Monitoring Airspace Complexity and Determining Contributing Factors

Daniel Weckler[*] and Bryan Matthews [†], Shayan Monadjemi[‡], Shawn Wolfe[§], Nikunj Oza[¶], and Hossein Eskandaripour[‖]
Data Sciences Group, NASA Ames Research Center, Moffett Field, CA 94035, USA

The national airspace has evolved over many years to accommodate increased traffic demand [1] while simultaneously maintaining air travel as one of the safest forms of transportation [2], [3]. One of the reasons for this success is the ability of the air traffic control system and the operators to adapt and accommodate to situations that routinely disrupt normal operations. These situations may include: adverse weather, delays, early arrivals, equipment outages, and other factors that are outside the operators' ability to control. These factors can lead to states where automation is unable to properly handle these issues, and therefore air traffic controllers and pilots have to intervene — ultimately increasing communication between operators resulting in higher workload. As controller workload increases to handle sub-optimal operating conditions, complexity increases. This is because, under these conditions humans are required to make tactical decisions in response to external factors. This results in a departure from the original strategic plan where operations would be more efficiently managed. Human operators manage airspace complexity under rigid regulations but in a constantly changing environment. The airspace is divided into sectors and the number of aircraft assigned to each controller is limited for safe handling. Some prior studies devised airspace complexity metrics in commercial aviation and related these metrics to controller workload (e.g., [4],[5]). The upper bounds on the system load are pre-determined. Such bounds on complexity make for a safe system, but the system cannot scale and adapt to autonomous, dense, and heterogeneous traffic — including the many types of Unmanned Aerial Vehicles (UAVs) envisioned to be added to the operations. We hypothesize that, as traffic density and heterogeneity grow, and other key metrics change, there will be phase transitions at which the way traffic should be managed changes significantly [6]. We offer a method for in-time detection of contributing factors that lead to phase transitions, characterized by increased complexity. To the best of our knowledge, there is no tool similar to ours that identifies such contributing factors or precursor patterns.

I. Introduction

Air traffic control is a critical component in maintaining a safe and efficient National Airspace. Controllers need to balance airline schedules while maintaining aircraft separation, avoiding convective weather, and minimizing disruptions to the traffic flow. As traffic flow management has shifted to using more precise and advanced automation to handle the control of an aircraft on a route, procedures such as RNAV STARs [7] have become predominately utilized to control commercial aircraft in the terminal airspace. These procedures allow the controller to issue a clearance on the arrival with the knowledge that the aircraft and crew will be complying with the lateral and vertical paths within the requirements of the procedure. This removes the need for them having to issue clearances while the flight is on the route. This gives controllers flexibility; allowing them to manage workloads more efficiently by monitoring the flights on these procedures instead of constantly issuing new turns, speeds, and altitude clearances to each aircraft. However, when there is a disruption to the system, these procedures can break down — requiring controller intervention to manage the flights with tactical vectoring commands. A study in 2018 showed that while most flights do not always fly the full STAR procedure to completion [8], the majority do adhere to the fixes within the common route of the procedure. A human factors study performed in 2019 [9] measured controller workload within the terminal airspace. The results

[*]Research Engineer, KBR, Inc.
[†]Senior Research Engineer, KBR, Inc.
[‡]Pathways Intern, NASA Ames Research Center
[§]Computer Scientist, NASA Ames Research Center.
[¶]Computer Scientist, NASA Ames Research Center.
[‖]Intern, USRA, NASA Ames Research Center.

of this study found that the tasks that contributed the most to high workload during disruptive operations were vector turning commands. In the context of this work, we are proposing that airspace complexity is a proxy for controller workload and can be indirectly measured by observing flights that deviate from normal expected paths. Furthermore, as emerging operations begin to integrate into the current system, although heavy reliance on automation can reduce controller workload, the situations where automation is no longer able to fully manage the system can quickly complicate operations for a controller. Understanding these factors is important to help design graceful degradation strategies into future automation so as to not overwhelm air traffic control when automation is not fully in effect.

To define the scope of this work we are proposing to measure complexity from the viewpoint of the Terminal Radar Approach Control Facilities (TRACON) controller's perspective. In particular, we are analyzing arrivals into KSFO. Factors such as air traffic density and airspace heterogeneity play a significant role in controller workload. These can therefore be assumed to be a proxy for complexity. We define airspace heterogeneity as how the arrival flights are distributed across the STARs. For example, a homogeneous airspace would only have flights sequenced on the same STAR route, while a completely heterogeneous airspace would have flights evenly distributed across all potential STARs. With safety as the top concern for airspace operators, it is important to recognize that, as density and heterogeneity grow, the focus of the system will change. During times of the day when the airspace has low density and heterogeneity, the flights will follow more efficient paths. On these paths, the aircraft move on established routes that are more or less directly to the destination. However, as density and heterogeneity increase, the system will begin changing focus to avoiding conflicts and prevent loss of standard separation. To accomplish this, controllers will route the flights in a more flexible way. Higher flexibility requires more communication and coordination between controllers and pilots which the current automation is unable to handle. This paper proposes a novel approach that monitors airspace complexity at multiple scales, uses a machine learning-based tool that predicts when operations will transition to a regime of greater complexity, and identifies actions that can reduce the complexity while still maintaining efficient and safe operations.

A. Related work

Previous research has looked to quantify airspace complexity. In Delahaye, et al.[10] the authors propose using a non-linear dynamical system model to construct a vector field based on historical flight tracks. This model describes the progression of state variables within the system and deviations from the expected system behavior indicates complexity. Another approach based on trajectory clustering was developed by Gariel, et al.[11]. This unsupervised learning technique consists of the following steps: (1) identify the general maneuvering areas (waypoints) by performing K-means or DBSCAN clustering on locations where aircraft frequently turn based on the surveillance radar track data, (2) map flight trajectories onto sequences of waypoints, and (3) cluster the sequences based on their common subsequences. Using the identified clusters the method measures complexity from the probability of an aircraft being a member of a nominal cluster. The resulting complexity measure is the sum of the entropy of aircraft inbound to an airport and the entropy of aircraft not inbound to the airport. From a high-level perspective, this model learns nominal operations in the airspace through the sequence of waypoints that are representative of where aircraft change direction and defines deviations from the nominal operations as *complex*. Although both these methods attempt to characterize the behavior of flights with respect to the historical data, they do not consider domain knowledge of procedural routes such as the STARs to quantify deviations from those expected paths. When a flight is following an RNAV STAR the controller has reduced workload and is primarily functioning as a system monitor even when flights are turning at fixes in the procedure. However, when a flight is required to deviate from an expected path, the role of the controller changes to an active manager with increased workload duties. This key piece of information is critical to consider when computing the complexity of the airspace.

II. Method

A. Dataset

We demonstrate our proposed approach using data from multiple complementary sources. This includes, but is not limited to: historical aircraft surveillance data from NASA's Sherlock Data Warehouse [12], METAR weather data, and airport configuration data from Aviation System Performance Metrics (ASPM) from the year 2019. The surveillance data includes both parameterized data such as Origin/Destination airports, top of climb, landing runway, flight plans, etc., and time series data describing each aircraft's flight path. These flight paths are sampled at a variable sample rate — increasing as the aircraft approaches the airport. This is due to how Sherlock manages flight track stitching between different radar facilities which have different sampling rates. The METAR weather data includes data such as visibility, temperature, and cloud covering. The performance data (ASPM) includes data such as the arrival rate of aircraft, meteorological conditions, and runway configurations. Both the weather and performance data are logged at defined intervals throughout the day at a coarser refresh rate than the surveillance data.

B. Star Routes

In addition to the logged data and metrics, we leverage pre-defined Standard Terminal Arrival Routes (STARs) procedures to characterize the path of each flight. Each flight files for one of these routes in the flight plan well before entering the terminal airspace. It approximately follows the route until it leaves the STAR — typically on the final fix of a STAR's runway transition. While most flights do not always fly the full STAR procedure to completion [8], the majority do adhere to the fixes within the common route of the procedure. Our approach leverages fixes in the common route of each of the STARs to build a reference path to the airport. This allows us to characterize the flight paths in what we are defining as the *maneuvering area*, which is where the airspace that connects the STARs to the final runway fix resides. Characterizing flights within the maneuvering area allows us to determine how off nominal the flights are, and in turn, calculate complexity score of the airspace.

Since unique parameters of STARs and runways are used to characterize the maneuvering area, it varies for each STAR/Runway pair. Most of the air traffic control work happens in this area. As such, the majority of our analysis happens within it. Portions of flight paths outside their respective maneuvering area (or flights outside it entirely) are not considered in this airspace analysis. Overflights and departures are also not included in the scope of this analysis. Not only is the maneuvering area different between STAR routes, it also depends on which runways the airport is using at that time. While the STAR fix for the maneuvering area remains constant, the final approach fix differs for each runway. For example, if an airport has four different parallel runways a flight could land on, then each STAR fix will have four different maneuvering areas. In this paper, we will be looking at arrivals into SFO, which have four possible parallel runway configurations. Each set of parallel runways have different final approach fixes however, since they are very close to each other, we will be considering both the left and right runway fixes as the same point.

Being able to define a maneuvering area for each flight is critical for our analysis. However, the surveillance data contains many flights that are incorrectly labeled (i.e. the flight flew a different STAR route than they initially filed for) or labelled "Unknown." To account for this, we defined a 2.25 nautical mile tolerance boundary across a 10 nautical mile slice of the circle (Fig.1) centered at the second fix on the common route for the STAR (for BDEGA3 this was the BDEGA fix). The rationale for selecting the second fix in the common route was to ensure that the flight began with the intention of flying the route for at least two consecutive fixes. While this was successful in correcting some STARs and Unknowns, there were still many unknowns that were unlabeled via this method. Upon taking a wider look at the map, it appeared that the vast majority of unknowns were coming from regional airports. These flights fly low enough not to need to be routed by the TRACON on a STAR route. Despite this, they add to complexity of the airspace (since they need to be merged with other arrivals in the region). Our solution was to create artificial STAR routes based on the airport they came from. To do this, we draw straight lines between SFO and each respective airport, and we define the new "pseudo STAR fix" as the point where it crosses the DYAMD4/Runway-28LR boundary (which is the furthest radial circle for KSFO). The vast majority of the Unknowns came from the Fresno airport (KFAT) and the Sacramento airport (KSMF). We denote these as KFAT|| and KSMF|| respectively. From here, we use them in our analyses just like we would any other STAR route.

C. Star Entropy

The first step we took to characterizing complexity was to measure the entropy of all the STARs and flights in the airspace. The formulas we used to determine entropy are described in Equations 1 & 2, where N is the total number of STARs, S is a particular STAR and F is the count of flights as a function of the STAR S. Equation 1 describes the proportions P_i of flights for each STAR given the total number of flights in the airspace during a 15 min time window. We use Python's entropy function found in Scipy's stats package to compute the entropy from the STAR proportions P_i.

$$P_i = \frac{F(S_i)}{\sum_{j=0}^{N} F(S_j)} \tag{1}$$

$$\mathrm{E} = -N \sum_{i=0}^{N} (P_i * log(P_i)) \tag{2}$$

In short, the more flights there were in the airspace spread across different STAR routes, the higher the entropy value would be. While this metric isn't entirely defined by the number of flights in the airspace, it is heavily influenced by it. In researching this topic we met with a former TRACON controller that emphasised that complexity should not be primarily driven by the number of flights, but by the nature of the operations. Because of this, we opted to make entropy a component used for predicting complexity that captures this unique operational behavior rather than the complexity metric itself.

D. Offset from path

Since more specialized vectoring maneuvers result in a higher workload, we start with a theory: each flight has a direct path it takes from the STAR's common route to the final approach's outer marker fix for the flight's landing runway. Fig 1 illustrates this radius of the BDEGA3/Runway-19LR maneuvering area. We measure the deviations of each flight from this path and create a distribution from these deviations. These distributions were created for each flight based on their *max offset distance* and *average offset distance*. We then use these distributions to characterize which flights to select for our nominal behavior.

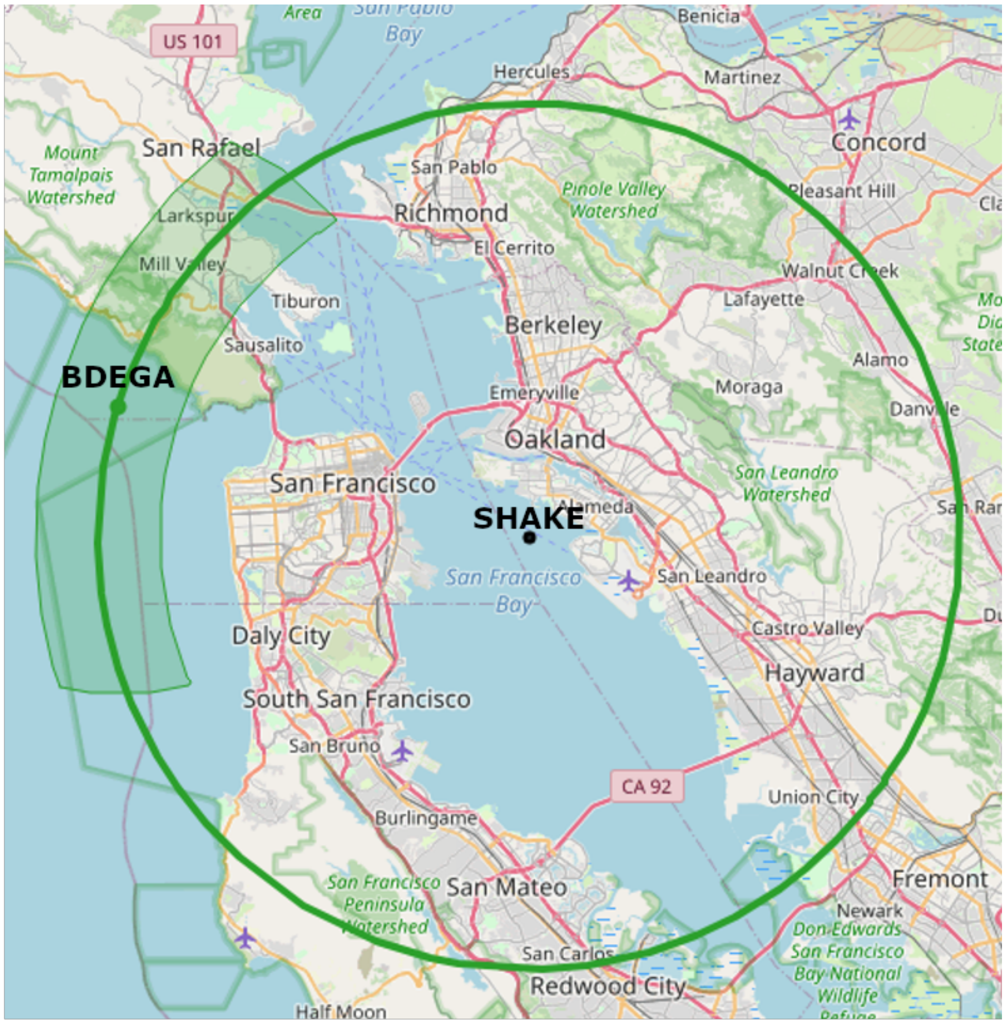


Fig. 1 STAR radius for the BDEGA3/Runway-19LR maneuvering area.

E. Representative Path

While using the direct path distributions to characterize the complexity score is an option, these distributions can be more noisy. A cleaner method would be to create a *representative path* to use for modeling the distributions instead. To create this, we define a *representative flight path* for each STAR route and runway pair. This flight is approximately the path the flight would take if there was a clear path with no other flights in the airspace. To create this path, we first identify the flights for a given STAR-runway pair using the distributions already created for the direct path. From these distributions, we select the flights that fall between the 44th-55th percentiles. This yields flights that conform to the most normal mode of operation. Each of these flights is partitioned based on the percent complete from the entry point into the maneuvering areas from 0% – 100% complete.

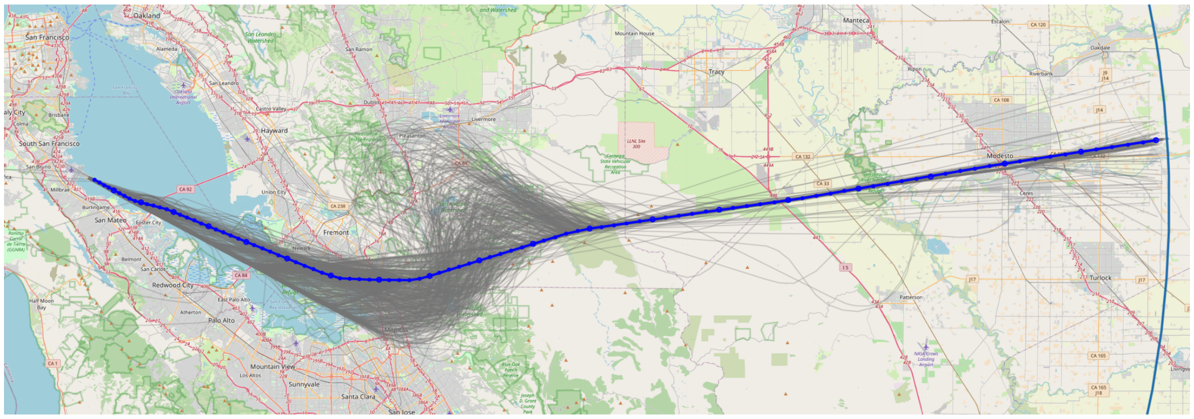


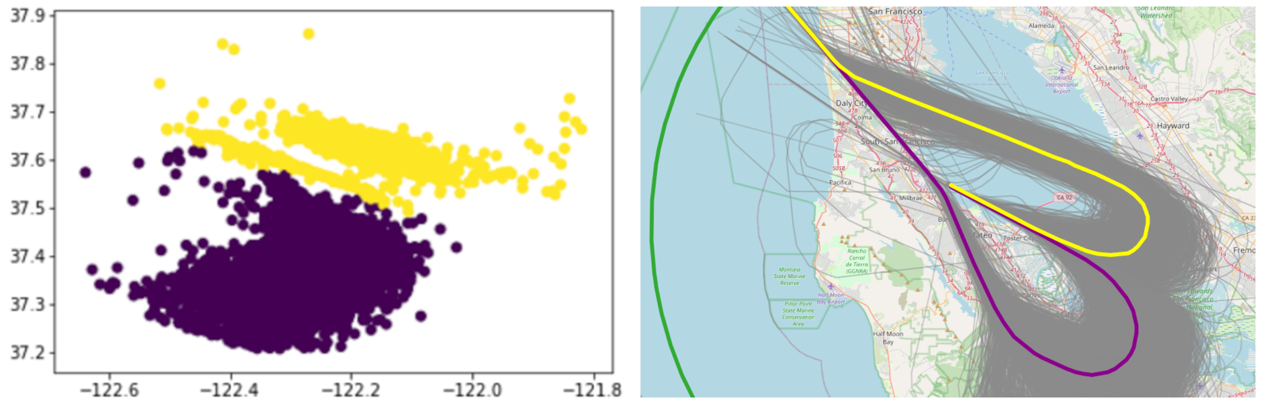
Fig. 2 Sample representative path for the DYAMD4/Runway-28LR

These percentages are defined by the amount of remaining time the flight has to reach the airport. For each percent bin, we take the median value of the flight's latitude/longitude coordinates, airspeed, and time remaining to the airport to construct a lookup table for each percent complete bin on a given route. Fig. 2 illustrates the representative path of a flow for the DYAMD4 STAR. As a flight enters the maneuvering area, we can use this path to find the estimated travel path, deviation from this path, and estimated arrival time of a flight to the airport. This is done by taking the flight's current position and finding the closest point on the representative path. From here, we can measure the flight's deviation from the representative path and create our distributions this way (much like we did for the direct path). There were some lesser-flown flight paths that did not have enough flights to create a smooth representative path. For these star-runway combinations, we used the direct path to approximate the distribution. In addition to this, some routes (such as BDEGA3 and STLER4) take two different paths to reach the airport. These deviations in routes and paths (once flights have left the STAR) do not appear to be documented anywhere in the surveillance data. To account for this, we used k-means clustering to attempt to separate the two paths. We took each flight's average position between 40-50% complete and applied the clustering algorithm. This is illustrated in Fig. 3a. This gave us a relatively clean separation and we were able to create a representative path for both the lower and the higher paths (Fig. 3b). To determine the offset distance for a flight on either of these split paths, we calculate it for both paths and take the minimum offset value as our distance.

F. Conflicts

Airspace conflicts are important feature in predicting complexity. We define conflicts as when two or more flights are projected to arrive at around the same time. For the purpose of this project, we used a window of 2 minutes. This situation can occur for a multitude of reasons (early or delayed flights, deviations due to weather, reduced runway capacity, etc.). If a conflict arises, preventative measures must be taken so that the corresponding flights arrive at different times. These measures include maneuvers such as S-turns and holding patterns that elevate the air traffic control workload, thereby increasing complexity. This increase in complexity is then captured by larger offset distances from the representative path. Because of this, we hypothesize it is possible to use these conflicts to predict a rise in complexity.

To calculate conflicts at a given point in time, we use the representative path to estimate the arrival times of each flight. Since we capture the median time it takes for the flights to land at each percent flown bin along their respective representative path we can use this estimate to predict the landing time for every flight in the maneuvering area. We map the current position of the flight to its respective bin on the representative path. This gives us the percent complete of the flight, which also gives us an estimate of the corresponding remaining time left on the path as described in Section



(a) k-means Cluster

(b) Dual Representative Paths

Fig. 3 Using a clustering approach to create 2 representative paths for BDEGA

II.E. After this point, we assume that the flight will follow the representative path to completion when deriving these estimates.

We can then compare these estimated arrival times against other flights for the same snapshot in time to identify potential conflicts. If more flights are estimated to arrive within our 2 minute tolerance window than there are runways available (for the airport's current runway configuration), then we have a potential conflict. We can use this derived measure along with other factors expected to contribute to disrupting the operations (such as weather and runway configuration changes) as an input to machine learning tools to detect precursors that increase complexity. This novel method will assist in uncovering insights into the contributing factors that lead to increased complexity that may allow for in-time responses to avoid reaching a high complexity state in the airspace.

G. Complexity Score Methodology: Creating Our Distributions

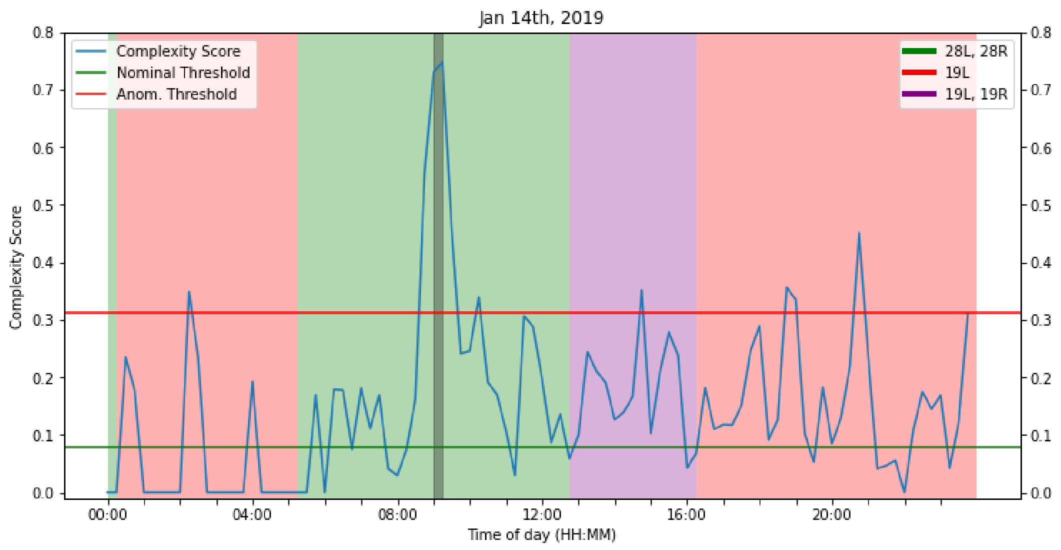


Fig. 4 Complexity score over one day. Runway configurations are highlighted in the background. The shaded area references the airspace in Fig. 5

It is important to note that the offset distances are only used as a reference. If the majority of the flights have a large, but consistent offset as compared to other routes, it does not necessarily mean that those flights have higher complexity. To account for this, we build a distribution based on the offset distances from a path for each STAR and runway pair.

This allows us to determine the normal mode of operations for that route. Flights that are in the upper tail of these distributions will result in higher complexity scores, while flights that fly in the median or below will represent the normal mode of operations and therefore will have lower complexity scores.

Some STAR/runway pairs were used infrequently and therefore were underrepresented in our data set. As a result, the observed empirical representation is not likely to be an accurate representation of the true latent distribution. To address this shortcoming, we used a hierarchical Bayesian modeling approach to increase the commonality among STAR/runway distance distributions by drawing their distributional parameters from a shared distribution. This allows us to have low data pairs less influenced by their few data points, and adhere more to the general trends observed across STAR/runway pairs. The more voluminous STAR/runway pairs often appeared to come from a gamma distribution, but with an additional mode. As a result, we modeled each STAR/runway pair as mix of two gamma distributions, a primary and secondary. The mean and variance of each of these gamma distribution was drawn from higher-level gamma distributions (for a total of four), which themselves were allowed to vary. These higher level distributions provided the mechanism for the high volume STAR/runway pairs to influence those with little data, and to introduce some regularity across the distributions. The mean and variance of these higher distributions were themselves drawn from fixed gamma distributions. As is often the case, we used the Metropolis-Hastings algorithm to find appropriate values for the various parameters.

Since flights following each STAR route take different paths to the airport, each STAR route has a different distribution, and we therefore can model these distributions to compute a complexity score from their respective normalized distributions. Fig. 4 shows an example complexity score over an entire day. The dark highlighted region indicates the snapshot in time where the complexity score is high. The corresponding flight track paths during the same 15 minute interval are shown in Fig. 5.

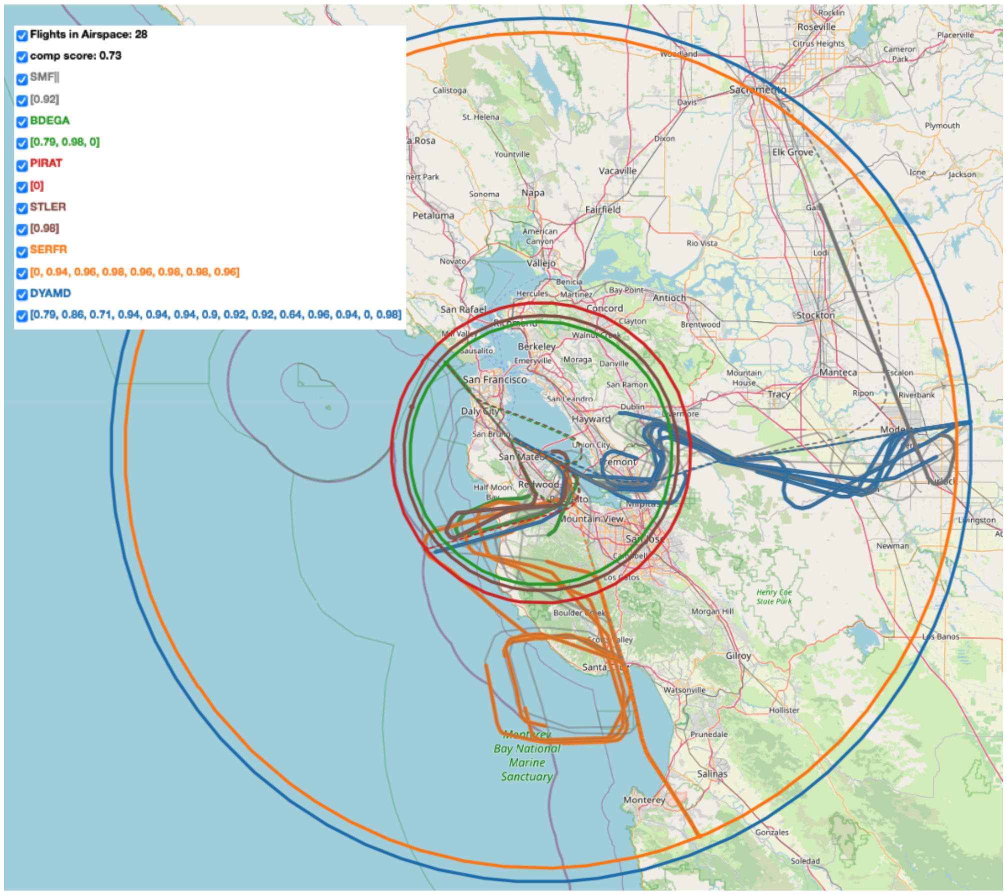


Fig. 5 Snapshot of flight tracks during the high complexity period.

The complexity score for an individual flight is calculated as follows: for each star-runway pair, a distribution using the offset from the representative path method is created. This distribution is then split up into percentiles. From here, a flight's position is measured, and the offset distance from the representative path is calculated. We then locate the position of this offset in the distribution and determine in which percentile it lies. This percentile is then used as the complexity score. To reduce the impact of low-complexity flights on the overall complexity score, we zero out any percentiles below 0.68 (i.e. any score below 0.68 is considered as fully nominal). In addition to this, we square the resulting complexity score to increase the spread between nominal and high complex periods. This results in the minimum complexity score a flight contributes to complexity to be around 0.462. To calculate the overall complexity score, we take the individual complexity scores of each flight for a time window and average them. There are sometimes situations where flights significantly deviate from the representative path, but the flights in the airspace is low. This is an inherently low complexity situation, but would average out to a high complexity score nonetheless. To reduce the impact of windows with low numbers of flights in airspace, we add a weight of two "zero" flights to the average (i.e. the average is treated as if there were already two perfectly nominal flights in addition to the other flights being averaged). The window with which we chose to calculate this complexity score is every 15 minutes (this aligns with the frequency at which the ASPM data is recorded). Fig. 6 illustrates the data processing pipeline through which the complexity score is computed.

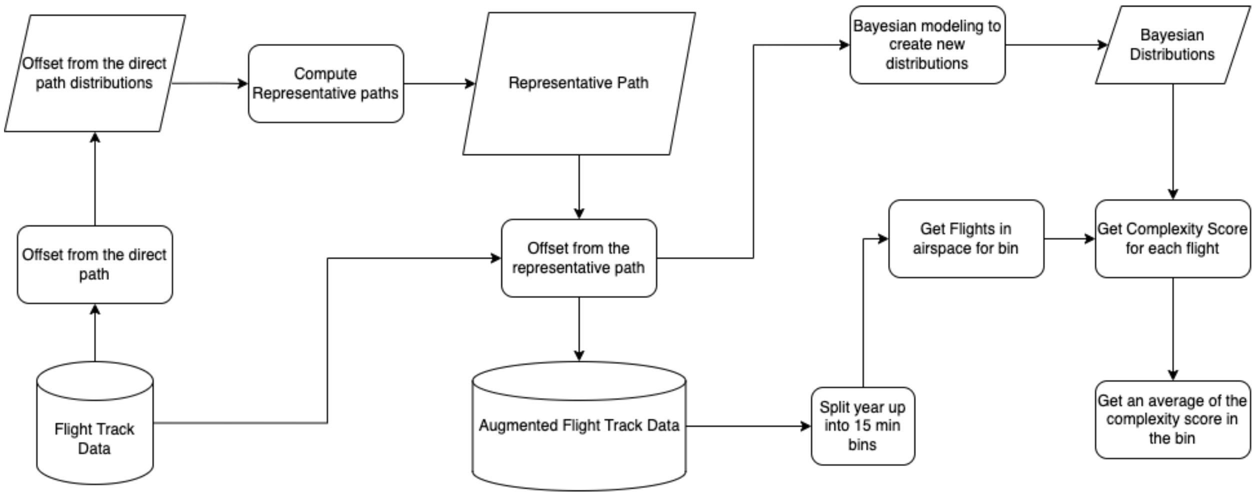

Offset from the direct path distributions
Compute Representative paths
Representative Path
Bayesian modeling to create new distributions
Bayesian Distributions
Offset from the direct path
Offset from the representative path
Get Flights in airspace for bin
Get Complexity Score for each flight
Flight Track Data
Augmented Flight Track Data
Split year up into 15 min bins
Get an average of the complexity score in the bin

Fig. 6 Data Processing Pipeline

H. Clustering Maneuvering Points: Measure α

To evaluate the effectiveness of our proposed airspace complexity metric, we will compare it against a baseline we developed inspired by prior work [11]. From a high-level perspective, this baseline model assumes that nominal flights have similar maneuvering positions in the airspace, whereas anomalous flights may exhibit uncommon maneuvering patterns. To implement this model, we selected a random set of 1000 flights from 2019 and identified geographic positions at which they had bank angles above a given threshold. Bank angle can be estimated from flight track data using the geometric formula shown in Equation 3. The formula uses the ground speed of the aircraft: V, the radius of the turn: R, and the gravitational constant: g. These values can be derived between consecutive radar recordings.

$$\Theta = \tan^{-1}\left(\frac{V^2}{gR}\right) \tag{3}$$

We refer to these points as *maneuvering points*. Then, we applied an unsupervised learning algorithm, HDBSCAN, to the set of *maneuvering points* to find clusters of common *maneuvering points*. We refer to these clusters of common *maneuvering points* as *waypoints*. Once trained, our model could consider the position of aircrafts while turning in real-time and label them as either one of the established waypoints, or detect them as an anomaly. We define a flight to be an *anomalous flight* if its most recent maneuver was not through an established waypoint. Moreover, we quantify airspace complexity at a given time as:

$$\text{complexity} = \frac{\text{\# of anomalous flights}}{\text{\# of total active flights}} \tag{4}$$

We visually validated our model by eliciting domain expert feedback, and deployed it to compute the airspace complexity metric for all of 2019 (at 30-second intervals).

I. Equating complexity with likelihood: Measure β

As an alternative to the above approach, we developed a simple probabilistic measure of complexity. The waypoints described above provide a discrete representation of a flight as a sequence of transitions through waypoints. Some transitions happen frequently, but some are rare, and for this method we view flights that make rare transitions as increasing the complexity, as this indicates some unusual situation.

For each flight, we used the observed transitions from all other flights as frequency of transitions. In addition, we assumed that each waypoint had an additional transition to every other waypoint, so that all transitions were represented. For example, assume that we have 31 waypoints, a given flight was observed to transit from waypoint A to waypoint B, and that in all other flights, there were 69 observed transitions from waypoint A, 9 of which were to waypoint B. In this case, we calculate the probability of transiting to waypoint B given a location of waypoint A as (9 + 1)/(69 + 31) = 0.1.

From these simple base probabilities, the overall likelihood of a flight was calculated as the product of the observed probabilities of all the flight's waypoint transitions. Likewise, the overall likelihood of the airspace at any given time is the product of the likelihood of all airborne flights. This yields a number from 0 to 1 (as it is a probability) which in practice is often nearly zero. We defined the complexity as the negative log of this likelihood, so that increasing scores indicate greater complexity. We further define the complexity to be zero when no flights are airborne.

We characterized the unusualness of a given flight by comparing it to other flights using the same STAR and runway, specifically, measuring where its distance from the representative path falls on a distribution of such distances. A straightforward approach would be to simply calculate the fraction of flights with a lesser offset. However, some STAR/runway pairs were rarely encountered, and so the observed distribution may very well be quite different from the true latent distribution, given the small sample size.

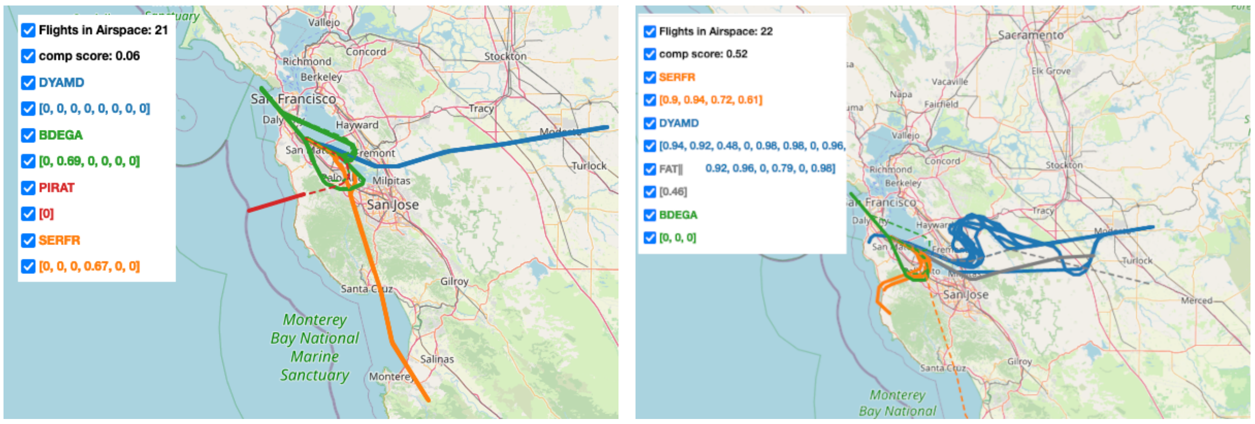
J. Precursor Analysis

The ultimate goal is to be able to predict an increase in complexity using an existing precursor detection tool: ADOPT[13]. At the core of ADOPT is the Deep Temporal Multiple Instance Learning algorithm, or DT-MIL. This algorithm combines multiple instance learning with deep recurrent neural networks to be used with time series or sequential data. DT-MIL is used to identify and explain precursors to anomalous events from time series data. To do this, we need to provide the model with a number of features to use in the prediction. For features we synthesized, the aforementioned Star Entropy, Conflicts, Measure α, and Measure β scores were used. For derived parameters, we included the ratio of flights off the representative path vs. the number of flights in the airspace, as well as the number of flights on the representative path. For raw parameters, we included the visibility, available landing runways, if there was a recent switch, flights in airspace, the temperature (in Celsius), the configuration of the arrivals, and whether or not there was a go-around event within the last 15 minutes. Since varied data sources naturally contain many different sample rates, we resample our data to 30 second intervals. All higher sample rates are downsampled, while lower sample rates (such as the complexity score itself) use "hold interpolation" (i.e. holding at a current value until it changes instead of traditional interpolation). We then take a 1-hour time window (henceforth referred to as the "prior" window) of these features to use to predict our "anomalous" event window — a 15 minute window where the complexity score is above our designated threshold of 0.312, which corresponds to the 85th percentile of complexity scores throughout the year. During the prior window, we ensure the complexity score does not exceed the anomalous score threshold. We also ensure the anomalous event window has at least five flights in its airspace to ensure enough flights are present to create a complex airspace. To ensure suitable separation, we select nominal "event" windows with a threshold no greater than 0.079 which corresponding to the 35th percentile of the complexity scores through the year. For the prior windows, we ensure the complexity score does not exceed the high complexity threshold as well. We then run ADOPT with these features before performing a feature ranking analysis after.

III. Results and Discussion

A. Complexity Metric Results

We computed the complexity measure over the year 2019 using our proposed technique. During low complexity times, the vast majority of flights can be found to closely follow their representative paths (see a sample snapshot



(a) Low complexity

(b) High complexity

Fig. 7 Comparison of high and low complexity flight track behavior for arrivals into SFO with their corresponding STAR routes.

Table 1 Confusion Matrix for ADOPT Analysis

		True Class		
		Nominal	Complex	Total
Predicted Class	Nominal	68	10	78
	Complex	11	67	78
	Total	78	78	156

from the airspace in Fig. 7a). While slight deviations can be observed in some flights, generally these deviations are considered nominal. There were also individual flights that exhibited large deviations (or even go-arounds at times), however they did not impact the paths of the other flights in the window and thus the overall complexity was still low. The high complexity scenarios are visually apparent as seen in a sample snapshot of the airspace in Fig. 7b. Many deviations, holding patterns, and go-arounds are present during various anomalous samples.

B. Model Results

We performed an ADOPT analysis on the dataset described in Section II.J. With 832 training samples, 556 validation samples, and 156 test samples, we achieved an AUC Train score of 0.92 and an AUC Test score of 0.91. We also achieved a precision of 0.87, a recall of 0.86, and an F1 score of 0.86 on the test set. This yielded a confusion matrix shown in Table 1:

Following this, we used ADOPT's feature ranking tool to analyze each feature and found the Off Path Ratio, STAR Entropy score, Measure α, and Number of Conflicts to be the most contributing parameters — with the Off Path Ratio being the overwhelming contributing factor in this analysis. Since multiple groups of factors could cause any of these high complexity scenarios, we used k-means to find different clusters between these feature rankings.

After performing k-means clustering on the features contributions for each precursor event we found 9 clusters. The normalized contributions were averaged for each cluster and the contributions that account for up to 90% of the total feature importance were identified. Table 2 list the identified features for each cluster as well as their overall percent impact for these clusters (note: the parameters listed are not exhaustive. They represent the parameters that were apparent in our clusters). The check marks indicates which top ranked corresponding features belongs to the each cluster. All clusters contained Off Path Ratio, STAR entropy, and cluster maneuvering as important features for prediction. It is not surprising that the Off Path Ratio is a strong predictor because it is looking at deviations from the representative path. This is analogous to the complexity measure but differs slightly with a more strict thresholds

Table 2 Top contributing precursor factor by cluster.

Cluster ID	Cluster Size	Off Path Ratio	STAR Entropy	Number of Conflicts	Measure α	Measure β	Go-Around	Runway Config	Visibility
0	53	✓	✓	✓	✓	✓			
1	92	✓	✓	✓	✓		✓		
2	91	✓	✓		✓				
3	114	✓	✓	✓	✓		✓		
4	125	✓	✓		✓				
5	125	✓	✓	✓	✓				
6	81	✓	✓	✓	✓		✓		
7	6	✓	✓		✓		✓	✓	
8	8	✓	✓	✓	✓				✓
Overall % impact	-	55.3%	12.7%	6.8%	7.7%	4.1%	4.5%	0.2%	0.4%

Table 3 Cluster groups based on similar feature contributions

Cluster Groups	
Group Description	**Cluster ID**
Flights are beginning to be vectored off the arrivals with increased arrival conflicts	{0,5}
Many flights already off path	{2,4}
Go-arounds disrupting the following arrivals	{1,3,6}
Changes in airport runway configuration	{7}
Poor Weather	{8}

and has no percentile magnitude information that the complexity measure measures. STAR entropy matches with our earlier hypothesis that the heterogeneity of the arrivals is an important factor in leading up to complexity. The cluster maneuvering metric is also a key feature for capturing the behavior of flights in the airspace and how they deviate from expected paths without utilizing the STAR assignments. The feature grouping were examined and clusters with similar contributing factors were manually grouped together. Table 3 lists the clusters that were found to have similar features of importance with high level descriptions of the cluster categories. The discovery of these cluster categories show how the precursor algorithm has the potential to identify various types of unique precursors that lead to high complexity.

IV. Conclusion and Future Work

We demonstrated a novel method for computing airspace complexity that combines domain knowledge from STAR paths with real flight data that represents the nominal mode of operations that may differ from the published STAR procedures. We have also derived useful parameters and metrics to assist in predicting high complex operations with high accuracy and discovered unique categories of precursor.

There are quite a few things that would be useful to refine. One thing to improve would be finding a way to handle STAR-runway pairs with low amounts of available data. Several pairs did not have enough flights throughout the year to be able to construct a smooth representative path. Augmenting this data with past or future flights would be one option to correct this. Another option would be to use a domain expert to manually draw a "best fit" representative path among the limited data. Limited flights also impacted the distributions for the complexity score. While the Bayesian distribution modelling ended up alleviating some of these issues, data augmentation in this case would help as well. Using clustering algorithms to see if these low-data distributions could be mapped to a bigger distribution would be an option as well. Another issue to tackle would be the robustness of the complexity score. The current iteration of the complexity score is still somewhat vulnerable to having many small deviations add up and falsely flag a low complexity window as complex — this is more often happening when there are a lower amount of flights in the airspace. Finding a different statistical algorithm (aside from a raw average with some weights) to calculate an overall score for a window would alleviate this. The prior issues described also cause the middle complexity score range (in between the nominal an anomalous thresholds) to be somewhat ambiguous. Solving the prior issues would likely make this less impactful — allowing for better separation and performance of the model.

V. Acknowledgments

This work was supported by NASA Ames Center Innovation Fund. We would also like to thank former TRACON controller Mike Santos for his insightful discussions on airspace complexity.

References

[1] National-Transportation-Safety-Board, "Annual Summaries of US Civil Aviation Accidents," , 2019. URL `https://www.ntsb.gov/investigations/data/Documents/AviationAccidentStatistics_1999-2018_20191101.xlsx`.

[2] National-Transportation-Safety-Board, "US Transportation Fatality Statistics," , 2017. URL `https://www.ntsb.gov/investigations/data/Pages/AviationDataStats2017.aspx`.

[3] Savage, I., "Comparing the fatality risks in United States transportation across modes and over time," *Research in Transportation Economics 2013,43,9–22*, 2013. https://doi.org/10.1016/j.retrec.2012.12.011.

[4] Keumjin Lee, E. F., and Pritchett, A., "Describing Airspace Complexity: Airspace Response to Disturbances," *Journal of Guidance, Control, and Dynamics, 32(1)*, Vol. 32, 2009.

[5] Anthony J. Masalonis, M. B. C., and Wanke, C. R., "Dynamic Density and Complexity Metrics for Realtime Traffic Flow Management," *MITRE Technical Report*, 2003.

[6] Alexandrov, N., "Control of Future Air Traffic Systems via Complexity Bound Management," *AIAA-2013-4419, ATIO Conference*, 2013.

[7] United States, "Aeronautical Information Manual," *Far/AIM: Federal Aviation Regulations*, Aviation Supplies Academics, Inc, Ashland, 2022, Chap. 5.

[8] Stewart, M., Matthews, B., Janakiraman, V. M., and Avrekh, I., "Variables Influencing RNAV STAR Adherence," 2018, pp. 1–10. https://doi.org/10.1109/DASC.2018.8569220.

[9] Lindsay, K., and Sutton, F., “Workload Assessment of Terminal Radar Approach Control Performance,” 2019. https://doi.org/10.2514/6.2019-0985.

[10] Delahaye, D., Puechmorel, S., Hansman, R., and Histon, J., “Air Traffic Complexity Based on Non linear Dynamical Systems,” 2003.

[11] Gariel, M., Srivastava, A. N., and Feron, E., “Trajectory Clustering and an Application to Airspace Monitoring,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, No. 4, 2011, pp. 1511–1524. https://doi.org/10.1109/TITS.2011.2160628.

[12] Eshow, M. M., Lui, M., and Ranjan, S., “Architecture and capabilities of a data warehouse for ATM research,” *2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*, 2014, pp. 1E3–1–1E3–14. https://doi.org/10.1109/DASC.2014.6979418.

[13] Janakiraman, V. M., “Explaining Aviation Safety Incidents Using Deep Temporal Multiple Instance Learning,” *KDD ’18: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 406–415. https://doi.org/10.1145/3219819.3219871.