Classifying Aircraft Approach Type in the National General Aviation Flight Information Database

Kelton Karboviak[1], Sophine Clachar[1], Travis Desell[1], Mark Dusenbury[2], Wyatt Hedrick[1], James Higgins[2], John Walberg[2], and Brandon Wild[2]

[1] Department of Computer Science, University of North Dakota, Grand Forks, ND 58202, USA, {kelton.karboviak,sophine.clachar,wyatt.j.hedrick}@und.edu, travis.desell@engr.und.edu,

[2] Department of Aviation, University of North Dakota, Grand Forks, ND 58202, USA, {dusenbur,jhiggins,walberg,bwild}@aero.und.edu,

Abstract. This work details the development of the "Go-Around Detection Tool", a tool for classification of aircraft approach types for the National General Aviation Flight Information Database (NGAFID). The NGAFID currently houses over 700,000 hours of per-second time series flight data recorder readings generated by over 400,000 flights from 8 fleet of aircraft and over 190 participating private individuals. The approach phase of flight is one of the most dangerous, and classifying types of approaches as stable or unstable, and if they were a *go-around*, *touch-and-go*, or *stop-and-go* is an especially important issue for flight safety monitoring programs. As General Aviation typically lacks the Weight on Wheels (WoW) technology and many others that exist within Commercial Aviation, there is difficulty in detecting landings and go-arounds as these need to be inferred somehow from the raw flight data. The developed application uses several airplane parameters reported by a flight data recorder and successfully detects go-arounds, touch-and-go landings, and stop-and-go landings as either stable or unstable with an accuracy of 98.14%. The application was tested using 100 student flights from the NGAFID, which generated 377 total approaches. Out of those approaches, 25.73% were classified as unstable. It was found that only 20.62% of all unstable approaches resulted with a go-around, which is far from the ideal 100% goal. Lastly, the application was parallelized and found to have a 9.75x speedup in doing so. The Go-Around Detection Tool can be used to provide post-flight statistics and user-friendly graphs on both an organizational- and individual-level for educational purposes. It is capable of assisting both new and experienced pilots for the safety of themselves, their organization, and General Aviation as a whole.

1 Introduction

The National General Aviation Flight Information Database (NGAFID) has been developed at the University of North Dakota as a joint university-industry-

FAA initiative that is responsible for the curation, dissemination, and analysis of flight data for the General Aviation (GA) sector of Civil Aviation [1, 2]. The objective of the NGAFID is to proactively identify accident precursors and mitigate risks associated with unsafe flight practices and aircraft maintenance issues within the GA community. This is achieved via non-punitive information sharing so as to educate operators on risks associated with their flights to encourage safer practices [1]. The analytical tools provided by the NGAFID are free and available to GA pilots who participate by uploading their flight data through the NGAFID web application [2] or the GAARD mobile application [3]. Subsequently, their flight data is preprocessed and analyzed using various queries. Many queries are based on threshold criteria called *exceedances*, which are predefined using known limitations of the make/model aircraft or the phase of flight. However, recent work has focused on developing more advanced analytics through machine learning and other holistic techniques [4–9]. Upon logging into the web portal, the user is provided with summaries of any unsafe events and is able to reanimate their flight(s) using X-Plane [10] or Cesium [11]. The intent is to educate participating pilots on any unsafe practices in their flight and maintenance issues with the aircraft which may contribute to an accident/incident. The overall goal of this initiative is to reduce the accident and fatality rates within the GA community.

GA is one of two branches of Civil Aviation, which pertains to the operation of all non-scheduled and non-military aircraft [12–15]. GA includes fixed-wing airplanes, helicopters (rotorcraft), balloons, dirigibles, gliders, etc.; and comprises 63% of all Civil Aviation activity within the U.S. [12, 14, 16]. Performing GA flight analysis is essential for making the GA community safer, as currently GA has the highest accident rates in Civil Aviation [15,17]. As of 2013, the total accident and fatality rates for GA were 5.77 and 0.99 per 100,000 flight hours, respectively; and 74.0% of GA accidents were caused by pilot actions [15].

Commercial Aviation aircraft have weight on wheels (WoW) sensors that are utilized for detecting when an aircraft is on the ground. For analyzing approaches for Commercial Aviation flights, all that needs to be done is detect when the WoW sensors are reporting the aircraft's approximate weight, which shows that the aircraft has completed a landing. Once a landing has been found, then the corresponding final approach is simply the time leading up to that landing. On the other hand, aircraft in GA typically do not have WoW sensors, which makes the process of detecting approaches and landings much more difficult. Also, as a cheaper alternative to traditional sensors and flight data recorders (FDRs), many GA pilots can now utilize smart phones and tablets (e.g., iPhone, iPad, Android devices, etc.) to record their flight data. Using a smart device severely limits the number of flight parameters that can be recorded as compared to a traditional GA FDR.

The Go-Around Detection Tool was created as an additional tool for the NGAFID to be used for detailed GA flight analysis. The basic question that provided the impetus for this research was, "How many unstable approaches result in a go-around being performed as a result?". The University of North

Dakota *Cessna 172S Standardization Manual* states that a go-around must be conducted if a stable approach is not achieved by 200 feet above ground level (AGL) [18]. The hopeful and theoretical answer to the question should therefore be 100% of unstable approaches result in a go-around, but this is very unlikely due to special circumstances and pilot misjudgment.

To the best of our knowledge, there is currently no existing research that has performed similar unstable approach analysis and landing detection within the scope of General Aviation. The challenge of developing the application lies in the fact that this type of tool does not, and cannot, rely on certain information and technology available to similar projects in Commercial Aviation. Since GA does not typically have this type of information available, a new method was required in order to detect go-around's. In addition to detection of go-around's, the application collects and analyzes many other useful statistics about the landing attempt including the landing type, unstableness, and reasons for unstableness (if applicable to the approach). By combining all the aforementioned features together, the tool aims to help pilots fix unsafe habits in their landings, which will reduce the number of pilot-caused landing accidents/fatalities in GA.

2 Related Work

Harris et al. [19] of MITRE Corporation mined accident and incident reports provided by the International Civil Aviation Organization (ICAO) in order to determine the specific attributes that were the cause in each kind of report and also needed to be considered "interesting" (i.e., anything that is an exception to commonly accepted knowledge among aviation experts) by the aviation expert who collaborated with them. They developed a system called Smithers, based on attribute focusing, that uncovered a correlation that having an advanced heads up display (HUD) can help reduce the amount of damage as a result of a runway incursion[3].

Matthews et al. [21] performed similar research in which their goal was to find anomalous data in flights. They differ in the fact that they used algorithms that could analyze at both a fleet-level and flight-level. Doing this allowed them to find anomalies for an entire organization or just a single flight, which makes it very useful in order to find patterns of problems. This idea is similar to the NGAFID project in which flight data can be analyzed on multiple levels while giving statistics for each.

In Dr. Ed Wischmeyer's paper *The Myth of the Unstable Approach* [22], he discusses how the term "unstable approach" is now becoming too vague to be used in accident and incident reports. He argues there are too many factors that play into an approach; therefore, labeling it solely as an "unstable approach" is not sufficient. This aligns with one of the goals of the Go-Around Detection Tool in that it was developed to detect unstable approaches and be able to state what

[3] Defined by the Federal Aviation Administration (FAA) as, "any occurrence at an aerodome involving the incorrect presence of an aircraft, vehicle, or person on the protected area of a surface designated for the landing and take off of aircraft" [20].

the specific parameter was that caused the approach to be unstable. In doing this, it allows for further statistics to be generated, which can reveal further patterns to be detected within an organization if it becomes a wide-spread problem.

Nazeri et al. [23] researched accident and incident data from several different commercial flight data sources in order to discover the factors that cause those events. They created eight high-level categories, each with sub-factors, for classification. They used an algorithm to analyze the data for correlations between different attribute-value pairs across the accident and incident data sets. A factor support ratio was calculated for each attribute-value pair and ranked in decreasing order to find the most significant factors. The following high-level factors were the four top ranked in order: company, air traffic control, pilot, and aircraft. They also did a time-series analysis of the data for the ten-year period in which the data was collected (1995-2004). This time-series data showed the pilot and aircraft factors are generally decreasing over time, while the air traffic control factors are generally increasing. By uncovering these patterns and analyzing them over time, they were able to find the factors that are leading causes for accidents/incidents and can address these factors for improvement.

Lastly, CloudAhoy [24] is a commercial product that is very similar to the NGAFID as it has the ability to collect a pilot's flight data then later replay the flight back to the user and analyze it graphically. It serves a similar purpose by giving pilots a fully automated debriefing after each flight. This allows for instant feedback where a pilot can review their flight and detect any issues that may have occurred during flight.

While many of these works give high-level views of how mining flight data could improve safety in the aviation community [25–27], they do not provide specific details on how to apply different mining techniques to the data in order to obtain interesting results. Further, most of these works focus on Commercial Aviation data, instead of General Aviation [21, 23]. In addition, the NGAFID project consistently receives data from several organizational fleets such as the University of North Dakota, Ohio State, and Oklahoma State. Some of the related works recognize fleet data [21]; however, there are others that do not such as CloudAhoy [24], which is a considerable disadvantage. Another advantage of the NGAFID is that it is free-to-use and is an open-source project, which is very unique within the flight data monitoring space. Lastly, to our knowledge this project is the first to create an automated analyzer that detects multiple stop-and-go's, touch-and-go's, and go-around's in a single flight, while being able to categorize each as stable or unstable.

3 Automated Go-Around Detection

The Go-Around Detection Tool provides two main features: approach quality analysis and landing type analysis. The approach quality analysis focuses on the slice of data when the aircraft is between 50 to 150 feet AGL and looks for parameters that have been exceeded, while the landing type analysis focuses on

the slice of data when the aircraft is below 50 feet AGL and determines the type of landing that was a result of the approach.

3.1 Approach Quality

The first feature, approach quality analysis, performs several different detections and analyses such as airport detection, runway detection, and unstableness analysis. The algorithm for detecting an aircraft's approach needs to iterate through all of the time values since there can be multiple approaches within a single flight. Once the algorithm detects the aircraft is one mile away from an airport and is less than 500 feet above ground level (AGL), it is determined that the pilot is beginning an approach and a unique approach identifier is generated in order to store meta-data later in the process. Next, the algorithm continues to iterate through time values until either the aircraft goes under 150 ft AGL or it goes back above 500 ft AGL, which is then recorded as a go-around. If the aircraft goes under 150 ft AGL, then it is determined to be on the final approach. At this point, the runway that is being approached can be detected using a combination of the aircraft's current geo-location and heading since the intended runway may not be closest to the aircraft. The aircraft is considered to be on the final approach while it is within one mile away from the airport and it is between 50 and 150 feet AGL inclusive.

The analysis for unstableness is performed during this final approach stage. During this analysis, several flight parameters are checked against predetermined thresholds to see if any were exceeded. The values used for the thresholds can be seen in Table 1, and were obtained from [18, 28]. No experimentation was performed on the values for the thresholds since these have been found to be the physical limitations of the aircraft by the manufacturer. Additionally, the FAA Airplane Flying Handbook [29] states that if the procedures and configurations it provides for approaches and landings conflict with those given in the manufacturer's flight manual, the manufacturer's recommendations should take precedence. The logical conditions used to determine if a threshold is exceeded are:

$$F_1 = 180 - |\,|\,\text{runway.hdg} - \text{airplane.hdg}\,| - 180\,| \leq 10^\circ \tag{1}$$

$$F_2 = |\,\text{crossTrackError}\,| \leq 50\,\text{ft} \tag{2}$$

$$A = \text{airplane.IAS} \geq 55\,\text{kts} \wedge \text{airplane.IAS} \leq 75\,\text{kts} \tag{3}$$

$$S = \text{airplane.VSI} \geq -1,000\,\text{ft/min} \tag{4}$$

$$U = \neg(F_1 \wedge F_2 \wedge A \wedge S) \tag{5}$$

A *true* value for a condition means the parameter is stable. Thus if any of the parameters are unstable, U will result to being *true*, meaning the entire aircraft is in an unstable state.

Once the aircraft either goes above 150 feet AGL or goes below 50 feet AGL, then the final approach is marked as finished, and the critical meta-data associated with the approach is stored. The control of the algorithm will then

Table 1. Exceedance thresholds for Cessna 172S [18, 28]

Parameter	Description	Value
F	Flight path correct	Less than 10° off runway heading, less than 50 ft left or right of the runway center line (crosstrack error)
L	Landing configuration correct	N.A.
A	Airspeed proper.	Indicated airspeed (IAS) within 55-75 kts
P	Power setting appropriate	N.A.
S	Sink rate appropriate	Vertical speed indicated (VSI) does not exceed -1000 ft/min

be passed to the landing analysis function, which will detect the type of the resulting landing. The algorithm for this analysis will be discussed further in the next subsection.

3.2 Landing Types

The second feature, landing analysis and type detection, is able to differentiate between a stop-and-go landing, touch-and-go landing, and a go-around. The landing analysis algorithm iterates through time values starting where the final approach analysis finished. It continues to iterate while the aircraft is below 500 feet AGL, or if it is the aircraft's final landing and the time values run out, then it stops analyzing. While the algorithm iterates through the time values, it checks if the aircraft's IAS is less than or equal to 35 knots. If this is true, then it is physically impossible that the aircraft is still flying, thus it is determined to be making a complete stop. In order to detect a touch-and-go landing, the previous five elevation readings are stored and their average is calculated. If it is found the aircraft is not making a stop-and-go landing, then the average elevation for the last five seconds is checked to see if it is less than five feet AGL. This means the aircraft is still at a flying speed (above 35 knots) and is also maintaining a stable elevation of five feet or less for at least five seconds.

Once the aircraft goes above 500 feet AGL or the time values run out, then the landing type is determined from the conditions found during the analysis. If it was found the aircraft was making a complete stop, then a value of 'stop-and-go' is stored. If it was not making a complete stop and had a stable elevation of 5 feet or less, then a value of 'touch-and-go' is stored. The final value type, 'go-around', is used as a fall-through since there are only three classifications. The three landing types and how they are detected are summarized in Table 2.

After the landing is classified, then the critical meta-data found during the analysis is stored. Lastly, the algorithm returns the time value of the landing's ending back to the approach quality analysis algorithm so that it can continue

Table 2. Landing types and their conditions

Type	Condition
stop-and-go	Aircraft's indicated airspeed speed (IAS) falls below 35 knots
touch-and-go	Aircraft is not making a complete stop and maintains a stable altitude of five feet AGL or less for at least five seconds
go-around	All other cases

to scan for the aircraft's next approach from that time value. These algorithms will continue to do their respective analysis until the flight has ended and all the time values have been scanned.

4 Implementation

4.1 Programming Language and Libraries Used

The Python programming language was used for implementation due to its ease of use, its reputable scientific and graphing libraries, and the ability to quickly produce a viable application. The libraries utilized are MySQLdb for interacting with the MySQL database, matplotlib for graphing flight parameters in the early stages of the application, NumPy for its scientific functions, and the geodesy scripts created by Chris Veness[4].

4.2 Parallelization

The application was originally created to process the flight data in a linear fashion. This proved to be fairly time consuming when running the application in batch mode with a significant number of flights contained in the NGAFID. In order to improve the performance and efficiency of the application, the built-in multiprocessing module was used. The parallel application uses the Producer-Consumer model in which the parent process acts at the Producer by enqueuing all of the unique flight identifiers onto a queue, and the child subprocesses act as the Consumers by dequeuing a flight identifier and processing it. The multiprocessing module was chosen over the built-in threading module due to the issue with Python's Global Interpreter Lock (GIL) effectively restricting bytecode execution to a single core [30]. This makes the threading module unusable for long-running CPU-bound tasks, which this application heavily relies on.

5 Results

5.1 Experiments

The experiments were run using Cessna 172S flight data produced by students at the University of North Dakota during the month of September 2015. Student

[4] http://www.movable-type.co.uk/scripts/latlong.html

Fig. 1. Example of using a KML file to visualize a flight path in Google Earth. This flight visualization is an example of a student flight that has multiple final approach phases.

flight data is ideal for unstable approach analysis testing as it contains very noisy data, which provides a diverse array of flying patterns. A random sample of 100 flights was chosen for the experiments.

First, the application was run against the 100 flights to obtain the automated analysis results. The same 100 flights were then manually analyzed in order to get human results, which could be compared to the automated results then determine the accuracy of the application. The test of the 100 flights was also run ten times each with the single-process version and the multi-process version as described in Sect. 4.2. This was done in order to compare and contrast the performance of the separate versions.

All results were gathered using a 2013 Mac Pro running macOS 10.11.6 with a 3.5 GHz 6 hyper-threaded core Intel Xeon E5 processor (for a total of 12 logical processing cores). The machine also has 32 GBs of 1866 MHz DDR3 ECC RAM.

5.2 Accuracy

The manual validation was performed using a combination of tools available on the NGAFID website: the Cesium flight reanimation tool and the Keyhole Markup Language (KML) generator to visualize the flight path on Google Earth [31] (see Fig. 1).

The Go-Around Detection Tool generated a total of 377 approaches for the 100 flights that were tested. As seen in Fig. 1, student flights typically consist of multiple approaches as this is something that needs to be practiced. Out of the total; there are 370 (98.14%) true positives, five (1.33%) false positives, and two (0.53%) false negatives. These results can also be found in Fig. 2. In the context of this application, a true positive is a case where the tool correctly indicates that an approach is occurring during a specified time frame. A false positive occurs when the tool indicates that an approach is occurring but is not in reality. Typically, a false positive occurs when the flight data has invalid

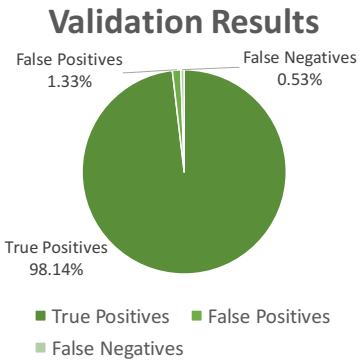


Fig. 2. Pie chart showing the manual validation results including true positives, false positives, and false negatives.

values for about the first ten rows, which then throws off the beginning of the algorithms. This happens infrequently, but could be accounted for in a future work by sanitizing the data before analysis. A false negative is the exact opposite where the tool indicates that an approach is not occurring but it is in reality. Typically, a false negative occurs when the approached airport's geological data is not contained within the database. These types of occurrences should stop once the airports database is expanded with more entries. Lastly, the tool misclassified the approached runway 13 times (3.45%). A runway is misclassified when the difference between the aircraft and runway headings is greater than 20°. This occurs during the runway detection portion of the approach analysis algorithm (Sect. 3.1), and the algorithm either returns a *null* runway or an incorrect runway due to the large heading difference.

In this same context, it is difficult to quantify the number of true negatives since these would be cases where the tool correctly indicates that an approach is not occurring. The difficulty lies in how to define a single occurrence. Should a single true negative be counted for every second the tool indicates that an approach is not occurring? If so, then this would create a numerous amount of true negatives and would dilute the percentages of the other statistics, which are more important in this application.

The validation results demonstrate that the Go-Around Detection Tool is exceptionally accurate in its ability to appropriately detect and classify most approaches in a flight.

5.3 Performance

A secondary aspect of this research was to test how parallelization can help improve the performance of an application in the domain of analyzing flight data. The results of the benchmarking tests showed that the linearly executing application ran for an average of 588.632 seconds over 100 randomly tested flights.

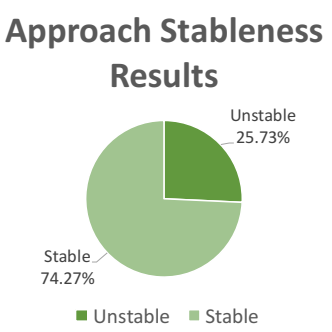
On the other hand, the parallel application ran for an average of 60.402 seconds over the same flights. This means the average per-flight execution times for the linear and parallel applications were 5.886 and 0.604 seconds, respectively. As a result, the parallelized application had a 9.75x speedup, which is fairly significant. This shows that the Go-Around Detection Tool can be used practically within the NGAFID as the system itself is currently transitioning into becoming a near real-time system.

As further evidence, the parallel application was tested on a larger subset of flights to see if the average execution time remained stable, in which it was tested on 5,272 flights. For this test, the parallel application was able to analyze the data and insert all the results into the database in 3,007.408 seconds. This gives a per-flight execution time of 0.570 seconds, which is slightly less than the average for 100 flights. The reasoning behind this can most likely be attributed to the fact that spinning up the sub-processes creates a substantial overhead. Thus, the longer the application is able to execute, the greater performance gain will be received. This will, of course, start to show diminishing returns as with any other parallel computing application.

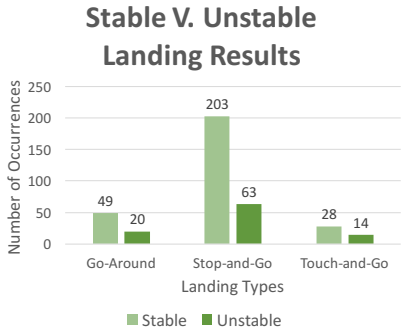
5.4 What Did We Find?

The results of the application have provided many possibilities for statistical analysis since numerous statistics can be calculated from the generated approach data. This can be seen in Fig. 3a to 3d in which a sample of the possible results were calculated for the experiments of the 100 flights performed for this research. With these various results, trends can be found in the data that has been analyzed. For example, we can see in Fig. 3a that out of the 377 approaches in the sample data, 74.27% (280) were stable and 25.73% (97) were unstable. By drilling down into that data, we can see the frequency for each of the landing types for stable and unstable approaches. Figure 3b depicts this more detailed information and shows that stop-and-go landings are the majority for both stable and unstable approaches. This result isn't very surprising for stable approaches; however, it is very undesirable for unstable approaches. If we look even further into the proportions for unstable approaches alone (Fig. 3c), we see that an unstable approach resulted in a go-around only 20.62% of the time. This is far lower than the hopeful 100%, but was expected to be approximately 20% by our aviation safety experts. As mentioned previously, this is largely due to pilot misjudgment since all the analyzed flights were piloted by aviation students; meaning they are still learning and are not professionals.

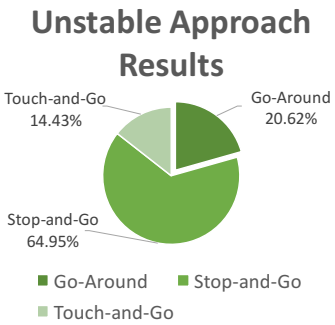
When looking at the unstable approaches and the parameters that caused them (Fig. 3d), additional interesting results can be found. We found the parameter that was exceeded the most was heading with 52 occurrences. Heading was not predicted to be the leading cause of unstable approaches, but our safety experts believe the 10° threshold (as defined in Table 1) may be too strict. Indicated airspeed was the second highest, but was predicted to be the leading cause since it was stated by our aviation safety experts to be a trend for UND's student pilots to be going too fast on final approaches.



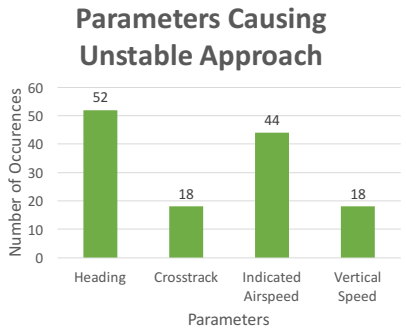
(a) Pie chart showing the number of stable approaches compared to the number of unstable approaches.



(b) Frequency of the occurrences of each landing type for stable and unstable approaches.



(c) Pie chart comparing the number of occurrences for each landing type after an unstable approach.



(d) Frequency of parameters that caused an aircraft to be unstable during an approach. Note that a single approach can have multiple unstable parameters, which causes the sum of the occurrences to not equal the total number of unstable approaches.

Fig. 3. Sample set of the statistics and trends that can be found from the automated analysis results.

5.5 Website User Interface

A new web page was implemented in the NGAFID for the purpose of dynamically displaying the results produced by the Go-Around Detection Tool to users (Fig. 4). The results are given in four tabs, one for each parameter, as histograms over a specified date range. A user is able to dynamically add additional date ranges, which will create an additional series in the chart for comparison. This

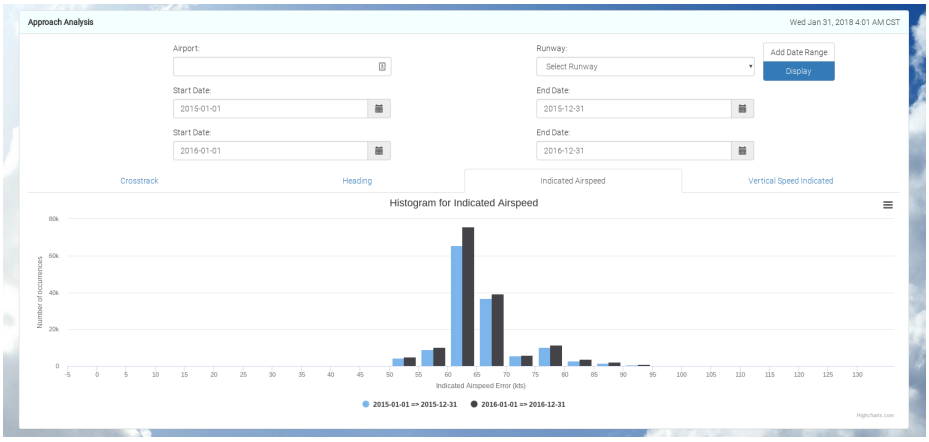


Fig. 4. A screenshot of the newly developed approach analysis tool on the NGAFID. It is showing the histogram for indicated airspeed error with two date range filters: 2015-01-01 to 2015-12-31 and 2016-01-01 to 2016-12-31. The frequency of exceedances can be seen with all values that fall outside of the 55-75 knots range.

feature can be used to detect changes in trends over time. A user is also, optionally, able to filter the results to an airport and further filter to a single runway. This will allow users to identify trends that are potentially occurring at a specific runway but not at any other runways.

6 Conclusion & Future Work

This paper presented the Go-Around Detection Tool, an application designed to augment the existing features of the National General Aviation Flight Information Database (NGAFID). The purpose of creating the application is to help further reduce General Aviation (GA) fatality rates since GA is the most dangerous branch of Civil Aviation. Additionally, the application was geared towards analyzing final approaches and landings because these phases of flight are where a majority of pilot-related accidents occur [15].

This research has provided many avenues for further work and refinement. First, the greatest constraint on the accuracy of the application is the accuracy of the instrument recording the flight data, whether that be a traditional flight data recorder or a smart device. This means that if data is recorded inaccurately, it is useless to the application and cannot be recovered. For example, in several of the 100 tested flights, the first 10 to 20 rows of data can have missing and/or spurious values due to the aircraft's sensors calibrating after first starting the flight data recorder. Thus, further work into filtering or sanitizing faulty data would be very beneficial.

Second, the greatest constraint on the execution time of the application is the algorithm for detecting when an aircraft is about to approach an airport.

Currently, the application uses a sequential search algorithm to calculate the closest airport based on the aircraft's geological coordinates every 15 seconds of flight data. Since the sequential algorithm has $\mathcal{O}(n)$ run time, and the database of airports that was used in the search contains over 16,000 airports, it quickly became the bottleneck of the application. It isn't immediately clear how traditional path algorithms may be useful in this context since in air transportation all nodes are fully-connected with (relatively) straight paths, and the distance between the aircraft and each of the nodes changes continuously over time. There was no extensive research done to find a more efficient algorithm for this kind of detection, so this is certainly one area that could be researched in the future.

Lastly, future research is planned on combining all the analysis tools currently available on the NGAFID website (high/low fast/slow analysis on final approaches and a self-defined glide path angle calculator) and developing an application which will calculate a "letter grade" based on some predetermined criteria for each approach within a flight. This will provide even more user-friendly methods for users to receive analysis and an overall score for their flights. This will be exceptionally useful for participating aviation universities in which a student pilot could record their flight, have it analyzed by the NGAFID, receive a score, then review the scoring breakdown with their flight instructor.

Once the Go-Around Detection Tool is fully integrated into the NGAFID, it will provide even more possibilities for data visualization and be easily accessible for both novice and experienced pilots. This will allow pilots on an individual or organizational level to become more aware of bad flight habits so they may correct them in future flights and help make General Aviation safer.

References

1. Clachar, S., Higgins, J., Wild, B., Desell, T.: Large-scale data analysis for proactive anomaly detection in heterogeneous aircraft data. Unpublished
2. National General Aviation Flight Information Database: Welcome to the national general aviation flight information database (NGAFID)
3. MITRE: Gaard–general aviation airborne recording device
4. Clachar, S.A.: Identifying and analyzing atypical flights using supervised and unsupervised approaches. Journal of the Transportation Research Board (2014) Published as part of an ACRP: Graduate Research Award.
5. Clachar, S.: Novelty Detection and Cluster Analysis in Time Series Data Using Variational Autoencoder Feature Maps. PhD thesis, University of North Dakota (December 2016)
6. Desell, T., Clachar, S., Higgins, J., Wild, B. In: Evolving Deep Recurrent Neural Networks Using Ant Colony Optimization. Springer International Publishing, Cham (2015) 86–98
7. ElSaid, A., Wild, B., Higgins, J., Desell, T.: Using LSTM recurrent neural networks to predict excess vibration events in aircraft engines. In: The IEEE 12th International Conference on eScience (eScience 2016), Baltimore, Maryland, USA (October 2016)
8. ElSaid, A.: Using long-short-term-memory recurrent neural networks to predict aviation engine vibrations. Master's thesis, University of North Dakota (December 2016)

9. Desell, T., Clachar, S., Higgins, J., Wild, B. In: Evolving Neural Network Weights for Time-Series Prediction of General Aviation Flight Data. Springer International Publishing, Cham (2014) 771–781
10. X-Plane: X-plane. More powerful. Made usable.
11. Analytical Graphics, Inc., Bentley Systems: Cesium
12. AOPA: What is general aviation (2009)
13. Allen, W.B., Blond, D.L., Gellman, A.J., Association, G.A.M., of State Aviation Officials (U.S.), N.A., MergeGlobal, I.: General aviation's contribution to the u.s. economy, General Aviation Manufacturers' Association (May 2006)
14. Federal Aviation Administration: The economic impact of civil aviation on the U.S. economy (November 2016)
15. Kenny, D.J.: 25th Joseph T. Nall report: General aviation accidents in 2013. Technical report, AOPA Air Safety Institute, 421 Aviation Way, Frederick, MD 21701 (2016)
16. Shetty, K.I., Hansman, R.J.: Current and historical trends in general aviation in the united states. Master's thesis, Massachusetts Institute of Technology (August 2012)
17. AOPA Air Safety Institute: 2014-2015 GA accident scorecard. Technical report, AOPA Air Safety Institute, 421 Aviation Way, Frederick, MD 21701 (2016)
18. UND Aerospace Foundation: Cessna 172S Standardization Manual. (Aug 2015)
19. Jr., E.H., Bloedorn, E., Rothleder, N.J.: Recent experiences with data mining in aviation safety. In: SIGMOD Record, Seattle, WA (June 1998)
20. Federal Aviation Administration: Runway safety: Runway incursions
21. Matthews, B., Das, S., Bhaduri, K., Das, K., Martin, R., Oza, N.: Discovering anomalous aviation safety events using scalable data mining algorithms. Journal of Aerospace Information Systems **10**(10) (2013) 467–475
22. Wischmeyer, E.: The myth of the unstable approach. International Society of Air Safety Investigators (August 2004)
23. Nazeri, Z., Donohue, G., Sherry, L.: Analyzing relationships between aircraft accidents and incidents. In: International Conference on Research in Air Transportation. (Feb 2008)
24. CloudAhoy: Cloudahoy: debriefing for pilots
25. Nazeri, Z., Bloedorn, E., Ostwald, P.: Experiences in mining aviation safety data. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data. SIGMOD '01, New York, NY, USA, ACM (2001) 562–566
26. Gallo, D.E.: Data Mining Applied to Aviation Data. PhD thesis, Universidad Politécnica de Madrid (June 2012)
27. Pagels, D.A.: Aviation data mining. Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal **2**(1) (2015) 3
28. Cessna Aircraft Company: Pilot's Operating Handbook and FAA Approved Airplane Flight Manual: Cessna Model 172S. 2nd edn. (November 2010)
29. Federal Aviation Administration: Airplane Flying Handbook 2nd Edition: FAA-H-8083-3A. Skyhorse Publishing Inc. (2011)
30. Beazley, D.: Understanding the Python GIL. In: PyCON Python Conference. Atlanta, Georgia. (2010)
31. Nolan, D., Lang, D.T. In: Keyhole Markup Language. Springer New York, New York, NY (2014) 581–618