Phase of Flight and Rule of Flight Calculator

Using National Offload Program Data

Charles Johnson
System Safety Section, ANG-E272
Federal Aviation Administration
William J. Hughes Technical Center, USA
charles.c.johnson@faa.gov

Adrian Rusu, Anthony Breitzman[§], John Bucknam, Nicholas LaPosta
Department of Computer Science, [§]Department of Mathematics
Rowan University
Glassboro, New Jersey, USA
{rusu, breitzman}@rowan.edu,
{bucknamj8,lapost48}@students.rowan.edu

***Abstract*— Calculating Phase of Flight is an analysis that could be widely used when determining patterns in air traffic or causes of airline accidents. However, currently there exists no way to determine Phase of Flight without being explicitly sent the data from the aircraft. The problem is that, typically, the FAA only retains access to surveillance data such as primary and secondary radar data from sources such as the National Offload Program (NOP). This paper discusses a preliminary statistics-driven classification system which only classifies basic phases of flight but could be expandable to classify a larger set of phases of flight for different aircraft types. It also discusses an approach to classify flights based on rules of flight (i.e. Visual Flight Rules (VFR) or Instrument Flight Rules (IFR)).**

Keywords-component; flight, phase, rules

I. INTRODUCTION

The Federal Aviation Administration (FAA) collects a large amount of data about aircraft/rotorcraft flying in the National Airspace via surveillance data sources (i.e. primary radar, secondary radar, airport surveillance, etc.). Typically this involves most of the automation systems that allow Air Traffic Control to track and efficiently separate aircraft. Over time, these systems have evolved and there now exists several different systems that provide tracking of aircraft in different environments (i.e. terminal, en route, surface, etc.). Today, the FAA collects data from each of these systems in order to monitor the efficiency and safety of the National Airspace System (NAS). While this permits analysis of simple metrics like overflights, airport arrivals and departures, it does not inform a safety analyst as to the level of detail about a flight at any given instance. It is the intent of the FAA to develop capabilities that allow for an integrated safety picture of the flight at any given time or point in space (i.e. from takeoff through departure, en route cruise, approach, and landing).

To begin to appreciate what data about a flight could mean to the average safety analyst, consider the following scenario faced by an FAA analyst investigating an accident or incident, or one who wishes to track metrics regarding any given flight. Ideally, he/she would want to know as much information about a flight in order to assess the safety of the NAS at both a micro and macro level. From a micro level, the analyst cares about an individual aircraft. At any given time, the more information about that aircraft and the context of its operation would serve to inform the analyst about any potential safe/unsafe conditions. In addition, when viewed at macro or global scale, aggregate data regarding aircraft operations would provide an analyst with a holistic view of the NAS allowing them to understand and propose changes to airspace, procedures, and equipage to better manage the flows of aircraft to and from their destinations. By determining two attributes of an aircraft (phase of flight and operating rules), an analyst can acquire additional domain knowledge that can only help to aid in the detection of safety events and efficiencies within the NAS.

One of the challenges for a safety analyst within the FAA today is that data exists in several disparate systems; systems that while designed to interface to each other, they are not designed with an analyst in mind. Often the data from these systems is in a unique or proprietary format that needs to be examined and reduced to something that is workable and reasonable from a data analysts' perspective. To further complicate matters, often the data is fairly noisy owing to the errors inherent in signal processing from the radars as well as the lack of high-resolution updates (i.e. some radar scans provide target updates of 8-12s) [1]. While there is quite a bit of research ongoing to leverage technologies that tie all of these systems together, there is not one single authoritative source for information for the average safety analyst, placing additional burdens on the average analyst.

Additionally, there is no silver bullet or automated solution that can quantify the attributes of an aircraft flying in today's NAS. There is no known way to automatically detect an aircraft's phase of flight or operating condition. Usually to determine this information, in the case of an accident, incident,

or use case for analysis, a safety analyst will often elicit the assistance of a subject matter expert (SME). This person has in-depth knowledge of how an aircraft behaves and can look at the data in order to determine an accurate picture of the flight condition surrounding a particular situation. Usually the SME is a pilot or air traffic controller. Additionally, this individual will often make use of other data sources external to the surveillance data (i.e. flight planning data, weather, and/or audio tapes of ATC transmissions with the pilot). This data can make it easier to determine phase of flight at any given time, but is not available for all aircraft in all phases of flight or classes of airspace.

For post-flight analyses and gate-to-gate measures as the FAA implements NextGen, it becomes increasingly difficult to quantify an individual aircraft and determine its phase of flight or operating condition (i.e. was the aircraft flying Visual Flight Rules – (VFR) or Instrument Flight Rules (IFR)). If there was a way to uniquely determine this information for any given aircraft and to associate individual radar tracks to a given aircraft, it would represent a huge benefit to the FAA. The datasets generated by radar data and other surveillance sources such as Automatic Dependent Surveillance Broadcast (ADS-B) are all voluminous (ADS-B messages are sent out at 1Hz) and the size of the surrounding dataset is quite large [2]. The FAA is currently exploring ways to examine the datasets associated with surveillance data sources in order to implement data mining techniques to the large amount of surveillance data. The analysis of "Big Data" in aviation is a unique and challenging problem and one that lends itself to defining and developing specific approaches to understand "attributes" of the data such as phase of flight and operating condition.

To begin to tackle this problem, the research team began to focus on an automated means of detecting phase of flight and operating condition for aircraft flying in the terminal environment. This was done for a number of reasons. First, since there are multiple FAA surveillance and ATC systems and all contain disparate data subject to unique formats, it would be impossible to develop, test, and implement phase of flight and operating conditions for each given the time and resource constraints inherent in the initial phase of the research effort. Second, there are certain phases of flight that only apply to each domain and for those that apply across several domains, often the algorithms for determining them are vastly different. Third, the ability to quickly and efficiently acquire, parse, and analyze data from several systems given time and resource constraints was not practical.

For all of these reasons it was decided to focus on the terminal environment. This was assumed to be the most interesting case from a research perspective since often phases of flight are changing in the terminal environment as aircraft are transitioning through approach to landing, taking off and departing or merely leveling off from an initial climb-out to altitude. In addition, in the terminal environment, operating condition often varies as aircraft on final approach to land will sometimes fly a "visual approach" whereby they change flight rules from IFR to VFR while in flight. This allows pilots and ATC to decrease separation standards and is prevalent in Visual Meteorological Conditions (VMC).

As an initial phase, the research team was provided with a large sample set of National Offload Program (NOP) data. NOP data is an offload of data received from radar returns from several terminal radars which provide secondary surveillance identifying aircraft with Mode C or Mode S transponders that are "seen" by ATC. This data is taken from different airports or stations around the country and has fields such as position, groundspeed, and altitude. For the purposes of this research effort, NOP data from one facility, Philadelphia TRACON (PHL) was the only one examined. This radar data is used to feed the ATC consoles and systems such as STARS (Standard Terminal Arrival System), and Automated Terminal Radar System (ARTS). However, the data is sometimes less than ideal since, unlike ADS-B, there is no unique transponder code for any given aircraft. Making things worse, ATC will often reuse codes from one aircraft to another based on several factors during a period of time. From a data perspective, this is problematic when trying to determine the Phase of Flight (PoF) metric because the classification relies on the data relevant to the aircraft being classified and accurate identification of an aircraft from within the dataset is paramount to accuracy of the data. However, there are ways to deal with these inconsistencies. One way is to examine the parameters in a point of data, and to utilize constantly defined parameters such as altitude, speed, heading, and beacon-code which are available for use. These compensate for parameters that are not available or able to be linked to an individual data point comprising an aircraft. Using these and similar parameters, we are capable of finding trends in different points of data that allows us to assume the Phase of Flight of different aircraft.

In the first section of this paper, we will discuss the different phases of flight and how they were described and calculated. We will then describe in detail each of the constant parameters that we are using for the analysis and some of the errors that may arise from either from recording or in-frequent absence. Next, we will discuss the process we used for establishing a unique identifier for a single aircraft, as well as issues encountered with that process. Afterwards, we will describe how we used these defined parameters to define the PoF for a point of time in flight. Lastly, we will share our results and approaches for future studies on this topic.

In the second section of the paper we will also discuss our planned approaches for calculating operating condition. This is relatively straightforward for certain aircraft (i.e. airliners), but can vary significantly for general aviation and rotorcraft. Due to time constraints on the initial phase of research, the team did not get to adequately explore this topic, however several approaches and methods have been proposed in order to examine this topic in greater detail with actual data.

II. PHASES OF FLIGHT

From ICAO taxonomy, the phases of flight are as follows:

- Standing
- Pushback/Towing
- Taxi
- Takeoff
- Initial Climb
- En Route
- Maneuvering
- Approach
- Landing
- Emergency Descent
- Uncontrolled Descent
- Post-Impact
- Unknown

However, these definitions denote actual phases of flight versus what the aircraft is actually doing at any given time. Since these require some domain knowledge and we are trying to approach the problem from a data-mining and automated perspective, the solution proposed was to further simplify the phases of flight into the following five sub-phases: For the purposes of the research effort, the team focused on the following five characteristics phases of flight:

- Taxi
- Climb
- Cruise
- Descent
- Unknown

It is important to note the distinction that these are not true phases of flight per the ICAO definitions, but they allow for a simplification of the problem to generate initial results. Afterwards, when the data mining algorithm is further refined, additional phases of flight could be added and the five sub-phases could be reworked into the ICAO phases of flight. This indeed, is the planned approach moving forward and allows for cases where the initial algorithm couldn't confidently determine one of the other classifications.

The five sub-phase PoFs are defined as follows:

- Taxiing - Taxiing happens on the runways so it has characteristics of very sudden heading changes, low altitude, and low speed.
- Climb - Climb is characterized by a dramatic increase in altitude over a long period of time or distance. This can happen during the flight or at the beginning right after Takeoff.
- Cruise - Cruise is characterized by long periods of level flight where the altitude varies very little. This happens at altitude once the aircraft has aligned onto the heading of its flight path.
- Descent - Descent is characterized by a dramatic decrease in altitude over a long period of time or distance. This can happen at any time during the flight and is most dramatic when approaching a Landing.
- Unknown - used when the algorithm cannot confidently determine the actual PoF to be one of the other four mentioned above.

Using these five sub-phases, allows for the establishment of the other PoFs over a longer period of time, which will be the focus of our future research.

III. RULES OF FLIGHT

By looking at individual periods of dips (immediate descents to ascensions and vice-versa), any form of shifting in altitude, and the constant height as guidelines, we can identify two Rules of Flight (RoFs) that are used by aircraft: Visual Flight Rules (VFR) and Instrument Flight Rules (IFR). Although these can be further categorized into sub-rules (Marginal VFR and Low IFR) based on weather conditions, the rules of flight do not change within these conditions. Thus, we decided to only classify flights based upon the two primary rules. We defined these flight rules as the following for our algorithm:

- Visual Flight Rules – Thousands of feet plus 500 feet. Odd number thousands of feet (2500, 3500, 4500, etc.) within our Altitude data. We also found that aircraft points with very low altitude were more likely to be VFR as well based on predefined points.
- Instrument Flight Rules - Even number of thousands of feet (2000, 3000, 4000, etc.) within our Altitude data. The exception to this is near the airport environment where 1000 feet is typically the altitude of the VFR traffic pattern.
- Unknown - In the case that our data was flawed or couldn't be defined as VFR or IFR.

Some issues encountered with defining these RoFs are that we are unable to use weather data since we are solely relying on NOP data. However, in the future this may change, and weather data, could be used in the future, if available, as a cross-check against the results of the algorithm. It is important to point out that weather data alone will not quantify whether a flight is IFR or VFR, as most aircraft operating in commercial aviation today, operate under IFR except for the approach and landing phase where they will often switch to VFR in accordance with ATC operations. In the future, we hope to develop a better fitting model to suit these definitions of PoF.

IV. DATA

Our data points came from one table with 15 GB of data from our database. The following parameters are consistently defined and are used within our application for our calculations:

- Beacon Code - An identifier used by radar to identify an aircraft at a given time. Different aircraft can share a beacon code though not at the same date and time within the same facility.
- Timestamp - Given to a point of data for identifying time during a flight.
- Altitude - The aircraft's height for a given point.
- Speed - The aircraft's ground speed for a given point.

Additionally, we defined and created two new parameters to our data, as described below.

- Row Number - The row number of our data point (Unique Identifier).
- Aircraft ID (ACID) - The identifier given to a set of points to identify the aircraft flight.

Although there are other parameters that may be used, such as Heading, (X,Y) position, AC Type, we felt that focusing on the parameters above would give us the best results, as some parameters such as AC Type weren't always defined in the complete dataset. Due to issues with the automation system and the NOP data, the information about which target corresponds to an aircraft type is not always available. This is dependent on the transponder of the aircraft and whether or not the flight plan information makes its way into the NOP data. However, if another source of data was used with higher-quality information, it is expected that the results would be of higher fidelity.

A. Database Details and Issues

In our database we had a full 24 hours of NOP data which contained 40 different column headings; however, a few of those columns were completely blank despite having a numbered header. We also encountered issues with a lack of any primary key or identification between different aircraft. There were also important columns that were sometimes defined or undefined, such as Flight Rules which would describe whether an aircraft was VFR or IFR; however if this problem didn't exist then the point of calculating RoF becomes meaningless.

The database used SQL/PL as its implementation language, which we found to be difficult to work with, especially when dealing with large datasets. For instance, when trying to edit all the rows on a table in our database, we found that due to the size of our table we weren't able to edit each row and would in fact crash our operation as a result. Due to this issue, we were required to process each part in smaller sets. Although in the end we were able to edit our tables, as a result it cost us a significant amount of time, requiring us to spend days on adjusting rows of data.

B. Finding a Unique Identifier

The most challenging issue in processing and arranging the NOP data was that it was not sorted and that Beacon Code values were reused throughout the data set, and therefore couldn't be designated as a Unique Identifier (UID). As mentioned earlier, we needed to average all data points, which would be problematic if data from two different planes were to be compiled together. This problem prompted us to search for a UID.

We initial looked at our dataset and wanted to see how different our values could be. By making box plots of major attributes such as Altitude and Speed, we are able to judge how drastic our changes can be. As seen in Figure 1 below, we see that our altitude has the most variety and outliers, also containing negative values fitting within our model.

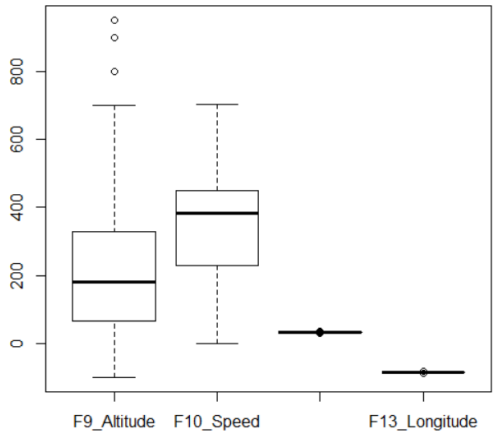

Figure1. Box Plot of Major Points

Figure 1.

Although this makes our altitude seem useless due to its large amount of variety, it can work surprisingly well when looking into separated spaces. One example is shown with the well-defined aircraft, primarily using one beacon code. In the listing below, we can clearly see the aircraft making its ascent.

	Time	Callsign	AcType	Altitude	Speed	Heading
199	00:00:02.480	JCY377	PC12	13	101	246
670	00:00:07.110	JCY377	PC12	14	100	248
1245	00:00:11.738	JCY377	PC12	14	101	242
1728	00:00:11.738	JCY377	PC12	14	101	242
2334	00:00:20.981	JCY377	PC12	15	103	246
2820	00:00:25.577	JCY377	PC12	16	111	261
2822	00:00:25.577			16	111	260
2823	00:00:25.577			16	111	257

Figure 2. JCY377 Flight Data

From these results, we decided to rely on timestamps and beacon codes. We sorted our database by Beacon Code and Timestamp ascending. Afterwards, we created two new values: Row Number and ACID. Row Number would act as a UID for each point of data, while ACID would uniquely define a set of points as an aircraft's flight.

As we see with our data, gathered by using a common beacon code, we found that our altitude can be used as a clear marker for phases of flight. Later, we would see this aircraft go from ascent to cruising in 35000 ft and then descending to cruise at 30000, then again at 17000. Using these and other data points, we may predict patterns to map basic rules that

can be used for predicting rule of flight along with basic flight patterns seen in our ascent to predict phases of flight.

To define Row Number for each point, we simply auto-incremented from 0 until we reached each point. As for ACID, we needed to use more comparisons in order to differentiate multiple aircraft from each other. In order to solve this, we initialized our first point with an ACID of 0. As we set each point, we checked whether the Beacon Code has changed, and if it did, then we incremented the ACID from the previous value, otherwise we kept it unchanged. Along with checking the Beacon Code, we also used the Timestamp to check the difference between the current point and the previous point. If the difference was greater than a defined amount (in our case, 100 seconds) then the ACID would increment by one from the previous point.

A critical issue with our current processing of ACID was that we initially believed the Beacon Code was incremented by radar up to 7777 and then returning back to 0. However, after processing we found that a large portion of ACIDs were 0 and were vastly larger than any other ACIDs. It also seemed that the sets of Beacon Code decreased over time, meaning that Beacon Codes are reused when not being used. Because of this, there is an issue with our ACID, meaning that it will require more information to accurately calculate for lower levels of Beacon Codes.

C. Additional Data Cleansing

Our algorithm is sensitive to missing data and outliers so an additional sweep of the data was made once we had a unique ACID for each flight. Because the fields are filled by various sensors there are cases where readings of 0 are given for altitude where the altitude is 20,000 feet just three seconds prior and 21,000 feet just three seconds later. In these cases the 0 can be replaced with confidence with the average of the surrounding values. There are similar cases with missing speeds. We also corrected cases where a record is more than 50% divergent from surrounding data. For example a speed of 300 MPH then 100 MPH then 300 MPH again will be replaced with 300 MPH.

The final cleanup of the data involved removing any flights with fewer than five records. There are cases where the flight is passing through a corner of the airspace and flies in and out of the radar range leaving a trail of only a few records. These are removed because we need about 30 seconds of flight data for one aircraft for the algorithm to make a determination of flight path.

V. PHASE OF FLIGHT ALGORITHM

For defining the PoF at a given time, we use our calculated ACID to retrieve the points of data from the database. From there, we insert a timestamp that exists within our received dataset and analyze all points 30 seconds before and 30 seconds after the timestamp. Knowing our range within the dataset, we look at the altitude and summarize the set into five numbers as follows:

- First - The first altitude when sorted from earliest to latest.
- Min - The smallest altitude in our dataset
- Max - The largest altitude in our dataset
- Avg - The average altitude in our dataset
- Last - The last altitude when sorted from earliest to latest.

We defined these five numbers as the five-number summary. We similarly generated a five-number summary for speed as well. After receiving these numbers, we then checked the difference between the first and last points of altitude. If it was less than 1000ft then it could be one of two pre-defined phases:

- Taxiing - If the max altitude was low (less than 500ft) and had low speed (less than 100 kts)
- Cruising - If the min and max altitude had a small difference (less than 1000ft)

If the difference was larger than or equal to 1000ft, then it could indicate one of two pre-defined phases:

- Climb - If the first altitude point is smaller than the last altitude point
- Descent - If the first altitude point is greater than the last altitude point

If the data couldn't fit into one of these categories, it would be classified as Unknown.

VI. RULE OF FLIGHT ALGORITHM

Similarly to the PoF algorithm, the RoF algorithm uses the calculated ACID to retrieve the points of data from the database. We then use an inserted timestamp to analyze the set of data of points 30 seconds before and after our or timestamp. After knowing this, we use the identified points to generate a five-number summary for altitude, as well as a separate one for speed, (First, Min, Max, Avg, and Last), defined as above.

After generating our two five-number summaries for altitude and groundspeed, we then round our altitude to the closest multiple of 5, from which we define our RoF as one of the following:

- Visual Flight Rule - Approximated average ends with a 5 or average is 0.
- Instrument Flight Rule - Approximated average ends with a 0.

If the data couldn't fit into one of these categories, it would be classified as Unknown.

VII. SOFTWARE SYSTEM STRUCTURE

We built a software system to facilitate the interaction with the data, to run the data-mining algorithms, and to output the results. Our software has three main parts: Calculation

Module, Intermediary Module, and Graphical User Interface (GUI). We decided to separate the problem into these modules so we could test the reliability of the system and to allow the Calculation Module, to function as an add-on module in other FAA systems.

The diagram depicting the structure of our software system is shown below.

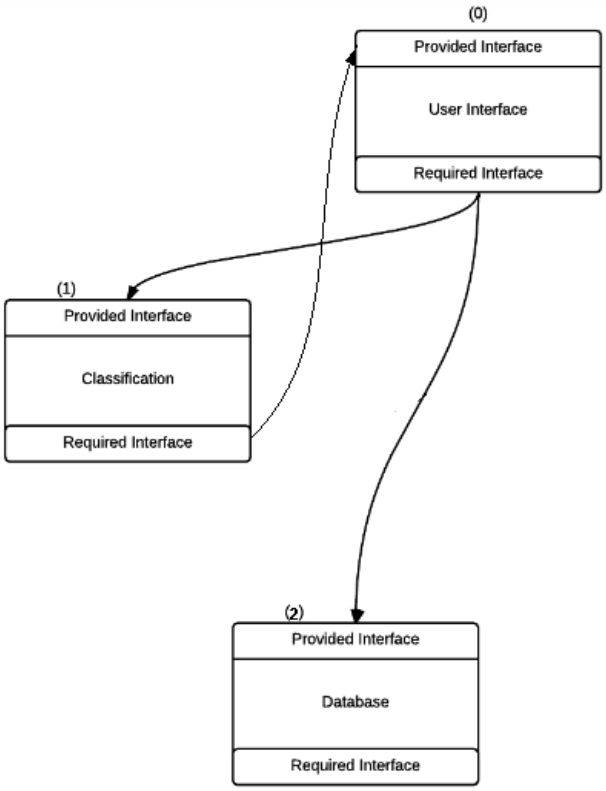

Figure3. Software System Structure

A. Calculation Module

Our calculations were done through a custom module in Python that would use our Rule of Flight and Phase of Flight algorithms for different classifications. Initially, we took in a JSON file that was created by our GUI with information gathered from the database using a timestamp and ACID. This JSON file would then be formatted before being inserted into any method in our module using the Python JSON package. The JSON file should be in the following format:

- [

  ...

  {“ts”:HH24:MM:SS.FF,“alt”:a,“speed”:v}

  …

  ]
- ts - Timestamp
- Alt - Altitude at a given point
- Speed - Ground speed velocity at a given point

Here, each set of ts, alt, and speed is a data point from the database.

After doing this conversion, we insert the formatted dataset into our module. There are two methods that can be used by the module for classification:

- phaseClassification(dataset, time) - Classifies the phase of flight for a given dataset surrounding the given time. Will return a phase of flight as a string.
- ruleClassification(dataset, time) - Classifies the rule of flight for a given dataset surrounding the given time. Will return a rule of flight as a string.

There is also an existing checkData() method as a possible redundancy for data, in case our formatted data has negative values, consistent “bad entries” or not fitting under a threshold for difference between two points of altitude or speed.

After doing all necessary formatting and possible checks, each individual classification method uses restructureDataToPeriods(dataset, time) to take the entire dataset and only look at a few values that we consider valuable. These values are 30 seconds before and after our timestamp that was sent through our method. This allows us to take a smaller, more unique image of our aircraft at a given time. After implementing our dataset into periods, we then run our periods through the classification method using the phase of flight and rule of flight algorithms described above.

B. Intermediary Module

We also implemented a script in between the Calculation Module and the GUI. We created this intermediate step because the function in Java that allows other languages to be executed internally only reads the output stream of the internal program. This means that for Java to get any value from executing the Calculation Module, the module would need to return values by printing them to the console. Obviously, we didn’t want that to be the standard return type of the Calculation Module since we want it to be flexible and importable into any project. Therefore, this function calls the Calculation Module functions, which return Strings, and prints the output so that the GUI can read and display them.

C. Database Module

The GUI module was set to interact with the database in order to add a level of abstraction between the source data and the Classification Module. We wanted to ensure that the Classifier would work despite the source of the data as long as it was provided the proper input format. This module is actually a part of the GUI program and was written using JDBC to connect and submit queries to the database. The GUI then would clean the received data in order to give nice, clean data to the Calculation Module. This cleansing is similar to the cleanData() method used by our module, but has some further checks based on our expected averages.

D. *Graphical User Interface*

The software system included a user interface to demonstrate the findings of the phase of flight and rule of flight algorithms. The GUI was written in Java. It has two input fields and communicates with the Python module. In the first field, by selecting some unique identifier, a user could select an aircraft to analyze. The user could then input a point in time they wanted to analyze in the second field. The GUI also had an output text area where it displayed the output of the Calculation module which made for a better display than a simple command line based user interface.

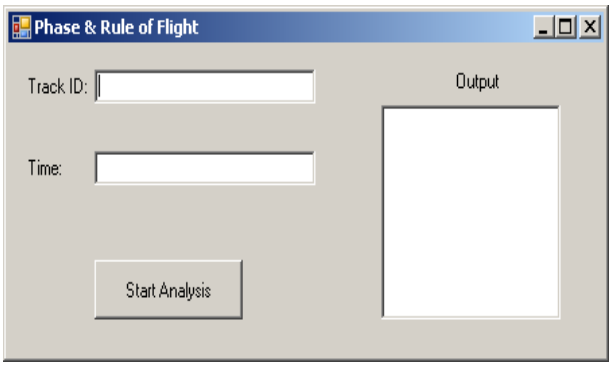

Figure4. Test Application

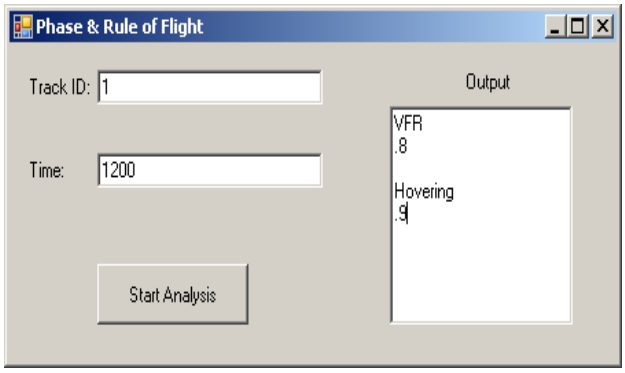

Figure5. Example Input and Output

VIII. RESULTS

From using the phase of flight algorithm with the given ACID, we found that when defined, the PoF had a good amount of accuracy. However, we did run into issues with frequent Unknown PoFs, forcing us to redefine our current definitions to have more accuracy as well as to use more of the parameters that are available to us. This includes rules of flight such as VFR and IFR, as well as Latitude & Longitude with coordinates for cross-checking our approach. However, some parameters, despite being informative, may be empty at times (Rule of Flight as an example).

IX. FUTURE CHANGES

There are several different areas of improvement that could lead to better overall results. We will break down the changes that could be made, by area. In short, the RoF algorithm could be improved to account for additional context defining RoF for aircraft flying today. Furthermore, the PoF algorithm could be expanded, and a Unique Identifier could be selected using better criteria.

We believe that these issues could be resolved by defining our current phases with phases from neighboring time periods, thereby creating more detailed information on phases of flight. Furthermore, our Rule of Flight, although is fairly simple and follows basic rules, may not match what is set by the aircraft. This is mainly because of our simplification as well as different conditions that could exist. These types of issues, including weather and other conditions are not considered, but can be included into future changes.

However, as an overall prototype, we found that this form of calculation certainly has promise and can be improved with either a strengthened algorithm or form of machine learning. This can be seen with phase of flight algorithm, which is able to identify key details for different results, matching with what was view in our data analysis.

A. *Rule of Flight*

Although we are capable of calculating the RoF, it isn't always correct. This is because of not only variations with our current definitions, but also due to conditions and possible rules for specific classes of aircraft. Due to this, we would need a more comprehensive algorithm capable of more logical assumptions. We also believe that our definitions could be improved upon for different headings in different locations, as well as more specific description on our classification of altitude. These flaws can be seen in some of our pre-defined Flight Rules in specific rows, where we are capable of receiving the RoF. Although we believe a majority will match, there are, specific occasions especially with incredibly well-defined aircraft rows with high altitudes. Along with this, we may also use the heading at a given point to better calculate the RoF at a given time.

B. *Phase of Flight*

Our current method of calculating the PoF only has a small list of classifications, but it is our hope that by using these classifications we are capable of defining other classifications as series of sub-classifications which would require more work by the PoF calculator. These classifications can be seen in the Phase of Flight: Definition and Usage Notes defined by the International Civil Aviation Organization (ICAO) [3]. This includes Standing and Towing as ground-level phases that precede Taxiing. An example of this is by describing a takeoff as a mix between our definition of taxiing and ascending. We also need to improve on the accuracy of our predictions by using more data and by having our classifications better

defined. For future research, we may also look into data at other airports and compare their traffic and look into possible ways of editing our calculations based on different regulations.

C. *Unique Identifier*

One of the major changes that we have to make is on our ACID. Although it is useful for data points with higher Beacon Codes, it is absolutely useless for Beacon Codes such as 0 or 1 due to the consistent use. Originally, we suspected that the Beacon Code simply incremented and went back to 0 after passing its max (7777 base 8). We were clearly wrong with our prediction, causing faults in our ACID, except with much higher values.

Some ways to fix this could be by using location and possible parameters that aren't always defined such as matching Rule of Flight and AC Type to find similar patterns. Ways that we can use Location is through the Haversine formula using Latitude and Longitude [5], allowing us to keep track of paths over time. Through these methods, we are capable of making better and more accurate ACID, which allows to better differentiate aircraft.

D. *Additional Changes*

Some things we are capable of adding would be the use of the Haversine formula in order to calculate distances between the given latitude and longitude. Based on the latitude and longitude of the plane along with the latitude and longitude range of the airport, we will be capable of reading the PoF and RoF with more certainty. This includes reading a large descent near the airport as a possible Approach or Landing phase, while having a similar situation farther from the airport possibly being an Uncontrolled Descent or Emergency Landing.

In fact, there are numerous packages in the statistical program R that we have not fully exploited for doing things with Latitudes and Longitudes. For example, in the package Geosphere [5], there are great circle functions for computing Haversine distances, heading, and predicted destination based on latitudes and longitudes. Most of these are built on the work of [6].

Along with this, we are able to use our measurements of this distance and heading, along with the regulation of different airports, to hopefully better calculate the RoF at a given time. By doing this along with the regulation of different airports, we should be able to become more accurate depending on location.

We may also try re-describing the periods that are used by our phase of flight and rule of flight algorithms. Currently, we are using a section of time from our dataset which acts as a good default, but there are moments where there could be drastic changes from either Taxiing to Ascending or Cruising to Descending where it may appear to be either one or another. To solve, we could use our timestamp as a point of adjustment and attempt to better section of points with similar altitudes and speeds, or at least similar deltas in altitudes and speeds. By doing this we may have our time range move from 30 seconds before and after our timestamp to being 45 seconds before and 15 seconds after, or vice-versa. This could act as a better range for our phases as well as possibly having some determination in rule of flight.

It also should be noted that because of time constraints a minimum of data cleansing was done before the five number summaries were computed. We don't know if we could obtain better results by doing data reduction techniques such as Principal Components Analysis (PCA) or other methods prior to computing these summaries [7].

Finally, we have not tried any advanced data mining methods such as fitting the results to a Neural Network to see if the predictions could be improved. The limiting factor in this case was the time it would take to build an extensive training set, but that could be implemented in a future version [8].

ACKNOWLEDGMENT

We thank Rowan University Computer Science students Phuc Tran, Winston Cheong, and Shaan Menon for their contributions to the development of the data-mining system, and to FAA employees Mike Paglione, Chu Yao, Tom Tessitore, and Tony Gurcsik for their contributions to setting up the collaboration and for their technical insight into the problem being addressed.

REFERENCES

[1] <http://www.faa.gov/air_traffic/>

[2] <http://www.faa.gov/air_traffic/technology/tamr>

[3] Chamberlain, Bob. "COMPUTING DISTANCES." *Computing Distances*. NYU Computer Science Department, n.d. Web. 12 Feb. 2016. <http://www.cs.nyu.edu/visual/home/proj/tiger/gisfaq.html>.

[4] *Phase of Flight*. N.p.: International Civil Aviation Organization, Apr. 2013. PDF. <http://www.intlaviationstandards.org/Documents/PhaseofFlightDefinitions.pdf>.

[5] Hijmans, R., Williams, E., Vennes, C., "Spherical Trigonometry R Package," Version 1.5-1, 2015

[6] Karney, C.F.F. "Algorithms for geodesics," *J. Geodesy* 87: 43-55. https://dx.doi.org/10.1007/s00190-012-0578-z.

[7] Barbara, D., DuMouchel, W., Faloutsos, C., Haas, P.J., Hellerstein, J.H., Ioannidas, Y., Jagadish, H.V., Johnson, T., Ng, R., Poosala,V., Ross, K.A., Servcik, K.C., "The New Jersey Data Reduction Report," *Bull. Technical Committee on Data Engineering*, 20:3-45, Dec. 1997

[8] Han, J., Kamber, M., Pei, J. "Data Mining Concepts and Techniques," Third Edition, 2012, Morgan Kaufman, Waltham, MA.