle will be separated as two clusters, and the two dots on the right will be linked together as a cluster. Similarly, if we want only two cluster, then all four dots on the left and all three dots on the right will be grouped as two clusters, respectively. This feature allows us to adjust how detailed we want our trajectory patterns to be.

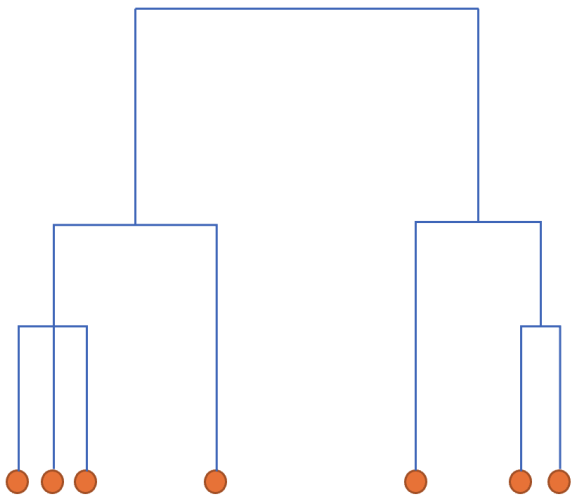


Figure 2.3. An example of dendrogram

In general, there are mainly six linkage methods (complete, Ward's, average, single, median and centroid). Each method has its own criteria for computing the dissimilarity between pairs of clusters and the pairs of clusters with the minimal dissimilarity are linked. For example, for the complete linkage method, the dissimilarity between two clusters is the maximal dissimilarity between the data within those clusters [21]. For the Ward's method, the dissimilarity between two clusters is the increase of within-cluster variance after they are merged [22]. Through practice, we find that Ward's linkage method works the best with our data so we choose it.

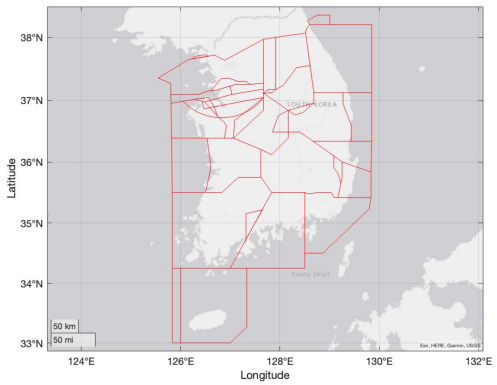
2.2 Data Description and Preparation

This section discusses the data used for demonstration and illustration of our framework and some required preprocessing procedures for the data.

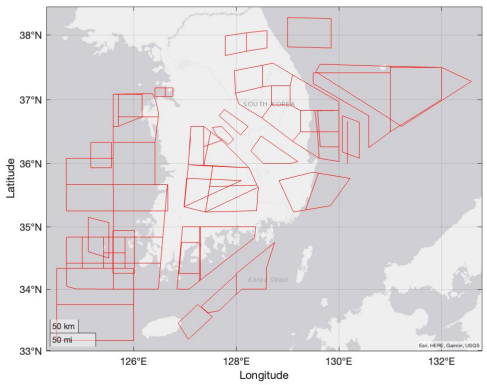
2.2.1 Aeronautical Information Publication Data

Our data is from Korea and it consists of the AIP data and the Automatic Dependent Surveillance-Broadcast (ADS-B) data for Incheon International Airport (ICN) and Gimpo International Airport (GMP). The AIP data contains airspace and routes information, where the airspace includes terminal maneuvering areas (TMA) (Figure 2.4a), military areas (Figure 2.4b), prohibited areas (Figure 2.5a) and sectors (Figure 2.5b).

The routes include Standard Instrument Departure Routes (SID), Standard Arrival Routes (STAR), approach and departure routes for ICN (Figure 2.6) and GMP (Figure 2.7).

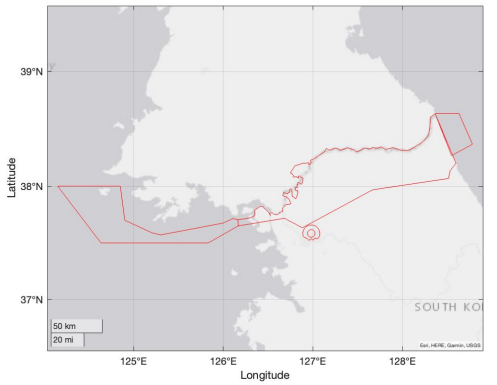


(a) Terminal Maneuvering Areas

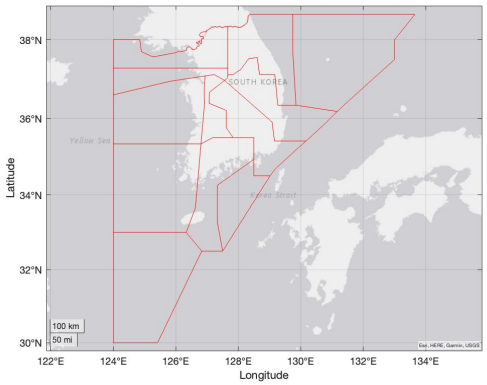


(b) Military Areas

Figure 2.4. Terminal Maneuvering Areas and Military Areas

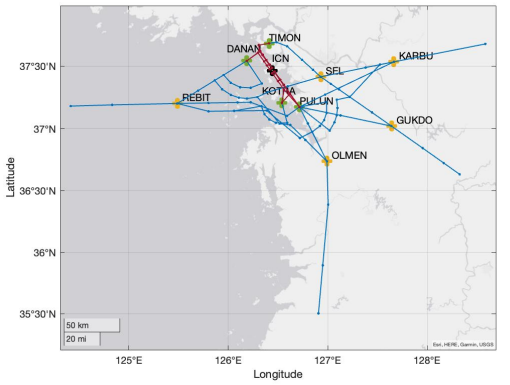


(a) Prohibited Areas

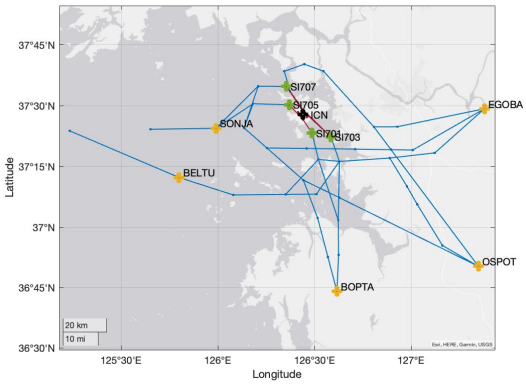


(b) Sectors

Figure 2.5. Prohibited Areas and Sectors

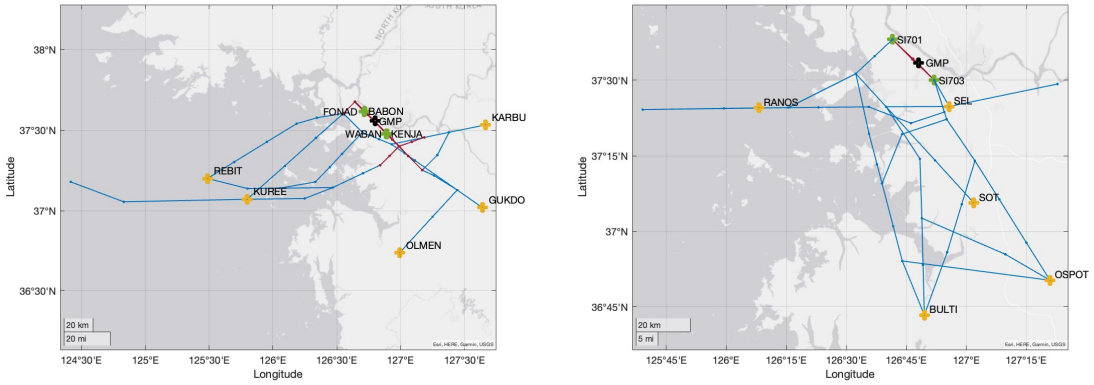


(a) STAR of ICN



(b) SID of ICN

Figure 2.6. STAR and SID of ICN

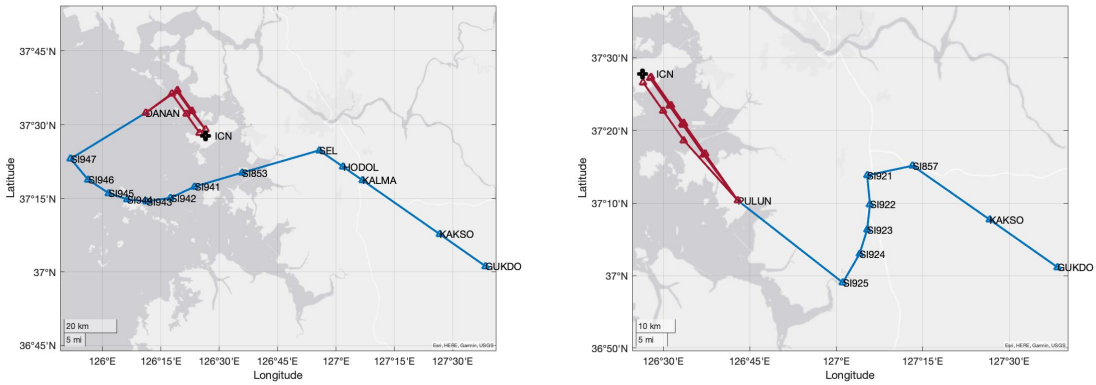


(a) STAR of GMP (b) SID of GMP

Figure 2.7. STAR and SID of GMP

Since the STAR of ICN is the most complex one, we chose to investigate the STAR of ICN. For the STAR of ICN, there are five entry fixes, shown as yellow dots in Figure 2.6a. All flights arriving at ICN must pass through one of those fixes and they must begin their approach phase from one of the fixes shown as green dots in Figure 2.6a. Each of those fixes also correlates with certain runways.

In order to demonstrate our clustering framework, a certain entry fix needs to be chosen. We choose GUKDO as the entry fix due to its complexity. GUKDO is the entry fix for two routes, GUKDO 1N (Figure 2.8a) and GUKDO 1P (Figure 2.8b). Flights using GUKDO 1N will land from the north and flights using GUKDO 1P will land from the south.



(a) GUKDO 1N (b) GUKDO 1P

Figure 2.8. Routes for GUKDO

2.2.2 Automatic Dependent Surveillance-Broadcast Data

Automatic Dependent Surveillance-Broadcast (ADS-B) data provides an aircraft's state information such as its position, speed and angle relative to the runway. For the scope of the data, we choose the data collected from January to February 2019 and there are 204,118 flights in total (Figure 2.9). We separate the flights based on the airport they arrived or departed using the position of ICN and GMP. Then, we used the altitude change to separate arrivals and departures (Figures 2.10 and 2.11). Since popular clustering algorithms like Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and k-means cannot deliver well-separated clusters for arrivals in ICN. We decided to tackle the arrivals in ICN and chose GUKDO as an entry fix to demonstrate our framework. From arrivals in ICN, we used the position of GUKDO to get arrivals that passed GUKDO.

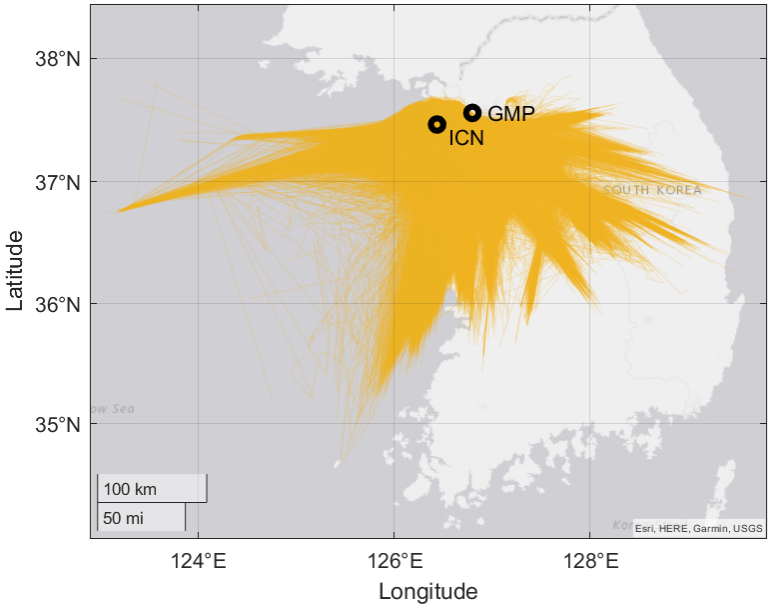
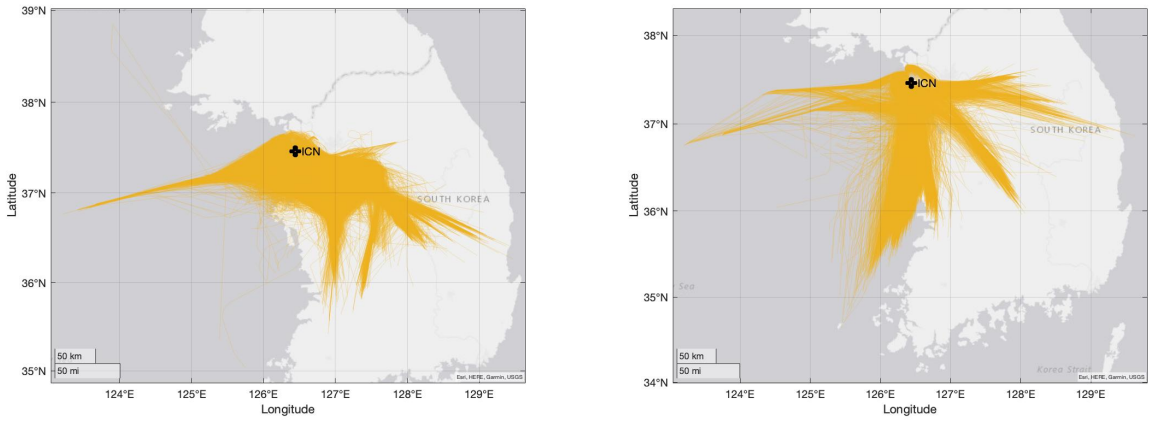


Figure 2.9. All Flights

2.3 Identified Trajectory Patterns

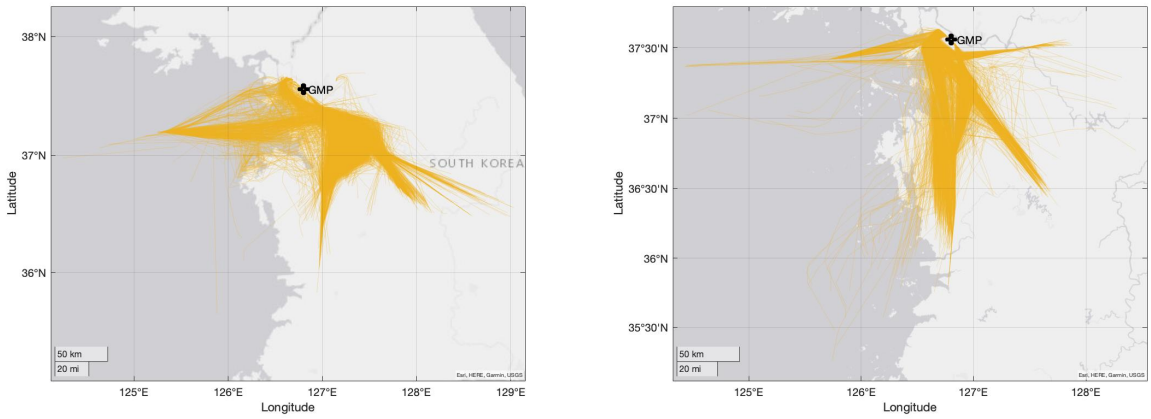
In this sections, we present the identified trajectory patterns by using our proposed framework. The scope of our demontration data is the ADS-B data recorded in ICN and



(a) Arrivals in ICN

(b) Departures in ICN

Figure 2.10. Arrivals and Departures in ICN



(a) Arrivals in GMP

(b) Departures in GMP

Figure 2.11. Arrivals and Departures in GMP

GMP between January and December 2019. There are 204,132 trajectories in total. By using our domain knowledge and AIP data, we separate the data into arrivals and departures for ICN and GMP, and then we further separate those trajectories by the entry fix that they pass through. In our case, we choose arrivals in ICN that pass through GUKDO (Figure 2.12) as our data to demonstrate the framework in Figure 2.1.

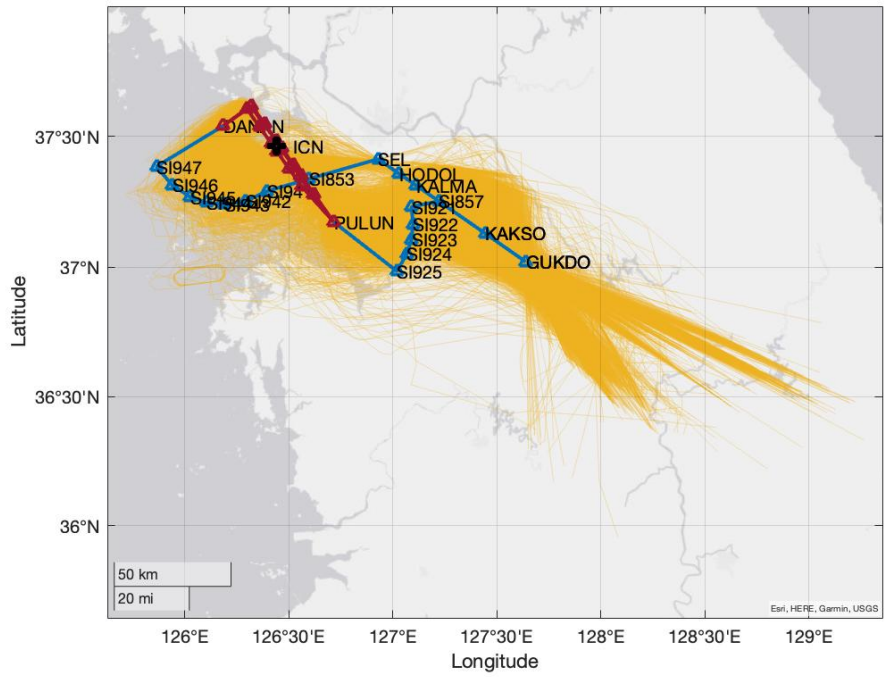


Figure 2.12. Trajectories that Pass Through GUKDO

As shown in Figure 2.8, GUKDO has two routes (GUKDO 1N and GUKDO 1P). In order to find out the trajectory patterns inside each route, we further separate the trajectories by the routes (Figure 2.13 and Figure 2.14) using their landing position.

2.3.1 GUKDO 1N

The total number of trajectories in GUKDO 1N is 2,967. After implementing our proposed framework, we found that there are three trajectory patterns in GUKDO 1N (Figures 2.15, 2.16 and 2.17). From AIP data, we know that the proper order of fixes that the aircraft using GUKDO 1N should follow is GUKDO, KAKSO, KALMA, HODOL, SEL, SI853, and

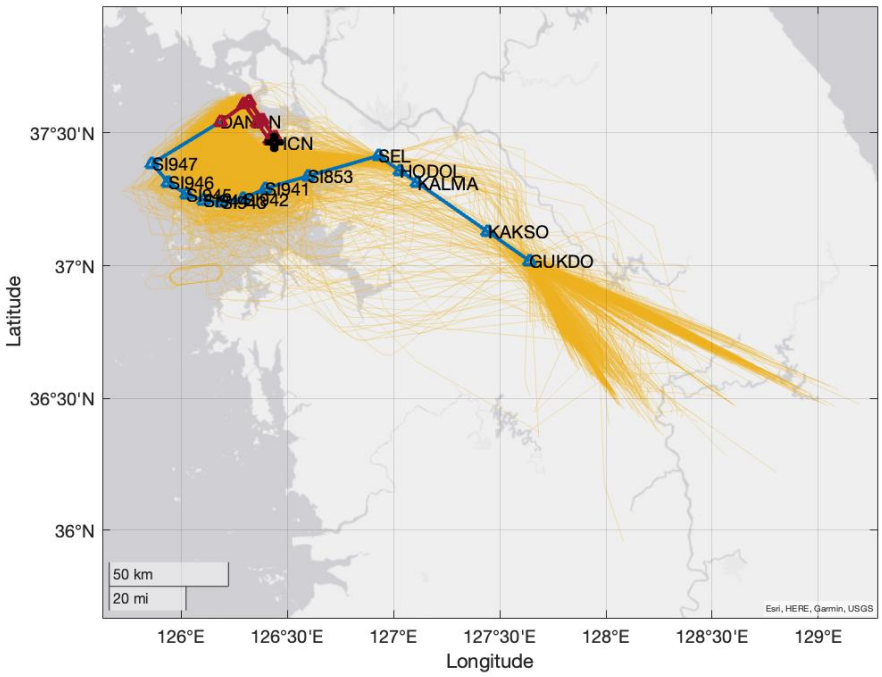


Figure 2.13. Trajectories in GUKDO 1N

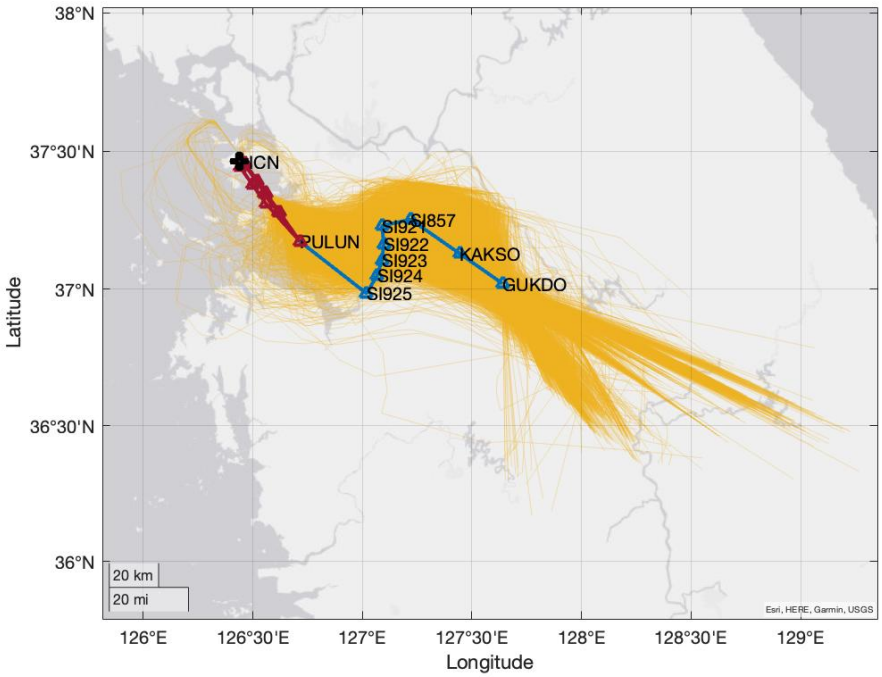


Figure 2.14. Trajectories in GUKDO 1P

then direct-to fixes (SI941 to SI947) where the pilot can choose to go to DANAN directly and entry the approach phase or go to the next direct-to fix. However, from the patterns that we have identified from the ADS-B data, most aircraft did not follow that order. Instead, they did direct-to from SEL or SI853 rather than one of the direct-to fixes. Such deviation from the order is called vectoring.

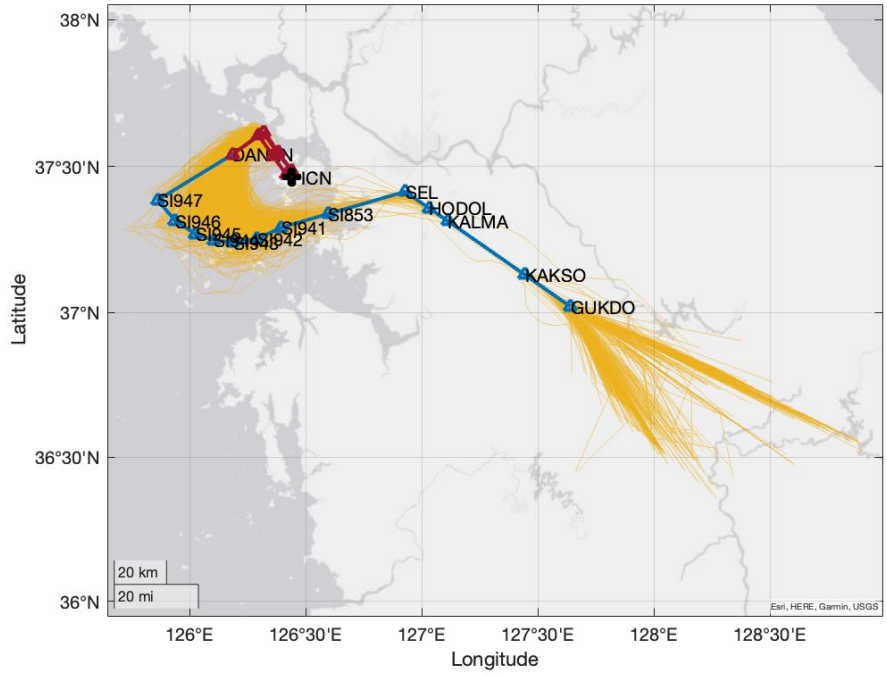


Figure 2.15. Trajectory Pattern 1 in GUKDO 1N (958 Flights)

2.3.2 GUKDO 1P

Compared with GUKDO 1N, GUKDO 1P is a rather popular route with a total number of trajectories in GUKDO 1P of 8,355. By using the framework, we found eight patterns for GUKDO 1P (Figures 2.18, 2.19, 2.20, 2.21, 2.22, 2.23, 2.24 and 2.25). The proper order of fixes for GUKDO 1P is GUKDO, KAKSO, KALMA, HODOL, SEL, SI853, SI941, direct-to fixes (SI942 to SI947) and then DANAN, where the aircraft enter the approach phase.

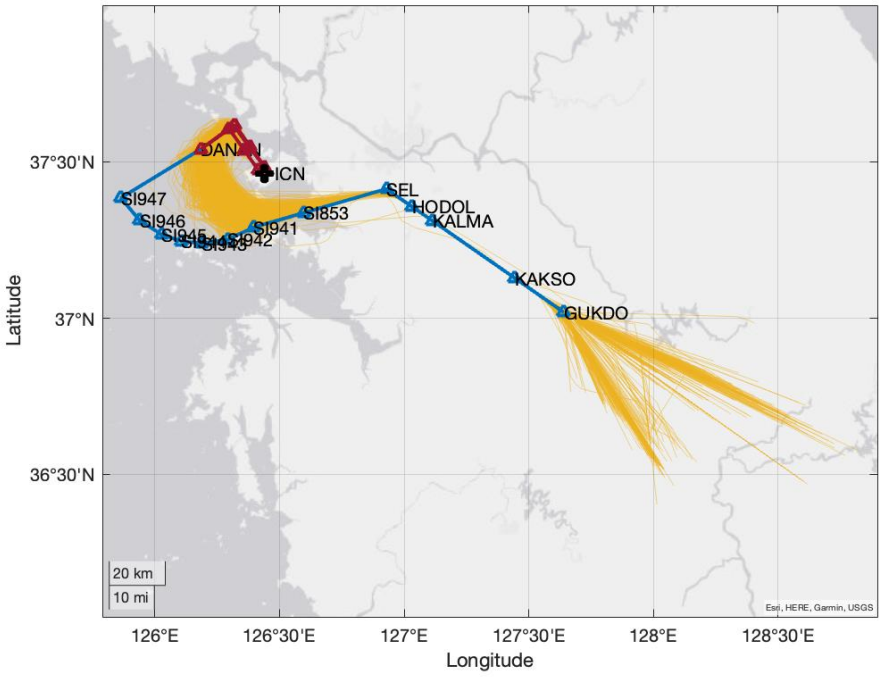


Figure 2.16. Trajectory Pattern 2 in GUKDO 1N (728 Flights)

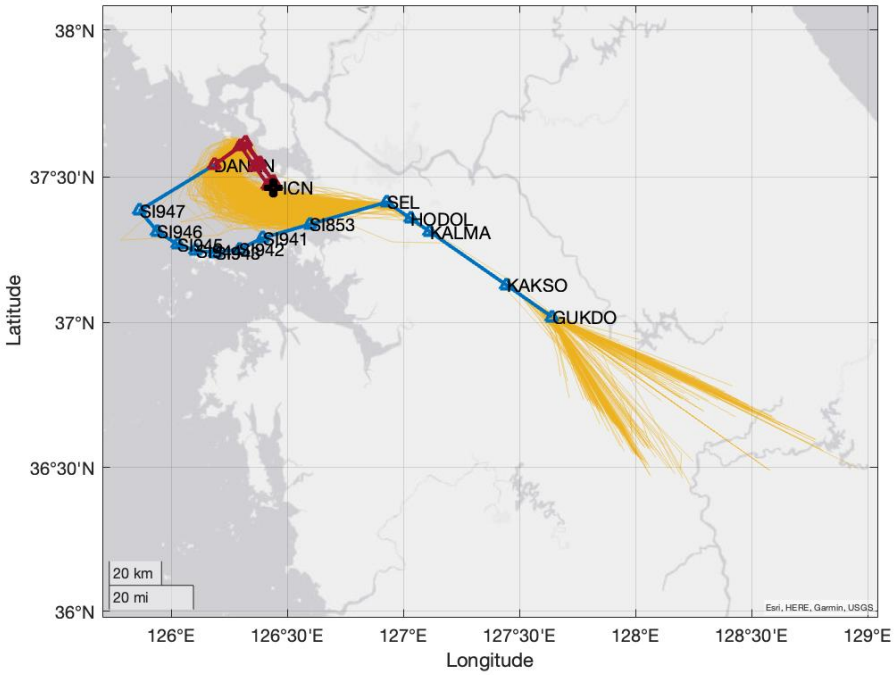


Figure 2.17. Trajectory Pattern 3 in GUKDO 1N (568 Flights)

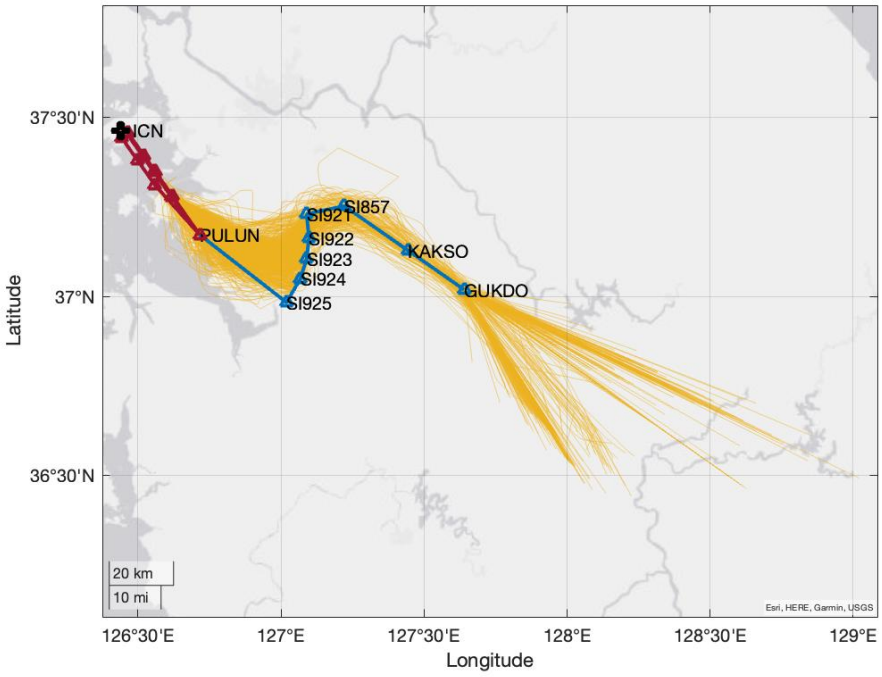


Figure 2.18. Trajectory Pattern 1 in GUKDO 1P (697 Flights)

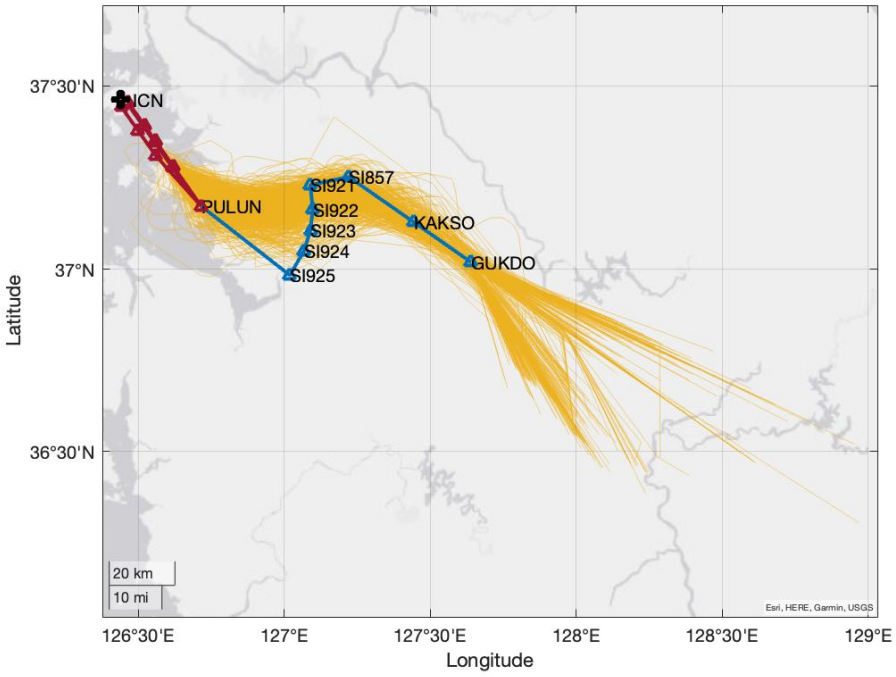


Figure 2.19. Trajectory Pattern 2 in GUKDO 1P (919 Flights)

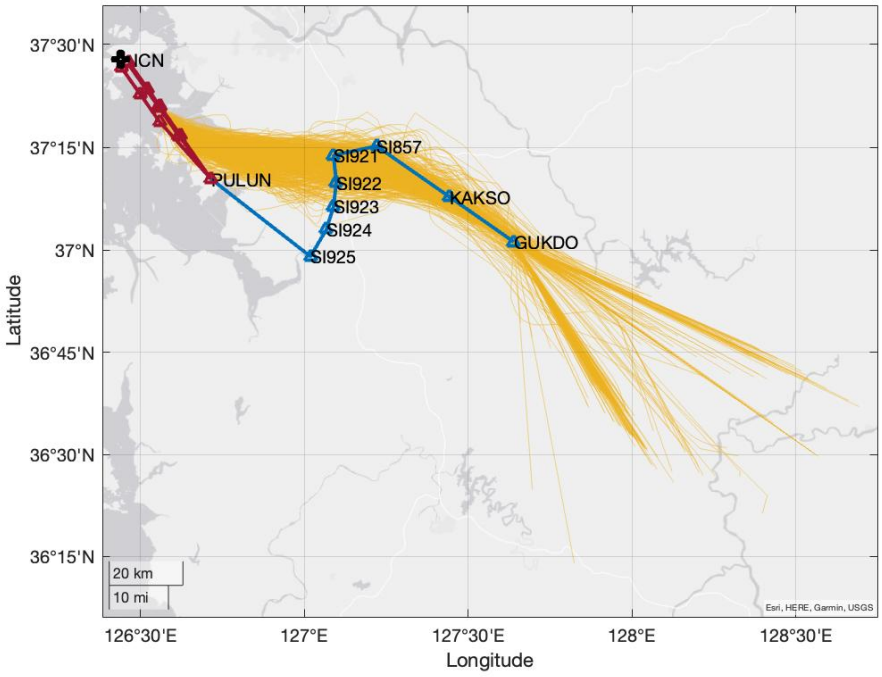


Figure 2.20. Trajectory Pattern 3 in GUKDO 1P (591 Flights)

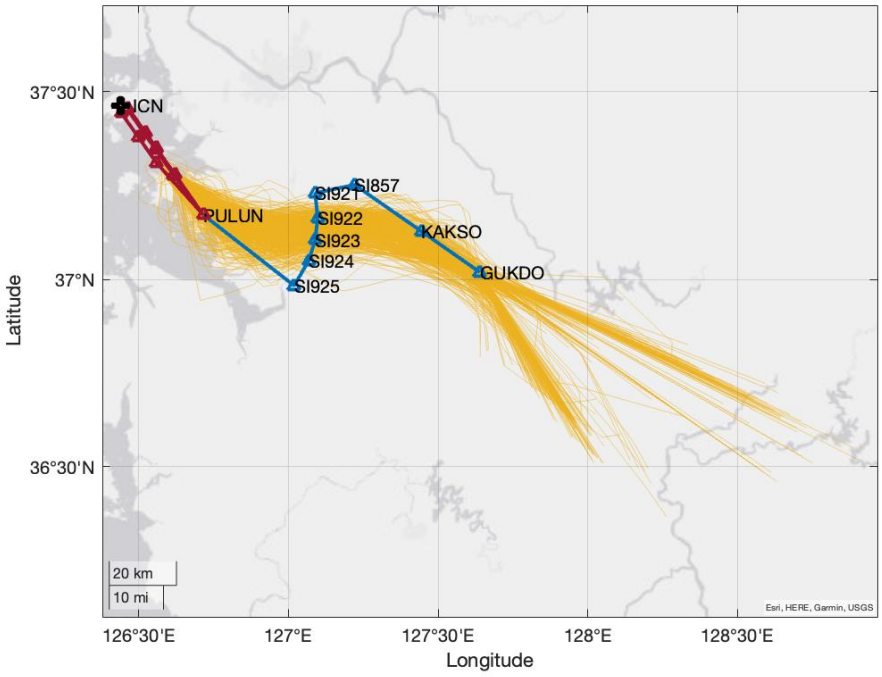


Figure 2.21. Trajectory Pattern 4 in GUKDO 1P (499 Flights)

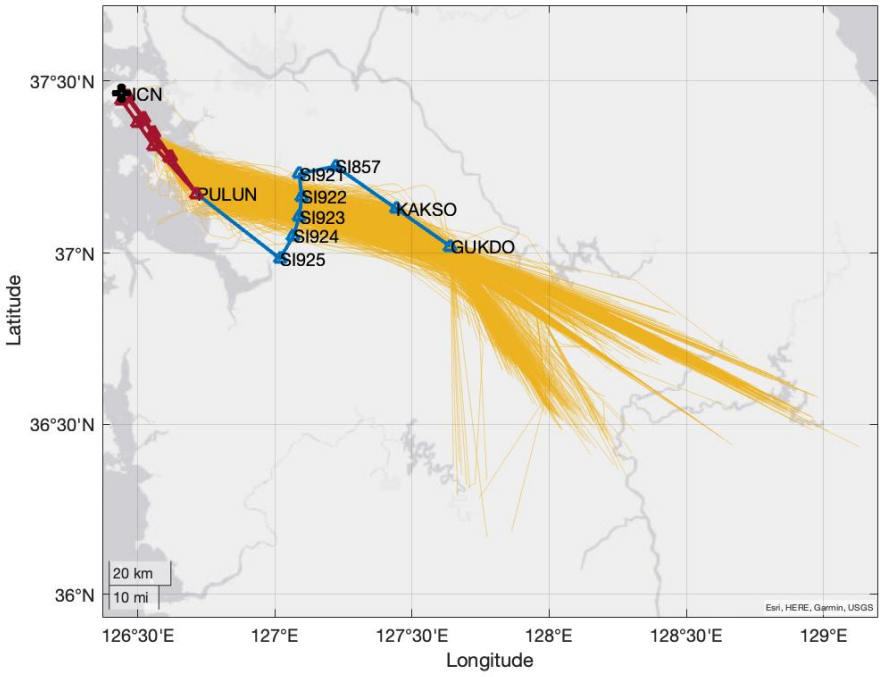


Figure 2.22. Trajectory Pattern 5 in GUKDO 1P (1886 Flights)

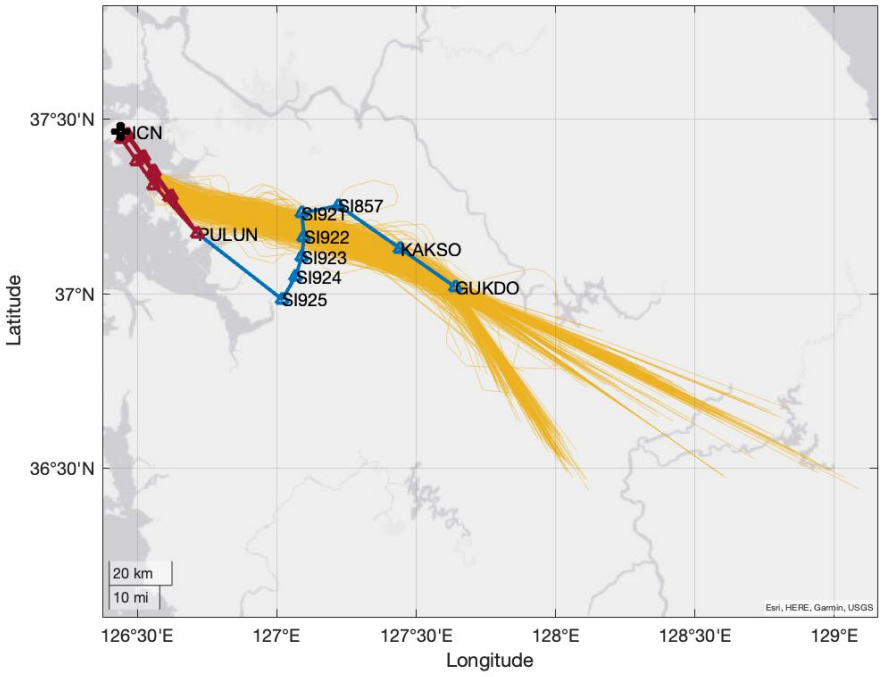


Figure 2.23. Trajectory Pattern 6 in GUKDO 1P (956 Flights)

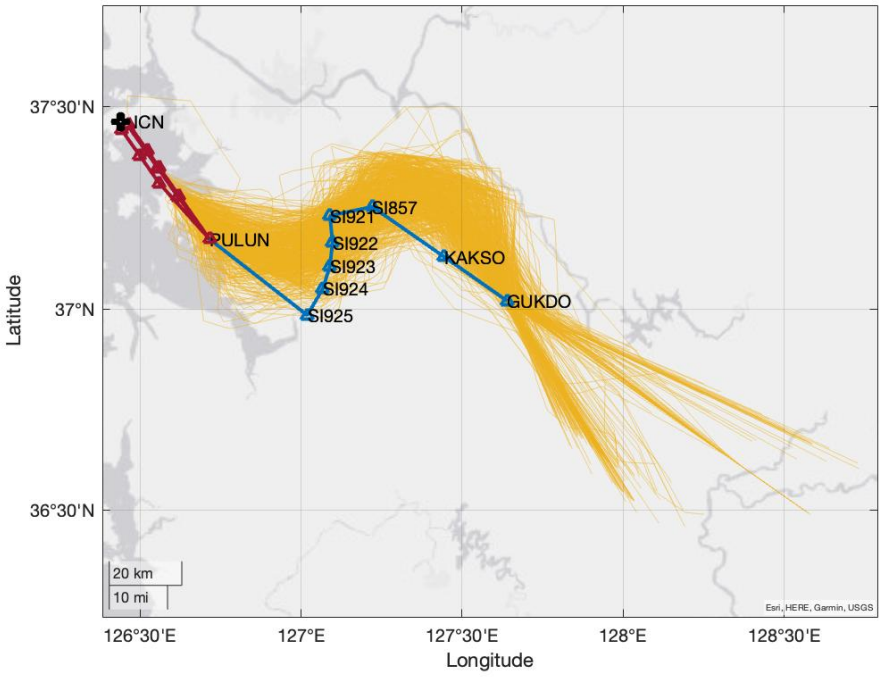


Figure 2.24. Trajectory Pattern 7 in GUKDO 1P (600 Flights)

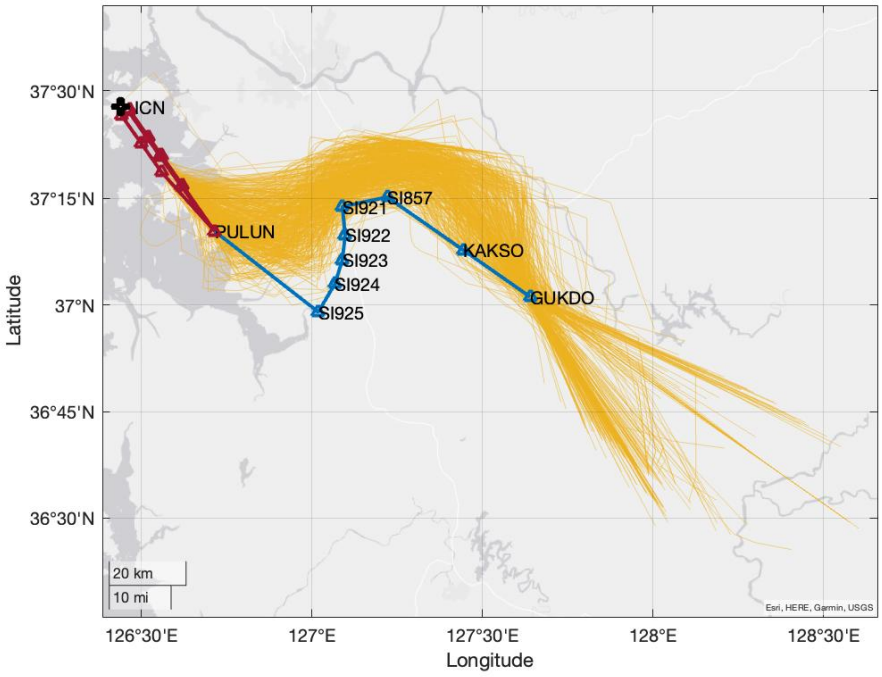


Figure 2.25. Trajectory Pattern 8 in GUKDO 1P (466 Flights)

2.4 Fixes-Based Binary Tree for Trajectory Patterns

As shown in Section 2.2.1, routes are constructed by fixes in a certain order. However, as shown in Section 2.3, aircraft often disobey that order, called vectoring. Therefore, multiple trajectory patterns can be identified from the historical data. Although vectoring is often allowed by ATCs, it inevitably increase ATCs' workload and the likelihood of making mistakes. One thing that we can do to help ATCs is to tell them as early as possible that which pattern an incoming aircraft is most likely going to follow by using a trained classification model. To better train the classification model and construct the trajectory patterns by fixes, we propose a fixes-based binary tree for trajectory patterns.

2.4.1 GUKDO 1N

Compare to GUKDO 1P, GUKDO 1N has less patterns. From the identified patterns, the fixes-based binary tree for GUKDO 1N is shown in Figure 2.26

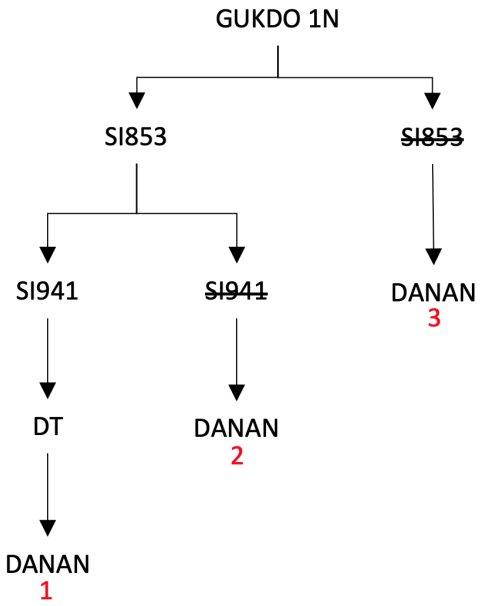


Figure 2.26. Fixes-Based Binary Tree for GUKDO 1N

In Figure 2.26, cach red label represents a pattern number. DT represents one of the direct-to fixes in that route. A fix name with a strikethrough means aircraft in that pattern

bypass that fix. Similarly, a fix name without a strikethrough means aircraft in that pattern pass that fix.

2.4.2 GUKDO 1P

Figure 2.27 shows the fixes-based binary tree for GUKDO 1P. Since patterns in GUKDO 1P is very complex, we need to use terms like *inside*, *outside*, *turn*, *up* or *down*.

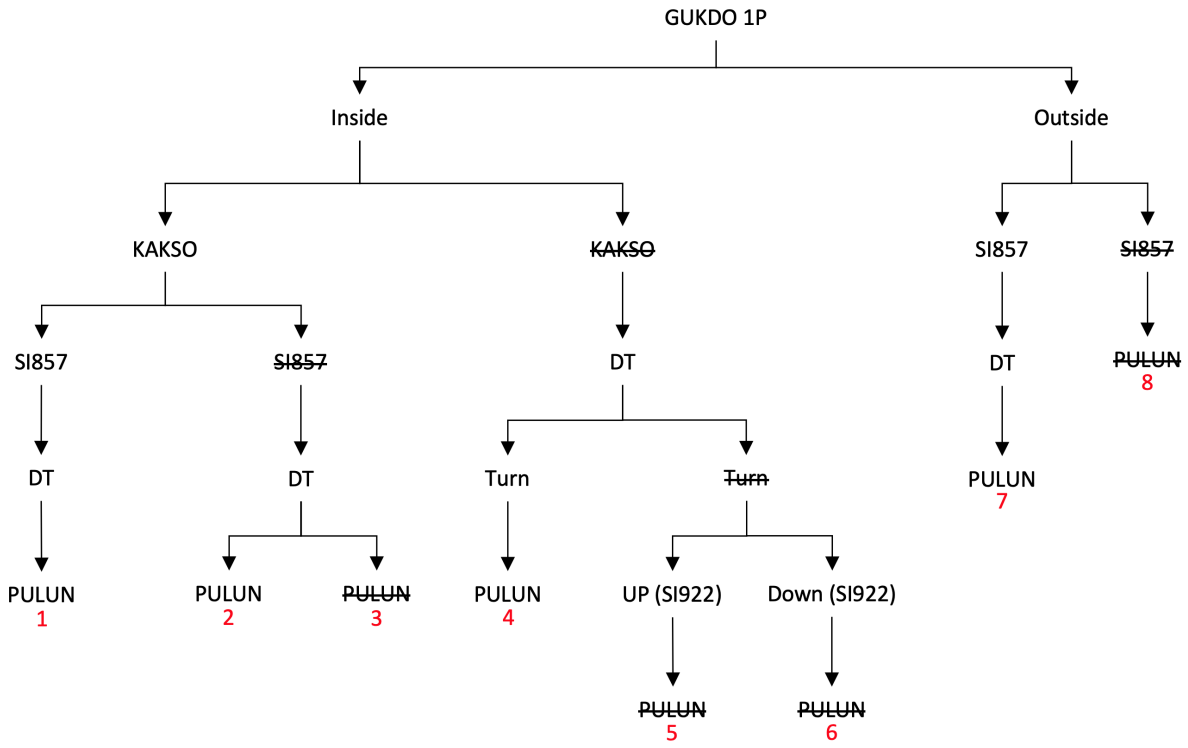


Figure 2.27. Fixes-Based Binary Tree for GUKDO 1P

Inside or *outside* can be more intuitive when comparing pattern 7 (Figure 2.24) and pattern 8 (Figure 2.25) to the remaining patterns. Pattern 7 and pattern 8 are *path extension* patterns and they extend the route *outside* GUKDO 1P before KAKSO. Other patterns, after they pass or bypass KAKSO, remain *inside* GUKDO 1P. *Turn* is a term we use to differentiate pattern 4 (Figure 2.21) from pattern 5 (Figure 2.22) and pattern 6 (Figure 2.23). After pattern 4, 5 and 6 bypass KAKSO and go to direct-to fixes, pattern 4 *turns* a little and then reach direct-to fixes while pattern 5 and 6 go to direct-to fixes in a very straight

way. Between pattern 5 and 6, though, there is *Up* or *Down* terms to further differentiate them. Most trajectories in pattern 5 go to direct-to fixes that are above SI922 and most trajectories in pattern 6 go to direct-to fixes that are below SI922. By dividing pattern 5 and 6 instead of combining them into a single pattern avoids having a wide pattern that not only decreases the classification accuracy, but also helps ATCs less.

3. TRAJECTORY PATTERN CLASSIFICATION

This chapter is organized as follow: in Section 3.1, we describe the preprocessing of the data for training the classification model. In Section 3.2, the architecture of the classification model is explained and presented. In Section 3.3, the result of applying the classification model to our demonstration data is presented and analyzed.

3.1 Data Description and Preparation

As shown in Chapter 2, the trajectory patterns in GUKDO 1N and 1P are identified using our proposed clustering framework. Since ADS-B data was recorded in various length, resampling the data into the same length using linear interpolation before using it as training data for the classification model is necessary. We take the mean of the length of all trajectories for GUKDO 1N and 1P and set the mean as the length for resampling, 137 and 148, respectively. We define the trajectory pattern classification problem as a binary classification problem. We use one binary classification model for each node that has two children in the fixes-based binary tree instead of using one multi-class classification model for all patterns in the binary tree. By doing so, the classification model converges earlier so that ATCs can make decisions as early as possible. As the blue dots and numbers shown in Figure 3.1, for GUKDO 1N, there are two nodes that have two children, so there are two binary classification models to classify flights in GUKDO 1N. Similarly, shown in Figure 3.2, for GUKDO 1P, there are seven nodes that have two children, so there are seven binary classification models to classify flights in GUKDO 1P.

Table 3.1. Number of Trajectories in GUKDO 1N

Model	Total	Training	Testing
1	2,254	2,028	226
2	1,296	1,166	130

The total trajectories for each classification model include all trajectories under the left and right child of that classification model. All trajectories under the left and right child are labeled as 1 and 0, respectively. For example, total trajectories for classification model 1 in

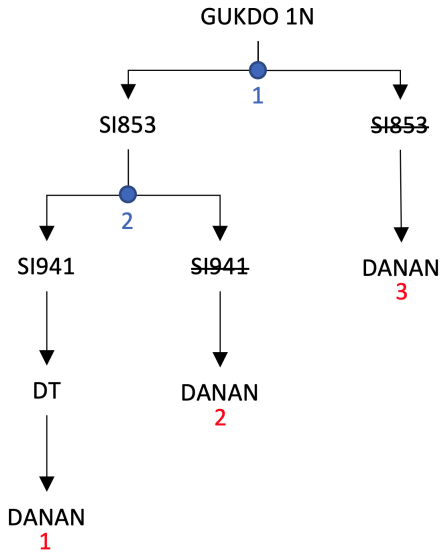


Figure 3.1. Fixes-Based Binary Tree for GUKDO 1N with Classification Models

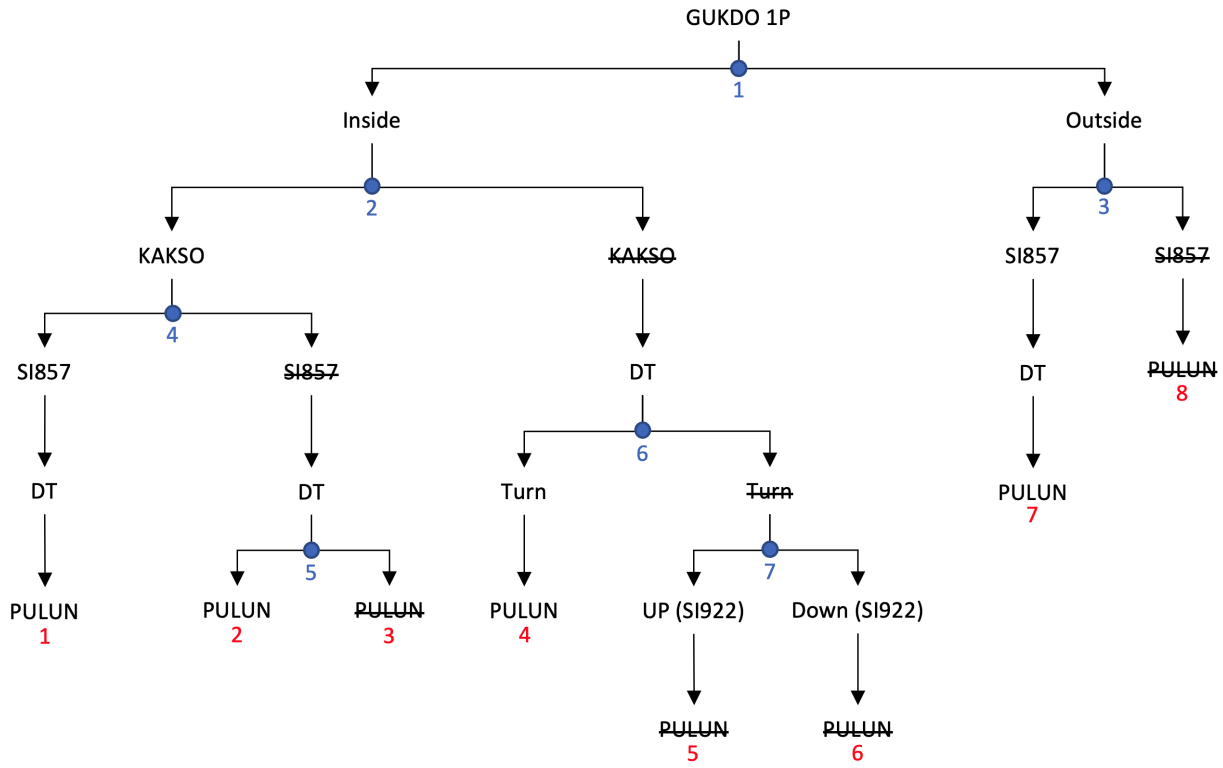


Figure 3.2. Fixes-Based Binary Tree for GUKDO 1P with Classification Models

Table 3.2. Number of Trajectories in GUKDO 1P

Model	Total	Training	Testing
1	6,614	5,952	662
2	5,548	4,993	555
3	1,066	959	107
4	2,207	1,986	221
5	1,510	1,359	151
6	3,341	3,006	335
7	2,842	2,557	285

GUKDO 1P includes trajectories in all eight trajectory patterns. Trajectories in patterns 1, 2, 3, 4, 5 and 6 are labeled as 1 and trajectories in patterns 7 and 8 are labeled as 0. For training each classification model, 90% of the total data for that classification model is used for training and 10% is used for testing. A more detailed information is shown in Tables 3.1 and 3.2. The output of the classification model ranges from 0 to 1.

3.2 Neural Network Architecture

In order to classify the aircraft in real-time, we use a Long Short-Term Memory (LSTM) and fully connected mixed neural network as the classification model. The LSTM is a type of Recurrent Neural Network (RNN) architecture that solves the vanishing gradient problem by using the forget gate [23]. A graph illustration of the structure of a LSTM unit is shown in Figure 3.3. c_{i-1} and h_{i-1} represent the output of the $(i-1)^{th}$ LSTM unit where h_{i-1} is also the output label of the $(i-1)^{th}$ LSTM unit. x_i represents the i^{th} value of the input data. At each time step, there could be more than one LSTM unit, which is called the hidden units. Increasing the number of hidden units increases the LSTM's learning ability but also the training time. In our case, we choose the number of hidden units as 512 to obtain the best results with an acceptable total training time. After the data is processed by the LSTM layer, we gradually reduce the dimension using multiple fully connected layers with Rectified Linear Unit (ReLU) activation function, and an output layer with sigmoid activation function (Figure 3.4). Loss function is chosen to be binary cross entropy because our classification models are binary classification models. The multiple fully connected layers

with ReLU activation function are used instead of one because a deeper neural network deals with non-linearity better [24].

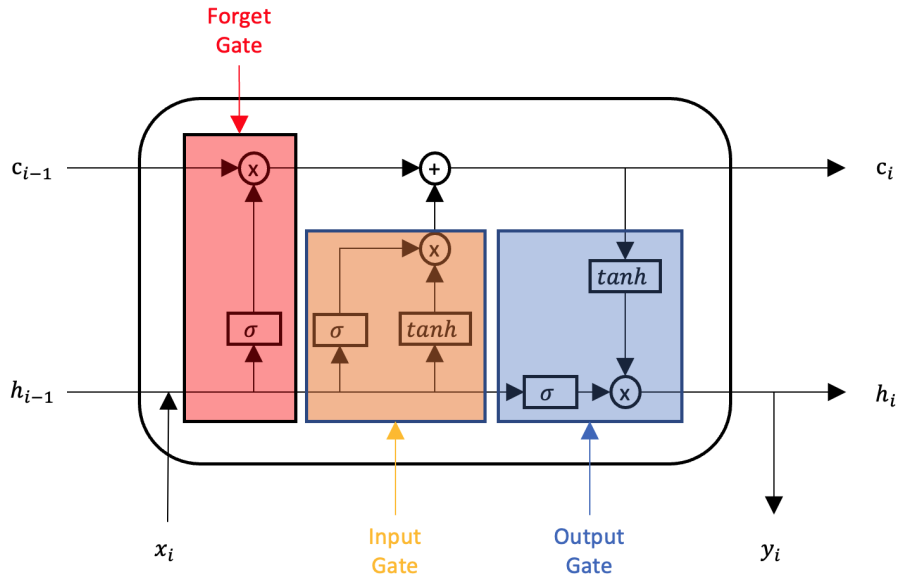


Figure 3.3. Structure of a LSTM Unit

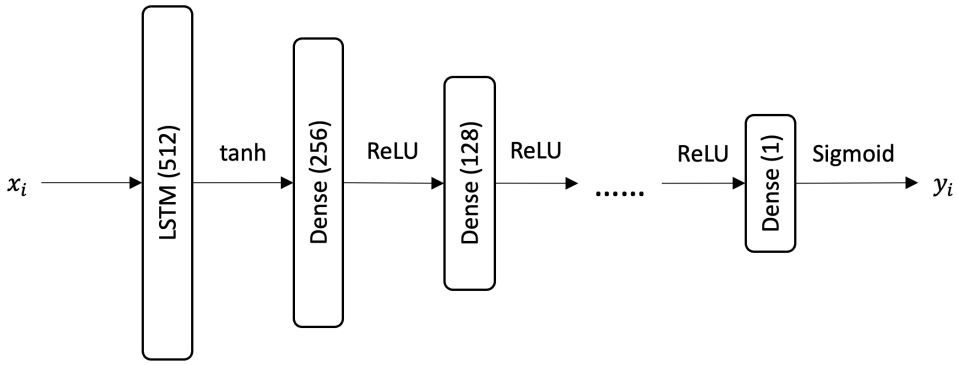


Figure 3.4. Neural Network Architecture for the i^{th} Time Step

3.3 Results and Analysis

This section presents the results from the classification models for GUKDO 1N and GUKDO 1P, and explanation and analysis of those results.

3.3.1 GUKDO 1N

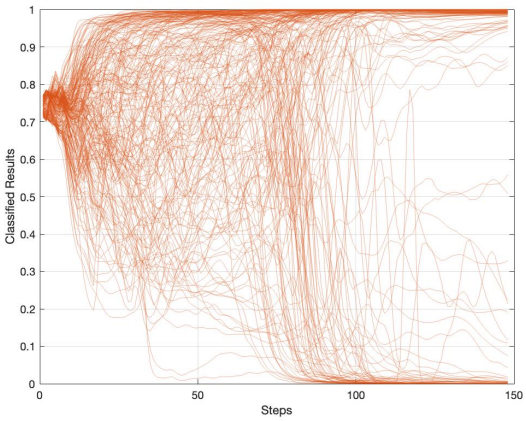
Figures 3.5 and 3.6 show the evolving value from the classification models as the trajectories develop from the first to the final point. It is clear that model 1 converges at an earlier step than model 2 does. It is because model 1 is used for determining if the flight will pass or bypass SI853 while model 2 is used for determining if the flight will pass or bypass SI941, and in Figure 2.13, we can see that SI941 is a fix after SI853. Therefore, model 2 can only converge after model 1 has converged. Figures 3.5c and 3.6c show the testing trajectories cut at 80_{th} and 100_{th} step, where model 1 and model 2 converge, respectively.

3.3.2 GUKDO 1P

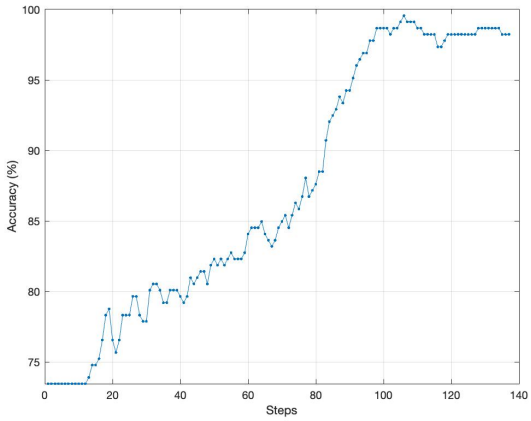
Figures 3.7, 3.8, 3.9, 3.10, 3.11, 3.12 and 3.13 show the evolving value from the classification models as the trajectories develop from the first to the final point.

3.3.3 Analysis

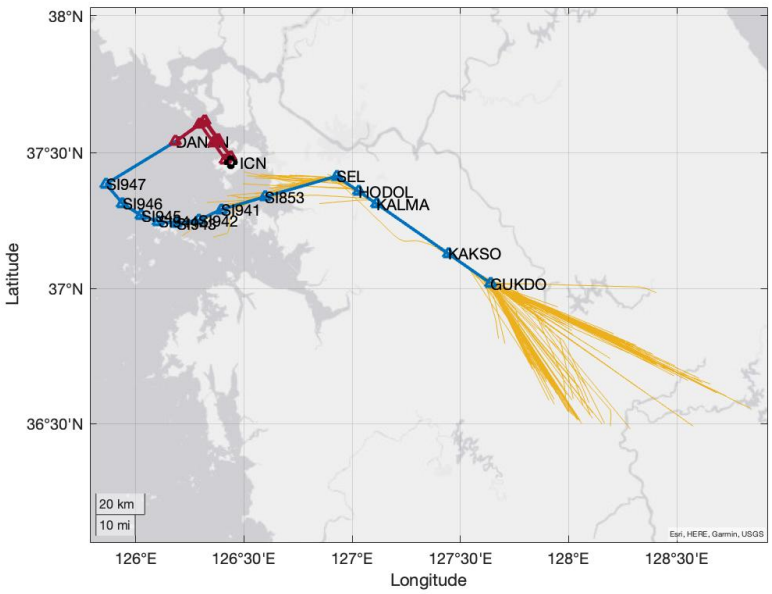
In real-time implementation, the model will be used based on the position of the aircraft. For example, model 1 for GUKDO 1N is used to classify whether the aircraft would pass or bypass SI853. Therefore, when the aircraft is around SI853, the classification model converges and the classification accuracy at around 87%, shown in Figure 3.7b. Since the fixes-based binary trees for GUKDO 1N only has two levels of classifier, if the classification result is below 0.5, ATCs know that the aircraft is most likely going to follow pattern 3. After the aircraft leaves SI853, model 2 is being used and determined whether the aircraft that have already passed SI853 would pass or bypass SI941 at an accuracy around 86%. Similarly, if the classification result is below 0.5, ATCs know that the aircraft is most likely going to follow pattern 2. A complete accuracy result can be found in Tables 3.3 and 3.4.



(a) Result of Model 1 for GUKDO 1N

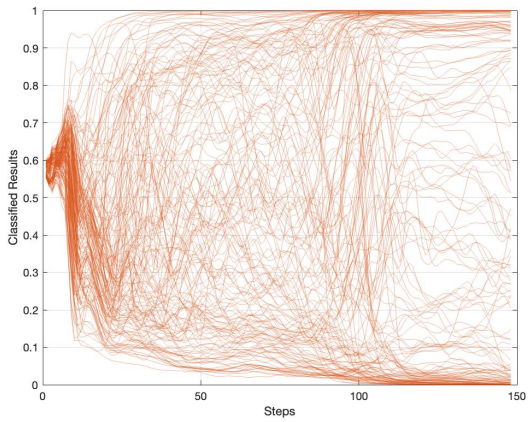


(b) Accuracy of Model 1 for GUKDO 1N

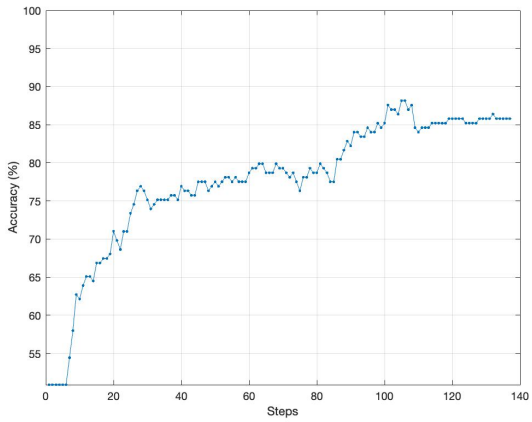


(c) Testing Trajectories of Model 1 for GUKDO 1N (Cut at 80^{th} Step)

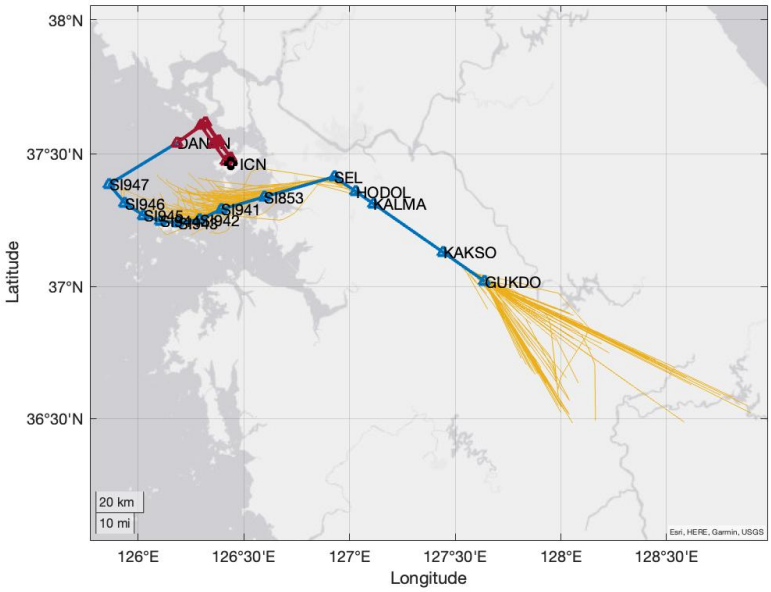
Figure 3.5. Model 1 for GUKDO 1N



(a) Result of Model 2 for GUKDO 1N

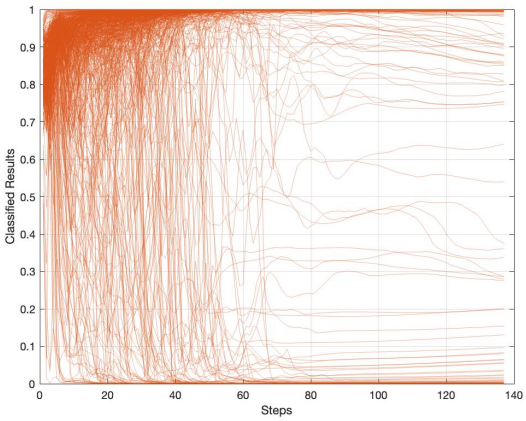


(b) Accuracy of Model 2 for GUKDO 1N

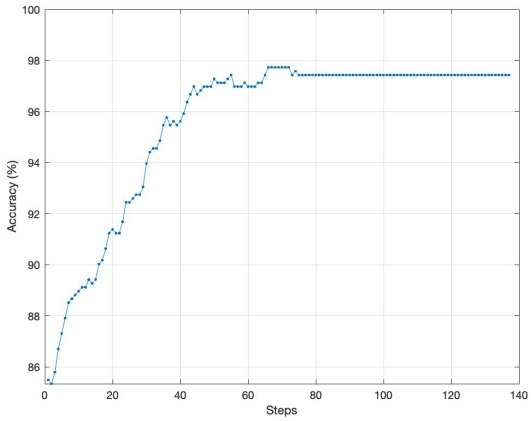


(c) Testing Trajectories of Model 2 for GUKDO 1N (Cut at 100^{th} Step)

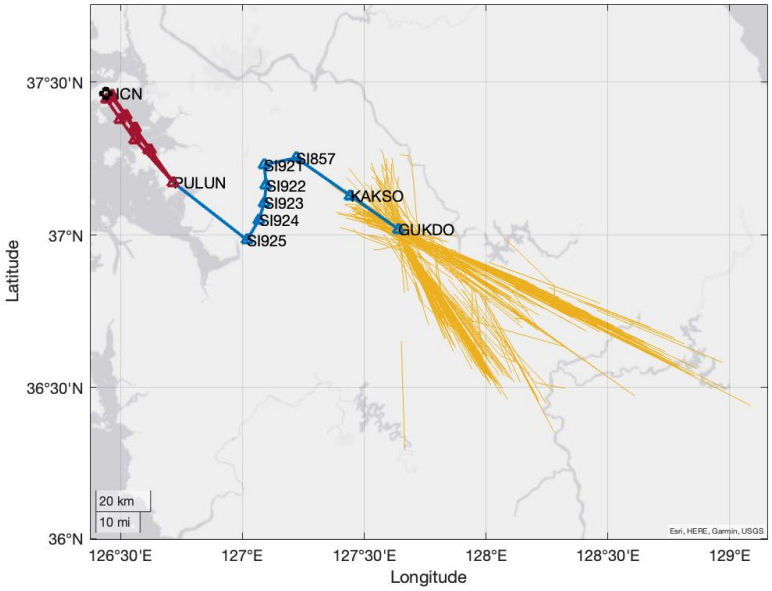
Figure 3.6. Model 2 for GUKDO 1N



(a) Result of Model 1 for GUKDO 1P

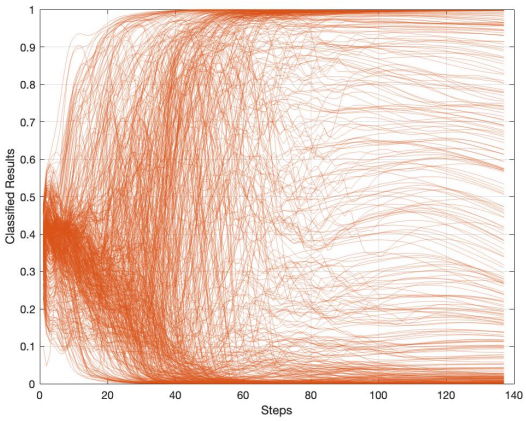


(b) Accuracy of Model 1 for GUKDO 1P

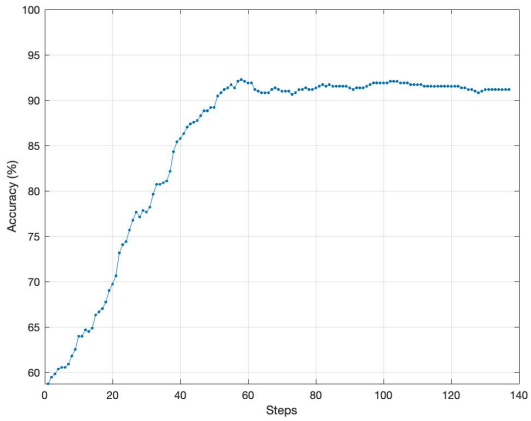


(c) Testing Trajectories of Model 1 for GUKDO 1P (Cut at 20^{th} Step)

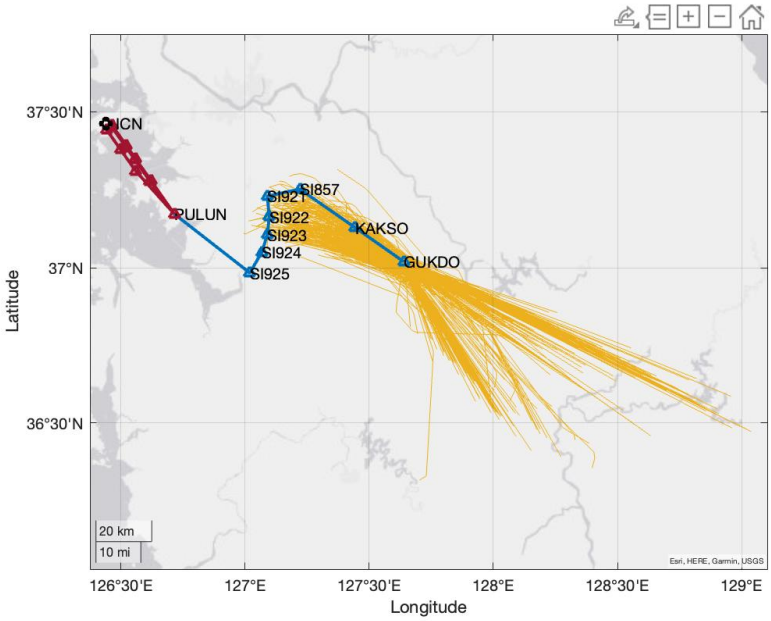
Figure 3.7. Model 1 for GUKDO 1P



(a) Result of Model 2 for GUKDO 1P

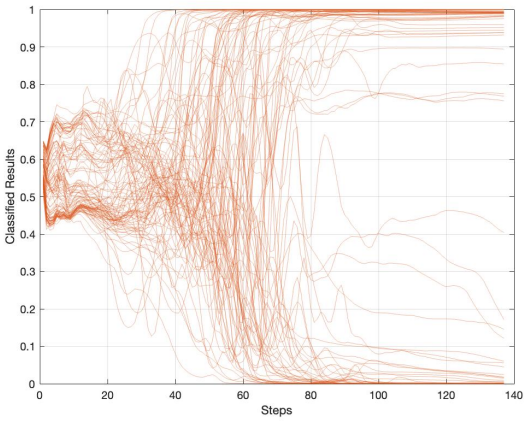


(b) Accuracy of Model 2 for GUKDO 1P

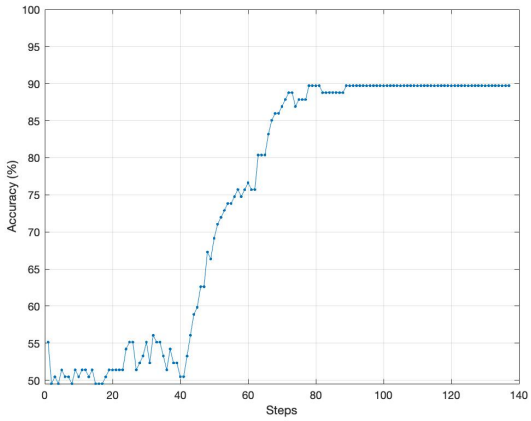


(c) Testing Trajectories of Model 2 for GUKDO 1P (Cut at 50^{th} Step)

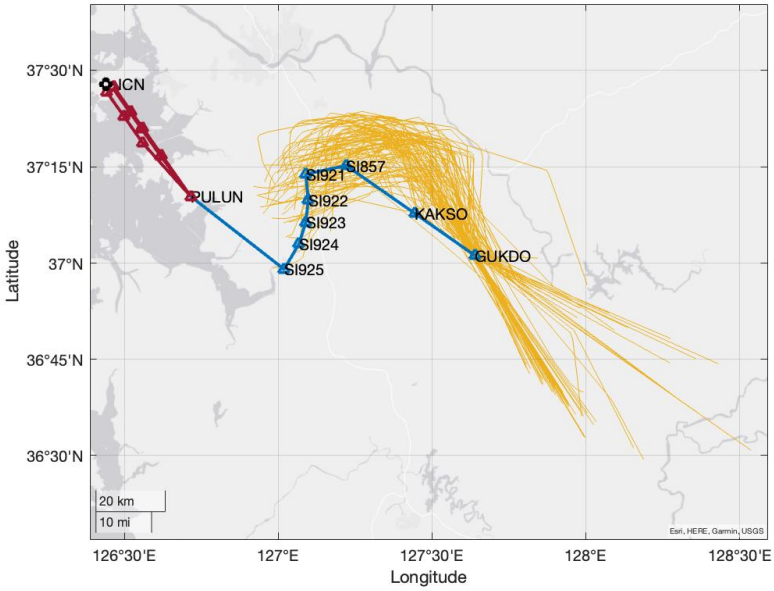
Figure 3.8. Model 2 for GUKDO 1P



(a) Result of Model 3 for GUKDO 1P

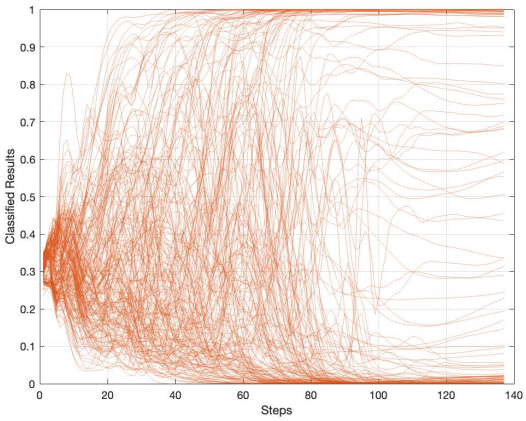


(b) Accuracy of Model 3 for GUKDO 1P

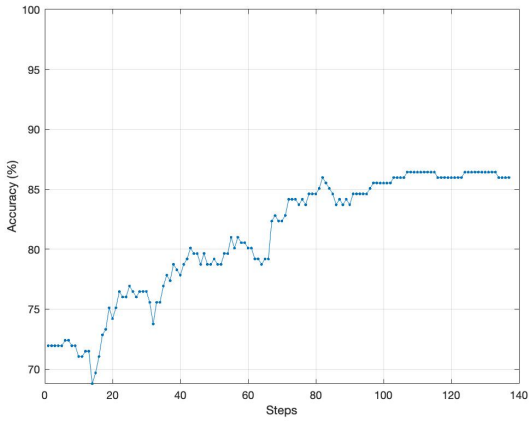


(c) Testing Trajectories of Model 3 for GUKDO 1P (Cut at 70^{th} Step)

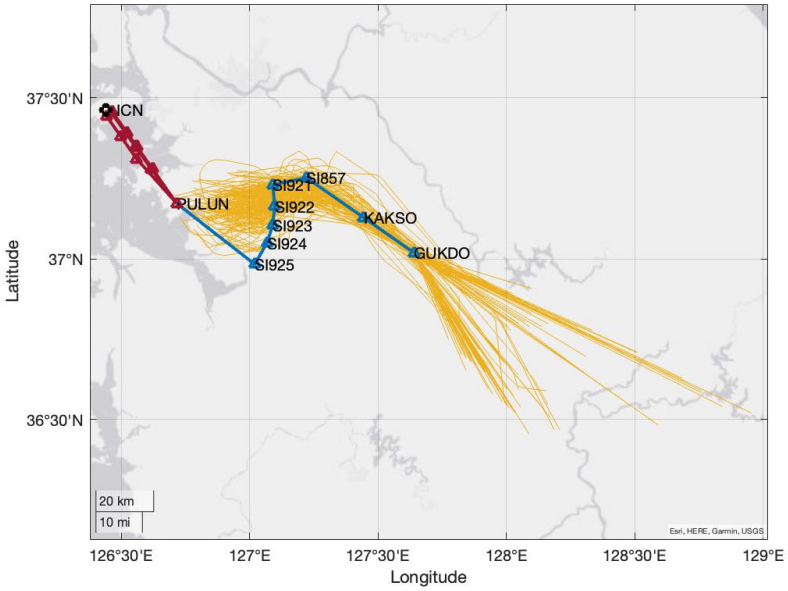
Figure 3.9. Model 3 for GUKDO 1P



(a) Result of Model 4 for GUKDO 1P

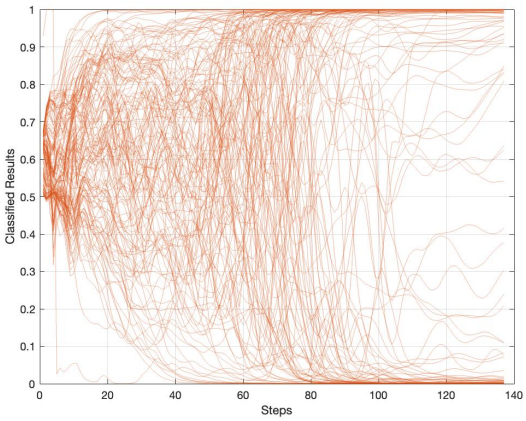


(b) Accuracy of Model 4 for GUKDO 1P

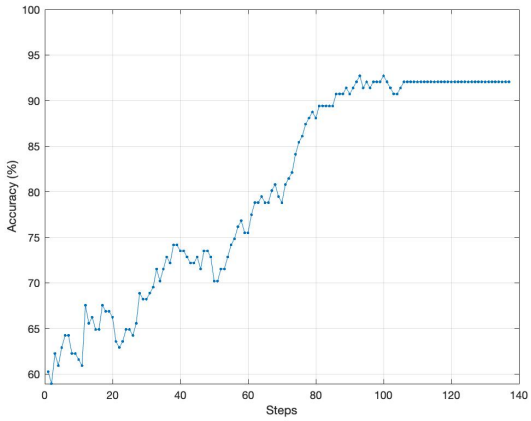


(c) Testing Trajectories of Model 4 for GUKDO 1P (Cut at 82^{th} Step)

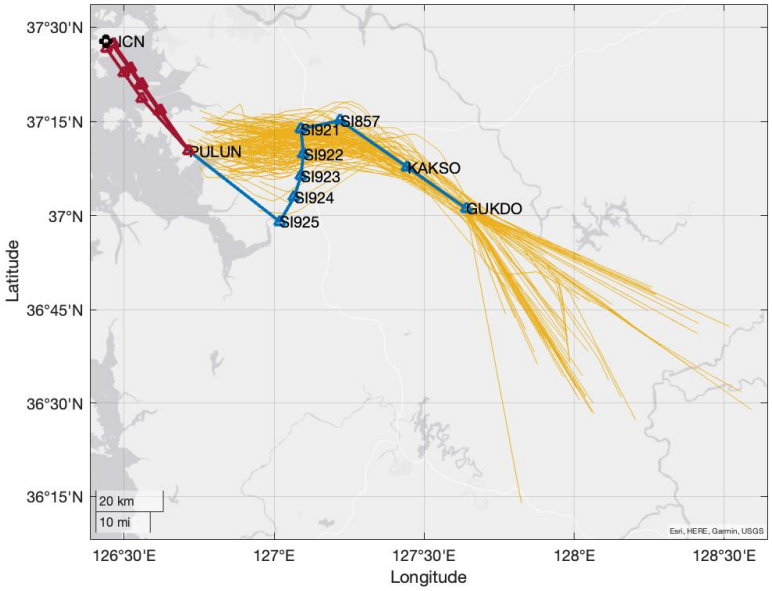
Figure 3.10. Model 4 for GUKDO 1P



(a) Result of Model 5 for GUKDO 1P

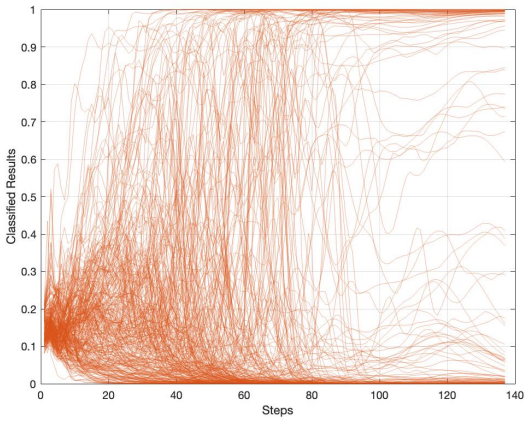


(b) Accuracy of Model 5 for GUKDO 1P

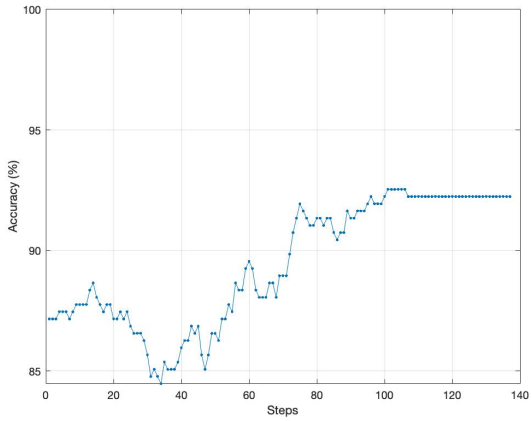


(c) Testing Trajectories of Model 5 for GUKDO 1P (Cut at 85^{th} Step)

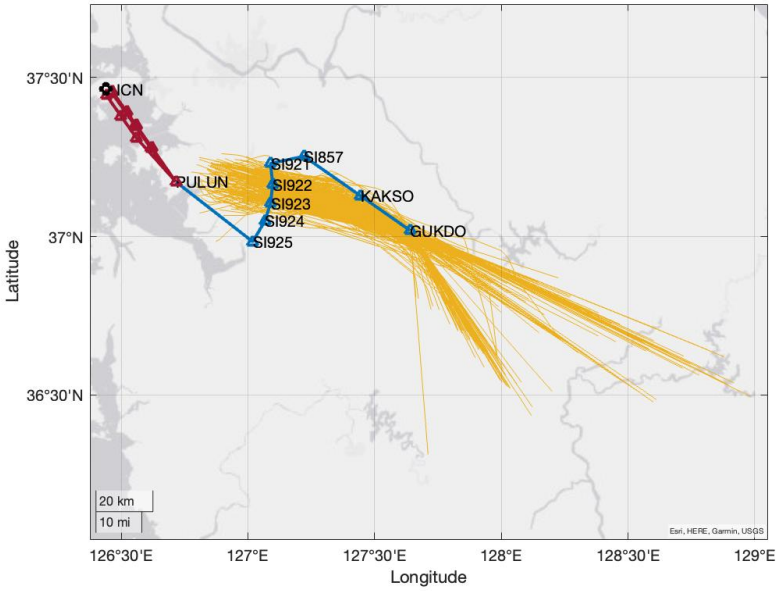
Figure 3.11. Model 5 for GUKDO 1P



(a) Result of Model 6 for GUKDO 1P

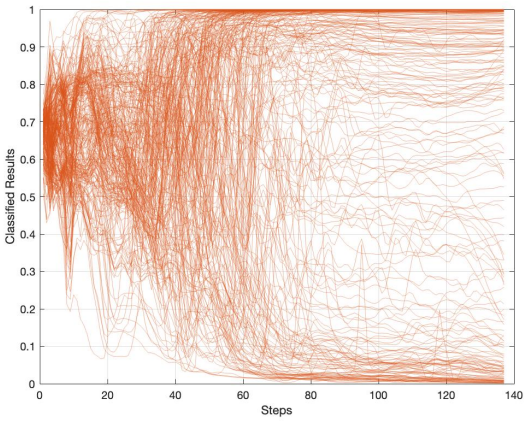


(b) Accuracy of Model 6 for GUKDO 1P

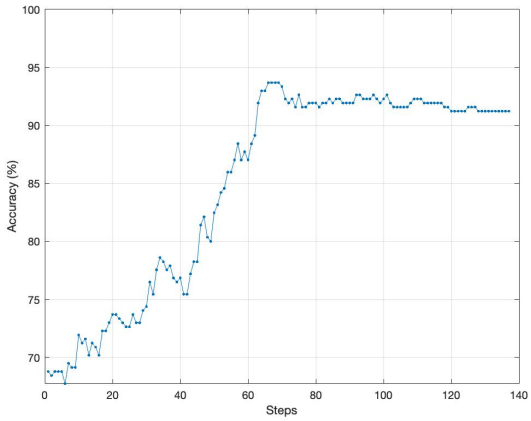


(c) Testing Trajectories of Model 6 for GUKDO 1P (Cut at 75^{th} Step)

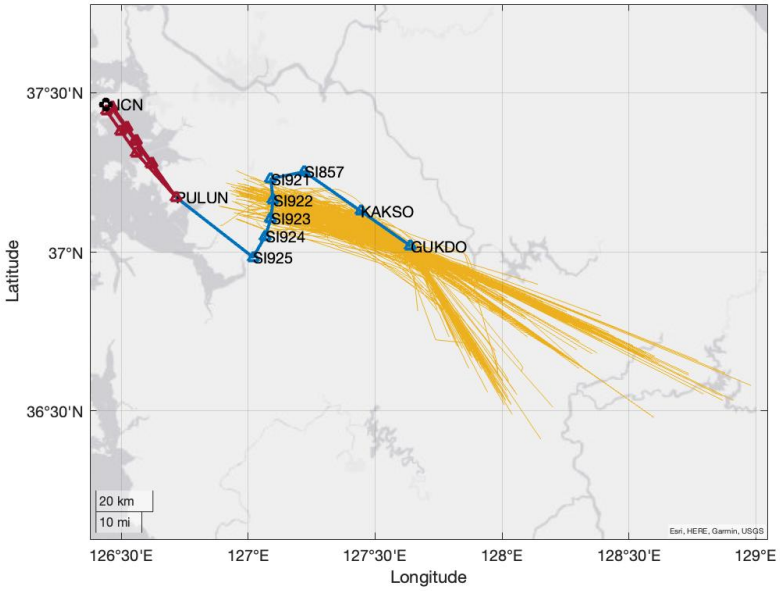
Figure 3.12. Model 6 for GUKDO 1P



(a) Result of Model 7 for GUKDO 1P



(b) Accuracy of Model 7 for GUKDO 1P



(c) Testing Trajectories of Model 7 for GUKDO 1P (Cut at 65^{th} Step)

Figure 3.13. Model 7 for GUKDO 1P

Table 3.3. Accuracy of Accuracy of Classification Models at Converged Point for GUKDO 1N

Model	Accuracy (%)	Converged Step
1	87	80
2	86	100

Table 3.4. Accuracy of Accuracy of Classification Models at Converged Point for GUKDO 1P

Model	Accuracy (%)	Converged Step
1	92	20
2	90	50
3	88	70
4	84	82
5	89	85
6	92	75
7	92	65

4. CONCLUSION

4.1 Concluding Remarks

This thesis has focused on providing a framework for trajectory pattern identification, a crucial data-preprocessing step for developing assistant tools for Air Traffic Controllers (ATCs) using data-driven methods, and trajectory pattern classification that can provide ATCs with real-time information of which possible trajectory pattern a new incoming aircraft could take. The trajectory pattern identification framework first computes the dissimilarity between trajectories using Dynamic Time Warping and then uses Ward's linkage method to link clusters of trajectories. When all trajectories are linked, number of clusters needs to be chosen and adjusted until all patterns are well-separated, patterns are combined based on fixes in the route to formed the final trajectory patterns. The trajectory pattern classification relies on Recurrent Neural Network and the fixes-based binary tree that categorizes the identified patterns by fixes. Both frameworks have been demonstrated with the Automatic Dependent Surveillance-Broadcast (ADS-B) data for Incheon international airport in South Korea that were collected between January and December 2019. The results have shown that the classification models can provide ATCs the possible trajectory pattern that a new incoming aircraft is most likely going to follow in real-time. Such probabilistic information can reduce ATCs' workload and the delay from their decision making process.

4.2 Future Work

For future work, there are mainly two parts. First, the proposed frameworks need to be applied to other routes in standard arrival route and standard instrument departures for an extensive test. Second, the result of the clustering framework, distinct trajectory patterns that previously could not be identified, can be used to improve the performance of applications such as estimated time of arrival or trajectory prediction, and to explore new domains like air traffic complexity prediction, sequencing or conflict detection and resolution. Therefore, we plan to utilize the identified patterns to show the improvements in numerical results.

REFERENCES

[1] S. Thomas and G. Athens, "Civilian and military air traffic control in the eu," 2000.

[2] *Air traffic controllers*, 15th, Bureau of Labor Statistics, Washington, DC: USA, 2014.

[3] M. Hawley, R. Bharadwaj, and V. Venkataraman, "Real-time mitigation of loss of separation events using reinforcement learning," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–6. DOI: 10.1109/DASC43569.2019.9081795.

[4] V. M. Janakiraman and D. Nielsen, "Anomaly detection in aviation data using extreme learning machines," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 1993–2000. DOI: 10.1109/IJCNN.2016.7727444.

[5] R. Deshmukh, D. Sun, K. Kim, and I. Hwang, "Temporal logic learning-based anomaly detection in metroplex terminal airspace operations," *Transportation Research Part C: Emerging Technologies*, vol. 126, p. 103 036, 2021, ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2021.103036.

[6] ——, "Reactive temporal logic-based precursor detection algorithm for terminal airspace operations," *Journal of Air Transportation*, vol. 28, pp. 155–163, 4 Oct. 2020, ISSN: 2380-9450. DOI: 10.2514/1.D0182.

[7] B. Matthews, S. Das, K. Bhaduri, K. Das, R. Martin, and N. Oza, "Discovering anomalous aviation safety events using scalable data mining algorithms," *Journal of Aerospace Information Systems*, Oct. 2013. DOI: https://doi.org/10.2514/1.I010941.

[8] L. Li, S. Das, R. J. Hansman, R. Palacios, and A. N. Srivastava, "Analysis of flight data using clustering techniques for detecting abnormal operations," *Journal of Aerospace Information Systems*, vol. 12, 9 Sep. 2015. DOI: 10.2514/1.I010329.

[9] M. C. Rocha Murça and M. de Oliveira, "A data-driven probabilistic trajectory model for predicting and simulating terminal airspace operations," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, 2020, pp. 1–7. DOI: 10.1109/DASC50938.2020.9256644.

[10] S. T. Barratt, M. J. Kochenderfer, and S. P. Boyd, "Learning probabilistic trajectory models of aircraft in terminal airspace from position data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3536–3545, 2019. DOI: 10.1109/TITS.2018.2877572.

[11] Z. Wang, M. Liang, and D. Delahaye, "Short-term 4d trajectory prediction using machine learning methods," in *7th SESAR Innov.*, 2017, pp. 1–9.

[12] M. C. R. Murça and R. J. Hansman, "Identification, characterization, and prediction of traffic flow patterns in multi-airport systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1683–1696, 2019. DOI: 10.1109/TITS.2018.2833452.

[13] K. P. Sinaga and M.-S. Yang, "Unsupervised k-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80 716–80 727, 2020. DOI: 10.1109/ACCESS.2020.2988796.

[14] B. Shi, L. Han, and H. Yan, "Adaptive clustering algorithm based on knn and density," *Pattern Recognition Letters*, vol. 104, pp. 37–44, 2018, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.01.020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865518300266.

[15] Z. Peng, J. Zhang, X. Gui, R. Hu, and D. Sui, "A data-driven probabilistic trajectory model for predicting and simulating terminal airspace operations," in *2020 International Conference on Research in Air Transportation*, 2020.

[16] R. Elankavi, R. Kalaiprasath, and R. Udayakumar, "A fast clustering algorithm for high-dimensional data," *International Journal of Civil Engineering and Technology*, vol. 8, pp. 1220–1227, 2017, ISSN: 0976-6316.

[17] M. Enriquez, "Identifying temporally persistent flows in the terminal airspace via spectral clustering," Oct. 2013.

[18] F. Liu, W. Bi, W. Hao, F. Gao, and J. Tang, "An improved fuzzy trajectory clustering method for exploring urban travel patterns," *Journal of Advanced Transportation*, vol. 2021, pp. 1–13, 2021, ISSN: 0197-6729. DOI: 10.1155/2021/6651718.

[19] X. Olive, L. Basora, B. Viry, and R. Alligier, "Deep trajectory clustering with autoencoders," Aug. 2020.

[20] N. L. Olsen, B. Markussen, and L. L. Raket, "Simultaneous inference for misaligned multivariate functional data," *Journal of the Royal Statistical Society*, vol. 67, pp. 1147–1176, 5 2016. DOI: 10.1111/rssc.12276.

[21] K. Sasirekha and P. Baby, "Agglomerative hierarchical clustering algorithm - a review," *International Journal of Scientific and Research Publications*, vol. 3, 3 2013, ISSN: 2250-3153.

[22] J. H. Ward, "Hierarchical grouping to optimize and objective function," *Journal of the American Statistical Association*, vol. 58, 301 2013, ISSN: 0162-1459. DOI: 10.1080/01621459.1963.10500845.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 8 1997. DOI: 10.1162/neco.1997.9.8.1735.

[24] B. Chakraborty, B. Shaw, J. Aich, U. Bhattacharya, and S. K. Parui, "Does deeper network lead to better accuracy: A case study on handwritten devanagari characters," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, 2018, pp. 411–416. DOI: 10.1109/DAS.2018.72.