Categorizing Flight Paths using Data Visualization and Clustering Methodologies

Yifan Song, Keyang Yu
Department of Computer Science & Engineering
The Ohio State University
Columbus, OH, USA
song.1221@osu.edu, yu.2049@osu.edu

Seth Young, Ph.D.
Dept. of Civil, Environmental, and Geodetic Engineering / Center for Aviation Studies
The Ohio State University
Columbus, OH, USA
young.1460@osu.edu

***Abstract*— This work leverages the U.S. Federal Aviation Administration's Traffic Flow Management System dataset and *DV8*, a recently developed tool for highly interactive visualization of air traffic data, to develop clustering algorithms for categorizing air traffic by their varying flight paths. Two clustering methodologies, a spatial-based geographic distance algorithm, and a vector-based cosine similarity algorithm, are demonstrated and compared for their clustering effectiveness. Examples of their applications reveal successful, realistic clustering based on human-in-the loop threshold determination, with geographic distance algorithms performing better for enroute portions of flight paths and cosine similarity algorithms performing better for near-terminal operations, such as arrival paths.**

Keywords-air traffic, clustering, data visualization, flight paths

I. INTRODUCTION

It is highly common for flights within the United States' national airspace system to fly flight paths between their origins and destinations that vary, or deviate, from their planned flight routes. While planned flight routes are quite predictable, most often following jet routes and departure and arrival procedures, all of which are published as standard by the Federal Aviation Administration, the actual paths flown are more stochastic, with less understanding of any patterns among paths flown between any given origin and destination airport.

These high levels of variability of flight paths over planned routes leads to inefficiencies in travel time, fuel burn, and aircraft utilization. If there was a method to identify how routes measure in their variability, the industry may be able to focus their efforts on creating efficiencies for these routes. One necessary element of such an analysis would be to have some robust method of categorizing a spectrum of flight paths into clusters. This paper describes a methodology that leverages the Federal Aviation Administration's NextGen System Wide Information Management (SWIM) Traffic Flow Management System (TFMS) data feed, a software tool known as *DV8* used for fast-time visualization of TFMS data, and two algorithms for determining route clusters based on the visualizations.

A. Background / Literature Review

Flight path (air trajectory) analysis is a crucial contributor to air traffic safety and efficiency. With growth in air traffic data feeds and the development of machine learning (ML) techniques, more and more ML methodologies have been introduced in flight route (trajectory) analysis. One of the early approaches is from de Leege et. al. to predict the landing trajectory in the landing process by training a generalized linear model [1]. Like de Leege, most traditional works used historical data in the prediction process. Recently, with the development in deep learning techniques, research as exampled by Liu and Hansen [2] consider a new model using, for example, convolutional and recurrent neural networks to predict the flight trajectories in a 4D, to include location, time, and the weather conditions in the vicinity.

The other approach to flight path analysis with machine learning is cluster analysis. There is a rich set of literature in this area. Gariel, et. al., were among the first to apply trajectory clustering into the aviation environment [3]. More recently, the use of newly available data and DBSCAN algorithm to cluster air traffic flows both in the enroute airspace (Duong, et. al. [4]), and in the airport terminal environment (Olive, et. al [5]). Another popular clustering algorithm, known as agglomerative hierarchical clustering, is used by Pusadan et. al. [6] to detect outlier flights with abnormal waypoints and Bombelli et. al. to identify and approximate well-traveled route in air strategic planning [7].

As an alternative to previous approaches, we propose a flight path clustering framework based on a flight data analysis and visualization tool known as *DV8,* developed by Young, et. al. [8]. Our clustering function leverages the effectiveness of interactive visualization and analysis. Instead of building a comprehensive model that may require detailed analysis over a single set of routes, our clustering function focuses on the efficiencies associated with interactive analysis, allowing a user to query, visualize, and ultimately cluster hundreds of flights in a matter of seconds. Our framework also provides the flexibility for users to select different cluster models and thresholds to generate different clustering results based on different analysis

needs and display the visualization and statistic results to let the users validate themselves.

II. Data and Visualization Model

A. FAA SWIM TFMS Data

The data used for this project emanates from the Federal Aviation Administration's SWIM program. Through the SWIM program the FAA disseminates large data sets describing the current air traffic environment in the U.S. National Airspace System [9]. The SWIM program offers several data sets. This project leverages the TFMS data set. TFMS provides real-time messaging of flight status (Such as flight plan created or modified, as well as other information including origin and destination airports, aircraft type, and aircraft operator), positioning, (latitude, longitude, speed, and altitude of aircraft in flight) and planned flight routes. The TFMS dataset disseminates its messaging through an .xml stream that may be ingested by external users upon completing the required FAA connection protocols [10].

B. DV8 Visualization Tool

To date, existing analysis tools for visualizing and analyzing large aviation data such as SWIM TFMS are in many ways inefficient: they are either time consuming, taking minutes even hours to query and process data in a static environment or visualizing in a one-shot manner. As authored by, Omidvar-Tehrani, et. al. [11], *DV8*, an interactive aviation data visualization and analysis framework, was developed to accommodate a series of query-response (interactive) analysis with historical data and dynamic comparison time-wise and location-wise. With novel techniques in caching, sampling (quantized granularity) and indexing, *DV8* can visualize and interpret the results in sub-seconds. *DV8* is implemented in Python as the computation engine with a PostgreSQL database and a web interface front-end in JavaScript. Being a web service, *DV8* has no requisite and can be executed with a browser on any platform. Fig. 1 illustrates the *DV8* system architecture.

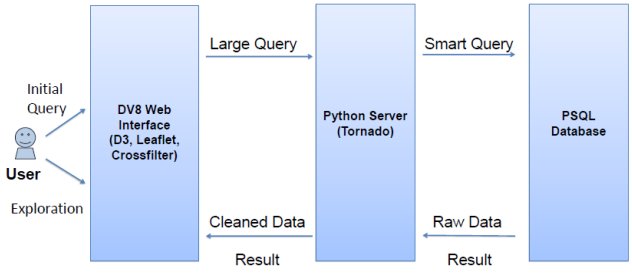

Figure 1: DV8 System Architecture

The key techniques of the *DV8* framework include:

Caching: Data is saved locally so repeat queries or sub queries (which are common in analytics) are completed immediately.

Quantized Granularity: Quantized Granularity displays every n^{th} point along a flight path. The Server delivers approximate data, but the relevant aspects of a flight path (shape, arrival/departure) are preserved.

Indexing: For instance, airport names are compared with equality operator and departure/arrival dates are compared with inequality operators thus hash and b-tree indexes are applied respectively for query speedup.

Fig. 2 illustrates an example of the *DV8* visualization of all flights from Columbus (CMH) to New York (JFK) in May 2014. *DV8* provides multiple visualization tools including color coding, dynamic filtering and toggle drawing. *DV8* also provides descriptive statistics of queried flights, as well as the ability for users to focus on a single flight and download processed data. From all these features, users can easily recognize flight patterns interactively via visualization and do a secondary analysis or development based on different needs. One of the most noticeable flight patterns is the flight path. From the figure, there are certainly multiple flight path choices and some outlier routes. This visualization motivates the need to develop clustering algorithms to categories flights by groups of routes. Thus is the motivation for this research.

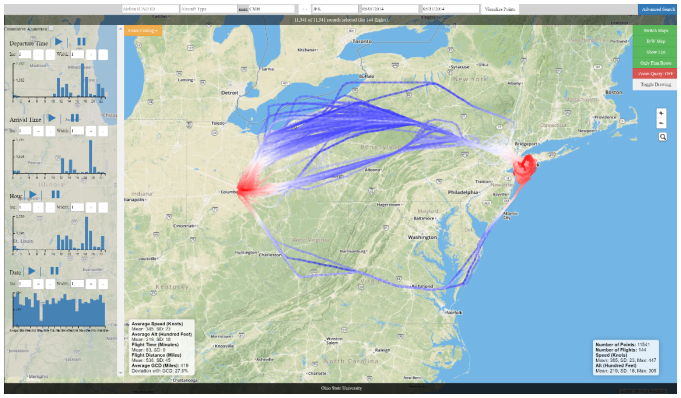

Figure 2: DV8 Visualization, CMH-JFK, May 2014

III. Route Clustering Methodologies

As discussed in Section II, *DV8* provides the ability to visualize multiple flight paths flown between two given airports. This visualization allows for easy human in-the-loop analysis consisting of visually and intuitively categorizing and clustering flight paths. However, for a human to do so for large numbers of flights over many airport-to-airport pairs, would be highly cumbersome. Therefore, using a machine learning algorithm to let perform the categorization of flight paths would automate this process and provide more information. One of the most direct and useful application for flight path learning is cluster analysis.

A. Hierarchical Clustering

Cluster analysis is an unsupervised learning task that is used to detect potential groupings within a dataset without the need to provide corresponding categorizations a-priori [12]. Cluster

analysis would assign the whole dataset into a number of groups that all data points within the same group are closer (more similar) to its counterparts in the same group and more distant (dissimilar) to the data points in the other groups. By using a clustering algorithm, our application categorizes flights into several groups according to their flight paths and detect possible outliers.

Our cluster analysis function is implemented as a hierarchical clustering algorithm, known as an agglomerative hierarchical clustering [13]. Fig. 3 provides an illustration of hierarchical clustering.

In agglomerative hierarchical clustering algorithm, initially, all data points are considered as an individual cluster. Then based on specific distance metric, the algorithm will compute the distance matrix for all pairs of data points. It will merge the two closet clusters and update the distance matrix. In each iteration, two closest clusters will be merged until there is only a single cluster remains or certain threshold is met.

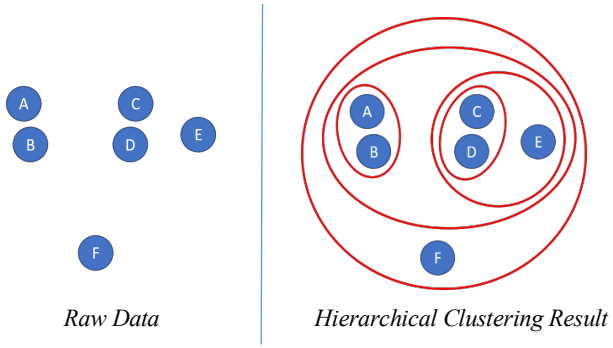

Figure 2: Hierarchical Clustering

The pseudo code of the algorithm is as follows:

Let each data point be a cluster

Compute the distance matrix between each cluster

Repeat

Merge two closest clusters as one

Update the distance matrix

***Until** certain threshold or only one cluster remaining*

The hierarchical clustering algorithm can often be visualized as a dendrogram, a tree like diagram that records the sequences of merges or splits, as illustrated in Fig. 4.

The advantages of using hierarchical clustering are that: 1) hierarchical clustering does not depend on many prior knowledge such as particular number of clusters; 2) the result, a specific number of cluster or a particular threshold, can be obtained by "cutting" the dendrogram at the proper level which provides more flexibilities. In our algorithm, we allow user to enter a distance threshold *t*, so that observations in each cluster have no greater a cophenetic distance than *t*.

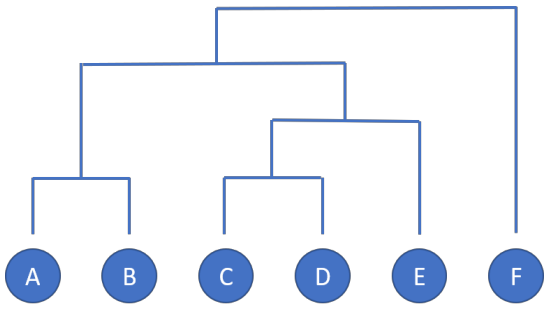

Figure 4: Dendogram

The "best" cutting level can often be determined by looping through all levels and find the one with best performance indices such as Davies–Bouldin index [14]. However, this process is computationally expensive which will reduce the efficiency and effectiveness of interactive visualization that *DV8* provides. Also, a result generated solely based on performance indices is limited to yield a singular results based on overall data, limiting opportunities for the human-in-the-loop to investigate different clustering experiments for the same query, such as detecting anomalies versus an overall categorization, focusing segments of the dataset, such as in the case of flight paths, the enroute flight segment versus departure and approach segments, etc. This will be discussed further in the next section.

Currently, using *DV8*, our clustering algorithms operates as a human-in-the loop function, with the user first selecting a clustering "algorithm", as well as an appropriate "threshold" for generating a clustering result. Based on the algorithm and threshold, our agglomerative hierarchical clustering algorithm cuts the dendrogram with a corresponding height and output the clustering result. The user can compare different clustering results based on different metrics and thresholds. Eventually, our research will allow the program itself to derive appropriate distance metrics and threshold values, based on a user's clustering intentions.

In our application, each record in the data is one flight path that contains multiple points consisting of latitude and longitude pairs. Based on the large size of data in each flight, to save computing time, our application extracts a small portion of data points from each flight path. By entering the specific number of points (N), our application divides each flight route equally into N segments and extracts one data point from each segment. Previous works, such as Satya, et. al. [15] have also found that point extraction decreases significantly computation requirements in the clustering function while preserving the original relevant information, such as shape and direction, and eventually reduces noise. In general, a large N will yield a more accurate result but take more computation power and time; recommend N values are 8, 16, 32 based on experiments.

B. Clustering Algorithms

After point extraction, our application considers two algorithms for comparative clustering computations: a spatial based algorithm based on geographic distance, and a vector-based algorithm using the concept of cosine similarity.

1) Geographic Distance: Spatial-based algorithm

The spatial based algorithm is based on geographic distance, more specifically, great circle distance (GCD). GCD is used to measure the distance between two points on the surface of Earth from latitude and longitude information. The Geographic Distance can be understood as an alternative of Euclidean Distance which is one of the most common distance functions, also often applied in air trajectory clustering. The agglomerative hierarchical clustering algorithm takes the average of great circle distance between each pairs of points for a given pair of flights. The threshold value is chosen by the user to compare how different thresholds would affect the cluster result. The formula is as follows:

$$d = \frac{1}{N}\sum_{i}^{N} GCD_i$$

$$Where\ GCD_i = R \cdot arccos\begin{pmatrix} sin(a_{1i})\, sin(a_{2i}) + \\ cos(a_{1i})\, cos(a_{2i})\, cos(|b_{1i} - b_{2i}|) \end{pmatrix} \quad (1)$$

where R is the earth's radius, a_{ij} and b_{ij} represents each latitude and longitude points from flight i point j respectively, the distance between two flights d is then calculated by taking the average of GCD between each pair of points. The geographic distance can be any real-value, in our case nautical miles (nm).

2) Cosine Similarity: Vector-based algorithm

Our second metric developed is based on a cosine similarity vector-based algorithm [14]. While the intuition of Geographic Distance is to measure the physical distance between two flights while the vector-based model is to measure the difference in their flying directions. For every two consecutive data points in a flight route, we subtract their longitude and latitude to get a vector of flight direction. We then calculate cosine similarity based on every pair of vectors to measure the direction similarity between two flight routes. The agglomerative hierarchical clustering algorithm takes the average of cosine similarity value between each pair of points extracted. The distance is calculated by 1 minus similarity. Equations (2) through (5) describe the algorithm.

$$X_i = (a_{i+1} - a_i, b_{i+1} - b_i) \quad (2)$$

$$s_i = \frac{A_i \cdot B_i}{\|A_i\|\|B_i\|} \quad (3)$$

$$S = \frac{1}{N}\sum_{i}^{N} s_i \quad (4)$$

$$d = 1 - S \quad (5)$$

Equation (2) shows how the vector is generated. Equations (3) and (4) are used to calculate cosine similarity for one pair of vectors where A_i is the i^{th} vector in the first flight and B_i is the i^{th} vector in the second flight. The distance between two flights is calculated by 1 – the average cosine similarity among pairwise vectors (5). The cosine similarity is the range of [-1, 1] while 1 representing two vectors with exact same direction, -1 for exact opposite direction, and 0 for perpendicular vectors.

IV. Example Applications of Clustering Methods

In this section, we will show some actual examples of how our clustering functions perform. We will show how different each algorithm's and associated thresholds yield different statistical results of the clusters and provide insight for our users. The thresholds may vary significantly from query to query. As such, the best way to validate is to experiment and check the visualization result. We recommend starting with a value between 40-80 nm for geographic distance and a value between 0.1-0.3 for cosine similarity. In future research, we plan to develop an automatic or semi-automatic way to recommend proper thresholds based on query and analysis need.

The first example comes from the following query: CMH (Columbus) to ATL (Atlanta) in the period of 06/01/2014 – 06/22/2014. In this example, two potential clusters are visualized. Most flights are in the east cluster that fly relatively similarly from CMH to ATL; the other small proportion of flights belonging to the west cluster choose to fly with greater variability. We will demonstrate how our agglomerative hierarchical clustering algorithm would perform to categorize such flight query.

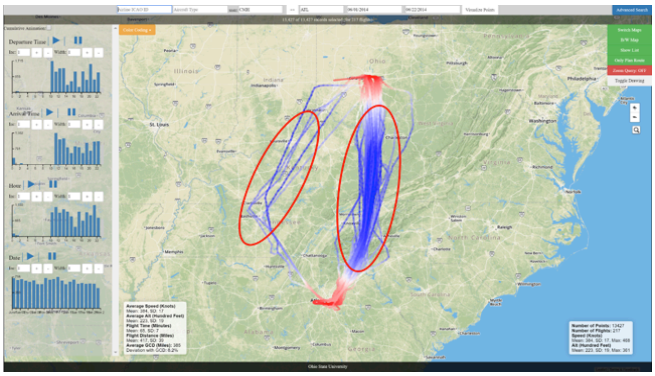

Figure 5: Flights from CMH to ATL, 6/1/2014 - 6/22/2014

Fig. 6 demonstrates the result of our clustering function using each algorithm, with a geographic distance threshold of 70 nm and a cosine similarity as the algorithm and 0.3 as its threshold. The result seems to have a perfect match with our intuitive manual grouping.

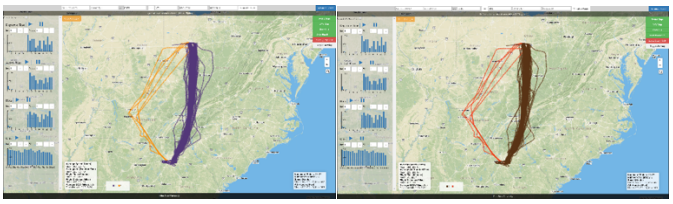

Figure 6: Clustering of CMH-ATL flights (Left: Geographical Distance Threshold 70 nm / Right: Cosine Similarity Threshold 0.3)

Table 1 describes relevant performance statistics of the two clusters generated by the agglomerative hierarchical clustering algorithm.

Table 1: Descriptive statistics of CMH-ATL categorized routes

Cluster 1		**Cluster 2**	
Number of Points: 133036	Number of Flights: 212	Number of Points: 391	Number of Flights: 5
Speed (Knots)		**Speed (Knots)**	
Mean: 382	SD: 17	Mean: 370	SD: 15
Altitude (Hundred Feet)		**Altitude (Hundred Feet)**	
Mean: 223	SD: 19	Mean: 232	SD: 12
Flight Distance (nm)		**Flight Distance (nm)**	
Mean: 420	SD: 45	Mean: 415	SD: 35
Deviation with GCD		**Deviation with GCD**	
10.0%		7.8%	

The above statistics demonstrate the effect of using our clustering function for generating the desired clustering results. Our clustering algorithm is able to identify a group of flights with larger deviation from GCD (great circle distance) between two airports. The clustering result also demonstrates one cluster with a larger variation in flight distance and lower average speed.

A. Geographic Threshold Clustering Example

In this next example, we demonstrate different clustering results and its corresponding statistical information by changing thresholds. We will use geographical distance metric in our example and change different thresholds to compare the clustering results and statistics information.

The example query we are using is SFO (San Francisco) to PIT (Pittsburgh) in the period of 07/19/2014 – 08/12/2014. In this example, by decreasing the threshold, the Geographical Distance model eventually generates three significant path groups from North to South which seems to have the best match intuitively. A relatively large threshold cannot distinguish while a relatively small threshold is providing inaccurate results. For example, as illustrated in Fig. 7, a geographical distance threshold of 100 nm yields a single cluster, while reduced distance thresholds of 80 nm and 50 nm, as illustrated in Fig. 8 and Fig. 9 respectively, yield 2 and 3 cluster results. Tables 2, 3, and 4 describe the performance of these clusters.

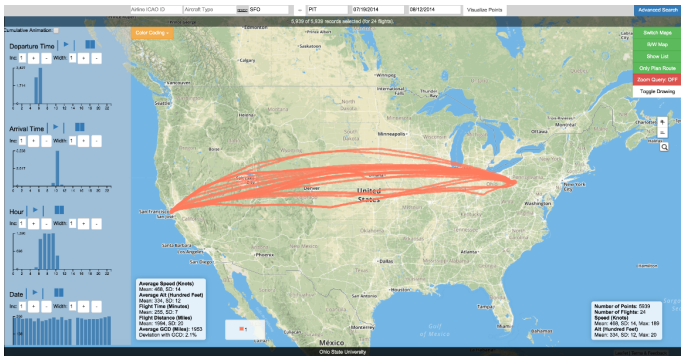

Figure 7: Clustering result of SFO-PIT, Geographical Distance threshold 100 nm – Single cluster result

Table 2: Descriptive Statistics - SFO-PIT Geographical Distance cluster threshold 100 nm

Cluster 1	
Number of Points: 5939	Number of Flights: 24
Speed (Knots)	
Mean: 467	SD: 14
Altitude (Hundred Feet)	
Mean: 334	SD: 12
Flight Distance (nm)	
Mean: 1994	SD: 20
Deviation from GCD	
2.1%	

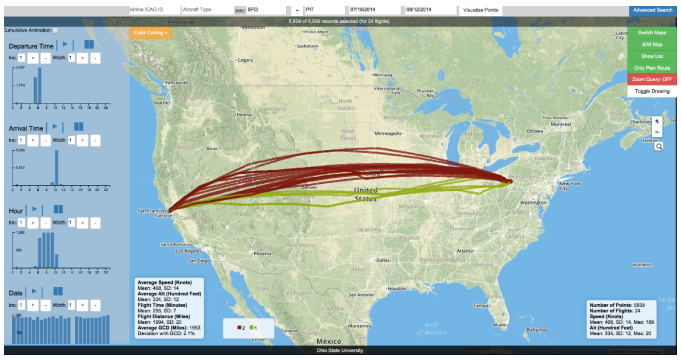

Figure 8: Clustering result of SFO-PIT, Geographical Distance threshold 80 nm - 2 cluster result

Table 3: Descriptive Statistics - SFO-PIT Geographical Distance cluster threshold 80 nm

Cluster 1		**Cluster 2**	
Number of Points: 4437	Number of Flights: 19	Number of Points: 1502	Number of Flights: 5
Speed (Knots)		**Speed (Knots)**	
Mean: 465	SD: 11	Mean: 478	SD: 18
Altitude (Hundred Feet)		**Altitude (Hundred Feet)**	
Mean: 337	SD: 12	Mean: 322	SD: 6
Flight Distance (nm)		**Flight Distance (nm)**	
Mean: 1990	SD: 25	Mean: 1997	SD: 17
Deviation with GCD		**Deviation with GCD**	
1.9%		2.3%	

Figure 9: Clustering result of SFO-PIT Geographical Distance threshold 50 nm - 3 cluster resul

Table 4: Descriptive Statistics - SFO-PIT Geographical Distance cluster threshold 50 nm

Cluster 1		**Cluster 2**		**Cluster 3**	
Number of Points: 3971	Number of Flights: 17	Number of Points: 1502	Number of Flights: 5	Number of Points: 466	Number of Flights: 2
Speed (Knots)		**Speed (Knots)**		**Speed (Knots)**	
Mean: 465	SD: 12	Mean: 478	SD: 18	Mean: 463	SD: 2
Altitude (Hundred Feet)		**Altitude (Hundred Feet)**		**Altitude (Hundred Feet)**	
Mean: 338	SD: 12	Mean: 322	SD: 6	Mean: 332	SD: 2
Flight Distance (nm)		**Flight Distance (nm)**		**Flight Distance (nm)**	
Mean: 1990	SD: 24	Mean: 1999	SD: 20	Mean: 1994	SD: 8
Deviation with GCD		**Deviation with GCD**		**Deviation with GCD**	
1.9%		2.4%		2.1%	

The above example demonstrates the effect of changing different thresholds for generating clustering results. By decreasing the threshold, the agglomerative hierarchical clustering algorithm can divide the existing clusters into two or more sub-clusters and provide more detailed statistical information for each cluster. Moreover, the clustering will also able to detect possible groups of air flights. For example, when threshold is 50 nm, the above example detects one group of flights as having a greater deviation from GCD and categorizes these flights to a separate cluster. However, decreasing threshold will not always provide the "optimal" result.

As illustrated in Fig 10, when the threshold is decreased to 30 nm, a "messy" result with 10 clusters is generated in our example. By experiment and visualization, our application is able to assist the user in finding the suitable threshold to acquire the "optimal" clustering result and its corresponding statistics.

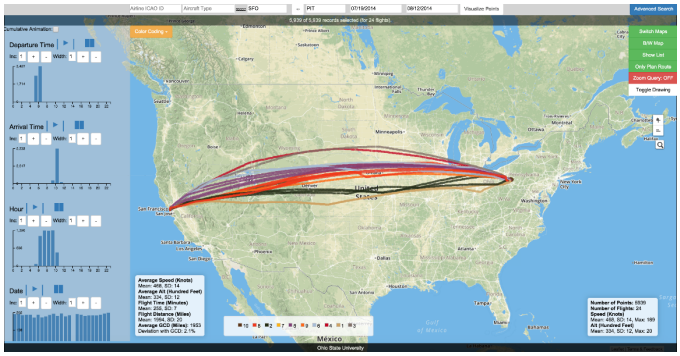

Figure 10: Clustering result of SFO-PIT, Geographical Distance threshold 30 nm

B. Cosine Similarity Clustering Example

In some cases, clustering using geographical distance may fail to achieve desired results, regardless of the threshold applied. An example of this is illustrated in Fig. 11 depicting flights between CMH and Philadelphia (PHL) from 10/1/2014 to 10/8/2014. Fig. 10 (left) illustrates a single cluster of flights when looked at the entirety of a flight, while (right) it is clearly seen that the arrival patterns into PHL follow two distinct paths (an easterly approach and a westerly approach). Regardless of what geographic distance threshold used, the spatial-based methodology fails to perform this clustering.

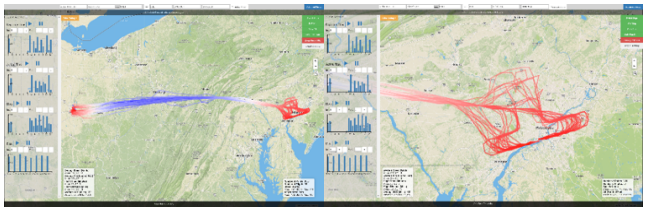

Figure 11: CMH-PHL 10/1/2014 - 10/8/2014, entire flight path (left), arrival paths into PHL (right)

The Cosine Similarity algorithm, however, generates a perfect clustering of these two approaches. As illustrated in Fig. 12, applying a threshold of 0.1 to the cosine similarity algorithm, results in these categories of approaches.

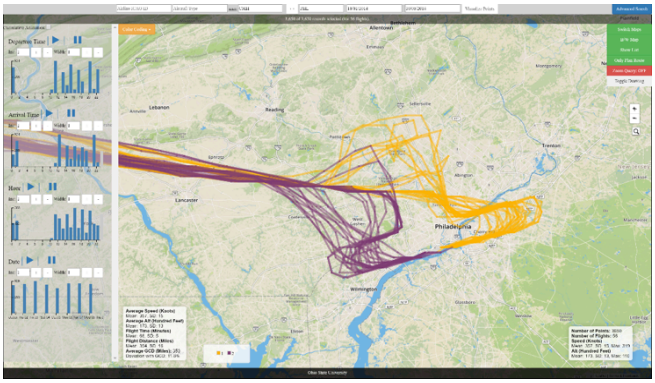

Figure 12: Clustering Result of CMH-PHL arrivals into PHL, Cosine Similarity threshold 0.1

C. Shortcomings of each algorithm

While each algorithm has its strengths, each encounter applications in which they are weak. In some cases, both algorithms reveal feasible results for the same group of flights. For example, referring back to Fig. 6, a reasonable clustering of flights between CMH and ATL was achieved by both geographical distance and cosine similarity algorithms.

In other cases, it is found that one algorithm may provide feasible results, while the other may not.

For example, in the case of a full flight route, such as in the SFO-PIT case, the cosine similarity algorithm fails to find appropriate clustering, due to the highly similar directional vectors of each of the flights visualized, as illustrated in Fig. 13.

Figure 13: Clustering result of SFO-PIT, Cosine similarity thresholds 0.03 (left), 0.01 (center), 0.003 (right)

Conversely, in the case of CMH-PHL, the clustering of approaches using the geographic distance algorithm either results in a failure to create any clusters or creates too many unrealistic clusters, as illustrated in Fig. 14. This is due to the fact that over the course of the flight, the relative distance between the routes is so small that the algorithm fails to pick up the relatively small variations in the approach paths relative to the entire flight.

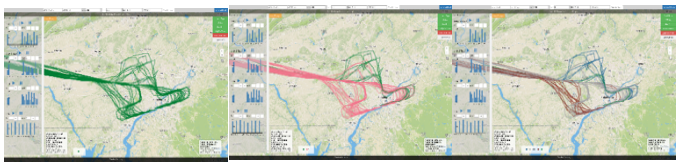

Figure 14: Clustering results of CMH-PHL, Geographical distance thresholds 30 (left), 20 (center), 10 (right)

From the above examples, we may conclude that the Geographical Distance algorithm is more accurate when clustering path groups that are significantly far away from each other though they may follow very similar shape or direction; the Cosine Similarity algorithm, however, is more sensitive to directional differences but cannot distinguish if the flights are in same direction. More generally, air flights often "near" each other during the departure and arrival phase of flight, but the difference between flight directions may be large to enter their planned flight routes. Therefore, the cosine similarity model may yield a good clustering result during the departure/arrival period just as the CMH-PHL example. For overall flights, and particularly clustering for enroute phases of flight, geographical distance is the more appropriate algorithm.

V. CONCLUSIONS AND FUTURE WORK

Using FAA TFMS data and the *DV8* data visualization tool, two distance functions were developed to categorize groups of flights by their flight paths using human-in-the-loop hierarchical clustering methodologies. Specifically, a spatial-based algorithm based on geographic distance between routes, and a vector-based algorithm, based on cosine similarity were developed and tested over three examples. The results found that while in some cases both algorithms produced reasonable results, the spatial-based algorithm performed better for overall flights, particularly all traveling in the same general direction, while the vector-based algorithm performed better in phases of flight where physical distance may be close, but directionality is more varied, as in the case of various approaches to an airport.

To take advantage of both cosine similarity's good performance during the departure/arrival period and geographic distance's good performance during the enroute flying period, future work includes a plan to segment flight routes into 3 segments, departure, enroute, and arrival and apply the appropriate clustering algorithm to each segment, creating a more diverse collection of clusters per route. Other advanced but computationally expensive distance functions to compute trajectory similarity would also be implemented, such as Fréchet distance [16], for experiments and comparison. *DV8* is a framework friendly to secondary development.

In addition, future work includes developing machine learning algorithms to reduce the "human-in-the-loop", by having the program itself selects the most appropriate algorithm and thresholds for each flight route based on users' intent while keeping the visualization efficiency. This becomes possible with

the ability to search and display many historical flights in fast-time. We will continue to develop these ML capabilities within the *DV8* framework, using current TFMS, as well as other available air traffic data feeds.

REFERENCES

[1] A. de Leege, M. Van Paassen and M. Mulder, "A machine learning approach to trajectory prediction," in *AIAA GNC Conference and Exhibit*, Boston, MA, 2013.

[2] Y. Liu and M. Hansen, "Predicting aircraft trajectories: A deep generative convolutional recurrent neural networks approach". Preprint: Research Gate publication 330035171, Preprint Dec. 2018

[3] M. Gariel, A. N. Srivastava and E. Feron, "Trajectory clustering and an application to airspace monitoring," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1511-1524, Dec. 2011.

[4] Q. Duong, D, Pham, T. Tran, A. Mai, "A Simplified Framework for Air Route Clustering based on ADS-B Data", Proceedings of the 13th international Conference on Computing and Communication Technology (RIVF 2019), 2019

[5] Olive, Xavier & Morio, Jérôme. (2019). Trajectory Clustering of Air Traffic Flows around Airports. Aerospace Science and Technology. 84. 776-781. 10.1016/j.ast.2018.11.031.

[6] M Y Pusadan et al. "Anomaly detection of flight routes through optimal waypoint", 2017 J. Phys.: Conf. Ser. 801 012041

[7] Bombelli, Alessandro & Segarra, Adrià & Trumbauer, Eric & Mease, Kenneth. (2019). "Improved Clustering for Route-Based Eulerian Air Traffic Modeling". *Journal of Guidance, Control, and Dynamics*. 10.2514/1.G003939.

[8] https://u.osu.edu/young.1460/dv8/

[9] https://www.faa.gov/air_traffic/technology/swim/overview/

[10] https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/pmo/industry_day/media/day_1_-_tfms_mark_novak_-_john_shea_-_final.pdf

[11] B. Omidvar-Tehrani, A. Nandi, N. Meyer, D. Flanagan, S. Young, "DV8: Interactive Analysis of Aviation Data", Proceedings of the 33rd IEEE International Conference on Data Engineering (ICDE), 2017

[12] L. Kaufman, P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Anlaysis*, Wiley publishing, 2009

[13] A. Athman Bouguettaya, Q. Yu, X. Liu X. Zhou, A. Song, "Efficient agglomerative hierarchical clustering". Expert Systems with Applications, Vol. 42, Issue 5, April 2015, p. 2785-2797

[14] D.L. Davies, and W. Donald, "A Cluster Separation Measure," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 1, No. 2, 1979, pp. 224–227. doi:10.1109/TPAMI.1979.4766909

[15] K.P.N.V.Satya sree1 , Dr.J V R Murthy, "Clustering Based on Cosine Similarity Measure", *International Journal of Engineering Science, & Advanced Technology*, Vol. 2, Issue 3. P. 508-512, 2012

[16] H. Alt, C. Knauer, and Wenk, C., "Matching Polygonal Curves with Respect to the Fréchet Distance," Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science, Springer–Verlag, Berlin, 2001, pp. 63–74.