Optimized Flight Safety Event Detection in the National General Aviation Flight Information Database

Aidan P. LaBella, Joshua A. Karns, Farhad Akhbardeh, Travis Desell
Rochester Institute of Technology
Rochester, New York
{apl1341,jxkvse,fa3019,tjdvse}@rit.edu

Andrew J. Walton
Liberty University
Lynchburg, Virginia
awalton2@liberty.edu

Zechariah Morgan, Brandon Wild, Mark Dusenbury
University of North Dakota
Grand Forks, North Dakota
{zechariah.morgan,brandon.wild,mark.dusenbury}@und.edu

ABSTRACT

This work presents a redesign of the National General Aviation Flight Information Database (NGAFID) to allow swift ingestion of flight data and calculation of flight safety events, as it is rapidly growing and currently housing over 750,000 hours of flight data. This redesign reduced memory and storage usage by a factor of 5, as well as reducing flight data import and event calculation time from over 3 minutes to 0.3 seconds by using a per-parameter compressed data representation in the database and a new pipelined data ingestion strategy. In addition to this, four new advanced flight safety event calculations were developed for proximity, self-defined glide path deviation, stall and loss of control in flight (LOC-I). A time and location bounded flight matching strategy was used to calculate proximity events for flights in on average 3.2 seconds per flight in the full 660,000 flight database. The stall and LOC-I event calculations show strong improvements upon prior work in the area on both a test flight flown for accurate stall time measurements and a historical sample of data independently annotated by two subject matter experts. The enhancements to the stall and LOC-I event calculations resulted in an increase in stall event detection from 23.1% to 83.1% in the historical data while reducing false negatives by a factor of 3. As the leading cause of aviation accidents worldwide is LOC-I, it is an especially important issue for flight safety monitoring programs, especially as other existing flight data monitoring (FDM) software lacks these new stall, LOC-I and proximity event detection capabilities.

CCS CONCEPTS

• **Applied computing** → **Aerospace**; • **Information systems** → **Information storage systems**; • **Software and its engineering** → **Software organization and properties**;

KEYWORDS

Computational Aviation Safety, Hazard Detection, Time Series Analysis, Big Geospatial Temporal Data

ACM Reference Format:
Aidan P. LaBella, Joshua A. Karns, Farhad Akhbardeh, Travis Desell, Andrew J. Walton, and Zechariah Morgan, Brandon Wild, Mark Dusenbury. 2022. Optimized Flight Safety Event Detection in the National General Aviation Flight Information Database. In *The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22), April 25–29, 2022, Virtual Event.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3477314.3507115

1 INTRODUCTION

In the United States, general aviation (GA) has historically been and continues to be relatively dangerous compared to other modes of transportation [6], with fatal accidents being much more common than in commercial aviation. Still, thanks to advancements in the field yearly fatalities have steadily decreased for the past four decades [6]. In an attempt to further improve the safety of general aviation, a wide array of flight data monitoring programs have been set up to allow pilots and instructors to view flight data retroactively and identify potentially dangerous flight conditions. According to a 2018 fact sheet published by the Federal Aviation Administration, loss of control events in flight (LOC-I), and in particular stall events, accounted for the vast majority of fatal incidents [4].

Loss of control in-flight (LOC-I) is a category of safety occurrence defined as "an extreme manifestation of a deviation from intended flight path" [10]. LOC-I is a broad category which is not strictly limited to stall and aerodynamic spin accidents. Nevertheless, analysis by Walton *et al.* [11] found that in instructional aviation, stall and spin accidents contributed to 81% of fatal LOC-I accidents, and the aviation safety community has therefore placed great emphasis on stall and spin prevention as a primary mitigation strategy. Thus, it is imperative that flight data monitoring programs provide tools to aide in the detection of such events.

One such flight data monitoring program is the United States National General Aviation Flight Information Database (NGAFID), which serves as a repository for GA flight data, with a web portal for viewing and tracking flight safety events for individual pilots as well as for fleets of aircraft. The NGAFID currently contains over 750,000 hours of flight data generated by 660,000 flights by 12 different types of aircraft, provided by 65 fleets and individual users, resulting in over 2.6 billion per second flight data records across 103 potential flight data recorder parameters. This database continues to grow daily as more data is uploaded. As the number of participating organizations and quantity of collected flight data has grown, a number of computational, memory and storage inefficiencies and bottlenecks swiftly became apparent in the prototype version of the NGAFID, and this work presents a performance based redesign of the system.

This work also presents new tools to aide in the detection of dangerous proximity, self-defined glide path deviation, stall and LOC-I events. To ensure the validity of the tools, the findings of the new tools are compared to those of flight safety subject matter experts (SMEs). Moreover, in order to provide a ground truth for comparison to the tool's findings, a video-recorded test flight was performed during which the flight crew performed a variety of different maneuvers, including stalls. Results present the performance improvements over the prior NGAFID implementation as well as greatly improved accuracy of stall and LOC-I calculations over prior state-of-the-art.

2 RELATED WORK

Much existing literature on flight data analysis aims to improve safety by improving retroactive feedback on flights. The detection of flight irregularities (*i.e.*, events and anomalies) is one common approach to this. In their 2019 work, Fernández *et al.* [5] use a long short-term memory (LSTM) based auto-encoder to detect anomalies during flight approaches, classifying abnormal flights with 74% accuracy. To aid in the analysis of specific flight phases, Liu *et al.* [8] developed a tool based on machine learning to identify different phases of flight. They trained a Gaussian mixture model using data from a flight data recorder which was able to assign the correct flight phase 90% of the time on average. Karboviak *et al.* [7] utilized data from the NGAFID to develop a "Go-Around Detection Tool," which classifies aircraft approaches as either a *go-around*, *touch-and-go*, or *stop-and-go*. Approaches are also categorized as either stable or unstable using similar means.

In regards to LOC-I, in a recent work by Harris *et al.* [6], data from the publicly available National Transportation Safety Board (NTSB) accident database was used to train a predictive regression tree model. The model predicts the efficacy of an automatic ground collision avoidance system (Auto GCAS). Their model found that out of 2302 fatal events, there were 530 deaths that could have likely been prevented with an Auto GCAS system, 126 of which were LOC-I events.

In 2017, Balogh introduced equations to calculate two values, referred to as loss of control in-flight (LOC-I) and stall probabilities, which served as the basis for these calculations in this work [2]. While these values aim to provide a statistical probability that represents the likelihood that one of these events will occur in flight, high values do not guarantee that an aircraft is certain to experience one. Rather, they are an aggregated measure of crucial flight parameters, all of which greatly affect the chances an aircraft will enter a stall or loss of control event. As such, the nomenclature was changed from 'probability' to 'index' for all purposes in the NGAFID. The stall index utilizes recorded G1000 (a flight data recorder) parameters to derive an estimated angle of attack and compare that to the critical angle of attack for the aircraft, allowing FDM analysts to see a calculated stall index. The LOC-I Index compares the stall index with a derived coordination index in order to show when the airplane was in a flight condition conducive to entering a spin.

3 ARCHITECTURE

The NGAFID 2.0 utilizes a data ingestion pipeline which separates the initial ingestion of uploaded data into the database from event calculation. First, NGAFID users upload flight data as zip files which contain multiple comma separated value (CSV) files, one file per flight. This upload process is resumable, as these files tend to be quite large, and uploading them as zip files reduces both upload time as well as storage usage on the virtual machine hosting the NGAFID. In addition to this new pipeline, the database was redesigned, as described in Section 3.1. The new database schema led to performance gains and enhanced storage efficiency, as well as improving the time required for event calculation, described in Section 3.3.

3.1 Database Redesign

The preliminary version of the NGAFID utilized a singular large table to handle all flight data records, where each row in the table was a per-second flight data recorder reading from a flight, with each column corresponding to a particular flight data recorder (FDR) parameter. This became problematic when more aircraft types were added to the database, as different aircraft have different FDR parameters. For example, Cessna 172s have a single engine and Piper Archer 44s have two engines, leading to more engine parameters. As a result, the table needed to be expanded to incorporate additional columns that were not yet present. This design also led to significant performance issues when performing queries, as multiple rows for particular flight parameters would need to be queried from the database to calculate the various tracked events.

Figure 2 presents a schema for the database redesign made to this singular flight data table as well as additions for improved event calculation. A major modification was that each column of data from a flight file is stored as an entry in either an `StringSeries` table for textual data, or `DoubleSeries` table for numerical data. These entries contain the name of the column, along with its given data type from the CSV (*e.g.*, feet, temperature, mph, date, etc.), and for the numerical data the minimum, average and maximum values which are calculated during the initial data ingestion step. The actual column values are stored as an array of bytes in a `BLOB`, which allowed for storing this data in a compressed format providing a significant reduction in storage as well as faster data ingestion, as I/O was the most significant bottleneck.

3.2 Compressed Archival Data

The columns contained in aforementioned flight files are very compressible and often repeat values many times, making them prime candidates for compression. As mentioned previously these columns are converted to either `StringSeries` or `DoubleSeries`. They are subsequently serialized into byte arrays and inserted into the database as BLOBs. Adding a compression step here was trivial: the byte arrays can be compressed before being inserted into the database as a BLOB, and inflated when they are being read from the database. The compression algorithm used is the one provided by the Java standard library, which is backed by zlib. While adding a

Figure 1: NGAFID 2.0's flight data ingestion pipeline.

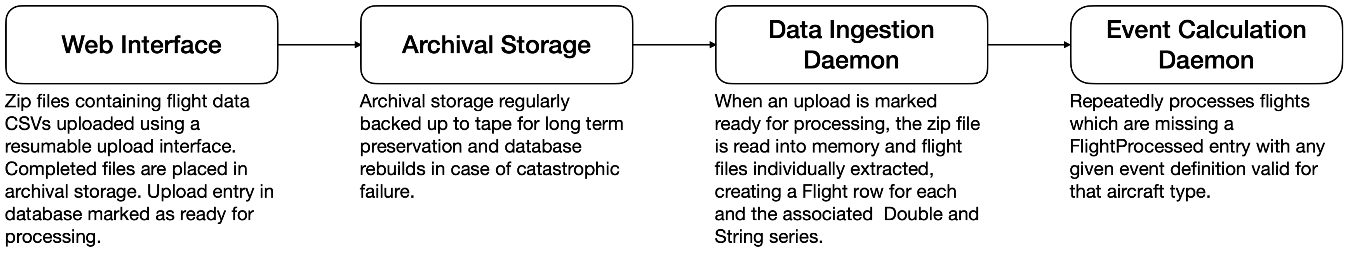
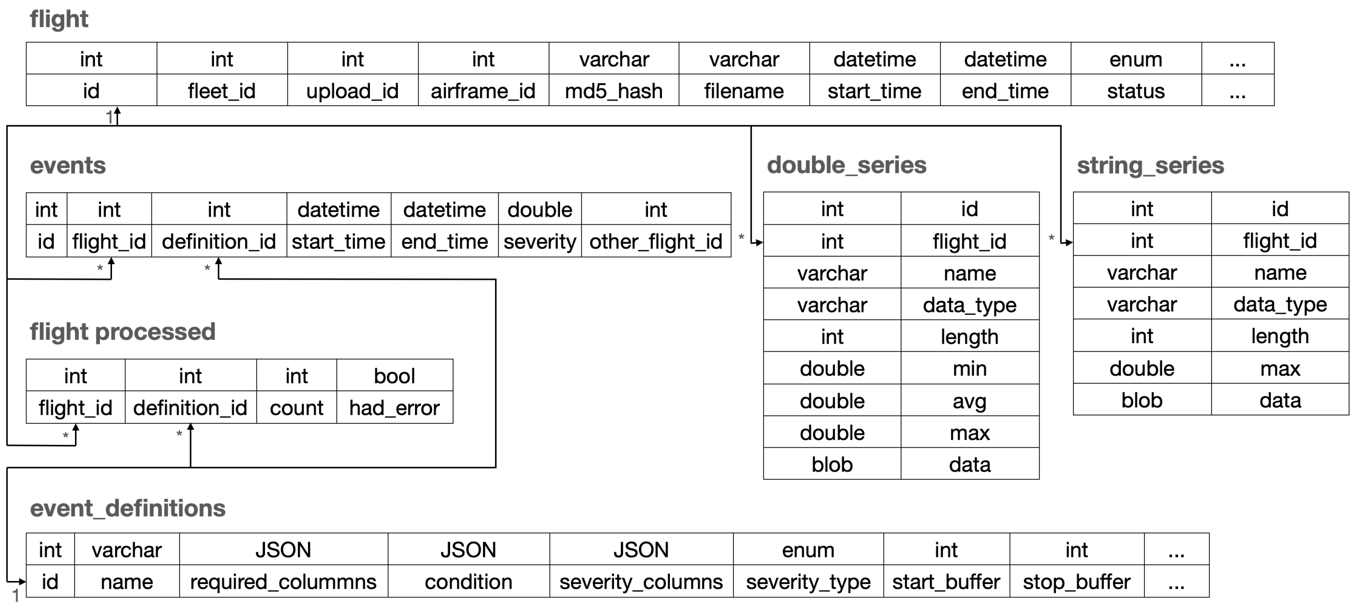

Web Interface
Archival Storage
Data Ingestion Daemon
Event Calculation Daemon
Zip files containing flight data CSVs uploaded using a resumable upload interface. Completed files are placed in archival storage. Upload entry in database marked as ready for processing.
Archival storage regularly backed up to tape for long term preservation and database rebuilds in case of catastrophic failure.
When an upload is marked ready for processing, the zip file is read into memory and flight files individually extracted, creating a Flight row for each and the associated Double and String series.
Repeatedly processes flights which are missing a FlightProcessed entry with any given event definition valid for that aircraft type.

Figure 2: Partial schema detailing the redesign to store data as compressed BLOBs and to process and track events generically.


flight
int | int | int | int | varchar | varchar | datetime | datetime | enum | ...
id | fleet_id | upload_id | airframe_id | md5_hash | filename | start_time | end_time | status | ...
events
int | int | int | datetime | datetime | double | int
id | flight_id | definition_id | start_time | end_time | severity | other_flight_id
flight processed
int | int | int | bool
flight_id | definition_id | count | had_error
double_series
int id; int flight_id; varchar name; varchar data_type; int length; double min; double avg; double max; blob data
string_series
int id; int flight_id; varchar name; varchar data_type; int length; double max; blob data
event_definitions
int | varchar | JSON | JSON | JSON | enum | int | int | ...
id | name | required_colummns | condition | severity_columns | severity_type | start_buffer | stop_buffer | ...
1
*
*
*
*
*
*
1

compression or inflation step does incur a computational overhead, the results in Figure 4 show that the reduced amount of disk usage lead to a speedup in flight processing.

3.3 Improved Event Calculation

The original NGAFID design calculated flight events, or *exceedences*, which represent potential flight or mechanical issues, as data was imported into the database. Most of these represent rules established by the Federal Aviation Administration (FAA) or system limits specified by the aircraft manufacturers, such as exceeding or going below airspeed; pitch or roll limits; or low oil pressure or low fuel. As new aircraft were added and events types were modified or added, this design was unsustainable as it resulted in an expensive and complicated process of determining which events needed to be calculated or recalculated for which aircraft. Additionally, it limited the addition of new advanced events presented in this work.

After the flight data has been imported to the database, an *Event Calculation Daemon* queries for the existence of rows by flight and event id in the `FlightProcessed` table to determine if a particular flight needs to be processed for a given event. This allows new events to be added dynamically with flights being automatically reprocessed. It also now generically processes the vast majority of flight events – all of them except for proximity events, as that requires data from two flights.

The database redesign contains an `EventDefinition` table, consisting of JSON objects that specify which flight data columns are required to calculate when an event is triggered in a flight (see Table 3). These consist of (potentially nested) conditionals of flight data columns and predetermined values along with their associated binary operators, *e.g.*, in Table 3, a *Engine Shutdown Below 300 ft* event is triggered when both an aircraft is between 500 and 3000 ft above ground level (AG), and either engine 1 (E1) or engine 2 (E2) RPM is below 100.

As the flight data is now stored by individual columns entries in the `StringSeries` and `DoubleSeries` tables, and the `DoubleSeries` table having pre-computed min and max values, for many events

Algorithm 1 Generic Event Detection

```
Input: Array of Required Data Columns: $\mathcal{RC}$
Input: Array of Severity Data Columns: $\mathcal{SC}$
Input: Event Trigger Conditional: $\mathcal{T}$
Input: Severity Calculation Function: $\mathcal{S}$
Input: Number of seconds of a trigger required to start an event: startBuffer
Input: Number of seconds without a trigger to stop an event: stopBuffer
Output: A list of events: $\mathcal{L}$

▷ Variables to track current event state
Event $\mathcal{E} \longleftarrow \varnothing$
startCount $\longleftarrow 0$
stopCount $\longleftarrow 0$

▷ Iterate over data elements to find events
for $l \longleftarrow 1$ to LENGTH($\mathcal{RC}[0]$) do
    ▷ Evaluate the conditional on the lth element of all the required data columns
    if $\mathcal{T}$.TRIGGERED($l, RC$) then
        ▷ Row triggered the event
        if $\mathcal{E} = \varnothing$ then
            ▷ Create a new event starting on this row
            $\mathcal{E} \longleftarrow$ new Event(l)

        ▷ Update the event's severity if it is worse than previously found
        $\mathcal{E}$.UPDATESEVERITY(l, $\mathcal{S}$, $\mathcal{SC}$)

        ▷ Increment the start count and reset the stop count
        startCount $\longleftarrow$ startCount +1
        stopCount $\longleftarrow 0$
    else
        if $\mathcal{E} \neq \varnothing$ then
            ▷ An event is being tracked
            stopCount $\longleftarrow$ stopCount +1
            if stopCount = stopBuffer then
                if startCount < startBuffer then
                    ▷ Not enough triggers to start an event, discard it
                else
                    ▷ Had enough triggers for an event and it has ended
                    $\mathcal{E}$.SETENDROW($l$)
                    $\mathcal{L}$.PUSH($\mathcal{E}$)
                $\mathcal{E} \longleftarrow \varnothing$
                startCount $\longleftarrow 0$
                stopCount $\longleftarrow 0$

Output: $\mathcal{L}$
```

the series data does not even need to be pulled from the database. If the flight data parameter(s) related to the event never exceed the specified threshold for that event, the event could not possibly have existed for that flight. This significantly speeds up the analysis process for the flight data, as flight data is only pulled from the database and flight events are calculated when they are possible.

Additionally, the preliminary NGAFID only specified which time steps in the flight data violated the event conditions, which did not provide an accurate view of when particular events actually occurred. For example, if an aircraft's altitude fluctuated above and below the threshold for a period of time, every time it dropped below the threshold and then back up would result in a new event – however that entire period would generally be viewed as a single incident. Using a *start buffer* which specifies how many seconds the data must violate event conditions for an event to occur, and a *stop buffer* which specifies how many seconds the data must satisfy the event conditions for the event to end, time periods involving the events could be more accurately presented. Algorithm 1 presents the generic algorithm for event detection, where the `triggered` method determines if the flight data at a particular time meet the conditions for the event.

Figure 3: Example JSON for event trigger conditionals. Groups and rules can be nested, allowing for any binary logic expression involving the data columns. The above will trigger events when the aircraft is between 500 and 3000ft above ground level, and the RPM from either engine drops below 100.

```
{   "name" : "Engine Shutdown Below 3000 Ft",
    "type":"GROUP",
    "condition":"AND",
    "filters":
        [{
            "type" : "RULE",
            "inputs" : ["AltAGL",">","500"]
        }, {
            "type" : "RULE",
            "inputs" : ["AltAGL","<","3000"]
        }, {
            "type":"GROUP",
            "condition" : "OR",
            "filters" :
                [{
                    "type" : "RULE",
                    "inputs" : ["E1 RPM","<","100"]
                }, {
                    "type" : "RULE",
                    "inputs" : ["E2 RPM","<","100"]
                }]
        }]
}
```

4 FLIGHT PROCESSING METHODS

Historically, events calculated by the NGAFID measure when certain metrics, tracked by the aircraft's flight data recorder (FDR), deviate from the envelope of values that would be considered safe and routine operation of the aircraft. This work introduces the process for calculating four new event types: *proximity*, *self-defined glide path deviation*, *stall*, and *loss of control in flight* (LOC-I), which represent some of the potentially most dangerous in flight events. These new events add additional complexity to the calculation process, as proximity events require data from two flights simultaneously, self-defined glide path requires knowledge of aircraft proximity to airport runways, and the stall and LOC-I events require calculations based on new time series data based on parameters derived from the flight data.

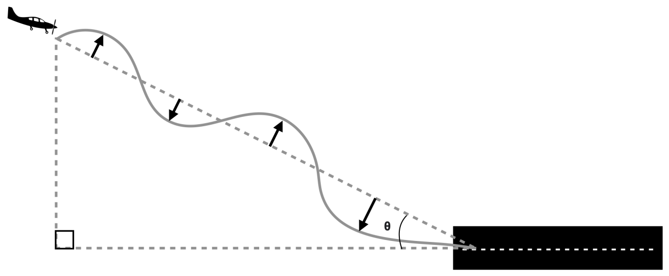
4.1 Proximity Events

A naive approach to calculating proximity would be $O(N^2)$ as each flight would need to be compared to every other flight to determine if they were within 500 feet of each other at the same moment in time while in the air 50 feet above ground level, to prevent false positives where aircraft are taxiing or in a hangar. This is prohibitively expensive as the NGAFID is currently scaling to millions of flights. However, the new database design allows for significantly improved detection of proximity events. As the entries in the `Flights` table contain the start and end time of the flight, it is first possible to select flights which could possibly be in the air at the same time: where s_i and e_i are the start and end times of the flight being processed, and s_j and e_j are the start and end times of potential matches, time matched flights can be found querying for $s_j \leq e_i \land e_j \geq s_i$. Following this, those flights can then be further filtered by latitude and longitude, in a similar fashion, as only flights with $(min(lat_j) \leq max(lat_i)) \land (max(lat_j) \geq$

$min(lat_i)) \wedge (min(lon_j) \leq max(lon_i)) \wedge (max(lon_j) \geq min(lon_i))$, given the precomputed minimum and maximum latitude values. Each of these query parameters are indexed as b-trees in the mySQL database to further improve performance. Given this filtered subset of flights, the proximity events can then be calculated for matching flights by skipping forward in time to the beginning of flight with the later start time and then for each moment in time calculating the two flights proximity given their altitude (above sea level), latitude and longitude, with a distance under 500 ft triggering the event when both flights have an altitude (above ground level) higher than 50 feet.

4.2 Self-Defined Glide Path Analysis

Figure 4: Geometry used to calculate the self-defined glide path, glide path angle, and glide path deviations.



When an aircraft approaches a runway for landing, the rate of descent should remain constant for the duration of the landing sequence. However, the exact rate of descent a pilot should aim for will change based on flight conditions, aircraft type, and geographic constraints. A self-defined glide path allows for safety analysis of flight landings regardless of these variables.

The self-defined glide path of an aircraft during landing sequence is a line formed by two points (depicted in Figure 4): the position of the aircraft when it first reaches 300 ft in altitude above the ground, and the point where the aircraft first touches the runway. This line forms the self-defined glide path, and deviations from the glide path are determined by calculating the minimum distance between the aircraft at a given point in time and the glide path. Extreme deviations from the self-defined glide path are useful indicators for flight analysts to flag unsafe flights. The self-defined glide path is also the resultant of two orthogonal vectors, one of which represents the altitude of the aircraft above the runway, with the other representing the horizontal distance to the runway threshold, as depicted in Figure 4. The self-defined glide path angle θ formed between the base of the triangle and the self-defined glide path is another useful metric for flight analysts, as extreme angles can be dangerous.

These calculations are done during the flight import process and stored in the database. This data is then used by the turn to final analysis tool, depicted in Figure 5. The tool is intended to be used for aggregated analysis of data over months. Data can be fetched by date, airport, and runway, and can also be filtered by a minimum roll value.

4.3 Preliminary Stall and Loss of Control In Flight (LOC-I) Indices

With the new architecture of the NGAFID, the processing daemon now adds additional derived columns of time series data, which are then be used to calculate stall and loss of control events. The initial loss of control in-flight (LOC-I) and stall indices which form the basis of this work comes from Balogh [2], which provided a methodology for first calculating angle of attack (AOA) and then using that to calculate an index representing the potential for a flight to enter a stall, and finally an index representing the potential for a flight to experience a loss of control (LOC-I) event. This methodology was refined using feedback from SMEs who have an extensive background in flight data monitoring.

From Balogh [2], the derived angle of attack, and the stall and loss of control indexes can be calculated as follows. For each FDR recording (second) of a flight, i, a density ratio ρ_i is calculated using the FDR's outdoor air temperature (τ, in degrees Fahrenheit) and barometric pressure recordings (ω, in inches of mercury). The constants 29.92 and 288 are the standard pressure and temperature values, respectively:

$$\rho_i = \left(\frac{\frac{\omega_i}{29.92}}{\frac{\tau_i + 273}{288}} \right) \tag{1}$$

which is used along with the FDR indicated airspeed (δ_i, in knots), to calculate true airspeed (γ_i, in knots), where the constant 101.27 is a conversion factor:

$$\gamma_i = 101.27 * \left(\frac{\delta_i}{\sqrt{\rho_i}} \right) \tag{2}$$

The density ratio ρ_i, true airspeed (γ_i, in knots), FDR vertical speed (ν_i, in feet per second) and FDR Pitch (α_i, in degrees), are used to calculate the angle of attack (β_i, in degrees):

$$\beta_i = \alpha_i - \left(\frac{180^\circ * \arcsin \frac{\left(\frac{\nu_i}{\rho_i}\right)}{\gamma_i}}{\pi} \right) \tag{3}$$

Then the stall index, ϕ_i, is calculated by determining how close the calculated angle of attack is to the critical angle of attack, $\beta_{critical}$, which is the angle of attack the manufacturer states the aircraft will enter a stall. The stall index is capped at a maximum value of 1.0:

$$\phi_i = \min \left| \frac{\beta_i}{\beta_{critical}} \right| 1.00 \tag{4}$$

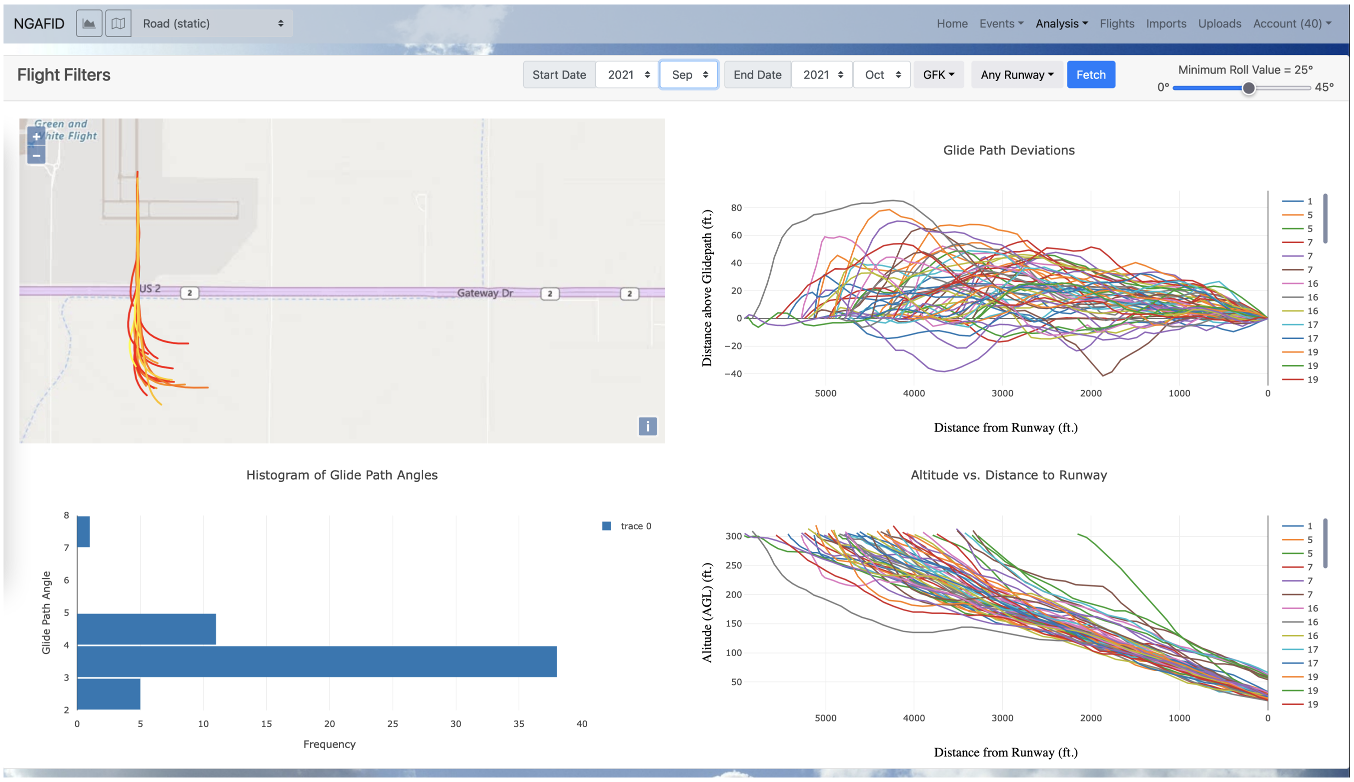
A yaw rate value, λ_i is calculated using the FDR heading value, κ_i:

$$\lambda_i = |180^\circ - |180^\circ - |\kappa_i - \kappa_{i-1}| \mod 360^\circ|| \tag{5}$$

which is used along with the FDR's roll, ψ_i to calculate a coordination index, χ_i, which is a measure of how far off the direction of the aircraft is (the yaw) is from the heading (the direction the aircraft is actually traveling):

$$\chi_i = \min \frac{100 \left(\frac{\pi \gamma_i \lambda_i}{10800} - 32 \sin \left| \frac{\pi \psi_i}{180^\circ} \right| \right)}{4} 1.00 \tag{6}$$

Figure 5: The turn to final analysis tool, which provides users with information about the self-defined glide path, glide path angle, and displays flight landings on the map.



which is then finally used to calculate the loss of control index (LOC-I), σ_i, for that point in time in the flight:

$$\sigma_i = \frac{\phi_i \chi_i}{100} \tag{7}$$

4.4 Refinements to the Stall and Loss of Control Indices

A preliminary review of randomly selected flights from the NGAFID showed that the initial version of the algorithm did a good job of identifying upset events after a stall had occurred. However, the SimpleAOA output seemed to lag behind the aerodynamic performance of the aircraft raising questions about whether the algorithm was correctly identifying the precursors to an aircraft stall, or only identifying the recovery from a stall, after an aircraft upset had occurred.

After analyzing the underlying G1000 parameters, two possible sources of error were identified in the stall index. One was that the original formula relied upon vertical speed information (VSI), which the FAA states can lag by 6-9 seconds. Additionally, the original formula used indicated airspeed rather than calibrated airspeed, meaning that high angles of attack would result in instrument errors which the formula was not accounting for.

4.4.1 Deriving Vertical Speed (VSI). In conversations with our subject matter experts and as detailed in the FAA's Pilot's Handbook of Aeronautical Knowledge [1], vertical speed FDR recordings can be lagged up to 9 or more seconds depending on flight conditions. As this lag is not possible to determine *a priori*, a modification to the previously published method was examined which instead uses a derived vertical speed, v'_i, in place of the FDR's vertical speed value in calculating the density ratio (Equation 3). v'_i is calculated by selecting the slope from a linear regression that is done over the previous, current and future Altitude MSL values, $\Upsilon_{i-1,i,i+1}$:

$$v'_i = linreg(\Upsilon_{i-1}, \Upsilon_i, \Upsilon_{i+1}) \tag{8}$$

4.4.2 Adjusting Indicated Airspeed. FDR recorded indicated airspeed does not account for instrument error for values below 70 in the Cessna 172 aircraft, the subject of this study. This was corrected for by taking a linear regression of the suggested indicated airspeed adjustments from the Cessna 172 manual [3], providing the following adjusted airspeed, δ'_i:

$$\delta'_i = \begin{cases} .70\delta + 20.667 & \delta < 70 \\ \delta & \delta \geq 70 \end{cases} \tag{9}$$

which is used in place of the indicated airspeed value δ_i in the true airspeed calculation (Equation 2).

4.4.3 Multiple Versions of LOC-I and Stall Indices. To demonstrate the improvements in accuracy when such changes are applied, three versions of the LOC-I and stall index formation were defined to analyze the change in results. Version 1 (V1) refers to the original calculation seen in Balogh's work [2]. Version 2 (V2) introduces the altitude-derived vertical speed reading, seen in Equation 8, in lieu of the FDR's vertical speed data. Version 3 (V3) incorporates the same changes seen in V2 plus an adjusted airspeed for lower airspeed ranges, seen in Equation 9. Stall and LOC-I events are triggered when the stall or LOC-I index reaches a value of at least 1.0 for 2 seconds, and these are differentiated by if the aircraft is at or above 1500 feet for high altitude or below 1500 feet for low altitude.

4.5 Stall and LOC-I Gradients and Metric Viewer

An important feature included in this version of the NGAFID is the ability for users to visualize the stall and LOC-I indices on a map using a gradient (see Figure 6). This helps users pin-point the location where an index reached a concerning value by using a color-coded scale from green (normal) to red (severe). In the event a user notices a severe portion of the flight path with this gradient, they have been given the ability to click anywhere on the flight path once the gradient viewing mode is activated. This renders a popup containing the flight metric readings at such given index, allowing for further investigation with the aid of additional parameters. Users are able to change which metrics are displayed, as well as the number of significant figures the metrics are rounded to. In addition, if the flight has at least one event at the given index, there will be an option for the user to view the description of such events.

5 COLLECTION OF EXPERIMENTAL DATA

5.0.1 Experimental Test Flight Data. One challenge in evaluating the accuracy of the angle of attack, stall and loss of control indexes is that they depend on an accurate vertical airspeed (VSI) recording which can be lagged by an indeterminate amount. In order to gather accurate ground truth data, a video-recorded test flight was performed to test that these refinements would more accurately flag stall and LOC-I events. The purpose of this test flight was to compare the timing of the stall indexes with when the aircraft actually stalled, as indicated by the video.

After obtaining the necessary organizational approvals, a GoPro was temporarily mounted inside the cockpit of a Cessna 172 equipped with a factory installed SafeFlight angle of attack indicator. The angle of attack indicator installed by Cessna uses a transducer on the leading edge of the wing to measure the aerodynamic stagnation point of the airflow around the wing, thus providing highly repeatable angle of attack information to the pilot.

The flight crew conducted a variety of slow flight and stall events, as well as takeoffs and landings, to provide a dataset of the types of maneuvers found during our initial investigation of stall and LOC events from the initial methodology. The recorded test flight contains multiple intentional stalls, including stalls with slow entries, rapid entries, extended recoveries, and imminent stalls with a smooth recovery with minimal VSI fluctuations. The first two stalls involved very slow decelerations, to see if the algorithm would correctly note the slowly increasing angle of attack. The airplane was then stalled in a variety of aerodynamic configurations, including combinations of flaps up, flaps down, power on, power off, slipping, and skidding; in order to later analyze the performance of the stall index in the full spectrum of flight configurations expected of a non-aerobatic aircraft.

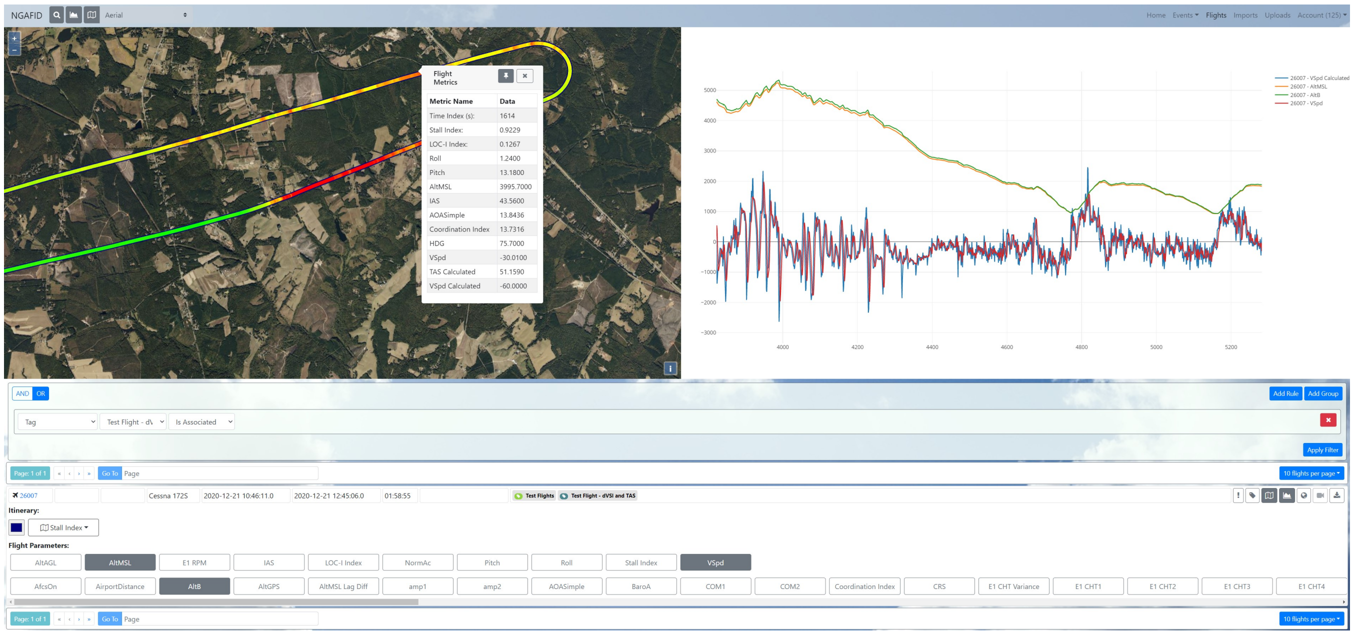
The flight crew also initiated stall recovery from different points. In some cases, recovery was performed very early, as soon as the stall warning was activated, to see if the derived angle of attack would correctly show that the aircraft had not reached critical angle of attack on that maneuver. Some stalls were recovered from an imminent stall (the aircraft was taken right to the edge of its critical angle of attack without fully stalling the airplane). If the stall index was working correctly, it should be calculated at 1.0 even though there were no large fluctuations in vertical speed. Normal recoveries (performed immediately upon exceeding the critical angle of attack) were performed, as well as delayed recoveries, to see if the stall index stayed at 1.0 longer in cases where the airplane was stalled longer. Falling leaf stalls were also conducted, where the airplane was deliberately kept at or beyond critical angle of attack for an extended period of time, to see if the stall index continue to read 1.0 in this situation with power off, partial power, and high power configurations. A series of accelerated stalls were also conducted where the airplane stalls at a higher g-force and higher airspeed, to make sure that the stall index would also read 1.0 at higher indicated airspeeds. Finally, a series of takeoffs and landings were performed using normal, short field, and forward slip techniques to ensure that the stall index would not generate too many false positives during routine training operations.

The pilot involved then reviewed the video footage, annotating the maneuvers performed and noting the timestamps in the video. The pilot provided two metrics: the aircraft's stall horn and whether the aircraft was aerodynamically stalled. Using the video from the aircraft's cockpit, the pilot indicated in two separate columns, as booleans, whether the stall horn was on and whether the aircraft was behaving in a manner consistent with aerodynamic stall. Almost always, the aircraft's stall horn, which is used to alert pilots to an imminent stall, will trigger before the aircraft reaches its critical angle of attack ($\beta_{critical}$). Since there is a gap in time from when the stall horn initially activates and when the critical angle of attack is exceeded, an acceptable range for when the stall begins was defined as the time from when stall horn first activates to when the aircraft first displayed aerodynamic cues consistent with exceeding the critical angle of attack. This delay had a sample average of $\overline{\Delta_t} = 8.3s$. As for determining the end of a stall event, it was decided that when the stall horn cuts out is the best indicator of when the danger of a stall is no longer present.

Then the video footage was matched to the G1000 FDR data, by going through the video file, noting each second whether the aircraft stall horn was audible (indicating a high angle of attack), as well as noting the time that the pilot believed that the aircraft was at or beyond the critical angle of attack.

5.0.2 Random Sample of NGAFID Flights. A second sample of data was collected using existing flights already in the NGAFID. The sample consisted of 23 flights, each with a varying number of stall and/or loss of control events. Flights were selected such that at least one of the stall or LOC-I events was triggered.

Figure 6: Flight Path with gradient based on the stall index and the new flight metric viewer, with a side by side data series plot.



Flight Metrics

Metric Name	Data
Time Index (s):	1614
Stall Index:	0.9229
LOC-I Index:	0.1267
Roll	1.2400
Pitch	13.1800
AltMSL	3995.7000
IAS	43.5600
AOASimple	13.8436
Coordination Index	13.7316
HDG	75.7000
VSpd	-30.0100
TAS Calculated	51.1590
VSpd Calculated	-60.0000

26007 Cessna 172S 2020-12-21 10:46:11.0 2020-12-21 12:45:06.0 01:58:55 Test Flights Test Flight - dVSI and TAS

6 RESULTS

6.1 Stall and LOC-I Accuracy on the Experimental Test Flight

Table 1 presents the results comparing the three methods (V1, V2 and V3) described in Section 4.3 to the ground truth data provided by the test pilots. From the test pilot annotations, if the calculated stall index reached a value of 1.0 in the timeframe between the stall horn activating and the aircraft exceeding the critical angle of attack, it was marked as valid (in bold). The end times were marked as valid (in bold) if the calculated end time fell within ±5s of the time the stall horn was last heard. Deviations from the valid ranges are show in parenthesis. Event #30 represents a false positive reported by V2 and V3. This table shows that moving from V1 to V3 greatly increases the valid start times from 37.9% to 93.1% and the valid end times from 72.4% to 93.1%. V2 performs better for end times, however this is within the range of error given the average absolute differences and standard deviations.

6.1.1 *LOC-I Index Results for Experimental Flight.*

Since LOC-I values are based on stall index values, plus a coordination factor (as demonstrated in Equation 6), these values were rated on a binary scale of either accurate or inaccurate. SME's used their knowledge of aerodynamics to determine if in fact a LOC-I event was imminent. As such, one pilot from the experimental flight rated every event using this scale with 30 out of 32 readings determined to be accurate.

6.2 Stall and LOC-I Accuracy on the NGAFID Flights

In order to gather ground truth data from the extracted NGAFID flights, both SMEs were tasked with finding start and end times for each stall events before being shown the stall and LOC-I calculations. To ensure these results were independent and bias-free, neither SME had access to the others annotations. It should be noted that for these flights, the SMEs only had access to the FDR data, so their results are more subjective as it was not possible to exactly tell when a stall horn went on, nor was there angle of attack indicator data.

As expected, discrepancies in the event annotations occurred between the SMEs. SME 1 marked 62 events while SME 2 marked 52 events. The SMEs had agreement on 49 of these events, and SME 1 found 13 events that SME 2 did not find, and SME 2 found 3 events that SME 1 did not find. This illustrates the challenge in determining the stall and LOC-I events solely from FDR data. However, for the events they both found, the start and end times were quite consistent, with an absolute difference average of 1.2 seconds for the start time, and 0.9 for the end time; with standard deviations of 2.4 and 1.7, respectively.

For each event that was recognized by the NGAFID, the start and end times of the event were compared to the SMEs start and end times by taking an absolute difference. Table 2 conveys the statistics for these deviations. To investigate how accurate the versions were with regards to detecting events and false negatives, Table 3 has multiple subcategories pertaining to the positive findings. A *true positive* is one in which the deviations from both SMEs start and end times is within the acceptable range of ±5s. An *acceptable positive* is

Table 1: Actual vs. Calculated Stall Events from the Experimental Test Flight

	Actual Stalls		V1 Stalls		V2 Stalls		V3 Stalls	
Event #	Start	End	Start	End	Start	End	Start	End
1	2316 – 2552	2567	**2519**	**2570 (+3)**	**2517**	**2565 (-2)**	**2520**	**2564 (-3)**
2	2565 – 2582	2589	2583 (+1)	**2590 (+1)**	**2574**	**2588 (-1)**	**2574**	**2587 (+2)**
3	2772 – 2775	2775	**2772**	2776 **(+1)**	**2773**	**2774 (-1)**	**2774**	**2774 (+1)**
4	2866 – 2901	2902	2909 (+8)	2909 (+7)	**2901**	**2903 (+1)**	**2901**	**2902**
5	3016 – 3024	3030	3028 (+4)	**3031 (+1)**	**3024**	**3028 (-2)**	**3024**	**3027 (-3)**
6	3133 – 3135	3140	N/A	N/A	**3134**	**3137 (-3)**	**3135**	**3136 (-4)**
7	3257 – 3266	3271	3272 (+6)	3276 (+5)	**3266**	**3271**	**3263**	**3271**
8	3282 – 3288	3293	N/A	N/A	3291 (+3)	**3292 (-1)**	**3288**	**3291 (-2)**
9	3308 – 3314	3324	3320 (+6)	**3324**	3316 (+2)	**3321 (-3)**	**3312**	**3320 (-4)**
10	3379 – 3383	3404	3374 (-5)	**3409 (+5)**	**3382**	**3403 (-1)**	**3382**	**3403 (-1)**
11	3419 – 3429	3448	**3429**	**3453 (+5)**	**3426**	**3448**	**3422**	**3447 (-1)**
12	3459 – 3472	3492	3476 (+4)	**3490 (-2)**	3473 (+1)	**3494 (+2)**	**3469**	**3493 (+1)**
13	3610 – 3617	3625	**3616**	**3625**	**3612**	**3622 (-3)**	**3617**	**3622 (-3)**
14	3628 – 3638	3644	**3637**	**3648 (+4)**	**3633**	**3644**	**3633**	**3643 (-1)**
15	3658 – 3663	3665	3666 (+3)	3675 (+7)	3664	**3670 (+4)**	3664 (+1)	**3670 (+5)**
16	3787 – 3792	3798	3802 (+10)	**3803 (+5)**	3794 (+2)	**3798**	3795 (+3)	**3798**
17	3818 – 3820	3824	**3819**	**3820 (-4)**	**3818**	**3824**	**3818**	3818 (-6)
18	3895 – 3902	3902	3905 (+3)	**3907 (+5)**	**3901**	**3904 (+2)**	**3898**	**3904 (+2)**
19	3923 – 3929	3934	3937 (+8)	3940 (+6)	3932 (+3)	**3933 (-1)**	**3928**	**3933 (-1)**
20	3950 – 3955	3956	3961 (+6)	3965 (+9)	**3954**	**3957 (+1)**	**3954**	**3957 (+1)**
21	3970 – 3987	3987	3993 (+6)	3993 (+6)	3988 (+1)	**3991 (+4)**	**3984**	**3991 (+4)**
22	4041 – 4047	4053	**4043**	**4054 (+1)**	**4045**	**4050 (-3)**	**4046**	**4050 (-3)**
23	4057 – 4059	4064	4063 (+4)	**4067 (+3)**	**4059**	**4063 (-1)**	**4059**	**4060 (-4)**
24	4070 – 4073	4082	4077 (+4)	**4086 (+4)**	**4072**	**4080 (-2)**	**4072**	**4079 (-3)**
25	4096 – 4104	4118	**4104**	4112 (-6)	**4102**	4107 (-11)	**4102**	4107 (-11)
26	4142 – 4149	4165	**4147**	**4166 (+1)**	**4146**	**4164 (-1)**	**4147**	**4163 (-2)**
27	4175 – 4181	4189	**4180**	**4188 (-1)**	**4180**	**4187 (-2)**	**4180**	**4187 (-2)**
28	4224 – 4226	4231	4230 (+4)	**4233 (+2)**	**4226**	**4231**	**4225**	**4231**
29	4252 – 4266	4271	**4259**	**4273 (+2)**	**4258**	**4270 (-1)**	**4263**	**4268 (-3)**
30	N/A – N/A	N/A	N/A	N/A	5367	5368	5367	5368
% Valid	–	–	37.9	72.4	79.3	96.6	93.1	93.1
Abs Avg	–	–	2.8	3.3	0.4	1.8	0.1	2.5
Std Dev	–	–	3.1	2.5	0.9	2.1	0.6	2.3

Table 2: Actual vs. Calculated Stall Event Time Deviation from NGAFID data

Ver.	SME 1				SME 2			
	Abs. Dev. From Start		Abs. Dev. From End Time		Abs. Dev. From Start		Abs. Dev. From End	
	Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
1	5.9	5.9	6.1	5.7	5.2	5.5	6.2	6.0
2	2.4	2.5	1.8	1.8	2.5	2.8	1.5	1.6
3	2.5	2.6	1.5	1.4	2.6	2.7	1.7	1.8

Table 3: Results for Acceptable Events from Random Data

Ver.	Positives				False Negatives	% Valid
	True	Acceptable	Delayed	False		
1	12	3	26	9	24	23.1
2	32	10	9	23	12	64.6
3	34	20	4	25	7	83.1

Table 4: Flight import speed and final database size.

Version Version	Min. Time	Max. Time	Avg. Time	DB Size
compression	1747s	1923s	1851s	4.2 GB
compression + LOC-I	1855s	2285s	2118s	4.9 GB
no compression	2048	2386s	2208s	22 GB
no compression + LOC-I	1868s	2560s	2152s	23 GB

differs from a true positive in the sense that an acceptable positive is where the aircraft was identified by only one SMEs to enter a complete aerodynamic stall within the valid deviation range. A likely scenario that would cause such disagreement is a premature detection of such event on the part of the calculation. Hence, these types of positives are considered acceptable since the calculations were correct in indicating that the danger of a stall or LOC-I event is eminent, and the FDM evaluating the flight should investigate further. Finally, a *delayed positive* is one in which the calculated start and end times exceed the acceptable deviation from both of the SME's calculated start and end times. A false negative is when a stall event was detected that neither SME had marked. These tables show that moving from V1 to V3 not only provides strong improvements on timing accuracy, reducing the average start differences by over 2x and end deviations by over 3x the SMEs, while also reducing the standard deviations by over 3x. Similarly, the accuracy (percentage of true or acceptable positives) increased dramatically from 23.1% to 83.1%.

6.2.1 LOC-I Index Results. Utilizing the same methodology to gather ratings as discussed in 6.1.1, SME 1 found 62/65 of identified LOC-I values to be accurate while SME 2 found 54/65 of identified LOC-I values to be accurate.

6.3 Data Processing Performance Improvements

Table 4 depicts the impact of the LOC-I index calculations on flight processing time and database size, with and without compression. For this work, mySQL version 5.6.50 was used as the database server. For each processing strategy, one months worth of flight data (8153 flights) was imported into a test database, with results reported for five repetitions. While using compression does incur a computational overhead, the performance improvements from reduced I/O provided a stronger performance payoff. The disk space saved was also quite significant, dropping by a factor of about 5. Overall, it only took $0.23s$ on average to import a flight (which average 4000 seconds or 1.1 hours in length). The performance of event calculation was also measured on this sample, which took $0.075s$ per flight on average to calculate all of the 47 events in the NGAFID (minus proximity). In total, it took on average $0.3s$ on average to import a flight and calculate events, minus proximity, which is a significant improvement over the prior NGAFID version which would take over 3 minutes to import and calculate events per fight by the time the design was retired. Proximity calculation performance was measured separately, using the full database of 660,000 flights (to accurately measure the database query costs), and required $3.2s$ on average, with each flight matching another 32.3 flights by time, which was reduced to 16.1 flights by location on average across a sample of 1,000 flights, which is a dramatic reduction over a naive implementation which would need to compare against every other flight in the database.

7 DISCUSSION

This paper presents a redesign of the National General Aviation Flight Information Database (NGAFID) to improve extensibility and performance, as well as reduce memory and disk space requirements as its growth continues. Using a compressed data representation within the database and a pipelined data ingestion strategy, data import and flight safety event calculation time was reduced from over 3 minutes in the previous implementation to $3.5s$ on average, while at the same time reducing memory and storage needs by a factor of 5. In addition, three new advanced safety events calculation methods were implemented to detect deviations from self-defined glide paths, potential stalls, loss of control in-flight (LOC-I) and proximity to other aircraft. The stall and LOC-I calculations were adapted from existing work [2] using input from subject matter experts (SME), and validated using both an experimental test flight with a wide variety of stalls, as well as historical data from the NGAFID independently reviewed by two SMEs. The enhancements to the stall and LOC-I calculations were shown to improve accuracy of event detection from 23.1% to 83.1%, reduce the number of false negatives by a factor of 3, and improve the accuracy of the timing of these events.

As demonstrated in the results, substantial improvements were found by utilizing an altitude-derived vertical speed over the FDR's vertical speed (VSI), although it was specific only to the Cessna 172 aircraft (the most common aircraft in the NGAFID). Future expansions will create an indicated airspeed to calibrated airspeed lookup table per airframe, or develop a methodology that uses change in groundspeed (corrected for wind speed and direction) in order to approximate true airspeed. Additionally, a number of false positives occurred when the aircraft entered a slip. This is likely due to the instrumentation error between indicated and calibrated airspeed increasing during uncoordinated flight. In all iterations explored in this work, we used a coordination index that looked for deviations (positive or negative) from coordinated flight. However, loss of control events are more probable during an aerodynamic skid (when yaw and roll are coupled) than a slip (when yaw and roll are decoupled) [9]. Future work will explore ways to discriminate between skids and slips, and apply airspeed corrections when the aircraft is in a slip to reduce the number of false positives due to slipping flight. We will also implement further performance improvements, in particular how to further speed up proximity detections which still take $\sim 3.2s$ per flight. As the bottleneck for this is the database I/O, further database optimizations or potentially using a distributed database system will be investigated.

REFERENCES

[1] Federal Aviation Administration and United States. Federal Aviation Administration. 2009. *Pilot's Handbook of Aeronautical Knowledge.* Skyhorse Publishing Inc.

[2] Shelby Balogh. 2017. *Use Of FOQA Data To Estimate The Probability Of Vehicle Upset Or Loss Of Control In-Flight.* Master's thesis. University of North Dakota, Grand Forks, ND, USA.

[3] Cessna Aircraft Company. [n. d.]. Cessna Skyhawk Pilots Operating Handbook. https://www.cpaviation.com/images/downloads/Cessn\a%20172M.pdf

[4] Federal Aviation Administration. [n. d.]. Fact Sheet – General Aviation Safety. https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=21274

[5] Antonio Fernández, D Martinez, Pablo Hernández, Samuel Cristóbal, Florian Schwaiger, and José Marıa Nunez. 2019. Flight Data Monitoring (FDM) Unknown Hazards detection during Approach Phase using Clustering Techniques and AutoEncoders. *Proceedings of the SESAR Innovation Days* (2019).

[6] J. G. Fuller and L. R. Hook. 2020. Understanding General Aviation Accidents in Terms of Safety Systems. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC).* 1–9. https://doi.org/10.1109/DASC50938.2020.9256778

[7] Kelton Karboviak, Sophine Clachar, Travis Desell, Mark Dusenbury, Wyatt Hedrick, James Higgins, John Walberg, and Brandon Wild. 2018. Classifying Aircraft Approach Type in the National General Aviation Flight Information Database. In *Computational Science – ICCS 2018*, Yong Shi, Haohuan Fu, Yingjie Tian, Valeria V. Krzhizhanovskaya, Michael Harold Lees, Jack Dongarra, and Peter M. A. Sloot (Eds.). Springer International Publishing, Cham, 456–469.

[8] D. Liu, N. Xiao, Y. Zhang, and X. Peng. 2020. Unsupervised Flight Phase Recognition with Flight Data Clustering based on GMM. In *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC).* 1–6. https://doi.org/10.1109/I2MTC43012.2020.9128596

[9] Rich Stowell. 2007. *The light airplane pilot's guide to stall/spin awareness: featuring the PARE spin recovery checklist.* Rich Stowell, Master CFI-A.

[10] Commercial Aviation Safety Team and ICAO Common Taxonomy Team CICTT. 2013. Aviation Occurence Categories-Definitions and Usage Notes.

[11] Andrew Walton, Carl Baumann, and Robert C. Geske. 2019. Fatal Flight Training Accident Report 2000–2015. *Aircraft Owners and Pilots Association* (2019).