In *Proceedings of the Eighth ACM SIGSPATIAL International Workshop on Computational Transportation Science*, Y. Huang and X. Chen, eds., Seattle, WA, November 2015.

# DICLERGE: Divide-Cluster-Merge Framework for Clustering Aircraft Trajectories

Samet Ayhan
Department of Computer Science
University of Maryland
College Park, MD 20742
sayhan@cs.umd.edu

Hanan Samet
Department of Computer Science
University of Maryland
College Park, MD 20742
hjs@cs.umd.edu

## ABSTRACT

Whether descriptive, predictive or prescriptive, most analytics applications pertaining to aircraft trajectory data require clustering to group similar trajectories and discovering a representative trajectory for all as a single entity. During the process, considering a trajectory as a whole may mislead, resulting in overfitting and a failure to discover a representative trajectory.

In this paper, we describe a novel clustering framework; divide-cluster-merge, *DICLERGE*, for the aircraft trajectory data, that divides trajectories into three major flight phases: climb, enroute, and descent. It clusters each phase in isolation, then merges them together. Our unique approach also discovers a representative trajectory, the model for the entire trajectory set. Our experiments use a real trajectory dataset with pertinent weather observations to demonstrate the effectiveness and efficiency of the *DICLERGE* algorithm to the aircraft trajectory clustering problem.

## Categories and Subject Descriptors

H.2.8 [**Database Applications**]: [Data Mining]

## General Terms

Algorithms

## Keywords

Aircraft trajectory clustering, divide-cluster-merge framework

## 1. INTRODUCTION

Air Traffic and Capacity Management constantly monitors aircraft operating within the National Airspace System (NAS) and aims to safely and efficiently manage its flow. Maintaining safe separation and sequencing of aircraft is a challenging task due to fact that the current approach of procedural control assumes that aircraft will fly along routes designated by *waypoints* or *fixes*. This approach is easy for air traffic controllers to understand; however, it can be inefficient for aircraft. To more effectively and efficiently address the pertinent challenges in the airspace with increasing volume, Trajectory Based Operations (TBO) is investigated in the context of *NextGen* in the U.S. [6] and *SESAR* in Europe [8]. The TBO concept uses four-dimensional aircraft trajectories as the base information for managing safety and capacity. In the heart of TBO resides the trajectory prediction process that relies on accurate clustering of aircraft trajectories, when a stochastic approach is used. Hence, clustering of aircraft trajectories is a key building block in probabilistic trajectory prediction process.

Although there is plenty of work in the area of data mining with regards to trajectory clustering in general [9, 13, 18, 20, 30, 31, 34], the vast majority address the issues pertaining to clustering of road networks [15, 16, 23, 32, 33]. Among the rest, only a handful of them deal with clustering of aircraft trajectories [10, 14, 19]. In fact, of those, to the best of our knowledge, there is no previous work addressing the clustering of aircraft trajectories taking three or more dimensions into account. Note that we do not deal with the issues of how the trajectories are generated (e.g., [29]), queries on the individual objects or waypoints (e.g., [22, 25, 27, 28]), and matching (e.g., [17, 21, 26]).

In this paper, we propose a novel framework for clustering aircraft trajectories that is based on divide, cluster, and merge tasks. The framework consumes a set of multidimensional trajectory points pertaining to a particular flight, divides them based on flight phases, clusters them in isolation, and merges them. By performing lateral and vertical smoothing, the framework generates a representative trajectory, the model, capturing the behavior of aircraft.

In summary, the contributions of this paper are as follows:

- Our unique approach enables accurately clustering of three or more dimensional aircraft trajectories by employing a divide-cluster-merge framework. Whereas the very limited amount of previous research address clustering of aircraft trajectories with two dimensions, only.
- We present a formal aircraft trajectory clustering algorithm based on well-known aviation domain facts. We also present a representative trajectory algorithm


*IWCTS'15,* November 03-06, 2015, Bellevue, WA, USA

http://dx.doi.org/10.1145/2834882.2834887.

that reveals the trajectory model based on lateral and vertical smoothing.

- We demonstrate by using a real aircraft trajectory dataset that our framework effectively performs clustering and discovers the representative trajectory.

The rest of this paper is organized as follows. Section 2 describes our novel algorithm for the aircraft trajectory clustering problem. Section 3 presents the results of experimental evaluation. Section 4 discusses related work. Conclusions directions for future work are outlined in Section 5.

# 2. AIRCRAFT TRAJECTORY CLUSTERING

In this section, we present an overview of our design. Section 2.1 formally describes the problem statement. Section 2.2 provides a skeleton of our trajectory clustering framework, *DICLERGE*. Section 2.3 elaborates on the AircraftTrajectoryClustering algorithm, which is a subtask of *DICLERGE*. Section 2.4 discusses the RepresentativeTrajectory algorithm, which is another subtask of *DICLERGE*, generating a representative trajectory.

## 2.1 Problem Statement

We implement an algorithm, *DICLERGE* that is based on the idea of divide, cluster, and merge. Given a set of multidimensional trajectory points pertaining to a particular flight, our framework executes a number of efficient and effective algorithms to generate a set of clusters as well as a representative trajectory. Aircraft trajectory, phases of flight, cluster and representative trajectory are defined as follows:

*Definition 1. (Aircraft trajectory)* An aircraft trajectory is a set of multidimensional points, where each point is defined by its 4-dimensions, latitude, longitude, altitude, and time. Additional dimensions such as airspeed and weather conditions may also be used.

*Definition 2. (Phases of flight)* Although there are at least five phases of flight which are taxi, climb, enroute, descent, and landing from the aviation standpoint, we will consider only three here as we are interested in wheels-up phases, only. These phases are climb, enroute, and descent. Climb phase refers to an aircraft's climbing to a certain altitude (typically 30,000 ft). Enroute is the level portion of aircraft between climb and descent phases. A descent during flight is the phase where an aircraft decreases altitude to approach landing.

*Definition 3. (Cluster)* A cluster is a set of multidimensional points pertaining to aircraft trajectories with commonality that are grouped together.

*Definition 4. (Representative trajectory)* A representative trajectory is a set of cluster centroids, filtered and connected together to best represent the underlying trajectories. It is an imaginary trajectory for representation purposes only.

Figure 1 shows the overall procedure of aircraft trajectory clustering in the divide-cluster-merge framework. To better illustrate the flight phases, a vertical profile of a flight is used. First, a set of aircraft trajectories are divided into climb, enroute, and descent phases. Then, points per phase are clustered in isolation. Finally, the clusters and representative centroids are accumulated together, forming the entire set for the original trajectories.

## 2.2 DICLERGE Algorithm

Algorithm 1 illustrates the skeleton of our trajectory clustering framework, *DICLERGE*. It goes through three phases by executing a number of algorithms to perform divide, cluster, and merge subtasks. In addition, Representative Trajectory is generated by applying lateral and vertical smoothing processes. These algorithms are elaborated in the following sections.

**Algorithm 1** DICLERGE (DIvideCLustermERGE)

**Input:** A set of multidimensional trajectory points, *D*
Number of clusters per flight phase, *kClimb, kEnroute, kDescent*
Max delta bearing, *maxDBearing*
**Output:** A set of clusters, *O*
A set of centroids, *CP*
A representative trajectory, *RTR*

**/* Algorithm AircraftTrajectoryClustering */**
**/* DIVIDE */**
1: **for each** (P in D) **do**
2: Divide the dataset D into a set of trajectories
3: Divide each trajectory TR into three phases
4: Accumulate each phase
5: **end for**
Get three sets of points
**/* CLUSTER */**
6: Perform k-means clustering on each set
Get sets of clusters per point set
Get a centroid per cluster
**/* MERGE */**
7: Accumulate sets of clusters, centroids
Get clusters, *O*
Get centroids, *CP*
**/* Algorithm RepresentativeTrajectory */**
8: Apply lateral smoothing
9: Apply vertical smoothing
Get a representative trajectory, *RTR*

## 2.3 Clustering Algorithm for Air Traffic Data

We now present our clustering algorithm for air traffic data. Given a set $D$ of multidimensional trajectory points, our algorithm generates a set of clusters, defined by their centroids. The input parameters for the algorithm are multidimensional observation set and a number of clusters per flight phase. Our algorithm is based on k-means clustering. However, instead of feeding in the entire dataset, the algorithm divides the input data into three chunks, each representing one of three flight phases and performs clustering on each chunk in isolation. The output clusters and representative centroids per chunk are then merged together.

## 2.4 Representative Trajectory

Our algorithm shares partitioning characteristic with the Mini Batch k-means algorithm [4]. However, unlike Mini Batch k-means, which randomly splits the dataset into smaller

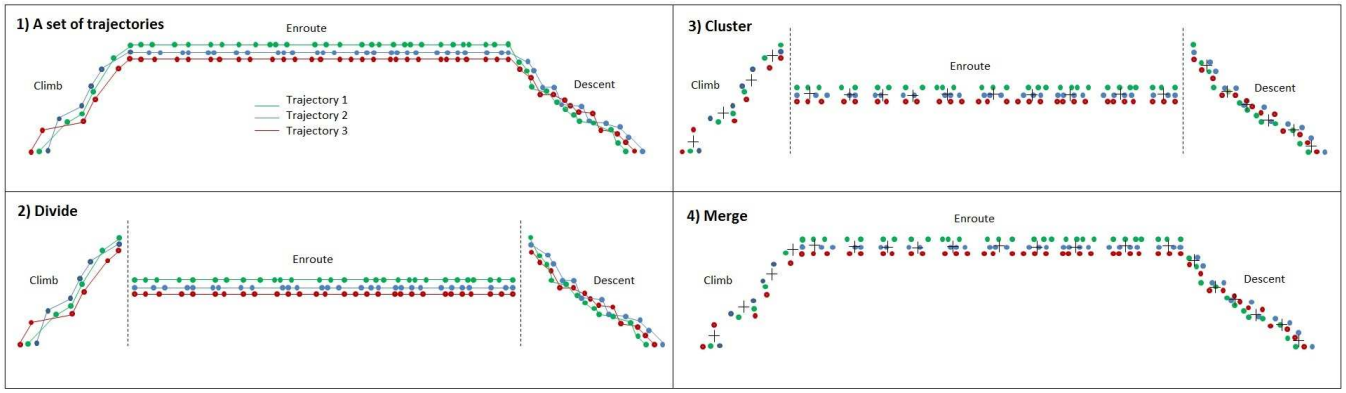



**Figure 1: The overall procedure of aircraft trajectory clustering using *DICLERGE* framework**

**Algorithm 2** AircraftTrajectoryClustering

**Input:** A set of multidimensional trajectory points, *D*
Number of clusters per flight phase, *kClimb, kEnroute, kDescent*
**Output:** A set of clusters, *O*
A set of centroids, *CP*

```
    /* STEP 1 */
    /* Determine departure and arrival points and
    all in between per flight.*/
 1: flight := [], flightClimb := [], flightEnroute := [], flight-
    Descent := [];
 2: previousDate := 01011001;
 3: for each (P in D) do
 4:   if (P.date > previousDate) then
 5:     previousDate = P.date
 6:     if (flight is not empty) then
          /* Set the enroute altitude */
 7:       enrouteAltitude = P(length(flight)/2).altitude
          /* STEP 2 */
          /* Divide the flight into phases */
 8:       enrouteStart = BinSearchLeftmost(flight,
          enrouteAltitude)
 9:       enrouteEnd = BinSearchRightmost(flight,
          enrouteAltitude)
10:       flightClimb.add(flight.slice(0, enrouteStart-1))
11:       flightEnroute.add(flight.slice(enrouteStart,
          enrouteEnd))
12:       flightDescent.add(flight.slice(enrouteEnd,
          length(flight)-1)
13:       flight = []
14:       flight.add(P)
15:     else
16:       previousDate = P.date
17:       flight.add(P)
18:     end if
19:   end if
20: end for
    /* STEP 3 */
    /* Compute cluster membership and centroids
    per phase */
21: centroidsClimb = kmeans(flightClimb, kClimb)
22: centroidsEnroute = kmeans(flightEnroute, kEnroute)
23: centroidsDescent = kmeans(flightDescent, kDescent)
24: centroids = centroidsClimb + centroidsEnroute + cen-
    troidsDescent
```

subsets, we divide the dataset with a well-known domain fact in mind; flight consists of three phases: climb, enroute and descent. Due to fact that aircraft's altitude increases at each observation during the climb, remains fixed during the enroute, and decreases during the descent phase, and that points per phase are spatially correlated, feeding the entire dataset at once to k-means results in overfitting, generating erroneous centroids.

Algorithm 2 shows AircraftTrajectoryClustering. Initially, all the points are assumed to be unclustered. As the algorithm progresses, clusters are formed and a centroid per cluster is computed. The algorithm consists of 3 steps: In the first step (lines 1∼7), the algorithm determines the first and last data recording for each trajectory by comparing its timestamp, as the timestamp of the first data point of the first trajectory will always be less than the timestamp of the first data point of the following trajectory (of the same flight which is usually scheduled for the next day). By splitting each sorted trajectory data in half, enroute altitude is determined, which is basically the median altitude of all trajectory points'. In the second step (lines 8∼20), the algorithm identifies the start and end indices of the enroute phase. This goal is attained by performing binary search on the sorted (by timestamp) list of data points. The leftmost occurrence of enroute altitude is the start, and the rightmost occurrence of enroute altitude is the end of the enroute phase of the flight. This enables the algorithm to compute the climb and descent phases of the flight, as index 0 to start of the enroute phase identifies the climb and the end of the enroute phase to the end of the flight identifies the descent phase. The algorithm retains a global array per phase, by adding the relevant data points per flight. This way, all data points for each phase are accumulated and stored in a relevant array. In the final step (lines 21∼24), the algorithm uses the array of data points along with a number of clusters per phase as input to the k-means clustering. The process fits the model using Euclidean distance and determines to which cluster each data point belongs. Upon convergence, the final set of centroids per flight phase are computed, which are merged to form the final list.

The representative trajectory of a cluster captures the overall movement of the aircraft based on trajectory data records. Once centroids are generated upon clustering, consecutive

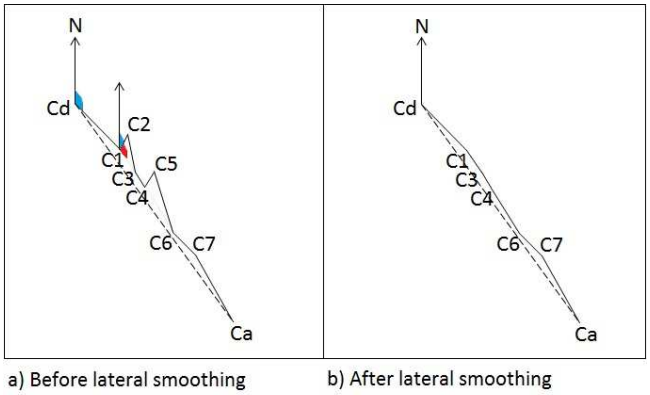



**Figure 2: Lateral smoothing**

centroids are connected to model the behavior of the aircraft. This requires omitting the abnormal centroids while connecting the valid ones to most accurately represent the overall movement of the aircraft with a single trajectory.

Algorithm 3 generates the representative trajectory in two steps. During the first step, (lines 1∼18), the algorithm performs the lateral smoothing. This goal is attained by keeping a domain fact in mind that aircraft usually fly straight and keep the delta bearing at minimal when they make turns. In the second step, (lines 19∼23), the vertical smoothing is done. This step takes advantage of the aircraft trajectory property: aircraft usually maintain the cruise level at or above 30,000 feet between the top of climb and the top of descent.

The input to the algorithm is a set of centroids sorted in the direction of departure to arrival airports. The default maximum delta bearing the aircraft is allowed to turn is a constant, *maxDBearing*, which is an input to the algorithm. The algorithm basically computes the bearing between two centroids at a time, as it traverses them. If the absolute value of the difference between the current and previous bearing is less than or equal to the maximum delta bearing, which is the lateral smoothing factor, the current centroid is appended to the representative trajectory. Otherwise, the current centroid is skipped and the next centroid is considered. The bearing is computed using the the *Haversine* equation [3] equation and the initial bearing is calculated using the latitude, longitude pairs of the departure and arrival airports.

The simplistic process is illustrated in Figure 2. The bearing between the departure (Cd) and arrival airports (Ca) is 154 degrees. The bearing between departure airport and the next centroid (C1) is 143 degrees. Given the delta bearing is 11 degrees, which is less than *maxDBearing*, 30 degrees, C1 is appended to the Representative Trajectory. The next centroid to be considered is C2. Hence, the algorithm computes the bearing between C1 and C2. Since the bearing between the two centroids is 65 degrees, the delta bearing is 143-65=78 degrees, which is greater than the maximum delta bearing. Therefore, the algorithm skips C2 and computes the bearing between C1 and C3. Due to fact that the bearing is 157 degrees, which means that the delta bearing is 157-143=14 degrees, the third centroid is appended to the Representative Trajectory. The process stops when the arrival airport is reached. The resulting set of centroids forms the lateral profile of the Representative Trajectory.

**Algorithm 3** RepresentativeTrajectory

**Input:** A set of centroids, each representing a cluster, *CP*
Max delta bearing, *maxDBearing*
**Output:** A representative trajectory, *RTR*

```
    /* STEP 1 */
    /* Apply smoothing in lateral plane */
 1: initial := True, repTrajectory := [], maxDeltaLateral =
    30.0, enrouteAltitude := 30000, prevPos = 0.0, 0.0, 0.0;
 2: prevBearing := haversine(deptAirportLon, deptAirport-
    Lat, arrAirportLon, arrAirportLat)
 3: for each (CT in CP) do
 4:    if (initial) then
 5:       prevPos = CT.latitude, CT.longitude, CT.altitude
 6:       repTrajectory.append(prevPos)
 7:       initial = False
 8:       continue
 9:    end if
10:    bearing = haversine(prevPos.lon, prevPos.lat,
       CT.longitude, CT.latitude)
11:    if (|(bearing-prevBearing) <= maxDBearing|) then
12:       prevBearing = bearing
13:       prevPos = CT.latitude, CT.longitude, CT.altitude
14:       if (prevPos not in repTrajectory) then
15:          repTrajectory.append(prevPos)
16:       end if
17:    end if
18: end for
    /* STEP 2 */
    /* Apply smoothing in vertical plane */
19: enrouteStart = BinSearchLeftmostGT(repTrajectory,
    enrouteAltitude)
20: enrouteEnd = BinSearchRightmostGT(repTrajectory,
    enrouteAltitude)
21: avgEnrouteAltitude = avg(repTrajectory.slice(enrouteStart,
    enrouteEnd).altitude)
22: setAlt(repTrajectory.slice(enrouteStart, enrouteEnd))
    = averageEnrouteAltitude
23: repTrajectory = repTrajectory.slice(0, enrouteStart-1)
    + repTrajectory.slice(enrouteStart, enrouteEnd) + rep-
    Trajectory.slice(enrouteEnd, length(flight)-1)
```

Vertical smoothing is done in the next step. The sorted centroids with regards to their lateral profile in the direction of departure to arrival airport are fed into a binary search function, where the function searches for the leftmost occurrence of altitude that is greater than or equal to 30,000 feet. The index of the leftmost occurrence is stored. The rightmost index is also searched and stored the same way. All centroids in between these two indices are sliced, forming the enroute phase of the Representative Trajectory. The mean value for their altitudes is computed. The mean altitudes for each of the enroute centroids are replaced with the original altitudes. This process completes the vertical smoothing and yields the final Representative Trajectory. Figure 3 captures the before and after view of vertical smoothing process in a simplistic way. The left figure shows ever changing cruise

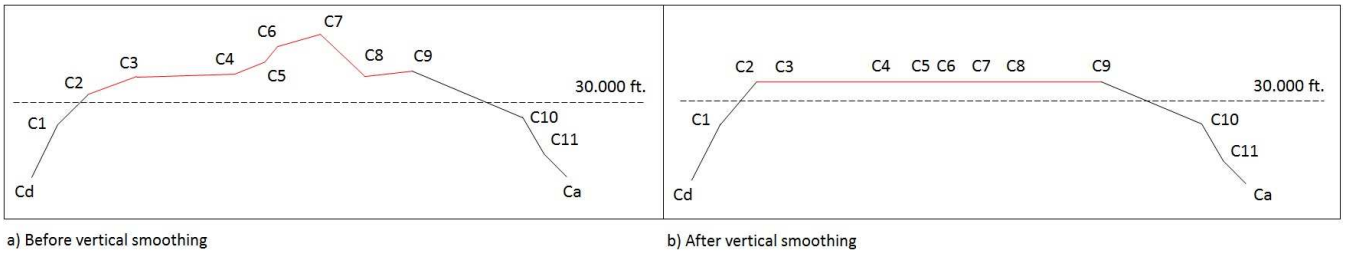



**Figure 3: Vertical smoothing**

altitude in the form of a rugged line in the vertical profile. In the right figure, the cruise altitude is fixed, presenting a realistic vertical profile.

# 3. EXPERIMENTAL EVALUATION

In this section, we conduct experiments to validate the effectiveness of our aircraft trajectory clustering algorithm, *DICLERGE*. We describe the experimental dataset preparation and environment in Section 3.1. We discuss the results for the clustering process as well as the effects of parameter values in Section 3.2.

## 3.1 Dataset Preparation

We used a real aircraft trajectory dataset: The Delta Airlines' flight DAL1865, departing from Hartsfield-Jackson Atlanta International Airport (KATL) and arriving at Miami International Airport (KMIA) for the duration of January through June 2015. The dataset has a total of 168 trajectories and 25036 points. ATL to MIA is one of the major routes in the NAS due to fact that the departure airport is the busiest airport in the United States and the flights are prone to frequent convective weather in the airspace controlled by Miami Air Route Traffic Control Center (ARTCC).

The source of the surveillance data is Aircraft Situation Display to Industry (ASDI) [1] provided in near real-time by the FAA. The surveillance part of the data was generated by merging a subset of various ASDI message types including: departure information, track information, and flight management information. In order to fuse weather parameters with surveillance data during the scheduled hours, we also obtained weather data. The source of the weather data was National Oceanic and Atmospheric Administration (NOAA) National Centers for Environmental Prediction (NCEP) Rapid Refresh product [5], an hourly-updated modeling system operational at NCEP. This way, each trajectory point for flight DAL1865 was associated with weather parameters yielding the following attributes per trajectory point: source center, source date, source time, aircraft id, aircraft speed, latitude, longitude, altitude, temperature, wind speed, and wind direction.

Our experiments were conducted on a Oracle VM VirtualBox v4.3.20 virtual machine running on Linux Ubuntu 14.04.2 64-bit LTS Operating System hosted by a computer with Intel Core i7-3840QM CPU @ 2.80GHz and 16GB memory, running on Microsoft Windows 7 Operating System. All the algorithms in our system were implemented in Python v2.7.

| $s_{regular}$ | $s_{climb}$ | $s_{enroute}$ | $s_{descent}$ |
|---|---|---|---|
| 0.678 | 0.557 | 0.702 | 0.565 |

**Table 1: Silhouette Coefficients**

## 3.2 Results for the Aircraft Trajectory Data

Our assessment included running k-means clustering with and without the *DICLERGE* framework on the DAL1865 trajectory data. For the clustering process, we used k=20 for both executions. Our attempt was to measure the clustering results quantitatively and qualitatively for each execution.

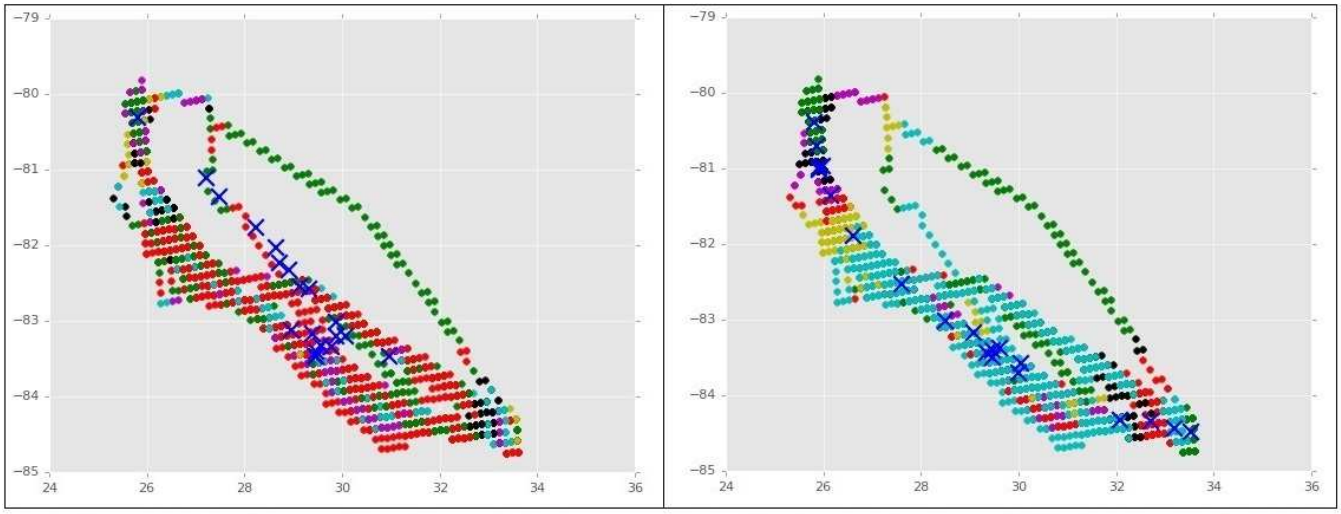
Unfortunately, there is no well-defined measure for quantitative evaluation due to fact that the ground truth labels are not known. Hence, we performed the evaluation using the model, itself. The *Silhouette Coefficient* is a way of such an evaluation, where a higher coefficient score relates to a model with better defined clusters [7].

We computed the coefficient for regular k-means clustering ($s_{regular}$). We also computed the coefficients for the *DICLERGE* framework on climb ($s_{climb}$), enroute ($s_{enroute}$), and descent ($s_{descent}$) phases. Table 1 indicated that the clustering quality of regular k-means was higher than the *DICLERGE* framework's climb and descent phases. However, it was lower than the *DICLERGE* framework's enroute phase.

For the qualitative measure, we used visual inspection and domain knowledge. The outcome of the execution without the *DICLERGE* framework is captured on Figure 4a. This execution treated the entire trajectory as a whole, generating centroids concentrated in the center. Whereas *DICLERGE* framework divided the trajectories into 3 phases first, clustered each set in isolation and merged them together at the end. For the $k$=20, we used *kClimb*=4, *kEnroute*=10, and *kDescent*=6 as the optimal parameters. The outcome of this execution is illustrated on Figure 4b.

It is obvious that the Figure 4 on the right presents a better clustering as the centroids follow a more linear pattern from departure to arrival airport, lining up toward the bottom of the trajectory set where the trajectory points are denser.

To generate representative trajectory, we ran our algorithm and obtained the result illustrated in Figure 5. The representative trajectory shown with red on Google Earth [2] is overlaid on top of the trajectory set, providing a model for the movement of the flight over the period of 6 months.




a) k-means clustering of aircraft trajectories, k=20

b) k-means clustering of aircraft trajectories using *DICLERGE* framework, k=20

**Figure 4: Clustering of aircraft trajectories with and without *DICLERGE* framework**

The heuristic in selecting the number of clusters per flight phase depends on the number of trajectory points per phase. We maintain the ratio of the number of trajectory points in each phase to the total trajectory points and map that to the pertinent parameters, *kClimb*, *kEnroute*, and *kDescent*. Our heuristic for selecting the lateral smoothing parameter is based on the level of smoothing we would like to perform. Due to fact that our trajectory is prone to convective weather, which means that aircraft may perform drastic turns to avoid the convection, we set our lateral smoothing to a relatively larger value which was 30 degrees.

## 4. RELATED WORK

Although there is a vast amount of literature in the data mining area, with regards to clustering trajectories, there is only a limited number of research on clustering aircraft trajectories.Of this research, there is no work addressing the clustering of aircraft trajectories taking three or more dimensions into account, to the best of our knowledge.

Clustering algorithms can be considered in four major categories: partitioning based methods (e.g., k-means [20]), density-based methods (e.g., DBSCAN [13]), hierarchical methods (e.g., BIRCH [34]), and grid-based methods (e.g., STING [31]). Once the aircraft trajectories are divided into flight phases, our *DICLERGE* framework uses k-means to perform clustering and merges them together.

One of the previous research, TRACLUS [18] proposes Partition-and-Group framework that is similar to ours in a sense that it partitions the trajectories into subtrajectories. However, in their approach, subtrajectories are represented by line segments and grouped together using a distance function. Besides, unlike ours, their clustering algorithm is density based.

In their research, Gariel et al. [14] proposed a way point based trajectory clustering to monitor the airspace and its

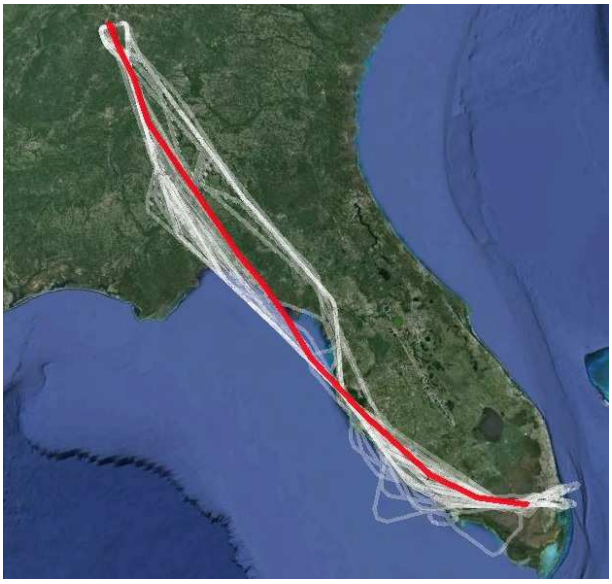


**Figure 5: Representative trajectory**

conformance. Although their approach was an efficient way to determine the compliance of flown trajectories with published trajectories, they focused on the 2-D coordinates of the way points in the (x,y) plane, disregarding the 3rd and higher dimensions of the trajectory points.

Eckstein [11] presented an automated flight taxonomy. With their approach, they first resampled the trajectories and then clustered using k-means on a reduced order model, attained by Principal Component Analysis (PCA). Eckstein used only first two PCs, omitting the others.

Leiden et al. [19] presented two trajectory clustering al-

gorithms for metroplex applications. The first algorithm they developed used a well-known image processing technique, ridge detection. One of the major shortcomings of their first algorithm was that it was unable to identify a backbone route where no ridge existed. Given the shortcomings of their first algorithm, they implemented a second algorithm that was based on k-means clustering. However, they only evaluated their results qualitatively. They never performed a quantitative evaluation using the model itself.

In their effort, Brinton et al. [10] investigated the automated partitioning of airspace into sectors so that sufficient resources could be allocated to meet the demand. They used Dynamic Density metrics and utilized a weighted euclidean distance function to address the constrained clustering problem. However, they never compared their results with a true optimal solution. So, the outcome appeared to be inconclusive.

# 5. CONCLUSION AND FUTURE WORK

This paper presented the *DICLERGE* framework for clustering multidimensional aircraft trajectory data and discovering the representative trajectory upon lateral and vertical smoothing. Our experiments with the real trajectory dataset showed the effectiveness and efficiency of *DICLERGE* framework and its advantages over the regular k-means clustering for the aircraft trajectory data.

Note that this study is just a milestone for trajectory prediction and we plan to utilize our work toward predictive and prescriptive analytics of multi-dimensional aircraft trajectory data. Other future work involves the introduction of a browsing of trajectories capability using techniques such as those used in the SAND Browser for spatial data (e.g., [12, 24]).

# 6. REFERENCES


[1] Aircraft situation display to industry. `http://www.fly.faa.gov/ASDI/asdi.html`. [Online; accessed 10-October-2015].

[2] Google earth. `https://www.google.com/earth//`. [Online; accessed 10-October-2015].

[3] Haversine formula. `http://rosettacode.org/wiki/Haversine_formula`. [Online; accessed 10-October-2015].

[4] Mini batch k-means. `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html`. [Online; accessed 10-October-2015].

[5] Ncep wmo grib2 documentation. `http://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc.shtml`. [Online; accessed 10-October-2015].

[6] Next generation air transportation system, nextgen. `https://www.faa.gov/nextgen/`. [Online; accessed 10-October-2015].

[7] Silhouette coefficient. `http://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient`. [Online; accessed 10-October-2015].

[8] Single european sky aair traffic management research, sesar. `http://www.sesarju.eu/`. [Online; accessed 10-October-2015].

[9] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1999), SIGMOD '99, ACM, pp. 49–60.

[10] Brinton, C., and Pledgie, S. Airspace partitioning using flight clustering and computational geometry. In *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th* (Oct 2008), pp. 3.B.3–1–3.B.3–10.

[11] Eckstein, A. Automated flight track taxonomy for measuring benefits from performance based navigation. In *Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS '09.* (May 2009), pp. 1–12.

[12] Esperança, C., and Samet, H. Experience with SAND/Tcl: a scripting tool for spatial databases. *Journal of Visual Languages and Computing 13*, 2 (2002), 229–255.

[13] Ester, M., peter Kriegel, H., S, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)* (1996).

[14] Gariel, M., Srivastava, A., and Feron, E. Trajectory clustering and an application to airspace monitoring. *Intelligent Transportation Systems, IEEE Transactions on 12*, 4 (Dec 2011), 1511–1524.

[15] Han, B., Liu, L., and Omiecinski, E. Neat: Road network aware trajectory clustering. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on* (June 2012), pp. 142–151.

[16] Hwang, J.-R., Kang, H.-Y., and Li, K.-J. Spatio-temporal similarity analysis between trajectories on road networks. In *Perspectives in Conceptual Modeling*, J. Akoka, S. Liddle, I.-Y. Song, M. Bertolotto, I. Comyn-Wattiau, W.-J. van den Heuvel, M. Kolp, J. Trujillo, C. Kop, and H. Mayr, Eds., vol. 3770 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 280–289.

[17] Jacox, E., and Samet, H. Metric space similarity joins. *ACM Transactions on Database Systems 33*, 2 (2008), 7.

[18] Lee, J.-G., Han, J., and Whang, K.-Y. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2007), SIGMOD '07, ACM, pp. 593–604.

[19] Leiden, K., and Atkins, S. Trajectory clustering for metroplex operations. In *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference* (2011).

[20] Lloyd, S. Least squares quantization in pcm. *IEEE Trans. Inf. Theor. 28*, 2 (Sept. 2006), 129–137.

[21] Nutanong, S., Jacox, E. H., and Samet, H. An incremental Hausdorff distance calculation algorithm. *PVLDB 4*, 8 (2011), 506–517.

[22] Nutanong, S., and Samet, H. Memory-efficient algorithms for spatial network queries. In *Proceedings of the 29th IEEE International Conference on Data Engineering* (Brisbane, Australia, 2013), pp. 649–660.

[23] Roh, G.-P., and Hwang, S.-w. Nncluster: An efficient clustering algorithm for road network trajectories. In *Database Systems for Advanced Applications*, H. Kitagawa, Y. Ishikawa, Q. Li, and C. Watanabe, Eds., vol. 5982 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 47–61.

[24] Samet, H., Alborzi, H., Brabec, F., Esperança, C., Hjaltason, G. R., Morgan, F., and Tanin, E. Use of the SAND spatial browser for digital government applications. *Communications of the ACM 46*, 1 (2003), 63–66.

[25] Sankaranarayanan, J., Alborzi, H., and Samet, H. Efficient query processing on spatial networks. In *Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems* (Bremen, Germany, 2005), pp. 200–209.

[26] Sankaranarayanan, J., Alborzi, H., and Samet, H. Distance join queries on spatial networks. In *Proceedings of the 14th ACM International Symposium on Advances in Geographic Information Systems* (Arlington, VA, 2006), pp. 211–218.

[27] Sankaranarayanan, J., and Samet, H. Distance oracles for spatial networks. In *Proceedings of the 25th IEEE International Conference on Data Engineering* (Shanghai, China, 2009), pp. 652–663.

[28] Sankaranarayanan, J., and Samet, H. Query processing using distance oracles for spatial networks. *IEEE Transactions on Knowledge and Data Engineering 22*, 8 (2010), 1158–1175.

[29] Sankaranarayanan, J., Samet, H., and Alborzi, H. Path oracles for spatial networks. *PVLDB 2*, 1 (2009), 1210–1221.

[30] Sung, C., Feldman, D., and Rus, D. Trajectory clustering for motion prediction. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* (Oct 2012), pp. 1547–1552.

[31] Wang, W., Yang, J., and Muntz, R. R. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1997), VLDB '97, Morgan Kaufmann Publishers Inc., pp. 186–195.

[32] Wang, Y., Han, Q., and Pan, H. A clustering scheme for trajectories in road networks. In *Advanced Technology in Teaching - Proceedings of the 2009 3rd International Conference on Teaching and Computational Science (WTCS 2009)*, Y. Wu, Ed., vol. 117 of *Advances in Intelligent and Soft Computing*. Springer Berlin Heidelberg, 2012, pp. 11–18.

[33] Won, J.-I., Kim, S.-W., Baek, J.-H., and Lee, J. Trajectory clustering in road network environment. In *Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on* (March 2009), pp. 299–305.

[34] Zhang, T., Ramakrishnan, R., and Livny, M. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1996), SIGMOD '96, ACM, pp. 103–114.