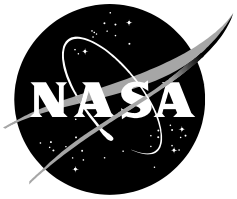
NASA/TM—2018–219964



RNAV Adherence Data Integration System Using Aviation and Environmental Sources

Ilya Avrekh
SGT Inc.
NASA Ames Research Center

Bryan L. Matthews
SGT Inc.
NASA Ames Research Center

Michael Stewart
San José State University
NASA Ames Research Center

June 2018

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

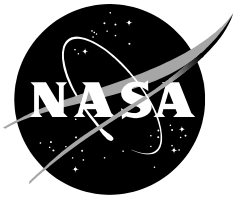
- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM—2018–219964



RNAV Adherence Data Integration System Using Aviation and Environmental Sources

Ilya Avrekh
SGT Inc.
NASA Ames Research Center

Bryan L. Matthews
SGT Inc.
NASA Ames Research Center

Michael Stewart
San José State University
NASA Ames Research Center

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA 94035-1000

June 2018

This report is available in electronic form at
http://

Table of Contents

1. List of Acronyms **7**

2. Basic Concepts **8**

3. Software and Hardware Used **9**

4. STAR Description Extraction **9**

5. Flight Data Extraction **11**

6. Wind Aloft Extraction **12**

7. Convective Weather Data Extraction **13**

8. Lateral Adherence Evaluation **15**

Technical Approach 15

Building and Labeling Planar Arrangements 16

Performing Point Location Queries 17

Evaluating Query Results 19

Arguments 22

9. Vertical Adherence Evaluation **22**

Arguments 22

Loading Snapshot Data 23

Loading the Lateral Adherence File

Loading the Wind Data

Loading the Convective Weather Impact Data

10. Flight-by-Flight Compartment Adherence **24**

Handling Data Quality Issues 24

Computing Cumulative Distance Traveled 24

Determining Late Entries and Early Exits 24

Stepping Through Compartments Flown 25

Features Based on Daily Aggregated Adherence 25

Features Based on Chart Cycle Aggregated Adherence 26

11. Synopsis **27**

12. Appendix

Appendix A: Data Fields in Extracted Files with STAR Descriptions 28

Appendix B: C Structures for STAR Arrangement Construction 30

Appendix C: Snapshot Values Logged at Each Compartment 33

13. References **38**

List of Figures

Figure 1: Processing pipeline for assessing lateral and vertical adherence **8**
Figure 2: STAR components **10**
Figure 3: Records in extracted data files with STAR descriptions **11**
Figure 4: Records in extracted flight data file **12**
Figure 5: Groups and datasets in HDF5 file with extracted weather data **14**
Figure 6: Fragment of the arrangement representing STAR tolerance zone **16**
Figure 7: In-memory structures supporting lateral adherence evaluation **17**
Figure 8: Tolerance zones around STAR common parts and transitions to runway 27 at KIAH **18**
Figure 9: Tolerance zones for TEJAS4 STAR common part, en route transitions, and transition to runway 27 at KIAH **19**
Figure 10: Records in the file containing Lateral Adherence Evaluation results **21**
Figure 11: Diagram of merging flights and miles in trail **26**

1. List of Acronyms

~~ATAC~~	~~Airborne Tactical Advantage Company~~
~~RD~~	~~R Documentation~~
AOI	Area of Interest
ARTCC	Air Route Traffic Control Centers
ASDE-X	Airport Surface Detection Equipment, Model X
ATM	Air Traffic Management
CDI	Course Deviation Indicator
CGAL	Computational Geometry Algorithms Library
CIFP	Coded Instrument Flight Procedures
CIWS	Corridor Integrated Weather System
CONUS	Contiguous United States
CSV	Comma Separated Value
CWAM	Convective Weather Avoidance Model
DCEL	Doubly Connected Edge List
FAA	Federal Aviation Administration
FMS	Flight Management System
GPS	Global Positioning System
GRIB	Gridded Binary
HDF5	Hierarchical Data File format 5
IFF	Integrated Flight Format
IFR	Instrumental Flight Rule
JSON	JavaScript Object Notation
LAEA	Lambert Azimuthal Equal-Area
LL	Lincoln Laboratory
MIT	Massachusetts Institute of Technology
NAVAID	Navigational Aid
NM	Nautical Mile
NOAA	National Oceanic and Atmospheric Administration
PBS	Portable Batch System
RADI	RNAV Adherence Data Integration
RNAV	Area Navigation
RNP	Required Navigation Performance
STAR	Standard Terminal Arrival Route
TRACON	Terminal Radar Approach Control
UTC	Coordinated Universal Time
VIL	Vertical Integrated Liquid
VOR	VHF (Very High Frequency) Omni-Directional Range
WAF	Weather Avoidance Field
WPID	Waypoint ID
XML	Extensible Markup Language

2. Basic Concepts

Standard Terminal Arrival Route (STAR) is an ATC-coded Instrumental Flight Rule (IFR) route established for aircraft arriving to certain airports [1]. In this report we study RNAV STARs, which can only be used by aircraft equipped with FMS or GPS.

A STAR provides a transition from the en route structure to an arrival waypoint-based 3D path through the terminal area. The STAR arrival route, also called the basic STAR procedure or the common route (we also use the term "trunk"), begins at the common arrival waypoint where all the various en route transitions merge together. The STAR common route has the name of this common waypoint with an addition of the STAR revision number (e.g. DOOBI2). The STAR en route transition name is formed as the name of the initial waypoint (NAVAID) of this transition prepended to the trunk name (e.g. AEX.DOOBI2).

Each STAR can have runway transitions to multiple runways from the common route ending at a fix, designated by ATC, which allows for radar vectors or connection to an instrument approach procedure. Each STAR runway transition is named by the runway it serves added to the trunk name (e.g. RW26R.DRLLR5).

STAR defines multiple "fly by" waypoints connected by straight line segments, which arriving flights need to follow, with altitude and speed restrictions for some route waypoints establishing overall descent and deceleration profiles.

STAR adherence has two components: lateral adherence (measure of deviating from STAR laterally); and vertical adherence (following descent and deceleration profiles).

This study considers RNAV1 STARs, which require the use of a course deviation indicator (CDI)/flight director, and/or autopilot. RNAV1 STARs have tighter required navigation performance (RNP) tolerances, which are defined as 1 NM lateral tolerance and ±300 feet vertical tolerance.

This technical manuscript describes a system called RNAV Adherence Data Integration (RADI) that fuses multiple heterogeneous data sources that take into account multiple factors to achieve this adherence assessment. Figure 1 illustrates the fusion of these data sources.

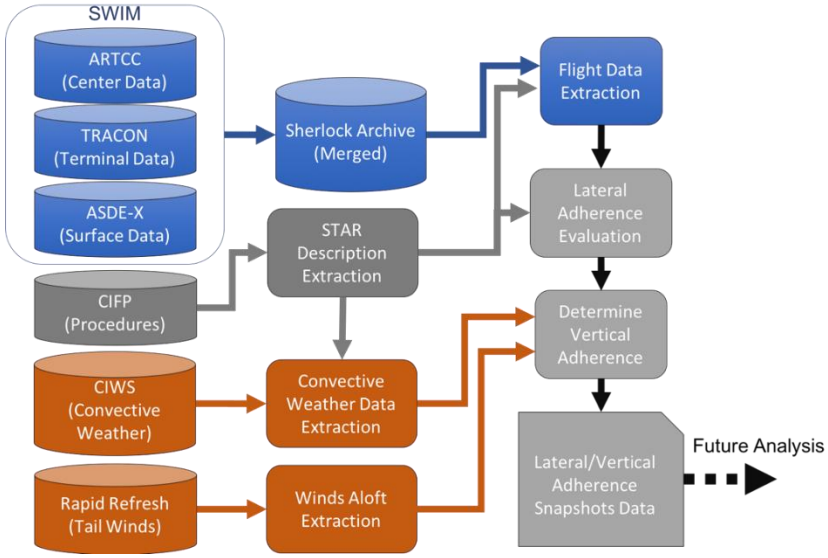


Figure 1 Illustrates the processing pipeline from various data archives and how they are fused to assess lateral and vertical adherence

3. Software and Hardware Used

The RADI system is designed to scale using a Linux high-end computing platform with multiple processes mapped to extract and fuse data for each day. Software was developed in both Python and C++. RADI utilizes the following software packages and hardware:

1. Tested on 64-Bit Linux Red Hat 6 architecture
2. Python 2.7 (Anaconda)
    a. Pyproj 1.9.5.1 [2]
    b. H5py 2.5.0
3. G++ (tested with 4.4.7)
4. CGAL 3.6.1 library (on Red Hat 6 distribution)
5. Cmake (tested with 2.8.12.2)
6. Linux cluster:
    a. PBS scheduler (tested with Torque 6.1.2)
    b. Linux cluster nodes with at least a total of 56 cores with 1.5Gb per core
    c. 1 additional Linux cluster node for master spark executor
    d. Apache Spark 1.6.0 Pyspark

4. STAR Description Extraction

STAR descriptions in XML form are produced by ATAC Corporation for each chart cycle as a single file for all US airports. The underlying data source that ATAC uses to prepare the XML files is the Coded Instrument Flight Procedures (CIFP). This data source was chosen because ATAC has already archived the CIFP data, whereas the FAA site only houses the current chart cycle. The STAR description extraction step extracts STAR description data for specified airports from the XML file, augments them with additional data, and saves them in the form used by subsequent processing steps.

For each specified airport the following comma-separated text files are created: the *waypoint* file (list of all STAR waypoints with their geographic coordinates and coordinates in other map projections used in subsequent steps); the *std_routes* file containing complete description of RNAV STARs for an airport, including vertical navigation constraints; and the *segments* file, which contains information about STAR segments with indicator points along these segments used for convective weather evaluation.

Waypoint information in the *waypoint* file includes the 5-character waypoint ID; latitude and longitude of the waypoint; x- and y-coordinates of the waypoint in the oblique stereographic projection with the tangent point at this airport; and waypoint 1 km grid coordinates in the Lambert Azimuthal Equal-Area map projection used in CIWS files. The first record of the *waypoint* file contains information about the airport such as latitude and longitude; airport elevation; airport magnetic variance; and maximum distance from the airport to the initial waypoint of en route transition, which defines our Area of Interest (AOI) around the airport.

Each STAR description contains three components as shown in Figure 2: the “trunk” (common) route description; en route transition descriptions; and runway transition routes descriptions. Each route component is a non-self-intersecting polyline of segments defined by the start and end waypoints. Some segments may belong to more than one route. The STAR description also includes vertical navigation constraints for each waypoint.

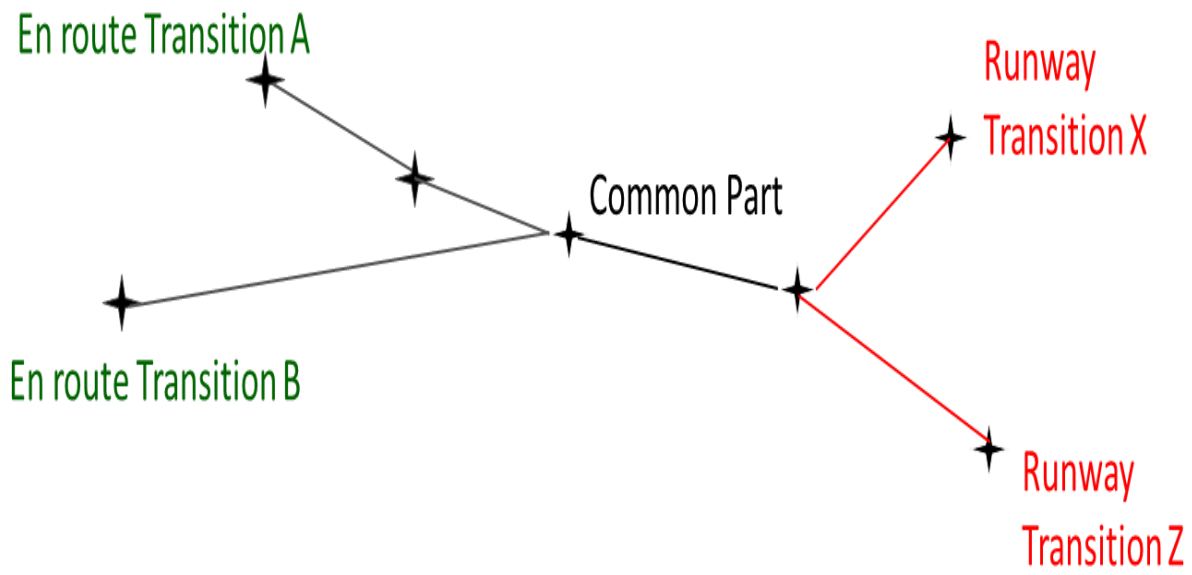


Figure 2 STAR components

The *segments* file contains records with a variable number of fields which augment each segment with a set of equidistant (2 NM) indicator points, defined by their 1 km grid coordinates in the Lambert Equal Area projection.

In some cases, the STAR is designed to serve certain airport runways but there are no respective runway transition routes specified. In such cases, STAR information in XML files is mapped to these runways via the *STAR_Rwy_Augm* JSON configuration file which is being maintained manually. Information in this file is used to create empty runway transitions in the *std_routes* file. Runway transition information has an important role in determining a STAR for a particular flight.

This processing step is implemented as the Python *std_rt_data_extract* module, which uses the Python module *xml.etree.ElementTree* for parsing and querying XML files, JSON for parsing JSON files, and PyProj to perform map projections.

This module is run from the command line and contains arguments defining which XML file is to be processed, the file containing the list of airports for which data extraction should take place, and the root of the directory structure that contains the extracted files (root/chart_cycle_id/airport_id). It also sends detailed processing and error information to the standard output which can be redirected to the log file.

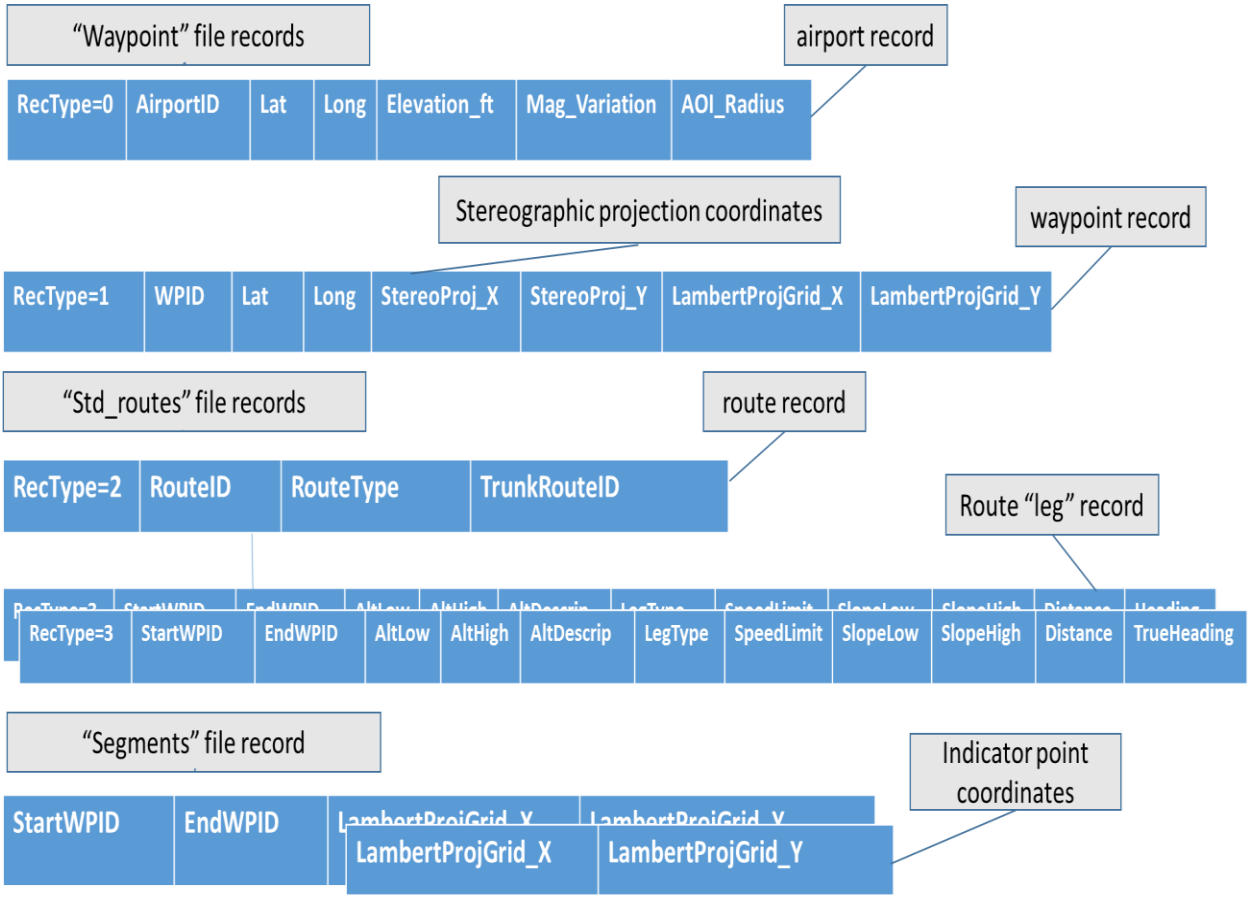


Figure 3 Records in extracted data files with STAR descriptions

5. Flight Data Extraction

The data for all US flights within a 24-hour period is contained in an Integrated Flight Format (IFF) file and corresponding meta data found in the RD file. These files are stored in the Sherlock ATM Data Warehouse. The flight data extraction step retrieves flight data for all arriving flights for specified airports from these files, forming one file for each airport for each 24-hour period.

The file with extracted flight data is a comma-separated text file that contains records of two types for each flight arriving to the given airport: a single flight header record, and multiple records for radar track points (flight path).

The flight header record contains the "flight key" (unique system identifier), flight number, aircraft type, aircraft performance category, aircraft weight class, destination airport code, landing runway, landing time, STAR filed in the flight plan, aircraft equipment list, and the number of track points for the flight.

Radar track point records contain the timestamp, aircraft latitude and longitude, aircraft x- and y-coordinates in the airport local stereographic map projection, aircraft altitude, climb rate, heading, and ground speed.

The meaning and format of the data fields are the same as corresponding fields in IFF and RD files, which are defined by the NASA Ames Sherlock project [3].

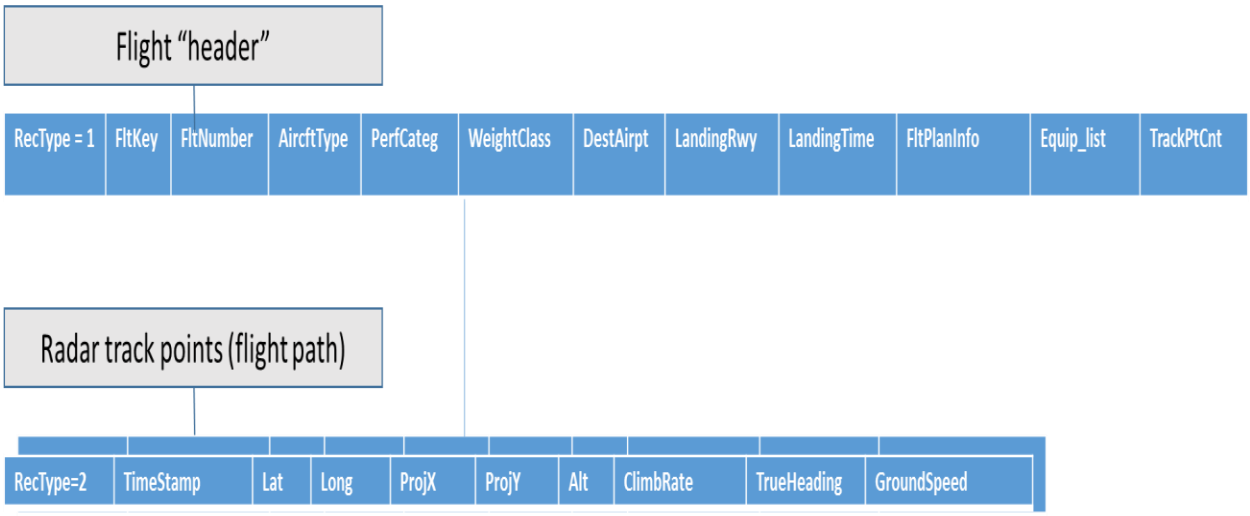


Figure 4 Records in extracted flight data file

The extracted flight path starts at the aircraft entry into an airport's AOI as defined during STAR data extraction and ends at aircraft touchdown. Extraction for all airports is performed in one pass through a pair of IFF and RD files.

In cases when elapsed time between two adjacent extracted track points is between 25 and 120 seconds, interpolated track points will be inserted between them to achieve a sampling rate of approximately 12 seconds. The interpolation algorithm is based on linear interpolation of the aircraft speed vector between two observed points with new points placed along a geodesic between the observed points. No interpolation is performed during the last 200 seconds before landing or in cases when the distance between two observed points is less than or equal to 2 NM.

This processing step is implemented as the Python *flight_data_iff_extract* module which uses the Python modules NumPy and PyProj, and JSON.

The module is run from the command line and specifies the IFF file to be processed and the file containing the list of airports for which data extraction should take place. The module uses a JSON configuration file to determine the root of the directory structure that contains the extracted files as well as the root of the directory structure that contains the extracted STAR descriptions. It also sends detailed processing and error information to the standard output which can be redirected to the log file.

6. Wind Aloft Extraction

Wind data is extracted from NOAA rapid refresh files stored in the ATM data warehouse. These files are in the GRIB1 file format and have 13 km grid resolution.

The file extraction code is written in Python and uses the Pygrib package to read the GRIB files. A list of the latitude and longitude for the airports of interest are defined, as well as the

altitudes of interest, in this case 4,000 ft to 40,000 ft in 1,000-foot increments. The code takes a list of GRIB files which have been organized into separate files for each hour UTC, as well as a list of position and altitude points to be queried. The airport latitude and longitude are looked up on the position grid in each of the files and the nearest grid point is identified. The default pressure range in hectopascals (hPa) in each GRIB file ranges from 100 hPa to 1,000 hPa in increments of 25 hPa. The pressures are then converted into pressure altitudes (ft) using the following equation:

Pressure Altitude = $(1 - (\frac{press}{1013.25})^{0.190284}) * 145366.45^2$

This figure serves as a lookup for the nearest altitude being queried. Once the altitude and position have been identified in the file for the given time sample, the temperature, humidity, and winds are pulled from the data structure. Winds are stored as U/V vectors where U is -East/West+ and V is -North/South+. The U/V vectors are converted to magnitude and standard wind direction (from true north) using the following methods:

Magnitude = $(u^2 + v^2)^{\frac{1}{2}} * 1.94384$

Direction = $\mathrm{mod}\ (270 - \frac{180}{\pi} * arctan^2(v, u), 360)$

Each line has a unique airport, date, time (hr), and altitude (ft) key that will be used to link the wind data with flights in the vertical adherence process. The files are stored in CSV format with the following columns:

- Airport
- Altitude
- YYYYMMDD
- HHMM
- Humidity (%)
- Temperature (K)
- Wind Speed (kns)
- Wind Direction (degs)
- Vertical Accuracy (ft)
- Horizontal Accuracy (NM)

7. Convective Weather Data Extraction

The goal of the Convective Weather Data Extraction step is to determine instances of severe convective weather along STARs for specified airports on a given day. This data is being used to determine weather impact on STAR adherence.

The data is being extracted from Corridor Integrated Weather System (CIWS) “gridded” data files. Gridded CIWS products are expressed as rectangular arrays with elements containing a data value obtained with uniformly spaced observations or computed results on a 2D surface. Gridded data arrays map to the earth’s surface for conterminous US states through a Lambert Azimuthal Equal-Area (LAEA) map projection with projection origin latitude and longitude at 38.0 and -98.0 degrees respectively. This data source was downloaded from the ATM data warehouse.

Two types of CIWS files are processed: 5-minute-resolution current contiguous United States (CONUS) precipitation (VIL), and current CONUS echo top datasets. Both types are encoded with NetCDF format [4], which, in turn, is based on Hierarchical Data File (HDF5) format [5]. The Vertical Integrated Liquid (VIL) measurement indicates the amount of atmospheric liquid as observed with radar and computed over the extent of a volume scan. The echo top value indicates the maximum altitude of observed radar returns computed over the extent of a volume scan. We use the Convective Weather Avoidance Model (CWAM) for terminal airspace developed at MIT [6] to map the pair of quantized VIL and echo top values to the probability of flight deviation to avoid convective weather area (Avoidance Probability).

To assess weather conditions over STARs, we extract VIL, echo top, and Avoidance Probability values for waypoints and indicator points along STAR segments created during STAR description extraction which have been stored with their LAEA grid coordinates in *waypoints* and *segments* files. These values will be considered only if Avoidance Probability exceeds the threshold of 40%.

Extracted values referencing STAR waypoints, segments, and compartments at 5-minute intervals for the given day will be stored in a separate HDF5 file for each airport. Waypoints and segments will only have records in the extracted file if the conditions described above exist within the 5-minute interval for that waypoint or segment. If there are no such waypoints or segments during the day in question, the file for the given airport will not be created.

This processing step is implemented as the Python *CIWS_data_extract* module, which uses the Python modules NumPy and h5py, which supports HDF5 file manipulation, and JSON. The module is run from the command line and specifies the following arguments: the day for which CIWS data is extracted; and the file containing the list of airports for which data extraction should take place. The module uses a JSON configuration file to determine the root of the directory structure that contains the extracted files as well as the root of the directory structure that contains the extracted STAR descriptions. It also sends detailed processing and error information to the standard output which can be redirected to the log file.

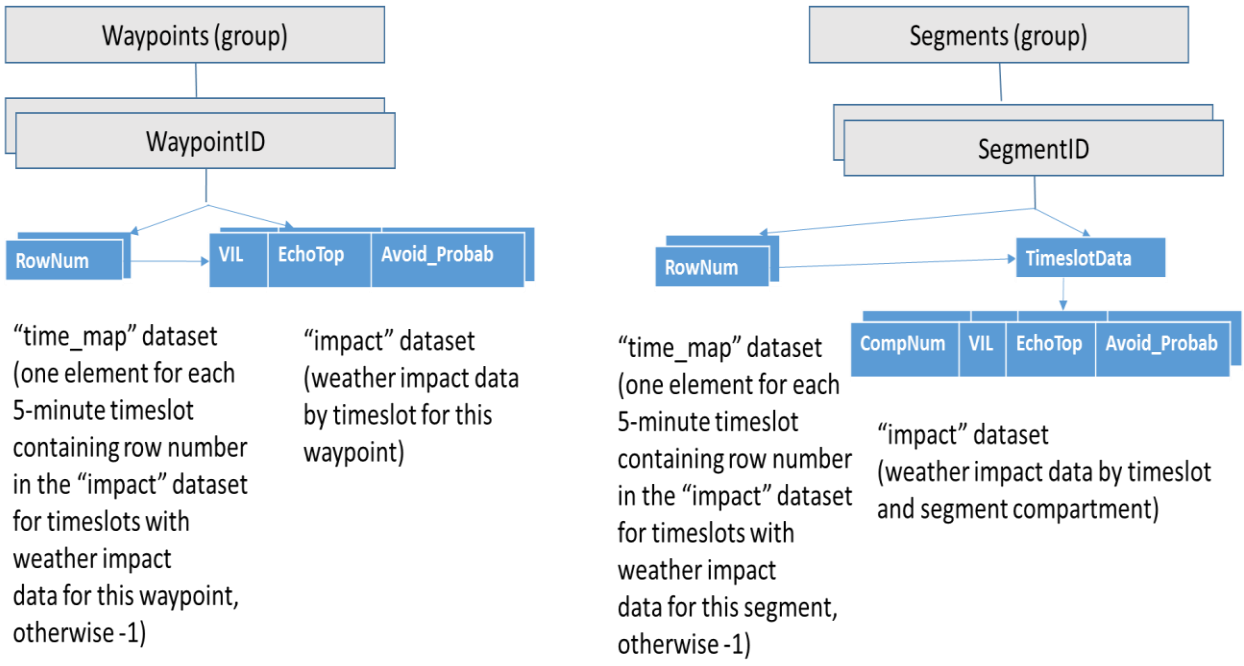


Figure 5 Groups and datasets in HDF5 file with extracted weather data

8. Lateral Adherence Evaluation

Technical Approach

The Lateral Adherence Evaluation step evaluates flight path adherence to a particular STAR instance which contains, in addition to a common part, specific en route transition and runway transition parts.

Because information about the STAR used by the flight is frequently unavailable and is often not the STAR filed with the flight plan, the actual STAR must first be determined. Once the actual STAR is known, the flight path's lateral adherence to the STAR can be evaluated. A straightforward approach would require checking all STARs for arrival airport by measuring distances from flight track points to STAR waypoints and segments, which could be very computationally expensive.

In this study, the actual STAR and lateral adherence will be determined by evaluating flight path containment in the navigation tolerance zone around STAR using "point location" queries. These zones are defined by Required Navigation Performance (RNP) tolerances for RNAV1 STARs as circular areas of 1 NM radius around waypoints, and rectangular strips of 1 NM width on each side of the STAR segments.

To facilitate this task, STAR navigation tolerance zones and flight track points are projected into 2D space using oblique stereographic map projection with the tangent point at a given airport, with a specific projection made for each airport [7]. This map projection is conformal with insignificant distance distortion (compared to tolerance zone size) within the airport terminal area. Projection for STAR waypoints and flight track points is performed in the STAR Description Extraction and Flight Data Extraction processing steps respectively.

A computational geometry approach implemented in a CGAL class library [8] is used to represent local stereographic projection of the STAR tolerance zones as a 2D arrangement. The arrangement A(C) is the subdivision of the plane into zero-dimensional, one-dimensional, and two-dimensional cells—called vertices, edges, and faces respectively—created by the curves (or their parts) in C. In this case, curves will be straight line segments forming borders of tolerance zones around STARs as described above. For simplification, circular areas around waypoints will be presented as octagons inscribed into these areas. To further qualify flight progression along the STAR, rectangular tolerance zones around STAR segments will be divided into "compartments" of 2 NM length along the segment. Note that the final compartment could be less than 2 NM in length if the total segment length does not evenly divide into 2 NM partitions. CGAL uses Doubly Connected Edge List structures (DCEL), commonly used to represent planar graphs, to describe arrangements.

Since there is a very low probability that track points will land exactly on vertices and edges, a CGAL Batch Point Location query algorithm is then used to determine the locations of the set of 2D projected flight track points (also referred to as query points) with respect to the STAR arrangement faces.

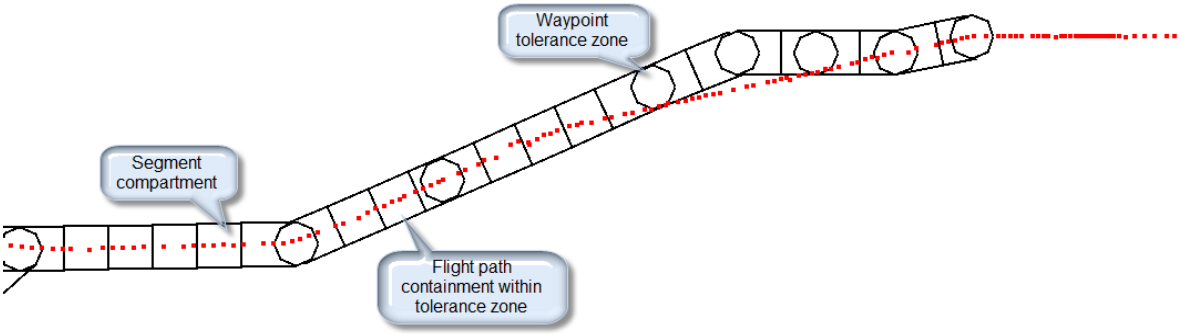


Figure 6 Fragment of the arrangement representing STAR tolerance zone

Building and Labeling Planar Arrangements

Lateral Adherence Evaluation is implemented in C++ using CGAL and utilizes CGAL-provided classes and methods to represent geometric primitives such as 2D points and linear segments as a linear geometric kernel, as well as planar arrangements of linear segments. A geometric kernel supporting floating point calculations is used since it satisfies the precision requirements for the task. The code creates objects that represent segments forming STAR navigation tolerance zones and then inserts them into an arrangement. It also provides custom implementation of the 2D point class to represent 2D points containing labels of type "long int".

STAR arrangement faces must be identified in a way that maps them to a respective waypoint or segment compartment. This allows the result of a point location query (i.e. point-face mapping) to provide detailed information of a flight track point's containment within the tolerance zone with resolution at the segment compartment and waypoint tolerance zone level. Face identification is achieved by using the "long int" number as the face ID and the *Arr_face_extended_dcel<Traits, long>* class template to associate the auxiliary data field type "long" with each face record in the DCEL. The face ID value is formed as follows:

1. Unique sequential number between 0 and 999 for waypoint
2. Segment compartment ID as concatenation of start and end waypoint serial numbers with compartment sequential number within this segment: (StartWP_ID + 1) * 100000 + EndWP_ID * 100 + CmptNum

By definition, the segment compartment ID will always be greater than 999, thus distinguishing it from the waypoint ID.

Associating the face ID with arrangement faces is done using a calibration point location query in which query points are placed at predetermined locations. These point locations are at waypoints for waypoint tolerance zones and one point in each segment compartment. The labels of these points are the IDs of their respective tolerance zones, either waypoints or segment compartments. When the results of a batch point query are returned, a point label is assigned to the face ID of the face on which the point is located.

See Appendix B for a list of several in-memory C structures which facilitate construction of STAR arrangements. The structures label faces and process query results by processing *waypoints* and *std_routes* files extracted from XML STAR descriptions.

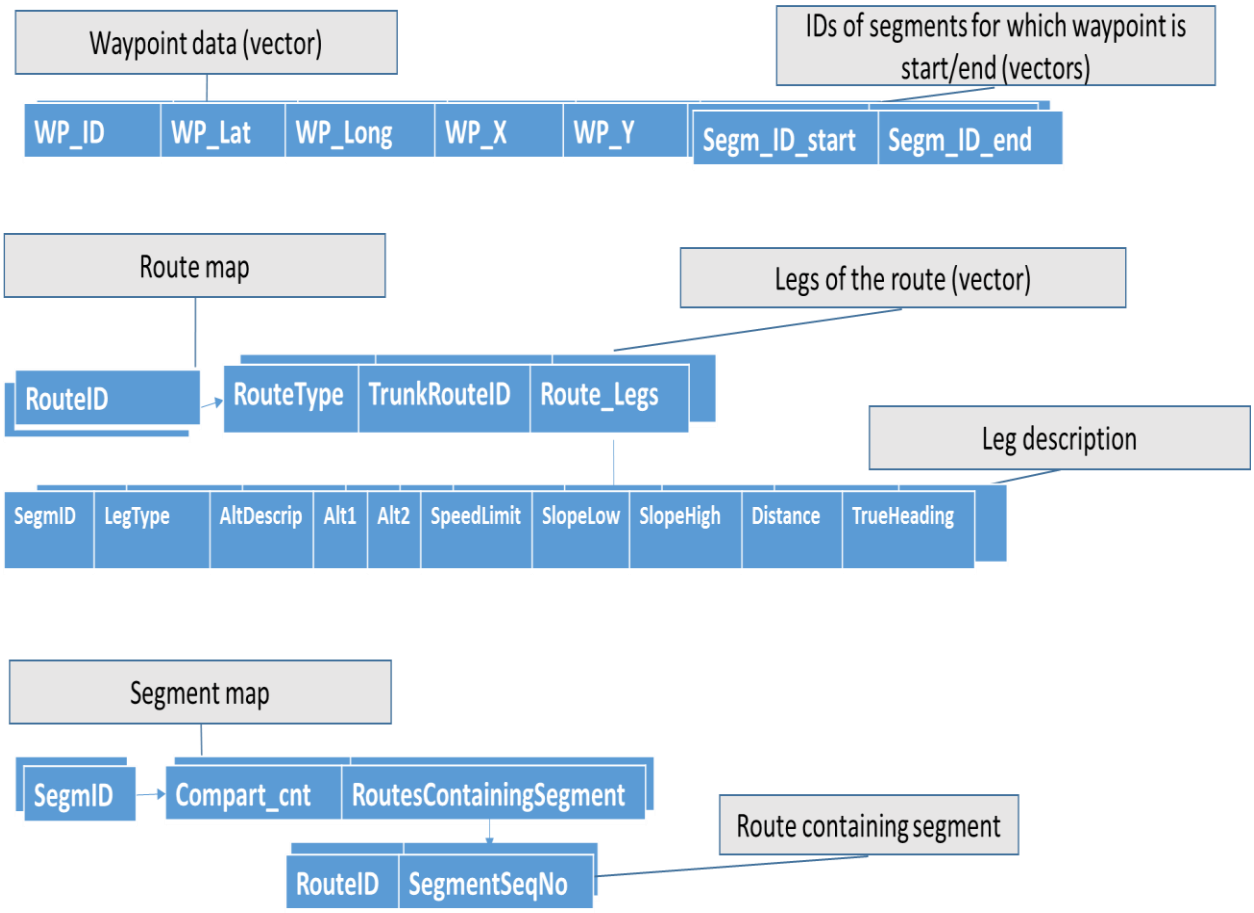


Figure 7 In-memory structures supporting lateral adherence evaluation

Performing Point Location Queries

The issue which arises when querying STAR arrangements is the overlap of the tolerance zones around different STARs or between different parts of the same STAR. Overlap can occur due to the existence of intersecting en route transitions for different STARs, the intersection of common parts of different STARs used in mutually exclusive airport flows, or the overlap of several en route transitions within the same STAR near the waypoint where they merge. Such overlaps create additional arrangement faces that belong to multiple segments or compartments of STARs, thus complicating their labeling and making track point location determination ambiguous.

This issue can be addressed by partitioning STARs for the given airport into subsets with no tolerance zone overlap and building separate arrangements for each of these subsets. Another solution is to perform STAR determination and evaluate lateral adherence for a given flight by

making several point location queries in which the result of one query is used to narrow the choice of arrangement for the next.

First a query is made against an arrangement based on STAR runway transitions to the flight's landing runway using the landing runway information collected during the Flight Data Extraction step. The arrangement also looks at the common parts of these STARs. This Step 1 query determines the STAR ID used by the flight. To support such queries, separate arrangements are built for common parts of STARs that have runway transitions to each particular runway, as well as a separate arrangement for the common parts of all STARs which don't have any runway transitions. These arrangements are built prior to performing queries and are stored in the map structure which allows for quick retrieval by runway ID.

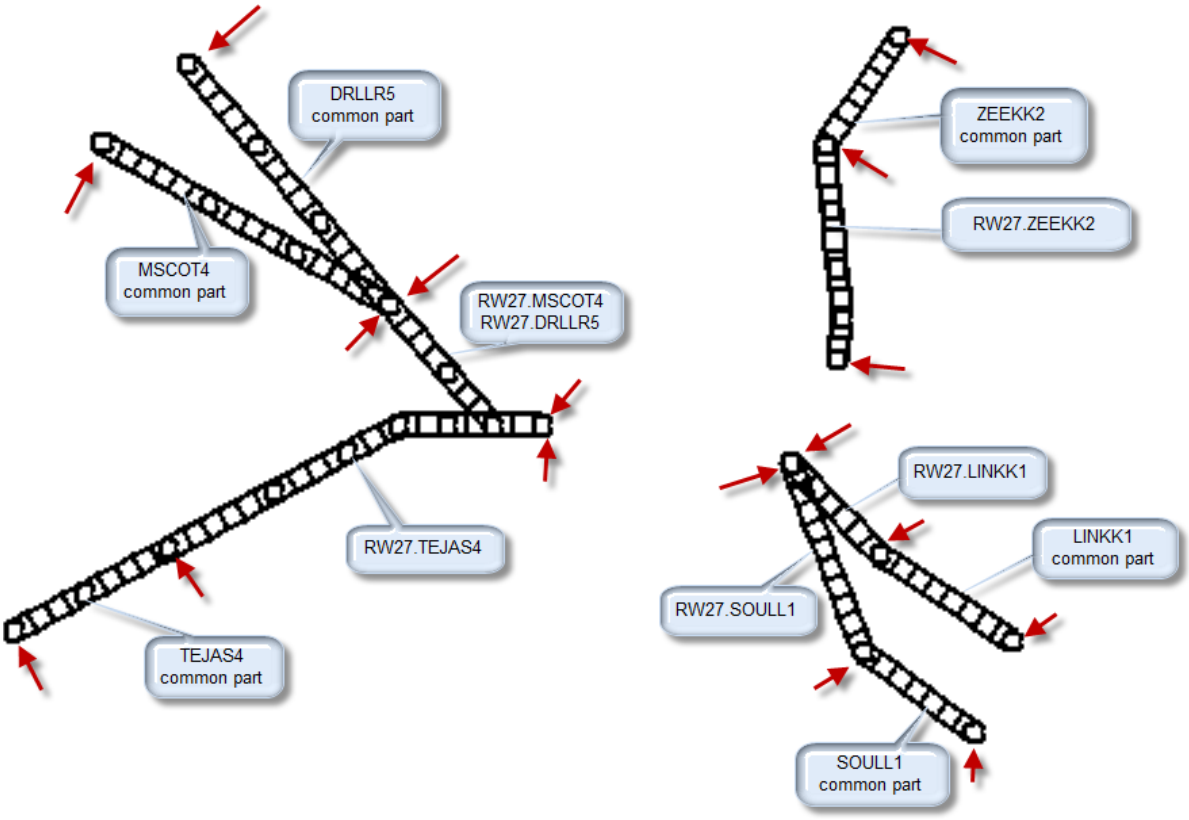


Figure 8 Arrangement, which represents tolerance zones around STAR common parts and transitions to runway 27 at KIAH

If the evaluation of the query results determines the STAR ID used by the flight, a Step 2 query is performed to determine the en route transition used. This query runs against an arrangement that includes the tolerance zone for the common part of the determined STAR, all its en route transitions, and the landing runway transition. Such arrangements for all STARs are also built in advance and stored in a separate map structure allowing for quick retrieval by runway ID and STAR ID.

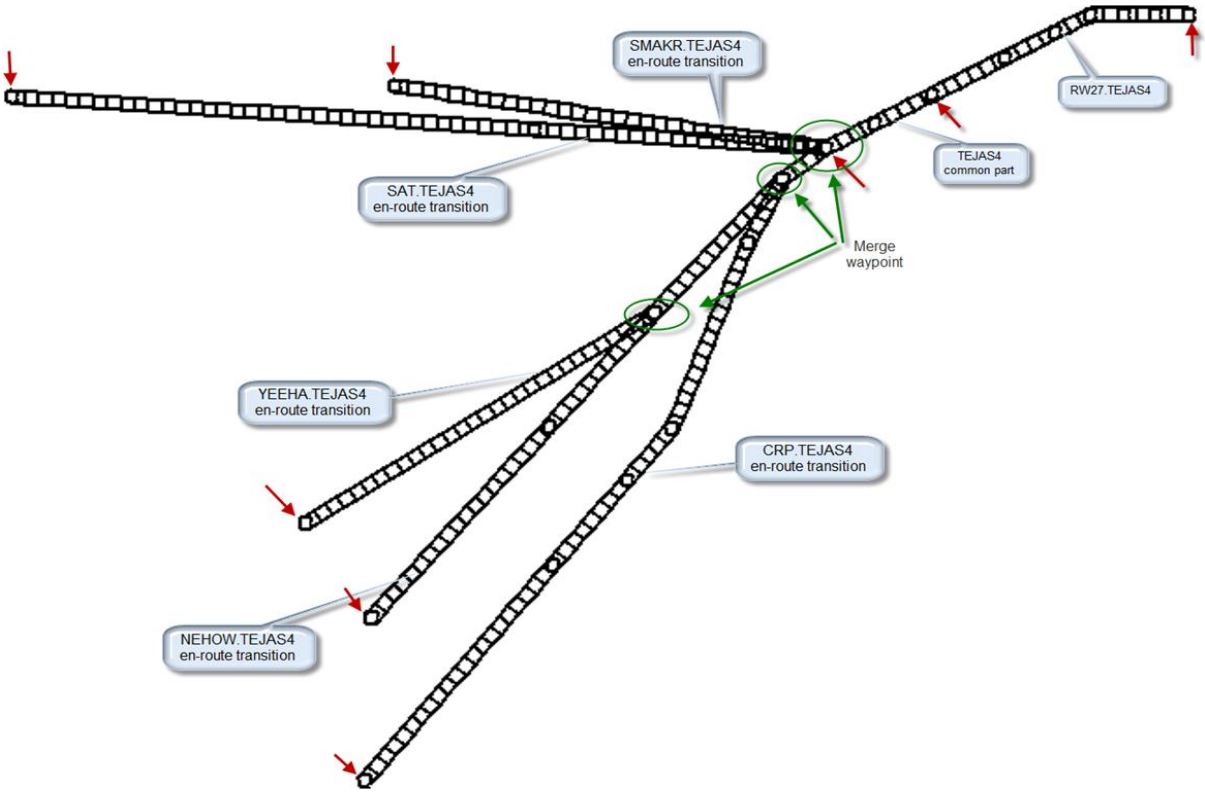


Figure 9 Arrangement representing tolerance zones for TEJAS4 STAR common part, en route transitions, and transition to runway 27 at KIAH

When en route transition has been determined by the Step 2 query, possible ambiguities are removed that are caused by the overlap of merging en route transitions that form acute angles near the merge waypoint. This removal is Step 3 and requires that a query is performed against an arrangement that contains tolerance zones for all parts of a determined STAR instance (i.e. determined en route transition, common part, and the landing runway transition). Each such arrangement is created when needed and stored in yet another map structure for quick retrieval.

Evaluating Query Results

Each query returns a list that specifies the locations of flight track points with respect to arrangement faces which, in turn, can be mapped to waypoint tolerance zones or segment compartments using a face ID. The query results are ordered lexicographically, i.e. in ascending order of the point's x-coordinate; for points with the same x-coordinate, results are further ordered by ascending y-coordinate. To order results chronologically, i.e. along the flight path progression, point labels are set as radar point timestamps. If the track point belongs to the unbounded face, its position is considered "undefined" or "in the void".

Further evaluation of the query results is performed by applying heuristic rules to determine which waypoints and segments can be considered to have been "flown". A waypoint or segment compartment is considered flown if there is a track point in its respective tolerance zone. The segment is considered flown if one or both of the following conditions are met: **both the start and end waypoints of the segment are flown, or the ratio of compartments flown to the total number of compartments in the segment exceeds the threshold value of 0.4. This**

heuristic was based on an inspection of a sample of flights with varying thresholds. It was determined that segment assignment could not be assessed visually with lower thresholds.

From the information about flown segments we determine the STAR common part in the Step 1 query or the en route transition in the Step 2 query using the segment map structure described above. This structure maps a flown segment to one or more STARs containing this segment. In cases where more than one STAR contains the segment, the list is pruned in two steps:

1. Removing STARs that cannot be part of the target arrangement of the query (e.g. STARs that don't have a runway transition which defines the arrangement in the Step 1 query).
2. Selecting the STAR with the maximum value of the ratio of the segments flown to the total number of segments in the route.

If no route information has been collected, the STAR for the flight is considered "undetermined" and no adherence evaluation can be performed. In some cases, the common part of a STAR can be determined, but the en route transition and/or runway transition used by the flight remain undetermined. This remains true even if the runway transition is implied by the landing runway.

The heuristics described above cannot theoretically guarantee STAR determination for all flight trajectories (for example, in contrived cases where a flight path skirts across multiple STARs), but provide reliable results for the vast majority of observed flights under regular conditions.

Results of the Lateral Adherence Evaluation are saved in a separate file for each airport with the following structure:

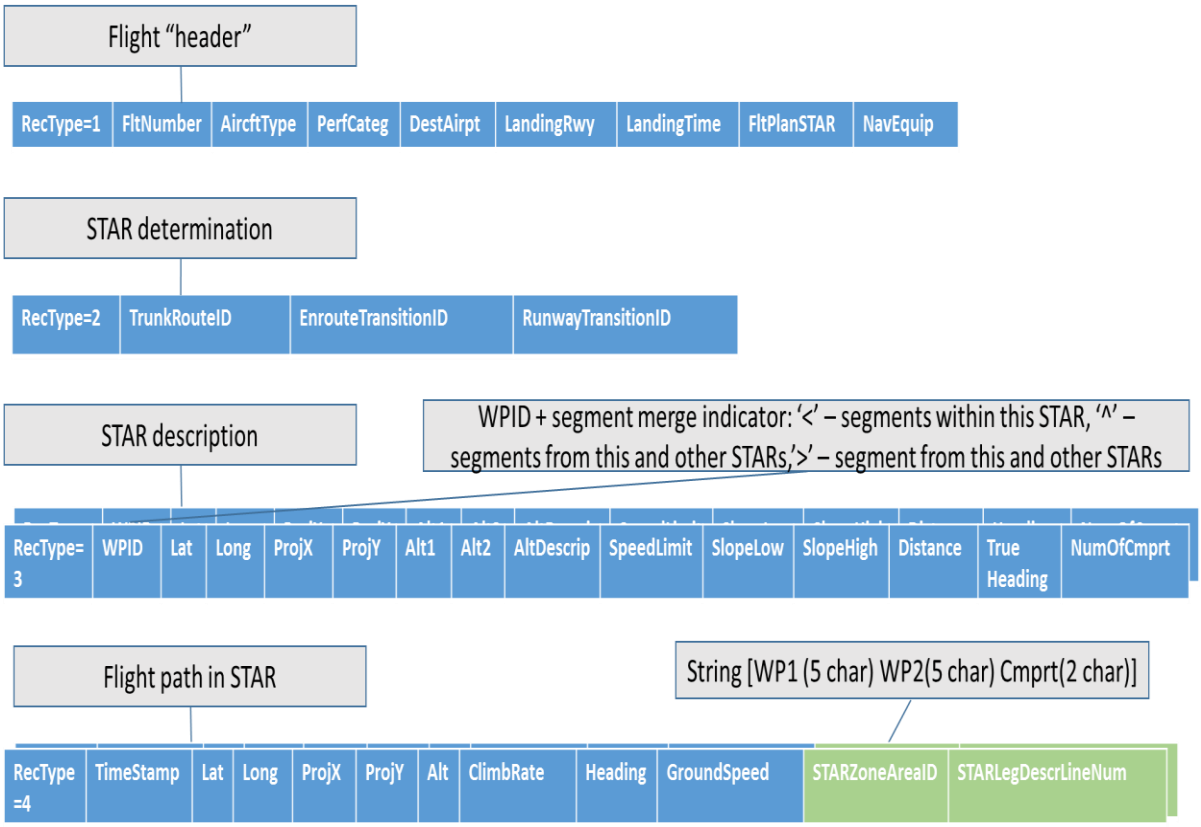


Figure 10 *Records in the file containing Lateral Adherence Evaluation results*

Data fields in *flight header* records (RecType 1) have the same meaning and format as the respective fields in the extracted flight data file (see figure 4).

Data fields in *STAR determination* records (RecType 2) represent STAR component IDs. In cases where a component has not been determined, the field will contain '*'.

Data fields in *STAR description* records (RecType 3) represent the description of the STAR determined during adherence evaluation and have the same meaning and format as the respective fields in extracted STAR description files (see table 1). Unlike the extracted STAR description files, the *STAR description* record also includes a field which contains the number of compartments for a particular segment (NumOfCmprt).

Data fields in *flight path in STAR* records (RecType 4) are the same as the respective fields in extracted flight data files (see figure 4) with the addition of the fields STARZoneAreaID and STARLegDescrLineItem. STARZoneAreaID identifies the waypoint or segment compartment containing the track point, and STARLegDescrLineItem contains a line number that maps it to the *STAR description* record for this flight that describes this specific STAR segment. If no STARZoneAreaID has been determined, the field will contain '*'; if no STARLegDescrLineItem has been determined, the field will contain '-1'.

If a STAR for the flight has not been determined, records with RecTypes 3 and 4 will not be created.

Arguments

The *StdRouteLatAdhEval* executable runs from the command line with the following arguments:

- <flight_data_timestamp> – character string, identifying flight data file day suffix (e.g. "20171201_050001_86398")
- <airport_list_file> – path to the file containing the list of airports for processing
- <FlightDataDir> – path to the directory containing the extracted flight data
- <STARDir> – path to the directory containing the extracted STAR data
- <ResOutputDir> – path to the directory containing files produced by the Lateral Adherence Evaluation
- -dbg – optional parameter, specifying that additional data will be added to the log file.

The executable also sends some detailed processing and error information to the standard output, which can be redirected to the log file. The level of detail is controlled by the presence of the "dbg" argument.

9. Vertical Adherence Evaluation

The purpose of the vertical adherence code is to fuse all the extracted data and generate snapshots throughout an RNAV STAR route of flights which have been identified as flying that STAR. The code combines the lateral adherence evaluation results, vertical restrictions from the procedural data, the winds aloft data, and the convective weather obstructions on the route. Snapshots are computed at the compartment level of approximately 2 NMs. Compartments can either be on a segment or at a waypoint on the route. Lateral non-adherence is defined as the flight track exceeding the 1 NM tolerance distance from the center line of a segment; in the case of waypoints, non-adherence is defined as the flight track exceeding the 1 NM tolerance radius from the waypoint. Anything outside these tolerance zones is considered not adhering to the route. Classifications of non-adherence are:

1. Late Entries: Flights that do not join STAR at a procedurally defined entry point
2. Skips: Flights that are adhering to the procedure, leave the STAR, and then return
3. Early Exits: Flights that leave the STAR at an undesignated exit point and never return

Arguments

This step is implemented in Python Apache Spark large-scale data processing framework version 1.6.0; each day of flights is processed in parallel across a 56-day chart cycle. The Spark code is run with the following arguments:

- <list_of_lateral_adherence_files> – text file list with absolute path to lateral adherence files
- <output_file_name> – name of final output csv file
- <stars_to_ignore_list> – text file list of STAR names that are ignored (i.e. do not have an iteration number). STARs are ignored if the STAR procedure has inconsistencies in the waypoint ordering for different transitions that flow through the same segments after merging (e.g. the Wynde navigation fix)

Loading Snapshot Data

Loading the Lateral Adherence File

The lateral adherence files are the starting point for generating the snapshots (see Figure *10* for schema). Each file contains flight data for a single day for each airport of interest. Each flight contains data on the inferred route and subsequent timeseries and lists whether or not the track point was inside a compartment on the route. The files are parsed sequentially for each flight.

Loading the Wind Data

The rapid refresh wind data is parsed by date and airport and loaded into memory for each lateral adherence file. When computing the tail wind components, each track point's Unix time stamp is converted into a date and hour to be looked up in the extracted wind file. When the relevant time has been found, the nearest altitude is identified and wind speed and direction are pulled. The heading on the route is used to determine the intended direction. The tail wind component of the wind is then computed using the following:

Tail wind = $\mathrm{mod}\ (270 - \frac{180}{\pi} * arctan^2(v, u), 360)$

This value is logged in the snapshot along with the temperature and humidity values.

Loading the Convective Weather Impact Data

The convective weather impact data is stored in HDF5 file format for each day and airport, and contains information about the Vertical Integrated Liquid (VIL), echo tops, and Weather Avoidance Field (WAF, which is defined by MIT LL). If there is no convective weather for a given airport and day, no file is created. If there is no file present the code assumes no weather impacts and sets default values ('NaN') for the weather obstruction features in each compartment. If the file does exist it is loaded and parsed after the compartment snapshots have been created. The convective weather is represented in three different characteristics.

1. En route Transition Obstruction: For late entry flights, the filed flight plan is parsed and the en route transition fix is identified. The input segment for that route is examined at the entry time on the route in the weather impact file. If an obstruction is found, the most severe compartment on that segment is identified and logged.
2. Obstruction At: For each compartment on the route, the weather impact file is checked to see whether or not an obstruction exists. If an obstruction exists the fields are logged. This applies to flights that are abeam to skipped waypoints.
3. Obstruction Downstream:
    a. For each compartment the code looks downstream on the route to determine the first instance that weather is obstructing a compartment. The VIL, echo top, WAF, obstructed compartment name, and distance measured in number of compartments are logged. This characterizes the front of the convective weather.
    b. For each compartment the code looks downstream on the route to determine if more severe weather obstructs any compartments beyond the initial front. The VIL, echo top, WAF, obstructed compartment name, and distance measured in number of compartments are logged. This characterizes the most severe portion of the convective weather.

10. Flight-by-Flight Compartment Adherence

For each flight the track points are examined to determine which compartments on the route have lateral adherence. A list of expected compartments is compiled from the defined route. The file labels each track point as either in a compartment (with the appropriate name), or outside ("in the void"). Compartments that are not on the route but exist in other routes are labeled as well, but a -1 flag is applied to indicate that these compartments should be ignored (or are "in the void") and considered as not laterally adhering to the route. Multiple track points may be inside the same compartment; in this case the consecutive positional data for these track points will be averaged.

To determine which compartments will be logged, a sequential list of compartment names is compiled based on a.) the number of compartments between waypoints in a segment, and b.) the first and last waypoint flown on the route. The compartments flown are then sequentially compared against the expected compartments on the route. A list of missed compartments will be used to log skipping behavior.

Handling Data Quality Issues

Filtering is applied to catch spurious track points that exit and return to the same compartment within one sample. In this case, the spurious track point will be relabeled as belonging to the preceding and succeeding compartments.

Due to noisy ground track parameters, a constraint exists such that the change between track points should not exceed a certain threshold. The initial speed is determined by the average speed over the first two waypoints that are crossed, removing any points that are above 700 kns. As the speed changes, points that are above 700 kns or below 100 kns are removed. Any point that is +- 50 kns compared to the previous compartment speed is held as the previous compartment speed.

Computing Cumulative Distance Traveled

The cumulative distance traveled along the flight path is computed across the track points from a 300 NM radius down to the closest point at the destination airport. If the flight originates within the 300 NM radius then it is also labeled a regional flight.

Determining Late Entries and Early Exits

The lateral adherence code will attempt to infer the en route transition flown based on the waypoint segments identified. If upstream from the first waypoint adhered to, any en route transition is ambiguous and will be marked as '*'. As with ambiguous en route transitions, ambiguous runway transitions will be marked as '*'. Presence of a '*' will result in early and late entry flags. In some cases the en route transition is inferred but the first waypoint on the transition is not flown. When the flown versus expected compartments differ and the first waypoint is identified as missed, the flight is flagged as a late entry. The same logic applies to runway transitions; if the last waypoint of an inferred runway transition is not flown, the flight is labeled as early exit.

Stepping Through Compartments Flown

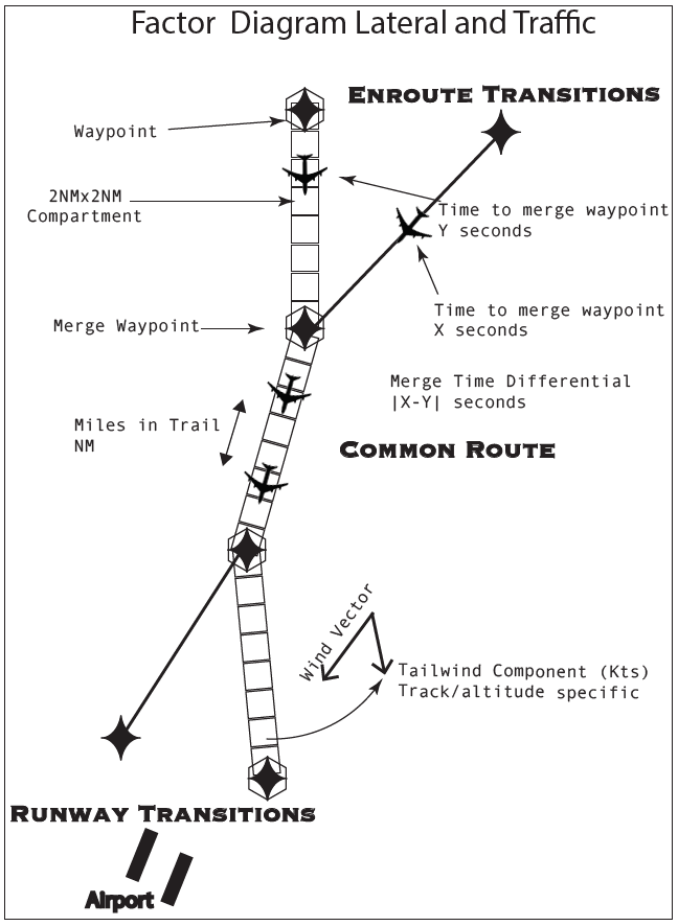
Compartment labels are parsed for each flight and the transitions are identified. The compartments flown between each transition are compared against the expected route; skipped compartments are identified beforehand so they can be handled when they are evaluated at each transition. The expected compartment pointer runs alongside when flown compartments are being assessed. If the flight begins repeating or flying compartments out of order, the repeat compartments are ignored and a flag is raised to identify this flight as having repeat behavior. When the compartment flown is labeled "in the void," the corresponding skipped compartment(s) are logged. The closest lateral distance is computed to the compartment centerline and positional data is averaged across the whole portion of the void. Time is uniformly distributed between start and end points across each compartment skipped to ensure a unique time stamp for each compartment. Compartments are handled in one of three ways:

1. **Waypoint crossing:** Vertical adherence based on the procedure is applied at this point. A waypoint's position above or below vertical restrictions is assessed and the magnitude is logged as +-300 ft. Other procedural information such as slopes and airspeed restrictions are also logged at this point. If procedural data is not known the values will be set to 'NaN'. All derived procedural information is "backwards looking", meaning the slope at a particular waypoint is the slope from the previous waypoints to that waypoint. Similarly, heading and deceleration ratios are also derived in a "backwards looking" fashion.
2. **Segment compartments:** Compartments are 2 NM apart on a segment, though the last compartment may be less than 2 NM if the segment is not evenly divisible by 2 NM. Snapshots from the positional information are derived from the flight and logged along with lateral adherence information.
3. **Void:** A void is defined when a track point is not labeled as inside a compartment on the route. When this corresponds with an expected compartment that is not flown through on the route, the expected compartments are labeled as skipped. One or more compartments can be skipped in a row and can be segment compartments or waypoints on the route. Skipped compartments are logged with the same behavior as segment compartments and waypoint compartments. Note: vertical adherence is not enforced on skipped waypoints.

Features Based on Daily Aggregated Adherence

Some features require aggregating data throughout the day to compute features that characterize traffic flow and the interactions between flights before merging waypoints.

1. **STAR Flow:** The first and last compartment for each flight on the route is logged, and for each flight and each compartment the flights are counted if the target time is between the start or end point.
2. **Merging Interactions:** After all flights have been mapped to compartments and their skipping behavior has been identified, the compartments that precede merge waypoints are examined. As a flight converges to a merge waypoint, the ground speed and remaining distance are used to compute the estimated arrival time for the flight given the current time at the compartment for all compartments and flights that are arriving at the next merge waypoint. Once these are computed, the flight steps through each



compartment and filters on other flights with a similar distance to the target merge waypoint (i.e., flights within +-1 NM of the waypoint). Then the estimated arrival time is compared with the target flight's estimated arrival time at the merge waypoint. The shortest arrival time differential (merge time delta) is identified and linked back to the candidate flight's current compartment. The difference between the tail wind and ground speed of the target flight and candidate flight are also computed at each respective compartment. Finally, the compartments between the current compartment and the merge waypoint are examined to determine whether either flight skipped any of the compartments. The results of this examination are labeled as 'N' if neither flight skipped, 'S' if the target flight skipped, 'C' if the candidate flight skipped, or 'B' if both flights skipped. The same evaluation is made for vertical deviations (see ments yields the miles in trail).

Figure 11 Diagram of merging flights and miles in trail

3. **Miles in Trail:** All crossing times are searched for each flight at each compartment. Flights that crossed the compartment before the flight at the target compartment are checked and the most recent one is logged. The difference between these crossing times is calculated, which yields the time since the most recent flight. The time the identified flight crossed the compartment is used to look up which compartment the target aircraft flew over. The difference in distance traveled between the two compartments yields the miles in trail.

Features Based on Chart Cycle Aggregated Adherence

After daily flights are assessed for all days within a chart cycle, additional features are computed based on statistical characteristics. For flights categorized as "late entry," the joining waypoint is logged. For each STAR the join waypoints are counted and the median is computed. All waypoints that had counts over mean+1 standard deviation are marked as a common join waypoint.

See Appendix C for a list of snapshot values logged at each compartment.

11. Synopsis

The RADI system described in this technical memorandum outlines a framework that can be used to fuse a variety of data including surveillance recordings, environmental observations, and procedural information to produce features that would otherwise not be observable by any single data source. The process is designed with scalability in mind so that large scale batch processing can be executed on a typical distributed cluster environment. This process was initially developed as a prototype to quickly assess adherence and iterate and engineer relevant features of interest that can assist in determining factors for non-adherence to procedural requirements. With this in mind there are areas where the prototype can be improved for efficiency, both in terms of file I/O access and storage, once the initial research questions have been answered and the process becomes part of an in-time decision support tool.

12. Appendix

Appendix A: Data Fields in Extracted Files with STAR Descriptions

Field Name	Description
RecType	Record Type Code (0-3)
AirportID	4-character airport code (e.g. KIAH)
Lat	Airport or waypoint latitude (decimal)
Long	Airport or waypoint longitude (decimal)
Elevation_ft	Airport elevation (feet)
Mag_variation	Airport magnetic variation (degrees)
AOI_radius	Airport AOI radius (nautical miles)
WPID	5-character waypoint ID (padded with blanks for Nav Aid)
StereoProj_X	Waypoint x-coordinate in local stereographic projection (meters, decimal)
StereoProj_Y	Waypoint y-coordinate in local stereographic projection (meters, decimal)
LambertProjGrid_X	X-component of the point 1 km grid coordinate in the Lambert Equal Area projection used in CIWS files (integer)
LambertProjGrid_Y	Y-component of the point 1 km grid coordinate in the Lambert Equal Area projection used in CIWS files (integer)
RouteID	Character string identifying common part, en route transition, or runway transition of the STAR
RouteType	1: Common part 2: En route transition *or* 3: Runway transition
TrunkRouteID	STAR common part name if RouteType = 2 or 3, or '*' if RouteType = 1
StartWPID	Start waypoint ID of the segment
EndWPID	End waypoint ID of the segment ('*****' for the last waypoint of the STAR component)
AltLow	Altitude lower limit (feet) at the start waypoint of the segment ('NaN' if not defined)
AltHigh	Altitude upper limit (feet) at the start waypoint of the segment ('NaN' if not defined)
AltDescr	A: Lower and upper limits are the same F: Only lower limit altitude is specified C: Only upper limit altitude is specified W: Both altitudes are specified N: None specified
LegType	Leg type code as defined in ATAC XML schema
SpeedLimit	Speed limit (knots) at the start waypoint of the segment ('NaN' if not defined)
SlopeLow	Flight trajectory slope along the segment calculated from lower altitude limits at start and end waypoints (degrees, decimal with 0.01 precision; 'Nan', if cannot be calculated)

SlopeHigh	Flight trajectory slope along the segment calculated from upper altitude limits at start and end waypoints (degrees, decimal with 0.01 precision; ‘Nan’, if cannot be calculated)
Distance	Segment length (nautical miles, decimal with 0.01 precision)
Heading (true)	True heading at start waypoint of the segment to end waypoint (degrees, integer)

Appendix B: C Structures for STAR Arrangement Construction

```
// start and end waypoints of the segment
struct leg_data_t
{
   int wp_start_ind;
   int wp_end_ind;
};
```

```
// airport info
struct airport_info_t
{
   std::string ap_id;
   double ap_lat;
   double ap_long;
   double elev;
   double magvar;
};
```

```
// waypoint info
struct wp_info_t
{
   std::string wp_id;
   double wp_lat;
   double wp_long;
   double wp_x;
   double wp_y;
   std::vector<long> segm_starts;
   std::vector<long> segm_ends;
};
```

```
// ‘leg’ info
struct route_leg_info_t
{
   long segm_id;
   std::string leg_type;
   std::string alt_descr;
   long alt1;
   long alt2;
   int speed_lim;
   double slope_low_to_next;
   double slope_high_to_next;
   double dist_to_next;
   int heading_to_next;
};
```

```
// Route description
struct route_info_t
{
   int rt_type;
   std::string trunk_rt_id;
   std::vector<route_leg_info_t>  rt_legs;
};
```

```
// segment as part of a route
struct segm_route_info_t
{
   std::string route_id;
   int segm_seq_no;
};
```

```
// segment information
struct segm_info_t
{
   int comp_cnt;
   std::vector<segm_route_info_t>  segm_routes;
};
```

```
struct flight_info1_t
{
   int flt_key;   // unique key assigned to flight
   std::string call_sign; // aircraft call sign
   std::string ac_type;  // aircraft type
   std::string perf_ctg; // aircraft performance category
   std::string weight_class; // aircraft weight class
   std::string dest_ap;  // destination airport
   std::string landing_rwy;  // landing runway
   long landing_time;        // seconds since midnight Jan. 1, 1970 (UTC)
   std::string flt_plan_STAR; // filed STAR
   std::string equip_list;    // equipment list
};
```

```
// Flight track point information
struct track_point_info_t
{
   double rec_time;
   double pt_lat;
   double pt_long;
   double pt_x;
   double pt_y;
   int    pt_alt;              // altitude in hundreds of feet
   long   pt_climb_rate;       // rate of climb in feet
   int    pt_course;           // heading (degrees north)
   int    pt_ground_speed;     // ground speed (knots)
   int    std_rt_bzone_type;   // -1 - undefined zone, 0 - "void", 1 - waypoint buffer zone, 2 -
segment compartment
   int    std_rt_bzone_wp_ind;    // waypoint index (defined, if std_rt_bzone_type = 1)
   long   std_rt_bzone_segm_id;   // segment id (defined, if std_rt_bzone_type = 2)
   int    std_rt_bzone_cmpt_ind;  // segment compartment index (defined, if
std_rt_bzone_type = 2)
};
```

Appendix C: Snapshot Values Logged at Each Compartment

Parameter Name	Description
Airport	Destination airport.
STAR	Inferred STAR route. 'NONE00' for flights that did not fly a STAR. A 'NONE00' compartment will only have airport waypoint and logged destination airport with landing time and cumulative distance traveled from 300 NM, or in the case of regional flights distance from initial radar track point.
Route_Name:	Inferred subroute name "Enroute Transition.Common Route.Runway Transition"
Waypoint	Compartment name. Segment compartments are defined as a concatenation of the 5-letter start waypoint (for VORs the value is the 3-letter code followed by two spaces) followed by the end waypoint with compartment number starting at '00', not to exceed '100'. Waypoints will be defined using just the 5-letter waypoint. Airport will be defined as the 4-letter airport code.
Date	YYYYMMDD
Unix_Time	Integer value.
Magnitude_of_Excursion (ft)	Vertical non-adherence above or below indicated as +- 300 ft. Only applies to waypoints with restrictions. Vertically adhering crossings are left empty.
Flight_ID	Unique flight number plus the last four seconds of the Unix time for landing.
AC_Type	Aircraft type, e.g. B739, CRJ7, etc.
Equipment	Equipment type, e.g. /L, /G. Filtered only on flights equipped for RNAV arrivals.
Flight_plan	Last amended flight plan.
Dest_RW	Landing runway as reported by Sherlock.
Alt_L	Lower Altitude restriction on waypoint. If none, defined as 'NaN'.
Alt_H	Upper Altitude restriction on waypoint. If none, defined as 'NaN'.
WP_Type	Floor ('F'), Ceiling ('C'), Window ('W'), At ('A'), None ('N'), Segment ('S'), or Airport ('NaN').

FAA_Slope_L	FAA defined lower slope. Looking backwards to last known lower bound. If previous waypoint has no restriction, continue back until lower bound is defined.
FAA_Slope_H	FAA defined upper slope. Looking backwards to last known upper bound. If previous waypoint has no restriction, continue back until upper bound is defined.
Window_Size	Difference between upper and lower bounded restrictions. Only applies to compartments with WP_Type of Window or At. (0)
Required_Slope_L_From	Flight-centric. Based on the flight's current altitude looking back to the previous lower bound altitude restriction and computing slope. May be in a segment compartment.
Require_Slope_H_From	Flight-centric. Based on the flight's current altitude looking back to the previous upper bound altitude restriction and computing slope. May be in a segment compartment.
Required_Slope_L_To	Flight-centric. Based on the flight's current altitude looking forward to the next lower bound altitude restriction and computing slope. May be in a segment compartment.
Require_Slope_H_To	Flight-centric. Based on the flight's current altitude looking forward to the next upper bound altitude restriction and computing slope. May be in a segment compartment.
Alt_at_WP	Current snapshot of flight's altitude.
Alt_Prev_WP	Carries over flight altitude value from the flight's previous waypoint.
Descent_Rate_At (ft/min)	Current snapshot of the decent rate at a waypoint. Computed by Sherlock.
Descent_Rate_Prev (ft/min)	Carries over decent rate snapshot from the flight's previous waypoint. Computed by Sherlock.
Grd_Speed_at_WP (kns)	Ground speed snapshot. Adjusted by data quality filter (see section above).
Grd_Speed_Prev (kns)	Carries over ground speed snapshot from the flight's previous waypoint. Adjusted by data quality filter (see section above).
Arrival_Rate_Past (n/15mins)	Count of the number of flights that landed at the specified airport within the last 15 minutes.
Tail_wind (kns)	Tail winds snapshot from RR file.
percent_humidy	Percent humidity snapshot from RR file at the altitude of the flight. Rounded to nearest 1,000 ft.
Temp (C)	Temperature snapshot from RR file at the altitude of the flight. Rounded to nearest 1,000 ft.
Perc_WP_Complete	The sequence of waypoints in the route divided by total number of waypoints. Behavior on ambiguous routes currently undetermined.
Perc_Dist_Complete	Distance across the sequence of waypoints and compartments in the route divided by total distance of the route.
Prev_Excursion_Count_On_Route	Running tally of the number of vertical excursions on the route before the current compartment.
Deviation_above	Boolean '1'. If the waypoint is a deviation above the restriction then '0', otherwise includes waypoints and/or compartments with no restrictions.

Deviation_below	Boolean ‘1’. If the waypoint is a deviation below the restriction then ‘0’, otherwise includes waypoints and/or compartments with no restrictions.
Deviation_Any	Boolean OR of deviation above|below represented as {1,0} for each waypoint and/or compartment.
Full_Vert_Comp	Boolean ‘1’. If all waypoints have deviation then ‘0’. Will be uniform across the full flight.
Max_Lat_Dev	Looks at the track points along a skip section. Computes the bearing and distance from the previous fix to each track point in the skipped section. Using the skipped compartment’s bearing, the lateral component is derived and the maximum distance is recorded.
Had_Late_Entry	Boolean ‘1’ for all compartments for a flight if the input transition is ambiguous or the first compartment is not flown on the inferred input transition, otherwise ‘0’.
WP_Skip	Boolean ‘1’ if a waypoint and/or compartment is skipped, otherwise ‘0’.
Had_Skip	Boolean ‘1’ if a skip occurs at any point along the route for a flight, otherwise ‘0’. Will be uniform across the full flight.
Had_Early_Exit	Boolean ‘1’ if the output transition is ambiguous, or the last waypoint in the output transition is not flown, otherwise ‘0’.
WP_Only_Full_Lat_Comp	Boolean ‘1’ if no skips, late entry, or early exits are found, otherwise ‘0’. Only looks at waypoints for lateral compliance.
Full_Lat_Comp	Boolean ‘1’ if no skips, late entries, or early exits are found, otherwise ‘0’. Looks across the full set of compartments.
Airspeed_Restriction (kns)	The airspeed restriction in knots as defined by the procedure.
Dynamic_Press (kns^2)	Airspeed restriction squared. This is derived to account for drag differences at different airspeeds.
Num_Speed_Restrict_in_Subroute	Total number of speed restrictions for the entire subroute.
Num_Alt_Restrict_in_Subroute	Total number of altitude restrictions for the entire subroute.
Dist_To_Last_Restrict (NM)	Distance in nautical miles to the last altitude restriction.
Cumulative_Dist_Travel_On_Route (NM)	Cumulative distance traveled for the route up to the current compartment. Includes skipped tracks from 300 NM.
Regional_Flight	Flag. Regional flights are defined as those flights which emerge within 300 NM of the airport.
Reverse_Flight	Flag. Reverse flights are defined as those flights which include compartments repeated out of order during the flight trajectory. Repeated compartments are ignored and not revisited when logging snapshots.
Repeated_WP	Labels compartments that were repeated during a reverse flight.
Level_Off_Before	Looking back four NM in flights with deviations at waypoint restrictions, Boolean ‘1’ if the altitude at the waypoint crossing is the same +-300 ft, otherwise ‘0’. Used to infer whether a flight was intentionally missing before the waypoint restriction.
Decel_Ratio (kns/NM)	Looks at the previous airspeed restriction and computes the difference in airspeed restrictions divided by the distance between waypoints.
true_airspeed	Ground speed with tail wind component added.

Time_since_last_flight (sec)	Computes the time difference (secs) between the last flight that flew over the compartment and the flight currently flying over the compartment.
Miles_in_Trail (NM)	Uses time_since_last_flight to look up the closest time stamp in the current flight the looks at the snapshot of cumulative distance traveled at the time the last flight flew over the compartment as well as the current time the flight is flying over the compartment. The distance is the Miles_in_Trail value.
STAR_flow	Count of all flights that are currently on the same STAR at the recorded time. Flights must have started the STAR after a late entry and be on the STAR before an early exit. Flights that are currently skipping are still counted as on the STAR because they will return.
All_STAR_flow	Count of all flights that are currently on any STAR at the recorded time. Flights must have started the STAR after a late entry and be on the STAR before an early exit. Flights that are currently skipping are still counted as on a STAR because they will return.
Arc_flow (degs)	Regardless of STAR adherence, all flights are checked at the 300 NM radius boundary for their bearing from the airport. Regional flights are checked for their initial point of entry into the airspace. The arc_flow radial is used as a label to determine the "slice" a flight belongs to. For each compartment, all flights within +- 22.5 degrees are queried to determine the flow of aircraft across the 45 deg arc.
Merge_WP (0/1/2/3)	Waypoint label '0' if not a merging waypoint; '1' if merge on a STAR; '2' if merge between STARS; and '3' if both merge on a STAR and between STARs.
Late_Entry_Join_Waypoint	Looks at the first waypoint for flights that were late entries for each STAR and computes the median waypoints used. Takes only those waypoints that occur over 1 std + mean or more. These waypoints will be common joining waypoints.
estimate_arrival_time_at_next_merge (Unix time)	Estimated arrival time at the next merge waypoint using current ground speed and remaining distance on the route. Result is added to current time (Unix time) to obtain arrival time.
target_merge_wp	Identifies the next upcoming merge waypoint.
merge_time_delta	For a given flight considers flights that are within 1 NM of that flight on a transition. Looks at estimated arrival times at the merging waypoint for the identified flights and finds the flight with the closest arrival time. Positive value indicates the flight is the lead flight, Negative value indicates it is the trailing flight.
grd_speed_merge_delta	For a given flight considers flights that are within 1 NM of that flight on a transition. Looks at estimated arrival times at the merging waypoint for the identified flights and finds the flight with the closest arrival time. Computes ground speed difference between both flights at that time snapshot.
tail_wind_merge_delta	For a given flight considers flights that are within 1 NM of that flight on a transition. Looks at estimated arrival times at the merging waypoint for the identified flights and finds the flight with the closest arrival time. Computes tail wind differential between both flights at that time snapshot.

dist_to_merge_wp	Remaining distance to the upcoming merge waypoint.
lead_flight_in_merge_conflict	Boolean '1' if the flight is the lead flight in the merging conflict.
Before_Merge	Labels whether there is an upcoming merging waypoint. Uses same labeling values to indicate the target waypoint as Merge_WP.
skip_before_merge	Identifies whether or not the flights in question skipped before the merge. 'N' if neither flight skipped; 'S' if the target flight skipped; 'C' if the conflicting flight skipped; or 'B' if both flights skipped.
vert_dev_before_merge	Identifies whether or not the flights in question had vertical deviation before the merge. 'N' = if neither flight deviated; 'S' if the target flight deviated; 'C' if the conflicting flight deviated; 'B' if both flights deviated.
conflicting_flight_id	Flight ID of identified conflicting flight.
VIL_downstream (kg/m^2)	Looking downstream, the first instance of WAF > 40% vertical integrated liquid content as recorded in CIWS data.
EchoTop_downstream (K-ft)	Looking downstream, the first instance of WAF > 40% echo tops content as recoded in CIWS data.
WAF_downstream (%)	Looking downstream, the first instance of WAF > 40%. WAF is computed in a lookup table based on VIL and echo tops content from CIWS data.
dist_weather_obstruct_downstream (N-Compartments)	Number of compartments to the first obstructed WAF > 40%.
compartment_downstream	Name of compartment obstructed by a WAF > 40%.
VIL_more_severe_downstream (kg/m^2)	Looking downstream, records the VIL if there is a higher VIL beyond the initial WAF > 40%. If none, 'NaN'.
EchoTop_more_severe_downstream (K-ft)	Looking downstream, records the echo top if there is a higher echo top beyond the initial WAF > 40%. If none, 'NaN'.
WAF_more_severe_downstream (%)	Looking downstream, records the WAF if there is a higher WAF beyond the initial WAF > 40%. If none, 'NaN'.
dist_weather_obstruct_more_severe_downstream (N-Compartments)	Looking downstream, records the distance in number of compartments if there is a higher WAF beyond the initial WAF > 40%. If none, 'NaN'.
compartment_more_severe_downstream	Looking downstream, records the compartment name if there is a higher WAF beyond the initial WAF > 40%. If none, 'NaN'.
VIL_on_filed_input-seg (kg/m^2)	Using the initial waypoint on the filed flight plan, looks at the compartments of the first segment. Records the VIL of the compartment obstructed with the highest WAF. If none, 'NaN'.
EchoTop_on_filed_input-seg	Using the initial waypoint on the filed flight plan, looks at the compartments of the first segment. Records the echo top of the compartment obstructed with the highest WAF. If none, 'NaN'.
WAF_on_filed_input-seg (%)	Using the initial waypoint on the filed flight plan, looks at the compartments of the first segment. Records the WAF of the compartment obstructed with the highest WAF. If none, 'NaN'.
compartment_on_filed_input-seg	Using the initial waypoint on the filed flight plan, looks at the compartments of the first segment. Records the compartment name of the compartment obstructed with the highest WAF. If none, 'NaN'.

WAF_at	Records the WAF value for each compartment at the time the flight is either in the compartment or abeam during a skip. If none, 'NaN'.
EchoTops_at	Records the echo top value for each compartment at the time the flight is either in the compartment or abeam during a skip. If none, 'NaN'.
VIL_at	Records the VIL value for each compartment at the time the flight is either in the compartment or abeam during a skip. If none, 'NaN'.

13. References

[1] "Federal Aviation Administration: Instrument Procedures Handbook," FAA-H-8083-16B, 2017

[2] Pyproj package: https://jswhit.github.io/pyproj/pyproj-module.html

[3] M. Eshow, M. Lui, S. Ranjan, "Architecture and Capabilities of a Data Warehouse for ATM Research", *Digital Avionics Systems Conference (DASC) 2014 IEEE/AIAA 33rd*, 2014

[4] NETCDF: https://www.unidata.ucar.edu/software/netcdf/

[5] HDF5: https://www.hdfgroup.org/

[6] Michael P Matthews, Rich DeLaura "Modeling Convective Weather Avoidance of Arrivals in the Terminal Airspace" MODELING CONVECTIVE WEATHER AVOIDANCE OF ARRIVALS IN THE TERMINAL AIRSPACE

[7] USGS: PROJ4 Cartographic Projection Procedures for the UNIX Environment—A User's Manual, Open-File Report 90–284, by Gerald I. Evenden, Revised in January 2003

[8] CGAL, Computational Geometry Algorithms Library, http://www.cgal.org