MTR120423

MITRE TECHNICAL REPORT

Threaded Track: Geospatial Data Fusion for Aircraft Flight Trajectories

Product 10-2.2-1

Adric Eckstein
Chris Kurcz
Marcio Silva

August 2012

Abstract

Previous studies demonstrated the capability of associating surveillance trajectories to produce a synthetic track with the best possible coverage and fidelity referred to as the threaded track. The previous process relied on synthesis of radar trajectories using identifying information (metadata) such as callsign, beacon code, and aircraft type. However, a substantial portion of general aviation flights contain no identifying information, and, thus were not included in the threaded track data repository. This study improves upon the previous methodology by removing the requirement of populated metadata for fusing flight surveillance trajectories. Instead, the updated threaded track process associates trajectories based on their temporal and spatial proximity. By employing this approach, a significant amount of data to the threaded track repository has been added and the complexity and computational cost has increased. This report describes the modifications to the threaded track process workflow and the characteristics of the resulting flights.

Table of Contents

1 Introduction **1-1**

2 Limitations of Previous Threading Algorithms **2-1**

3 Methodology **3-2**

3.1 Source data 3-4

3.1.1 NOP Data 3-4

3.1.2 ASDE-X Data 3-4

3.1.3 ETMS Data 3-4

3.1.4 ADS-B Data 3-4

3.2 Preprocessing and Filtering 3-5

3.2.1 Segmentation 3-5

3.2.2 Coasting and Outlier Detection 3-6

3.2.2.1 Low Variability Filter 3-6

3.2.2.2 Coasted Tracker Filter: ARTCC 3-7

3.2.2.3 Coasted Track Filter: ASDE-X 3-7

3.2.2.4 Epoch Time 3-7

3.2.2.5 Stationary Point Filter 3-7

3.2.2.6 Identical Position Filter 3-7

3.2.2.7 Identical Time Filter 3-7

3.2.2.8 Radar Registration 3-8

3.2.2.9 Long Range Filter 3-8

3.2.2.10 Lateral Outlier Filter 3-8

3.2.2.11 Vertical Outlier Filter 3-8

3.2.2.12 Speed and Heading Estimate 3-8

3.2.2.13 Ground Point Filter: ARTCC 3-8

3.2.2.14 Ground Point Filter: ARTS 3-9

3.2.2.15 Ground Point Segmentation 3-9

3.2.3 Radar Registration Errors 3-9

3.3 Geospatial Joining 3-10

3.3.1 Geohashing the Track Segments 3-11

3.3.2 Correlating pairs 3-12

3.3.3 Community Detection 3-13

Phase 1: Sub-Networks. 3-14
Phase 2: Network Splitting. 3-14
Phase 3: Ground Segment Joining. 3-15
3.4 Track Synthesis 3-15
3.4.1 Filter Stages 3-16
3.4.2 Windowed Least Squares Filtering 3-17
3.4.2.1 Windowing Function 3-18
3.4.2.2 Sensor Models 3-19
3.4.3 Adaptive Piecewise Least Squares Filtering 3-20
3.4.4 Resampling 3-21
3.5 Post Fusion with ETMS 3-22
4 Performance 4-22
4.1 Computational Tools 4-22
4.1.1 Apache Hadoop 4-22
4.1.2 Apache Avro 4-23
4.1.3 Apache Oozie 4-23
4.1.4 Matloop 4-23
4.2 Execution Plan 4-24
5 Data Output Characteristics 5-26
6 Future Improvements and Recommendations 6-30
6.1 Missing Features 6-30
6.1.1 Segment fusion across gaps 6-30
6.1.2 Addition of new data sources 6-30
6.1.3 Clock error compensation 6-31
6.1.4 ADS-B altitudes 6-31
6.1.5 Phases of Flight 6-31
6.2 Performance 6-31
6.3 Verification and Validation 6-31
6.4 Obstacles to Production 6-32
7 References 7-1
Appendix A Radar Registration Equations A-1
A.1 Radar Registration Correction: A-1

A.2 Slant Range Correction: A-1
A.3 Slant Range Error Propagation: A-2
A.4 Vertical Error Model: A-2
Appendix B Cross Track Models B-1
B.1 Straight Least Squares Model: B-1
B.2 Turn Least Squares Model: B-1
B.3 Mixed Model Solution: B-2
Appendix C Acronyms C-1

List of Figures

Figure 1. Threaded Track Process Workflow. 3-3
Figure 2. Coasting and outlier detection workflow. 3-6
Figure 3. Derived radar declination of NKX sensor from SCT TRACON during 2011. 3-10
Figure 4. Example illustrating geospatial joining. 3-12
Figure 5. Distribution of lateral correlation scores versus time (top) and marginal lateral distributions by metadata quality (bottom) between ARTS and ARTCC facilities. 3-13
Figure 6. Splitting a sub-network into 5 distinct groups. 3-14
Figure 7. Lateral (left) and vertical (right) plots of the trajectory segments from each of the groups in the sub-network splitting. 3-15
Figure 8. Windowed least squares filtering process. 3-17
Figure 9. Least Squares models for cross track filtering. 3-18
Figure 10. Windowing functions for cross track and along track filters. 3-19
Figure 11. Change point estimation filtering process. 3-20
Figure 12. Sample vertical trajectory and its least squares segmented profile. 3-21
Figure 13. Percentage of CPU time for the map-reduce workflow. 4-25
Figure 14. Input and output data sizes for the map-reduce workflow. 4-25
Figure 15. Optimal wall clock time for the map-reduce workflow in CAASD's Hadoop Cluster. 4-26
Figure 16. Categorized segment group count (left) and track point count (right) 5-27
Figure 17. Lateral plot of track groups, colored by data set. 5-28
Figure 18. Comparison of associated and unassociated data for the fused ETMS set. 5-29
Figure 19. Sample flight showing its sensor coverage (left) and the fused trajectory (right). 5-29

List of Tables

Table 1. Derivation of synthetic trajectory parameters. 3-16
Table 2. CAASD's Hadoop Cluster Characteristics. 4-23
Table 3. Physical execution plan for map-reduce jobs. 4-24

1 Introduction

Historical analysis of aviation systems often requires aircraft flight trajectories that traverse multiple regions of coverage, preferably in high fidelity gate-to-gate measurements. However, the surveillance system which Air Traffic Control (ATC) uses to monitor the National Airspace (NAS) in real time was not designed to capture all of a flight's surveillance data. In addition, aircraft are not assigned a unique identifier which persists for the duration of the flight[1]. Due to the lack of unique flight identifier, constructing an end to end flight trajectory given position reports from Terminal Radar Approach Control (TRACON), Air Route Traffic Control Centers (ARTCC) and Airport Surface Detection Equipment, Model X (ASDE-X) facilities in the NAS is a non-trivial task.

In 2011, CAASD began the development of the Threaded Track which is the compilation of all available surveillance sources into a synthetic trajectory that represents an optimal representation of an aircraft's end to end trajectory [1]. It was envisioned that the Threaded Track would become the universal trajectory data source to support a wide range of safety, security, and efficiency analyses. In this first iteration, the scope of the problem was confined to the fusion of flights which could be linked through identifiable information (callsign, aircraft type, etc), which enabled a simpler methodology for the process. However, this limitation removed a significant portion of General Aviation (GA) traffic, which is often required for many of these analyses. In order to capture these flights a fundamental shift in the Threaded Track process was required.

This paper describes a new process for associating position reports across a variety of surveillance sources that form the Threaded Track. We have developed a robust and efficient methodology to associate data from distinct surveillance sources simply by examining spatial and temporal proximity. In short, two tracks from different facilities belong in the same flight if they overlap in time and the reported positions from each source are close to one another. By considering the problem in this way, we remove the need for flight identifiers to be supplied by the surveillance system. Unlike previous versions of Threaded Track, this methodology requires application of network analysis techniques to determine the set of surveillance data that will be transformed into a Threaded Track.

2 Limitations of Previous Threading Algorithms

The first versions of Threaded Track linked radar trajectories from disparate sources using flight metadata fields which included callsign, aircraft type, arrival/departure airports, and mode-A beacon code. While the common set of this metadata varied between data sources, a requirement was imposed that each dataset have the callsign of the aircraft. The callsign was then used as a primary identifier to construct candidate links between trajectories and source dependent business rules were applied to establish a collection of trajectories from distinct sources.

The callsign requirement minimized the false positives caused by linking separate flights with poor or missing metadata fields. However, this requirement also greatly reduced the amount of

[1] While there are flight identifiers (such as callsign and beacon code), these are not available in all flights, and often exhibit instances of corruption, modification, or non-uniqueness.

surveillance data used to form the Threaded Track since two-thirds of the input data contains little or no metadata. In what follows, we will refer to these trajectories without callsign as *unassociated*. The unassociated radar data can typically be classified as:

- (Visual Flight Rules) VFR and GA traffic
- (Instrument Flight Rules) IFR over-flights and redundant coverage
- Noise

Adding VFR traffic to the Threaded Track data was a primary focus since the majority of these flights do not exist in the current Threaded Track. These flights are of particular interest since a significant number of safety and security events (e.g. Traffic Collision and Avoidance System (TCAS) Resolution Advisory (RA) messages) involve a VFR aircraft. The unassociated IFR over-flights also offer great potential to improve the fidelity of the fused trajectory though higher accuracy and redundant coverage.

While the unassociated data contains a substantial amount of useful information as indicated above, it also contains a substantial portion of noise. This noise contains non-aircraft measurements, ghost hits, and other errors. Separating out these features from the desired information is not trivial and requires a great deal of filtering and processing to remove the noise while preserving valid trajectories. As a consequence of including unassociated data, it will become critical for many analyses to classify which fused trajectories are noise and which are useful flights.

3 Methodology

In order to fuse the unassociated trajectory data into the Threaded Track, we have developed a methodology which fuses trajectory data requiring only spatial and temporal correlations between data sources (but using metadata when possible to support the data fusion). Therefore this approach does not require persistent and distinct metadata fields. Accurate and efficient means for fusing these sources based on the trajectory alone required substantial modification to the current Threaded Track process and is illustrated in Figure 1. Each component in Figure 1 is described in further detail in the following sections.

The Threaded Track process begins with capture of all pertinent surveillance data within a specified period of time (see Section 3.1 for a description of the data). Phase 1 begins when data from each source is formed into trajectory segments which can be thought of as primitive tracks that exist only within the bounds of the source facility (see Section 3.2.1). These trajectory segments are filtered and processed to remove corrupted and outlier points (see Section 3.2.2). Next, a similarity measure (based on proximity point by point) is assigned to all airborne trajectory segment pairs that are deemed close enough to one another (see Sections 3.3.1 and 3.3.2). Given the network of all trajectory segments linked by similarity measure, segment groups are defined as a collection of trajectory segments using a community detection method (see Section 3.3.3). Ground segments are then added to these flights and they are filtered and smoothed to form Threaded Tracks (see Section 3.4). Phase 2 is the final step where Enhanced

Traffic Management System (ETMS) segments are associated with Threaded Tracks by reapplying the same processing geospatial fusion steps using the output of Phase 1 in Phase 2.

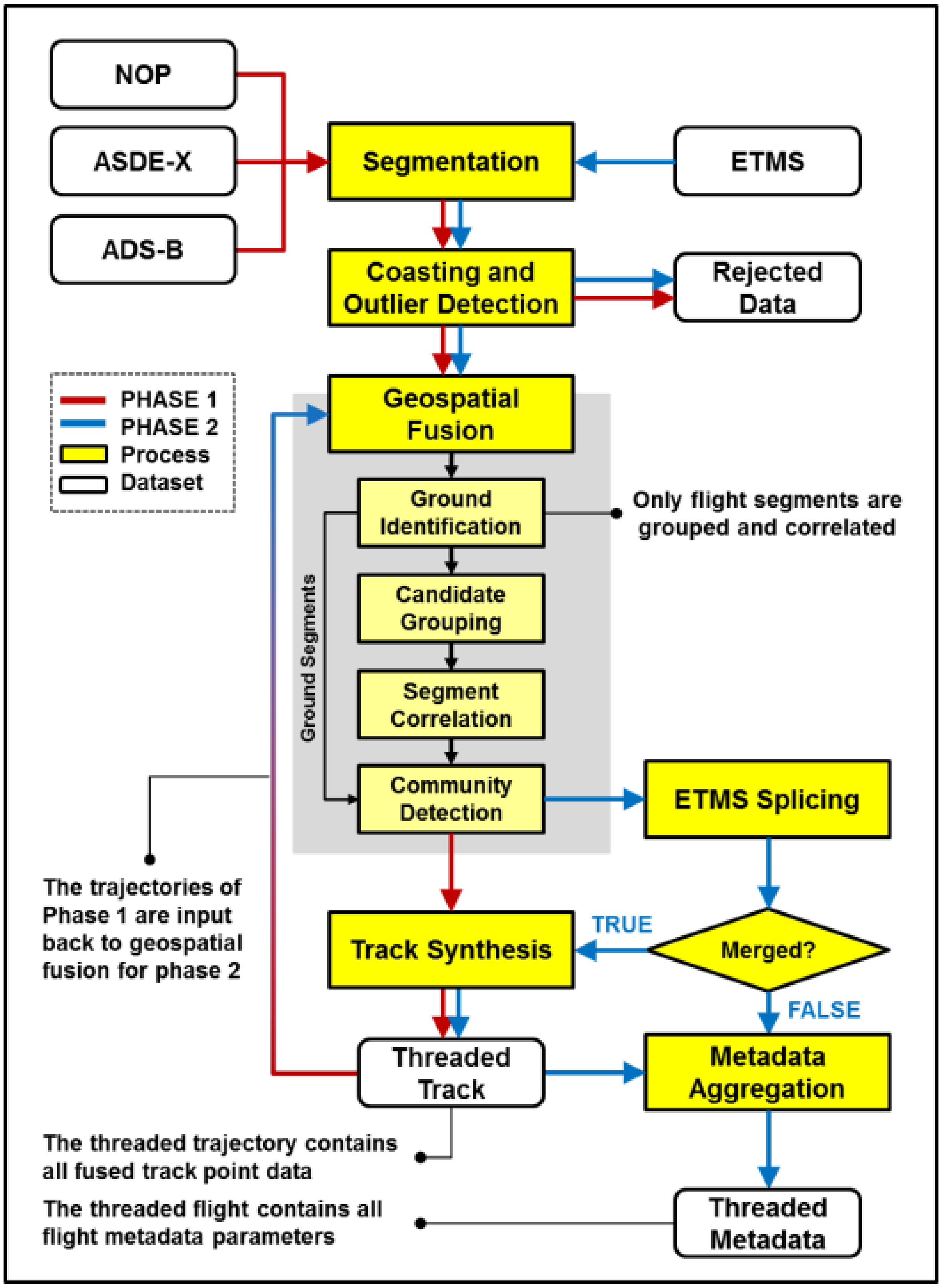

Figure 1. Threaded Track Process Workflow.

3.1 Source data

The Threaded Track is a fused set of information from all available aircraft surveillance data sources. The minimum requirements for the data set are a set of consecutive latitude/longitude position updates identified at specific times. Altitude is not a requirement, allowing for non-transponding aircraft to still be identified and fused.

3.1.1 NOP Data

The current Threaded Track is derived solely from radar surveillance sources. One of the primary data feeds to CAASD is the National Offload Program (NOP) which provides coverage at 158 TRACONs and 20 ARTCCs. The ARTCC facilities provide a post-processed set of Common Message Set (CMS) data, whereas the TRACON facilities come directly from Standard Terminal Automation Replacement System (STARS) or Automated Radar Terminal System (ARTS) facilities, each reporting a separate post-processed data format (108 ARTS facilities and 50 STARS facilities). Both ARTCC and STARS facilities compute multi-sensor fused reports, whereas the ARTS facilities provide individual radar sensor reports. ARTCC data has the largest spatial coverage per facility, with high quality flight plan and metadata reported at 12 second updates. TRACON facilities have higher fidelity position measurements reported at roughly 4 second updates, but are generally limited in this coverage to about 60 Nautical Miles (NM) of the airport and contain sparser flight metadata.

3.1.2 ASDE-X Data

In addition to NOP radar data, the ASDE-X data provides coverage of airport operations including ground movements for 35 airports. The ASDE-X data feed to CAASD is raw binary data captured by SENSIS and parsed into a comma separated values (csv) format. The trajectories generated by the ASDE-X system are fused from a range of measurements including: the surface movement radar (SMR) which provides highly accurate ground position reports at 1 second updates, the multilateration array which provides highly accurate positions (subject to the antenna geometry) within the first couple miles of flight around the airport at 1 second updates, and the airport surveillance radar (ASR) which is the same source that feeds the TRACON facilities, and will provide coverage to roughly 10 NM from the airport in ASDE-X.

3.1.3 ETMS Data

The Enhanced Traffic Management System (ETMS) also provides surveillance data in a CAASD derived data set from ASDI, in roughly one minute updates. This dataset also originates from the same CMS data as NOP Center coverage, but integrates flight plan information from CMS, enabling trajectories to be stitched across large coverage gaps (eg Hawaii flights).

3.1.4 ADS-B Data

Automatic Dependent Surveillance Broadcast (ADS-B) data provides a unique data set to this process in that its measurements are not derived from radar systems. ADS-B position reports are downlinked messages from onboard aircraft instrumentation which are recorded from ground based radio stations receiving the signals. The locations of these radio stations determine the coverage of the ADS-B data. ADS-B reports contain one second updates and provide a very consistent data quality throughout a flight's coverage. A parallel study has demonstrated the benefits in the application of this methodology to fuse ADS-B data into the Threaded Track [2].

3.2 Preprocessing and Filtering

Several processing and filtering steps are required to transform each of the raw surveillance data sets in a common format suitable for subsequent processing. These steps are discussed in detail below.

3.2.1 Segmentation

We refer to *segmentation* as the process of transforming individual position reports from a single surveillance source into a time ordered sequence which yields a segment trajectory (or segment for short) within the source facility bounds. Furthermore, this process constructs a unique identifier (segment id) for each segment.

The NOP and ASDE-X data sources are in a raw csv text format with one row per radar return. There is no explicit column to define which radar returns belong to a given flight. The segmentation process attempts to avoid merging two flights whenever possible, even at the expense of splitting a single flight.

This process uses different criteria for assigning points to a segment depending on the data source. The segmentation process begins by grouping the by sensor and date and then sorts the points within a group in time ascending order. After the grouping and sorting, the segmentation criteria is applied to each point in turn, and points in the same segment get assigned the same segment id. The criteria for each source are provided below:

- ARTS and STARS: two points belong to the same segment if they share the same track number[2] and are within the required[3] spatial and temporal bounds of each another.
- ARTCC: two points belong to the same segment if at least two of the callsign, computer id, or beacon code fields do not change and are within the required spatial and temporal bounds of each another.
- ASDE-X: two points belong to the same segment if they are within required temporal bounds and have the same mode S code and track number.
- ADS-B: two points belong to the same segment if they are within required temporal bounds and have the same mode S code

[2] The track number is the identifier used by each surveillance source's tracking system, which is typically a recycled integer value.

[3] The threshold values are not discussed in this paper, but rather the inputs and characteristics of each algorithm are provided. Further testing and validation is required prior to defining the specific values.

3.2.2 Coasting and Outlier Detection

Following the segmentation step (see Figure 1), the segments pass through *coasting and outlier detection*, consisting of a series of data filters. This section defines the only place within the process where data is removed, as shown in the rejected data block from Figure 1.

Since each surveillance source has distinct data characteristics, we have developed a processing workflow of operations for each source, shown in Figure 2. Each of the yellow blocks describes a preprocessing operation, which have been grouped into 5 distinct categories. In the following subsections we provide details on each processing step but we briefly describe the nuances of each surveillance source.

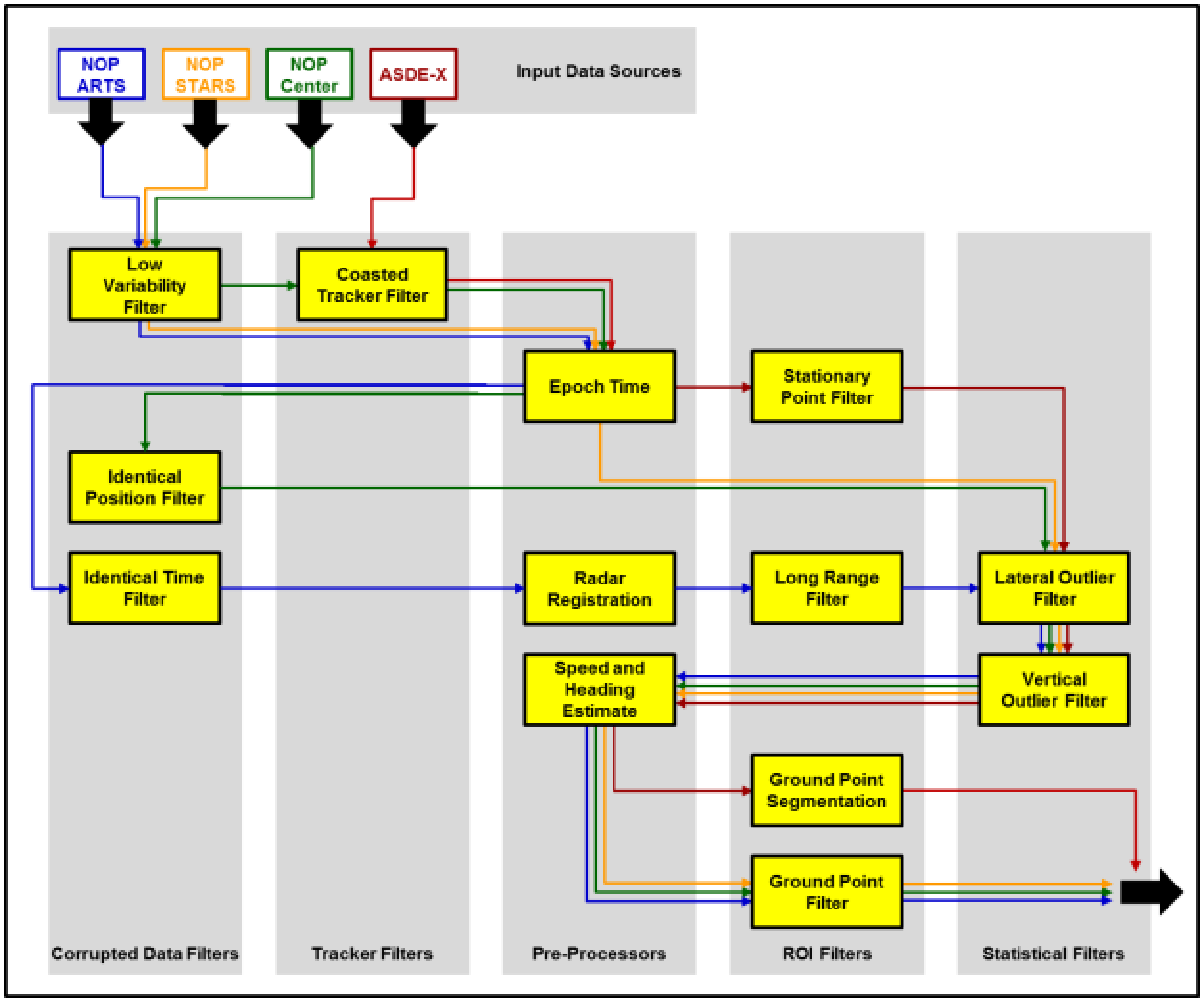

Figure 2. Coasting and outlier detection workflow.

3.2.2.1 Low Variability Filter

The purpose of the low variability filter is to remove non-aircraft trajectory segments that appear as a set of stationary points. These generally appear as very long (24 hour or longer) segments that bounce around between a distinct set of positions. The filter counts the number of distinct position and the total number of position reports in the segment. If the number of distinct positions and the number of positions reports are above specified thresholds, the entire segment is removed.

3.2.2.2 Coasted Tracker Filter: ARTCC

The ARTCC surveillance data passes through a tracker before captured by the NOP process. Under certain circumstances, the tracker can provide position updates that are synthetic and only based on the last known position, heading and speed of the aircraft. These trajectories often appear saw toothed (containing points which are collinear). In the NOP ARTCC data these points are identified by the CMS 153a field. Any point identified as coasted or any point within a specified number of updates of this point are removed.

3.2.2.3 Coasted Track Filter: ASDE-X

The ASDE-X system provides surveillance updates from a multi-sensor fusion process at a one Hertz rate, however, the update rates from the underlying sensors can be significantly greater. When this occurs, the system reports interpolated positions. Since these points are contained by other surveillance sources we remove them in the ASDE-X segments.

3.2.2.4 Epoch Time

We compute the epoch time (elapsed milliseconds since January 1, 1970) associated with each position report and the elapsed time from the first position report. This provides a simple linear scale against which to measure events and provides a more consistent standard across surveillance sources.

3.2.2.5 Stationary Point Filter

There is a large quantity of ASDE-X data on the ground for aircraft which are stationary (at the gate, taxiing, etc.). Since these points do not contribute any additional information to the trajectory, but take up a considerable amount of space, these points are removed, creating gaps on the ground where the end points of the gap are at the same location and zero speed. These stationary points are identified based on ground speed estimates of the trajectory.

3.2.2.6 Identical Position Filter

Similarly to the *Identical Time Filter*, we remove all points with identical position reports. These points are removed since they do not provide any additional information about the trajectory (typically the result of coasting or corruption for ARTCC data).

3.2.2.7 Identical Time Filter

We require each position report to be associated with a unique time. To enforce this condition, we order each segment by time and compute the time difference between consecutive points. We remove all points within a specified tolerance of its closest neighbors which removes the ambiguity of the aircraft being at two distinct locations at the same time.

3.2.2.8 Radar Registration

The NOP ARTS data reports the position in both Cartesian (x,y) and geodetic (latitude,longitude) coordinates. The origin of the Cartesian coordinates system is the location of the radar. We project the Cartesian coordinates into geodetic coordinates assuming a spherical earth model and using the aircrafts altitude and position of the radar sensor. We linearly interpolate position reports that have been flagged having no, zero or outlier altitudes using the nearest valid altitude reports. In Section 3.2.3, we described how the radar sensor's location is derived from the surveillance data.

3.2.2.9 Long Range Filter

NOP ARTS data sometimes contains trajectory points which are outside the physical range of the radar (approximately 60 NM). These points are assumed to be invalid and are removed.

3.2.2.10 Lateral Outlier Filter

The positions within a segment can contain large lateral biases. This filter detects lateral positions in the trajectory that are outlier and removes them. For each position update we consider a windowed portion of the trajectory and fit it to a simple trajectory model. If the deviation of the point from the model is large enough, the point is deemed an outlier and removed. To determine the outlier positions, an iterative weighted least squares method is applied where weights are determined by a window tapering function and the residuals at each iteration. Finally, the point is considered an outlier if it deviates from the model significantly more than the other points in the window.

3.2.2.11 Vertical Outlier Filter

Altitudes from radar surveillance sources are derived from the mode C transponder. It is possible for a lateral position to be reported with no corresponding altitude. This filter begins by marking all position reports that have either no or zero altitude. For altitudes associated with all other position reports we use a simple altitude model and mark all altitudes that deviate significantly from the model. We note that the underlying model is identical to that used in the lateral outlier detection with different input parameters selected.

3.2.2.12 Speed and Heading Estimate

This filter computes an estimated speed and heading associated with each position report. These values are used by subsequent filtering and fusion processes and are derived solely from the positions. The speed reported by the surveillance system can suffer from large errors and the heading is reported with respect to magnetic north and not true north. Thus, comparing across facilities requires specification of magnetic declinations at every facility.

3.2.2.13 Ground Point Filter: ARTCC

Under certain circumstances ARTCC surveillance data will contain position reports while the aircraft is on the ground. These position reports often do not correspond to the aircrafts true location and can persist for long periods of time at the end of NOP ARTCC segments. This filter identifies the position report where the coasting begins and removes all points from the first coasted point to the end of the segment. Working backward temporally from the end of the segment, we identify the first position report where the estimated speed is greater than a specified tolerance. This point to the end of the segment represents the candidate portion of the segment which can be removed. Within this candidate region we work forward in time and find the first coasted point reported by the CMS 153a field and remove this and all later points.

3.2.2.14 Ground Point Filter: ARTS

In some cases it is possible for NOP ARTS surveillance data to report positions while the aircraft is on the ground or is very close to the ground. Often these reports can suffer from significant biases but may not be removed by prior filtering. Identifying position reports on the ground is extremely important since the segment could have a very high number of segments from other flights in close spatial proximity. This can cause problems when forming flights from collections of segment. Since the radar sensor location is known, we simply remove all points within a specified cylinder centered at the radar sensor location.

3.2.2.15 Ground Point Segmentation

The ASDE-X data contains high quality segments or portions of segments where the aircraft is on the ground. These segments should become part of the Threaded Track, however, they must be isolated from the airborne trajectory segments due to the large number of segments within close proximity of one another on the ground. This filter splits an ASDE-X segment into airborne and ground segments.

3.2.3 Radar Registration Errors

NOP ARTS facilities provide a “single source” data feed, giving a trajectory measurement from each of the radars within a facility. For these sensors, it is possible to post-process the measurements to provide corrections to the reported positions. The radar registration in this context is used to identify the relevant parameters used to transform the raw range/azimuth measurements into latitude/longitude positions. These features include:

- Radar position (latitude, longitude, elevation)
- Radar declination (angle between true north and magnetic north)
- Range biases
- Temperature lapse rate

The radar positions are static and only need to be updated when new sensors are added. The declinations are approximately the magnetic declination of the radar, but will vary depending upon the specific calibration of the sensor and they vary over time due to changes in the Earth’s magnetic field. Similarly biases in range and temperature lapse rate, which will vary with the local atmospheric conditions.

The declinations and bias corrections are obtained by correlating sensor measurements between different facilities. The relative position error between different radar sensors is used to solve a set of least squares equations for these parameters, described in Appendix A. Figure 3 shows the derived radar declination over time for the Southern California TRACON (SCT) Miramar Marine Corps Air Station (NKX) sensor (covering San Diego International Airport). The error varies greatly over time, where around August the declination shifts by about 0.7 degrees. Variations on the order of a week of about 0.1 degrees are also apparent during the first 7 months.

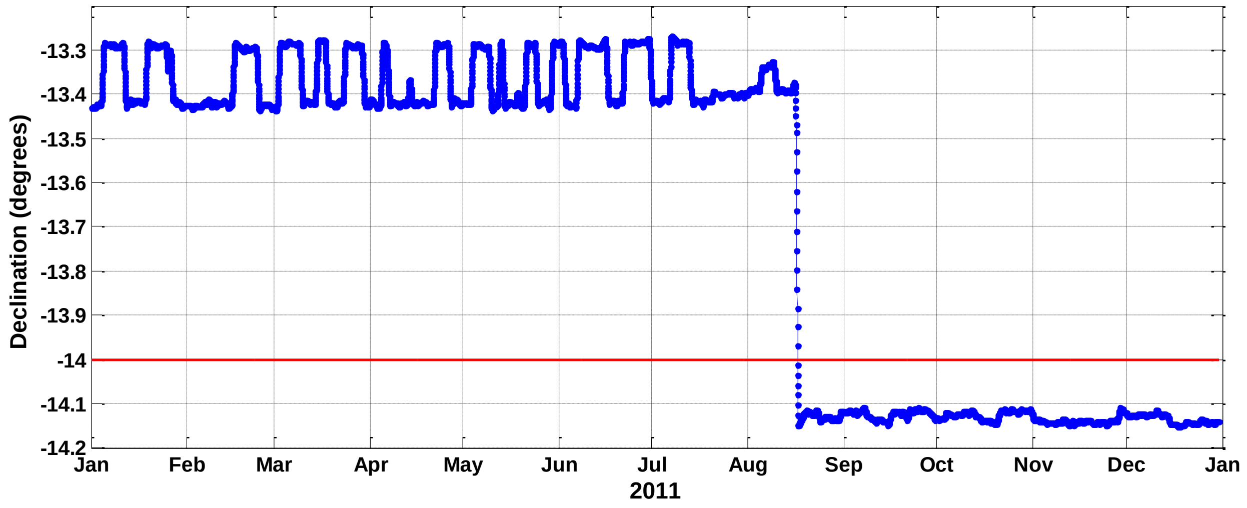

Figure 3. Derived radar declination of NKX sensor from SCT TRACON during 2011.

The NOP data uses a static value of 14 degrees to convert the Cartesian coordinates to geodetic coordinates throughout the time period in Figure 3. Thus, the geodetic coordinates provided in the NOP data will suffer from errors which become larger near the TRACON boundary. Several sensors were identified where the NOP value disagreed by several degrees. Since the geospatial association between segments relies on accurate position measurements, this post-acquisition calibration becomes a critical part of the Threaded Track process.

3.3 Geospatial Joining

We refer to geospatial joining as the process which associates trajectory segments simply by spatial and temporal proximity. Two segments are associated with one another if they overlap in time and the point wise distance between position reports is small. A straightforward implementation would require comparing every segment pair in a point wise fashion yielding a computationally intractable problem. However, this problem is inherently sparse since most airborne segments never come within close proximity to one another and, thus, need not be compared. However, at a given airport, ASDE-X ground segments come in close proximity with all other ground segments at that airport. Thus, ground sub-segments are not considered in the geospatial fusion process and are only associated with their parent segment.

In this section, we describe a computationally tractable method for determining the candidate set of segments which are close enough to require a point by point comparison. After segment comparisons we describe a method for linking segments and forming the set of segments which will be transformed into the Threaded Track.

3.3.1 Geohashing the Track Segments

This section describes how we ensure that only relevant pairs of trajectory segments are compared. To begin, we coarsen the time series of representation of a segment to a sequence of bins representing a quantized region of space (lateral and vertical). In the lateral direction, we use a method referred to as geohashing [3] and in the vertical and time direction simply use discrete bins. To prevent problems occurring near bin boundaries, we pad all segments bins with their neighbors. For a given segment, the unique set of bins provides a reduced representation and allows fast association between pairs of segments which then require a point to point comparison. We note that if the segment does not contain a valid altitude, we associate it the only a lateral bin and allow it to belong to any vertical bin. Figure 4 shows an example of geohashing applied to two trajectory segments. These segments would be linked based on their overlapping bins.

A fundamental tradeoff with this approach is the four-dimensional bin size versus the total number of segment comparisons. On one extreme, if the bin size excessively large then every segment will be contained within this bin and thus all segment pairs will be compared. However, if we make the bin size too small then computational cost of determining if two segments share a bin becomes large.

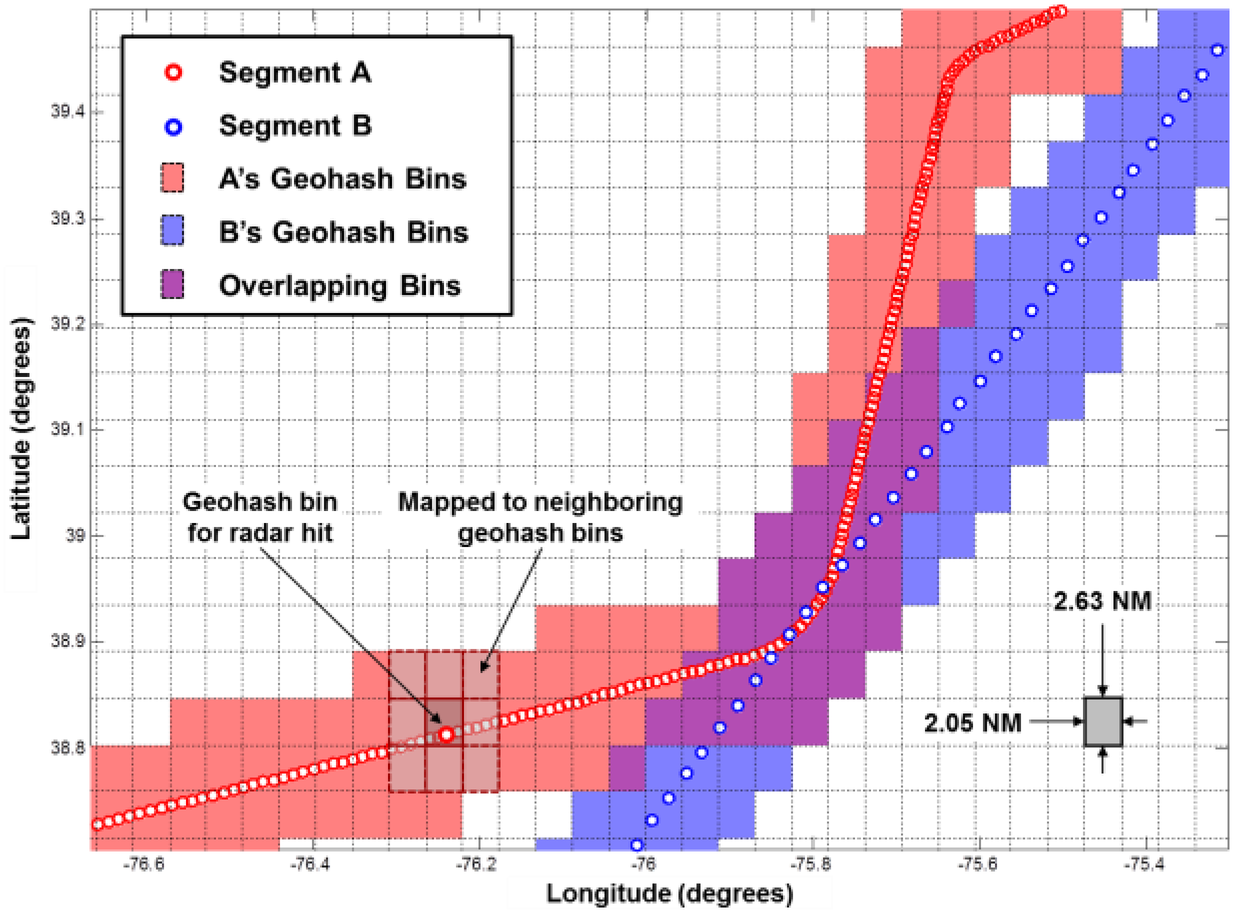

Figure 4. Example illustrating geospatial joining.

3.3.2 Correlating pairs

Once two segments have been shown to share a common bin, a finer level of comparison is required. In this step, we determine if two segments overlap in time. If they overlap, a point by point comparison is made and we construct a similarity score based on their point by point lateral and vertical proximity. The similarity score is a statistical metric based on the lateral and vertical differences between two segments. To associate points between segments we interpolate each segment to the same time and compare position and altitude. To prevent interpolation artifacts skewing the results of the similarity score, we only interpolate points within a required time duration of a valid position in the segment. In the case where a segment does not contain an interpolated value, no comparison between the segments is made. We note that since altitude is not a required field for each position report, the number of lateral and vertical comparison may be different.

The similarity score is source dependent with tighter tolerances for more accurate data sources and looser tolerances for poorer quality data sources (e.g. a 0.2 NM difference between two ASDE-X segments will have a lower similarity score than a 0.2 NM difference between two

ARTCC segments since the fidelity of the ASDE-X data is much greater than the ARTCC data). In Figure 5, we provide an example comparing segments in a TRACON and ARTCC facility. The top of Figure 5 shows the average lateral distance (log scale) between segment pairs versus the overlap time (colored by density of comparisons) and the bottom shows the distribution of segment pairs whose metadata agreed and disagreed as a function of average lateral distance. In this example there is a clear partition between segments with a small distance which belong to the same flight (toward the left) and those with a large distance that do not (toward the right).

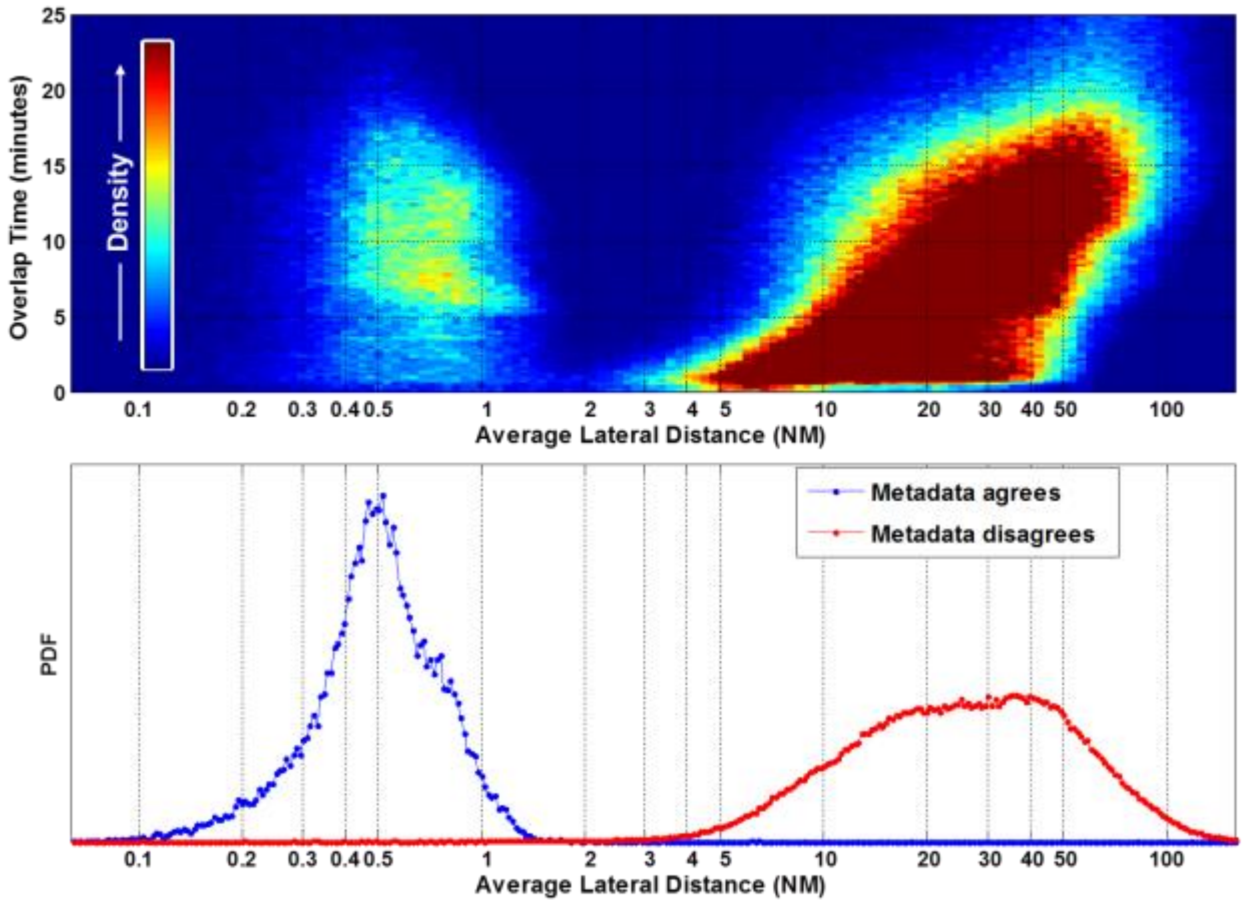

Figure 5. Distribution of lateral correlation scores versus time (top) and marginal lateral distributions by metadata quality (bottom) between ARTS and ARTCC facilities.

3.3.3 Community Detection

Given the similarity scores between segments, the next step is to determine the set of segments which form the basis for a Threaded Track. To perform this task we exploit network analysis techniques where the network nodes are segments and the edges are the similarity scores between segment pairs. We pose the problem of determining the flights from this network as a community detection problem, i.e. to find collections of nodes (a community) which are highly linked to one another and weakly linked to other communities [4].

Phase 1: Sub-Networks. In order to reduce the complexity of the problem, the network of all segments and scores is subdivided into a smaller set of sub-networks. A threshold score is used to define the upper bounds by which two nodes might be directly related. After removing all edges which do not meet this threshold, the nodes are grouped into sub-networks, where a sub-network represents the collection of nodes which have a path between every pair of nodes and share no path which another sub-network. This sub-network may contain multiple flights which come within proximity of one another, but there will not be multiple sub-networks that contain nodes from a single flight. Therefore each of these sub-networks can then be considered individually.

Phase 2: Network Splitting. A sub-network may contain more than one flight. Joined flights are detected when sub-networks contain nodes that overlap in time, but do not have an edge between them (e.g. they overlap in time but not space), referred to as no-link pairs. Edges in each sub-network are then iteratively removed until there no longer exists a path between each of these no-link pairs, while maximizing the network connectivity.

Figure 6 shows an example of a sub-network which is split into 5 distinct groups. The graph is organized to maximize separation between no link pairs. The distance between nodes therefore represents weaker connections. In this instance, 5 edges are removed in the splitting algorithm. Note that one of these groups contains only one node (segment), whereas the others to be collections of multiple nodes.

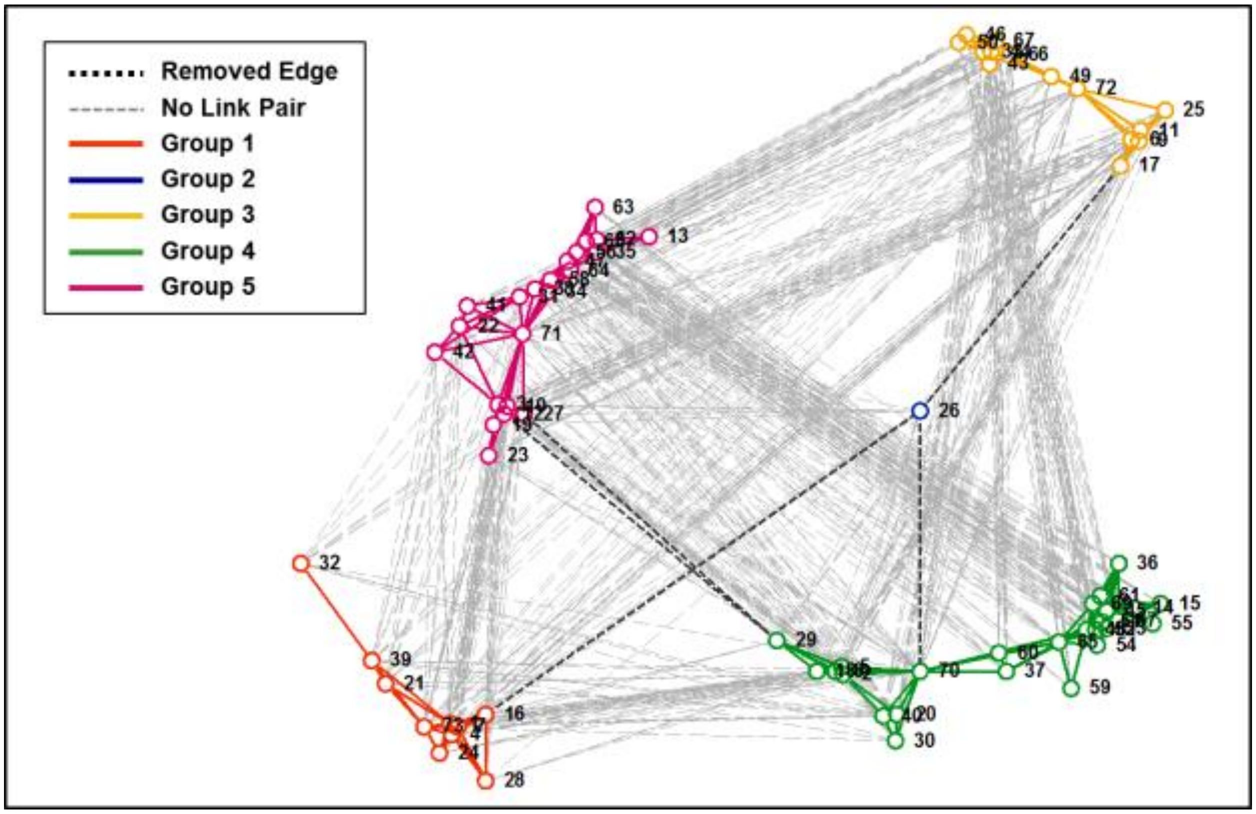

Figure 6. Splitting a sub-network into 5 distinct groups.

Figure 7 shows the lateral and vertical trajectories for the same groups from the sub-network. This sub-network contains 4 closely spaced arrivals to Hartsfield-Jackson Atlanta International Airport (ATL). Group 2 contains the singleton segment, which is the result of a tracker error in which a single segment tracking the flight in Group 4 switches to Group 3. Currently, the algorithm only removes edges, but it would be desirable to either re-segment this segment into its two components, or delete the segment from the final output. In addition, it is noted that this process does not currently rely on any flight metadata to split out the segments, but could easily be used when available to increase the fidelity of the splitting logic.

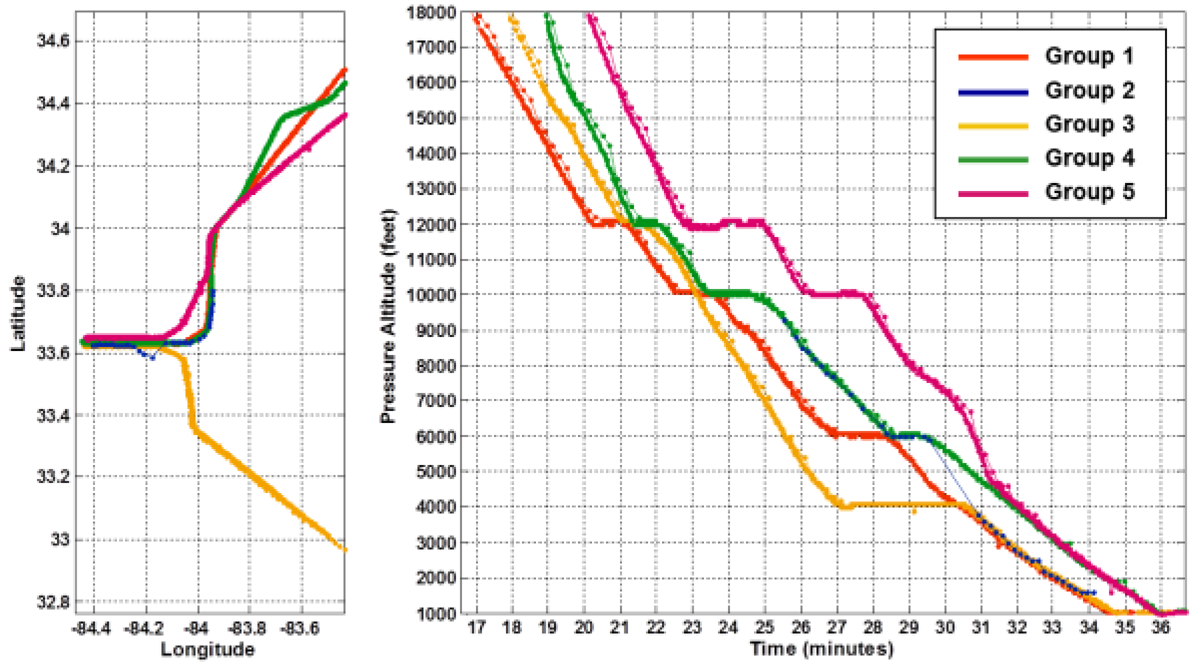

Figure 7. Lateral (left) and vertical (right) plots of the trajectory segments from each of the groups in the sub-network splitting.

Phase 3: Ground Segment Joining. These sub networks include only air portion of a flight but before starting the next process in the Threaded Track workflow the ground segments are merged. We do not enforce a one to one mapping between ground segments and air segments, thus two distinct flights can contain the same ground segment For example, consider a segment corresponding to an aircraft which taxis from the arrival runway to the gate, waits at the gate and then taxis to a departure runway. In the case, this ground segment would belong to the arrival flight and departure flight.

3.4 Track Synthesis

Track synthesis is the process by which all of the segments associated with a particular flight are fused into a single consistent synthetic trajectory. This process applies a series of filtering operations to remove noise and bias errors from the raw data without removing desired physical aircraft maneuvers. These filtering stages are defined as follows:

1. Cross Track Filtering
2. Along Track Distance
3. Along Track Filtering
4. Along Track Adjustment
5. Vertical Track Filtering

3.4.1 Filter Stages

Each of the filter stages produces one or more parameters of the derived synthetic trajectory, shown in Table 1 (each component on the left and which stage it is calculated in on the right). The track point times of the synthetic trajectory are identical to that of the raw track points - all error is modeled in the position components. In the first stage, *cross track filtering*, the lateral path is determined, providing filtered latitude and longitude measurements and the heading and curvature estimates along the path. In the second stage, the distance along the lateral path is integrated over each of the track points, calculating the along track distance. In the third stage, *along track filtering*, the along track distance is filtered as a function of time. This step also derives the ground speed and acceleration. In the fourth stage, the residuals of the along track distance filtering is used to correct the position, heading, and curvature along the path. Finally, in the fifth stage, *vertical track filtering*, the vertical profile is smoothed, providing a filtered pressure altitude as a function of time, as well as its derivative with respect to time, the climb rate (the climb gradient is also then provided as the ratio of the climb rate to the ground speed). The algorithms used in each of these filtering stages are described below.

Table 1. Derivation of synthetic trajectory parameters.

	→ Filter Stage →					
Parameter	**Input**	**1**	**2**	**3**	**4**	**5**
time	Raw					
latitude	Raw	Filtered			Corrected	
longitude	Raw	Filtered			Corrected	
pressure altitude	Raw					Filtered
along track distance			Derived	Filtered		
track heading		Derived			Corrected	
track curvature		Derived			Corrected	
ground speed				Derived		
ground acceleration				Derived		
climb gradient						Derived
climb rate						Derived

3.4.2 Windowed Least Squares Filtering

The *cross track filtering* and *along track filtering* stages use an aircraft dynamic model and apply a weighted least square method to the raw track positions to determine the lateral path and along track distance. This process is shown in Figure 8. Starting at the top block in Figure 8, the filter iterates over each of the trajectory point times (τ). This time is used to define the window function which is used to localize the solution at τ (described in section 3.4.2.1). Next, the filter iterates over each of the segments which has computational complexity of $O(N_p{\cdot}N_s)$ where N_p is the number of points and N_s is the number of segments. Within each of these interations, the segments points are first used to calculate the segments weights at τ. These weights are the product of the window function and the sensor weights. The sensor weights are source specific *a priori* estimates of accuracy, described in section 3.4.2.2.

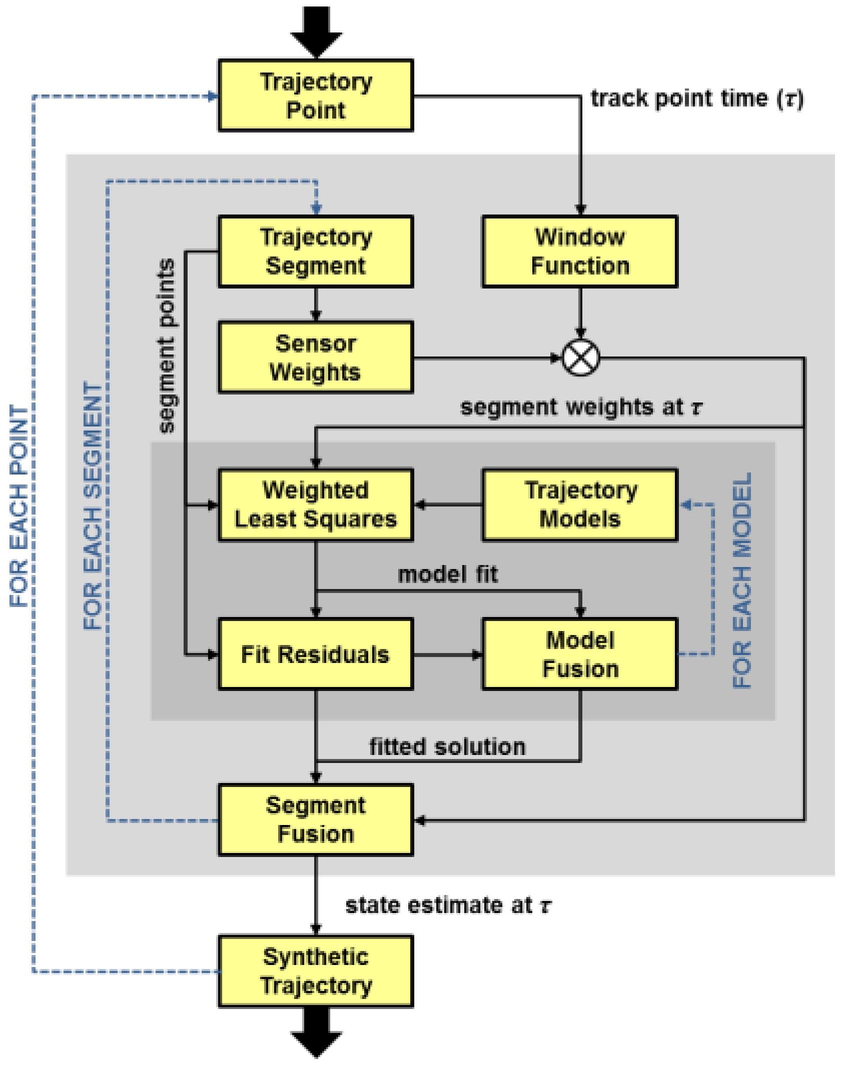

Figure 8. Windowed least squares filtering process.

Next, the segment weights and points feed into the inner block, which defines the least squares filtering operations. The *cross track filter* uses a constant heading (straight) model and constant radius (turn) model to the aircraft positions, providing a first order and second order approximation of the aircraft state (see Appendix B). Figure 9 shows an example of these two least squares models being applied to the lateral positions at a single point in the cross track filtering. Similarly, the along track filter applies a constant speed (first order) and constant acceleration (second order) model to the along track distances. Each model is then used to solve a weighted least squares solution. The final solution (for each segment at τ) provides a mixed state, weighted by the residuals of each model fit, described in Appendix B. The dark black line in Figure 9 shows the final least squares solution applied to each of the sample point times for this segment.

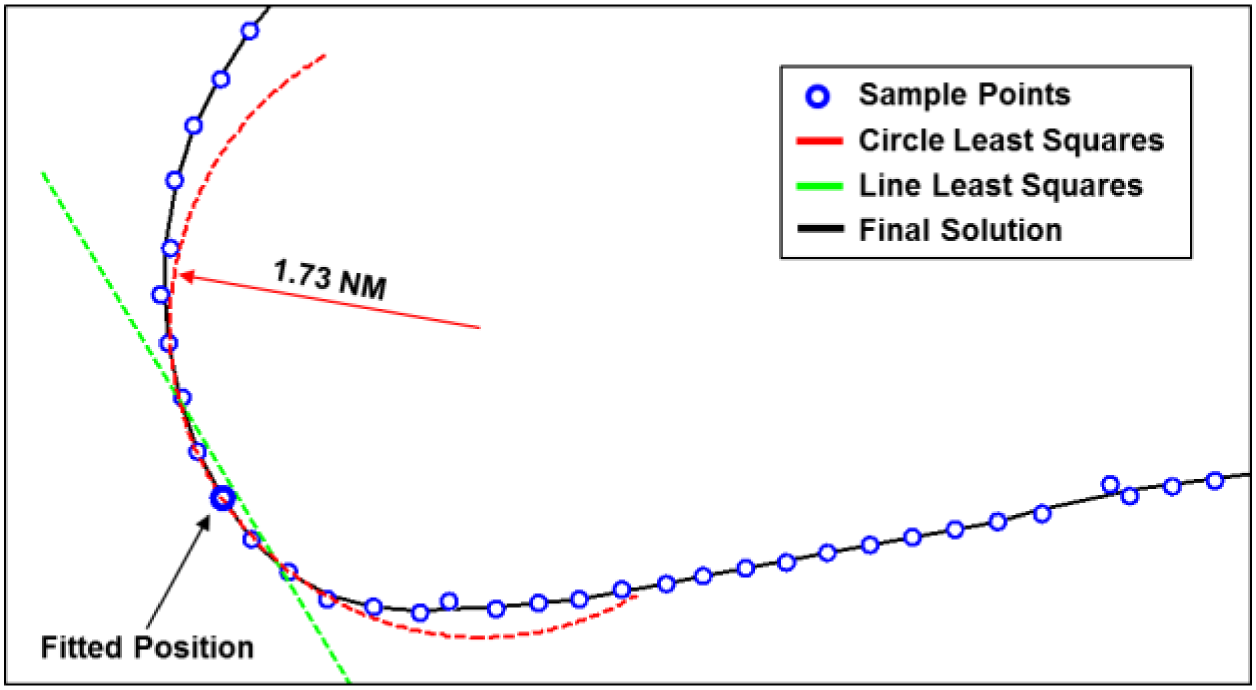

Figure 9. Least Squares models for cross track filtering.

After calculating each sensor's fitted state at τ, the solutions are fused based on a weighted average, where the weights are derived from the windowed sensor weights, as well as the fit residuals. Position and distance metrics apply a simple weighted average, whereas derived parameters (e.g. heading, speed, etc.) apply the product of these same weights with a distance weighted mean. This distance weighted mean is a measure of central tendency that applies higher weights to more central values and lower values to the tails of the distribution [5].

3.4.2.1 Windowing Function

For the least squares filter input, Gaussian window functions are used with standard deviations of $1.67/\upsilon$ for the cross-track filter and $2.83/\upsilon$ for the along-track filter (where υ is the sensor's sampling rate). These functions were determined empirically to balance the filter bandwidth of the current data sources accuracy and expected aircraft maneuvers. Wider windows will produce a smoother solution, but can smooth out smaller maneuvers. Narrow windows are much more sensitive to small maneuvers, but are also more sensitive to random noise. Figure 10 shows this window function overlaid onto representative track points for the bandwidths of the cross track filter and along track filter. The effective width of each is the width of the rectangular window with equivalent area. This gives an effective width of four track points in the cross-track direction, and seven track points in the along-track direction.

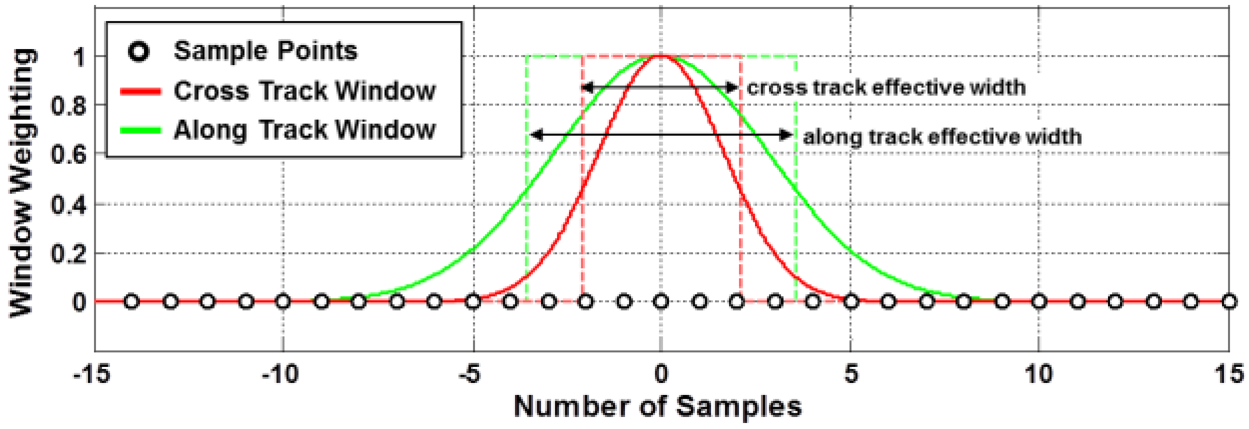

Figure 10. Windowing functions for cross track and along track filters.

3.4.2.2 Sensor Models

ASDE-X Model: The lateral weights for ASDE-X are based on the three distinct sensor regions for the surface movement radar (SMR) the multilateration array, and the airport surveillance radar (ASR). Within each of these regions, a constant accuracy is provided based on sensor specifications and approximations of range, dilution of precision, etc. The vertical weights are based on a standard mode C error.

ARTS Model: The lateral weights for ARTS are based on two independent error distributions along the radar range and along the radar azimuth. The azimuth radar typically dominates the far field error, whereas near field errors are dominated by the range. These marginal error probabilities are projected into the cross track and along track directions using the difference between the radar azimuth and the track heading. The vertical weights are based on a standard mode C error.

STARS Model: The lateral weights for STARS are divided into two distinct regions, a near field with higher update rate Tracon measurements, and a far field with measurements from en route radars. Unlike ARTS measurements, the range from the radar is unknown for any point, so instead constants based on expected ranges are used to provide accuracies in these two regions. The vertical weights are based on a standard mode C error.

ARTCC Model: The lateral and vertical weights for ARTCC measurements are based on constant assumptions and standard mode C errors.

ADS-B Model: The ADS-B model is still being developed, however, there are parameters within ADS-B data to describe the integrity and accuracy of each of the position reports. Special treatment is required for the vertical measurements below FL180, which are typically standard pressure reference altitude, whereas radar uses local pressure correction.

3.4.3 Adaptive Piecewise Least Squares Filtering

The fifth stage, vertical track filtering, does not apply the same least squares models as the other filtering stages. Alternatively, a filtering process subdivides the vertical trajectories into discrete regions that conform to a specific trajectory model. This algorithm is based on a hybrid variant of the Douglas-Peucker algorithm [6]. This process is shown in Figure 11.

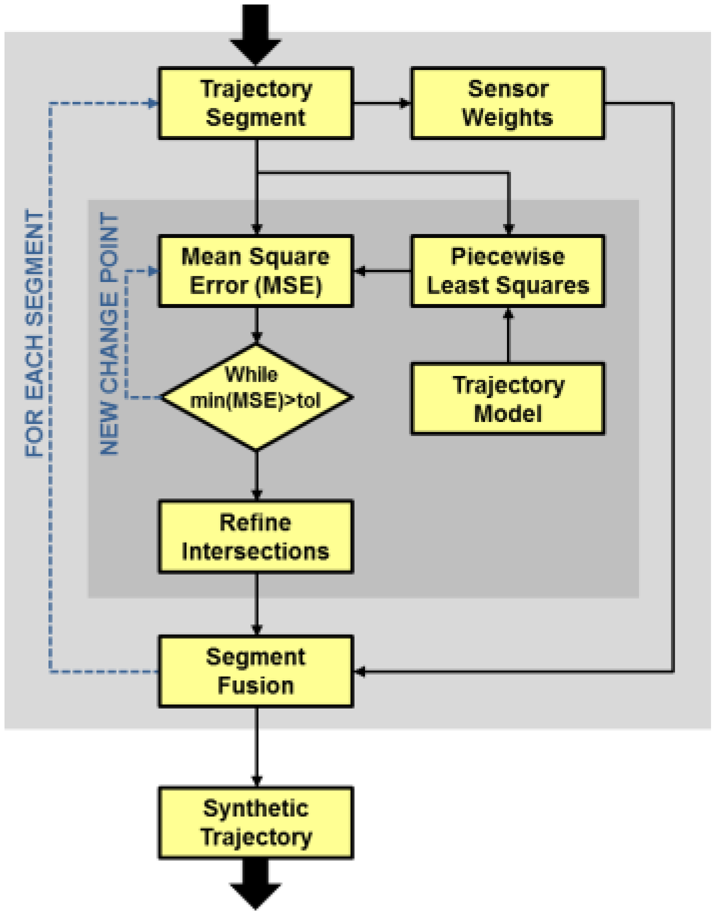

Figure 11. Change point estimation filtering process.

The process begins by estimating the piecewise least squares function using the trajectory model. P*iecewise* signifies that for a given time (τ) within a segment, the least squares model is applied to all points prior to τ, and then to all points occurring after τ, independently (such that there is a discontinuity in the first derivative at τ). Effectively, this tests how well a given point serves as a break in the trajectory model. The residuals from both fits are used to construct a Mean Square Error (*MSE*), which is applied for each time (τ) within the segment.

The time where *MSE* is minimized represents the optimal place to subdivide the segment. If the maximum(*MSE*) is greater than a specified tolerance, ε, then the segment is subdivided at the location where *MSE* is minimized, creating a *change point*. This implies that as long as the residuals from the trajectory model fit are relatively high, the algorithm will continue to iteratively subdivide until the fit residuals are less than the specified tolerance.

For the *vertical track filtering* stage, the trajectory model is a linear fit of pressure altitude as a function of time (i.e. constant climb rate). Figure 12 shows an example where the altitude profile has reached convergence, and the red circles show the change points that were identified.

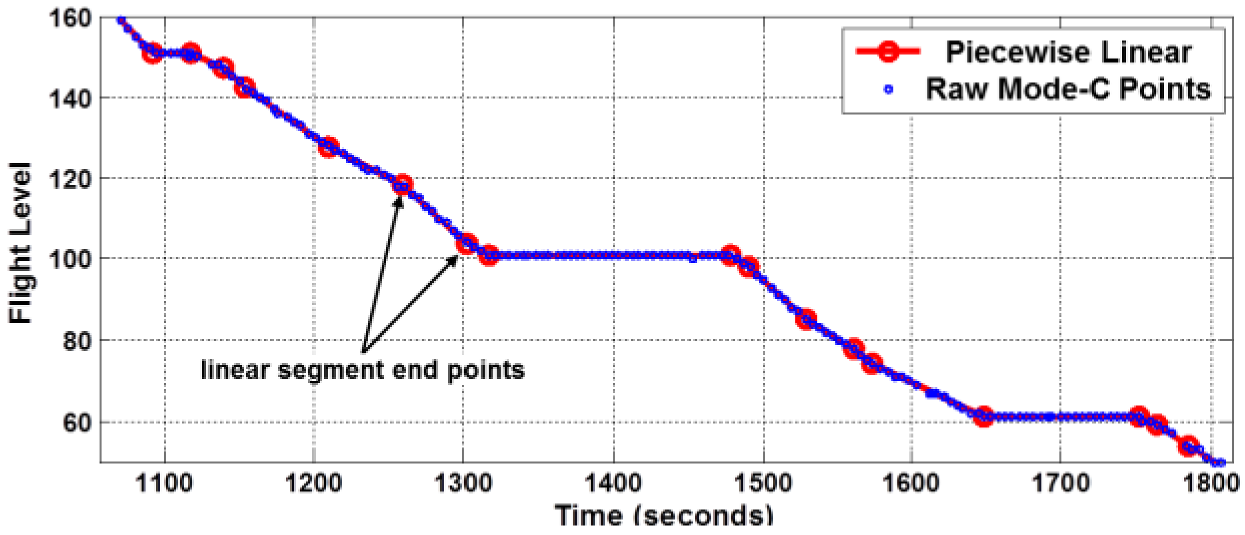

Figure 12. Sample vertical trajectory and its least squares segmented profile.

Since the change points can only occur at the discrete sampling of the segment's points, these points are refined using the intersection between respective region's trajectory model fits, providing exact intersection points, during the *refine intersections* step.

This process is iterated over the segments, creating a piecewise linear trajectory for each segment. These fitted solutions are averaged together using sensor weights and fit residuals in a manner similar to the windowed least squares filtering. The sensor weights are calculated in section 3.4.2.2.

3.4.4 Resampling

Since the noise filtering process evaluates each input track point, the output synthetic track can have a highly non-uniform sampling rate, depending on the specific characteristics of each sensor. In order to provide a more consistent sampling rate and reduce unnecessary storage of redundant information, the synthetic trajectory is resampled to a new time interval.

The output sampling rate was chosen to be that of the highest update rate sensor at any given time. This provides a sampling period ranging from 1 second to 12 seconds. In addition, to minimize interpolation errors, when only one active sensor is contributing to the synthetic trajectory, the final point times are identical to the source data times (e.g. no resampling).

3.5 Post Fusion with ETMS

Compared to other surveillance sources, Enhanced Traffic Management System (ETMS) data requires special treatment when fused into the Threaded Track. ETMS trajectory points (given at one minute updates) are directly joined (without smoothing) into the Threaded Track where there are gaps in the coverage of other data sources, or where ETMS is the only coverage source (outside the NAS). Unlike the other trajectory sources, ETMS provides end-to-end trajectories, and is capable of linking across large time gaps (such as Hawaii to NAS flights) using flight plan information. However, it is often the case for domestic flights that ETMS provides redundant coverage over NOP and ASDE-X regions, and no ETMS track points will be fused into the trajectory[4].

Because of the low sampling rate and large coverage of ETMS trajectory segments, fusion with ETMS occurs in a second phase of geospatial joining. ETMS positions are correlated with the fused trajectory output from phase 1 (see Figure 1). This process contains many of the same processes as phase 1, but is applied to different data sources. The ETMS data supplies the Threaded Track with flight plan information, arrival/departure airport information, and an ETMS identifier providing the ability to associate it with other data in the CAASD Repository System (CRS).

4 Performance

4.1 Computational Tools

The implementation of the methodology makes use of externally developed open-source and commercial tools and frameworks, as well as CAASD-developed software. The large volume of data being processed and the computational complexity of the fusing algorithms led us to the Hadoop ecosystem of tools for implementing these fusing processes. These tools provide a platform for reliable, scalable, and data driven computation.

4.1.1 Apache Hadoop

The Threaded Track process workflow is implemented as a series of Map/Reduce applications using the Apache Hadoop™ distributed processing framework. Previous studies have demonstrated the use of Hadoop for storing and processing surveillance data.

The CAASD Hadoop cluster has grown substantially in the past few years, and its updated description can be found in Table 2.

[4] While the trajectory points are not defined by ETMS, the link to the ETMS flight information is still provided.

Table 2. CAASD's Hadoop Cluster Characteristics.

Data Nodes	42
CPU Cores	544
RAM	1.5 TB
Disk Capacity	489 TB
Map Slots	320
Reduce Slots	152

The Map/Reduce jobs primarily fall into two categories: compute-intensive jobs applying some computation to every datum in a series, and I/O intensive jobs joining and sorting different types of records together. The compute-intensive jobs often perform the bulk of their computation in the Map phase, and many times do not include a reduce phase. The break-down of methodology steps into Map/Reduce jobs is described in section 4.2.

4.1.2 Apache Avro

Apache Avro is a data serialization system that provides a compact binary data format as well as human and machine readable schema information. It was designed for use with Hadoop, and has cross language support for integration with other tools. Avro data files support efficient access within Map/Reduce jobs, as the files contain schema information and the data within files are organized in a block-oriented format that supports compression.

Unlike many other serialization systems, Avro supports dynamic typing which facilitated the implementation of generic data processing steps within the Threaded Track workflow. The fast serialization and deserialization APIs, along with the flexible generic record features, provide a performant data exchange mechanism between processing steps.

4.1.3 Apache Oozie

Apache Oozie is a job scheduling and workflow system for Hadoop. It represents workflows as a directed-acyclic graph (DAG) of tasks, where each task can be a Map/Reduce job, control-flow step, file operation, or external process. The Oozie system can manage the execution of complex multi-step workflows on a Hadoop cluster, and can trigger workflows based on the existence of data dependencies.

Oozie is used to orchestrate the Threaded Track workflow and manages its execution on the CAASD Hadoop Cluster.

4.1.4 Matloop

Some of the software components used in the methodology are implemented using MathWorks' numerical analytic suite, MATLAB®. CAASD has designed a unique toolset, *Matloop*, to enable running these MATLAB components within the Hadoop environment.

Matloop allows deployment of compiled Matlab programs as Map/Reduce jobs within the cluster. To support the development efforts described in this report, Matloop was extended to support the use arbitrary AVRO objects for input and output..

4.2 Execution Plan

The Threaded Track process workflow is organized into 22 map-reduce jobs. These jobs represent the physical execution plan, described in Table 3, which shows the relationship of each map-reduce job to the phase and components in Figure 1 (not a one-to-one mapping). Components that are labeled *Shuffle Data* do not directly map to one of the components in Figure 1, but are required steps to reorganize the data for subsequent map-reduce jobs. Jobs 3-11 are identical to jobs 14-22, but operate on different sets of data (although there are some data specific features).

Table 3. Physical execution plan for map-reduce jobs.

Job #	Map-Reduce Job	Phase	Component
1	Filter NOP	1	Coasting and Outlier Detection
2	Filter ASDEX	1	Coasting and Outlier Detection
3	Geohashing	1	Candidate Grouping
4	Distinct Pairs	1	Candidate Grouping
5	Group Pairs 1	1	Candidate Grouping
6	Group Pairs 2	1	Shuffle Data
7	Score Pairs	1	Segment Correlation
8	Union Find Graph	1	Community Detection
9	Collect Groups 1	1	Shuffle Data
10	Collect Groups 2	1	Shuffle Data
11	Splitting & Synthesis	1	Community Detection
			Track Synthesis
12	Filter ETMS	2	Coasting and Outlier Detection
13	Filter Phase 1	2	Coasting and Outlier Detection
14	Geohashing	2	Candidate Grouping
15	Distinct Pairs	2	Candidate Grouping
16	Group Pairs 1	2	Candidate Grouping
17	Group Pairs 2	2	Shuffle Data
18	Score Pairs	2	Segment Correlation
19	Union Find Graph	2	Community Detection
20	Collect Groups 1	2	Shuffle Data
21	Collect Groups 2	2	Shuffle Data
22	Splitting & Synthesis	2	Community Detection
			Track Synthesis

Figure 13 shows the percentage of total CPU time and Figure 14 shows the input/output sizes for each of the 22 map-reduce jobs. The phase 1 *Splitting & Synthesis* job (#11) takes roughly one third of the total process time, which is computationally intensive, but not I/O intensive. In contrast, the phase 1 *Group Pairs 2* job (#6) takes 25 percent of the time, which is an I/O heavy job due to the large number of segment comparisons. Since most trajectory data has been fused in phase 1 and only ETMS is being fused in phase 2, we see that phase 1 takes 86% of the total processing time.

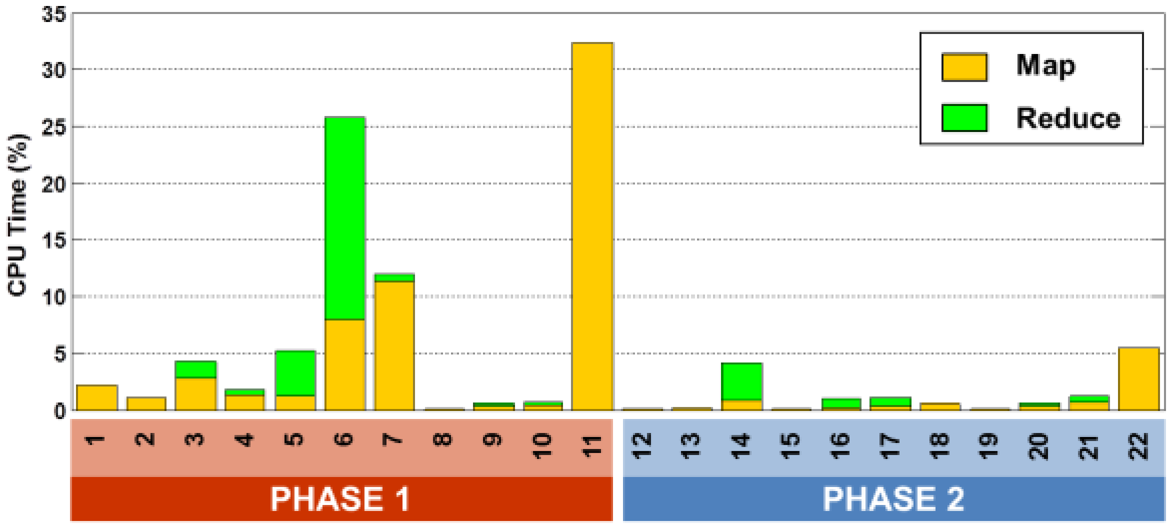

Figure 13. Percentage of CPU time for the map-reduce workflow.

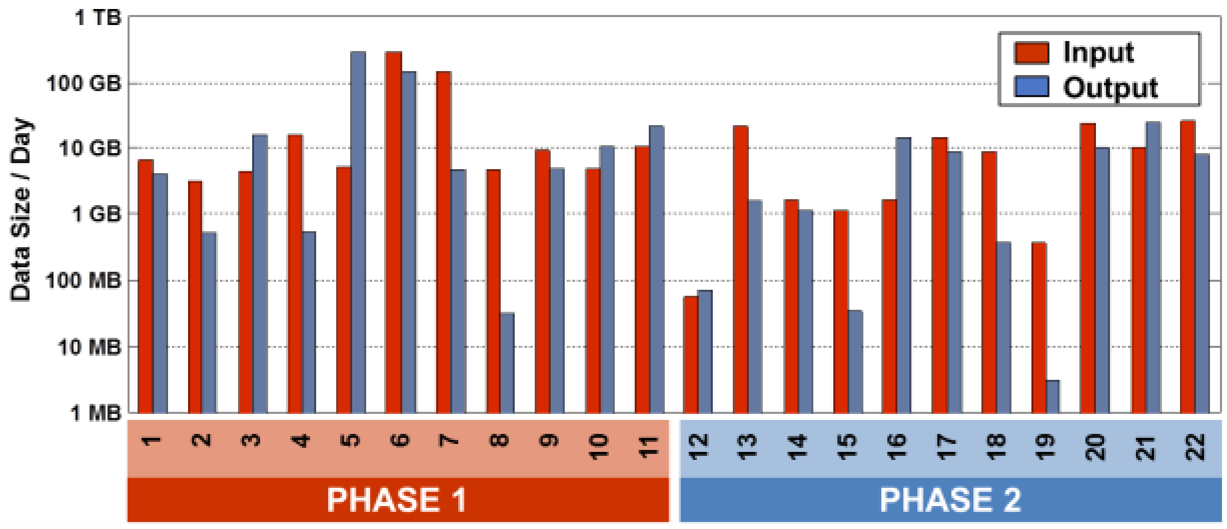

Figure 14. Input and output data sizes for the map-reduce workflow.

Figure 15 shows the optimal execution time for each map-reduce job given the number of map and reduce slots available on CAASD's Hadoop cluster, described in Table 2. The execution time is normalized for a day of NAS-wide surveillance data.

Accumulating the computational time from the processes in Figure 15, each day of data requires approximately 3 hours to process (optimally). In order to process the current 2 years of historical data will therefore take at least 3 months of dedicated processing time. This estimate also does not account for hadoop overhead time, added data at boundaries, and nonlinear accumulation of processing time (all of which will increase the actual processing time), which could easily increase this estimate by a factor of two or more.

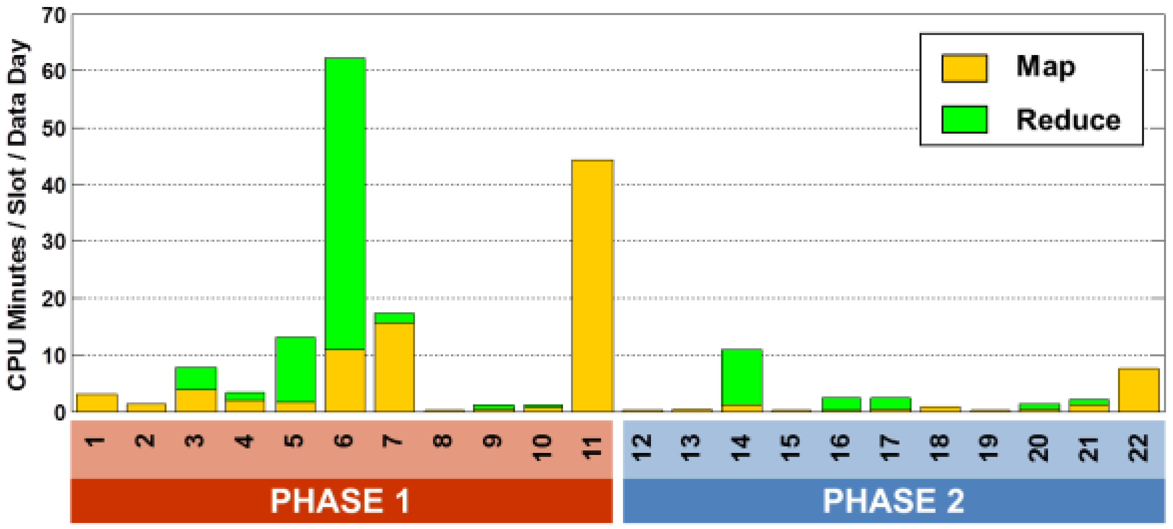

Figure 15. Optimal wall clock time for the map-reduce workflow in CAASD's Hadoop Cluster.

5 Data Output Characteristics

Since no metadata is required within the geospatial fusion process, the data output characteristics are quite different than previous versions of the Threaded Track. Figure 16 shows a chart of the number of groups[5] (left) and number of points (right) in the final output for five distinct types of data. The *fused ETMS* accounts for fused groups which merge a series of radar track data with an ETMS segment. *Associated* data accounts for groups which merge one or more radar segments with identifying information (i.e. callsign), but are not paired to an ETMS segment. *Unassociated* data accounts for groups which merge one or more radar segments with beacon codes, but do not contain any identifying information or ETMS segments, whereas *primary only* contains fused radar segments without any beacon codes. The *single ETMS* group is also included, which accounts for ETMS segments which were not matched to any radar data (e.g. London, Canada, Guam, and other regions of distinct coverage).

[5] The term *group* is used to signify the trajectory from the fused data set. The term flight is not used here to prevent confusion since these groups may consist of both merged flights, split flights, noise, etc.

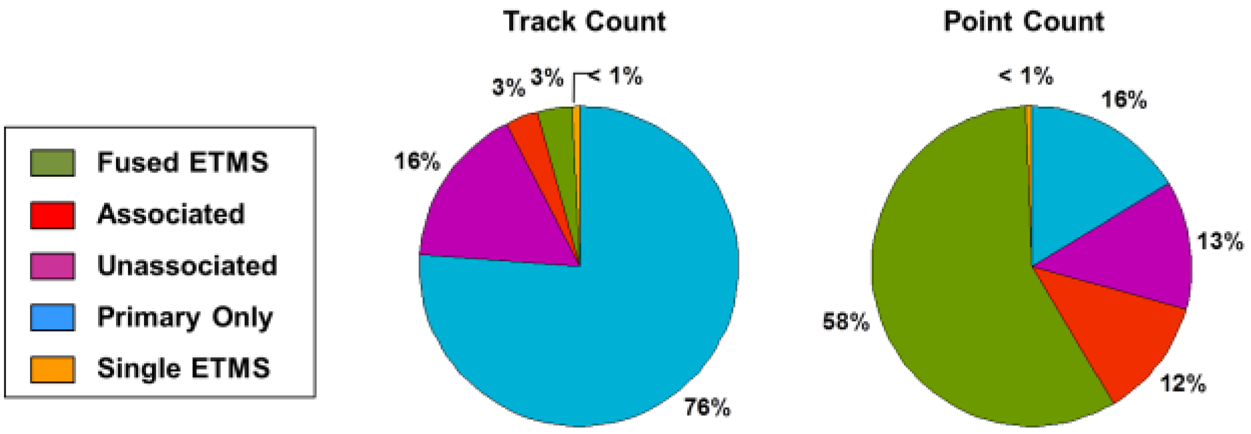

Figure 16. Categorized segment group count (left) and track point count (right)

Both the unassociated and primary only sets represent segment groups which were not present in previous versions of the Threaded Track. Note that these sets accounts for about 29% of the track points and 94% of the segment groups. Also, while the *fused ETMS* contains only 3% of the fused groups, it contains 58% of the track points, suggesting that these are longer duration, more complete flights. In contrast, the primary only contains 76% of the groups, but only 16% of the points.

Figure 17 shows a lateral (latitude, longitude) plot of the fused groups over Florida, colored by each of these five different output sets. The *fused ETMS* set contains the bulk of the commercial traffic between domestic and international city pairs along well structured routes. There is also a small degree of *associated* data scattered across these same regions.

Again, the *unassociated* and the *primary only* sets consist of data which was not in the previous version of the Threaded Track. The unassociated appears to be mostly VFR traffic centered around satellite airports and along the coastline. The primary only, however, is mostly composed of a sparse scatter near the radar sensors, indicative of a high degree of noise, as well as ASDE-X surface data. Since the methodology described in this paper fuses only in flight regions of data, vehicles and other airport surface movements may result in a high degree of split data, which would also account for why these show up as 75% of the fused groups.

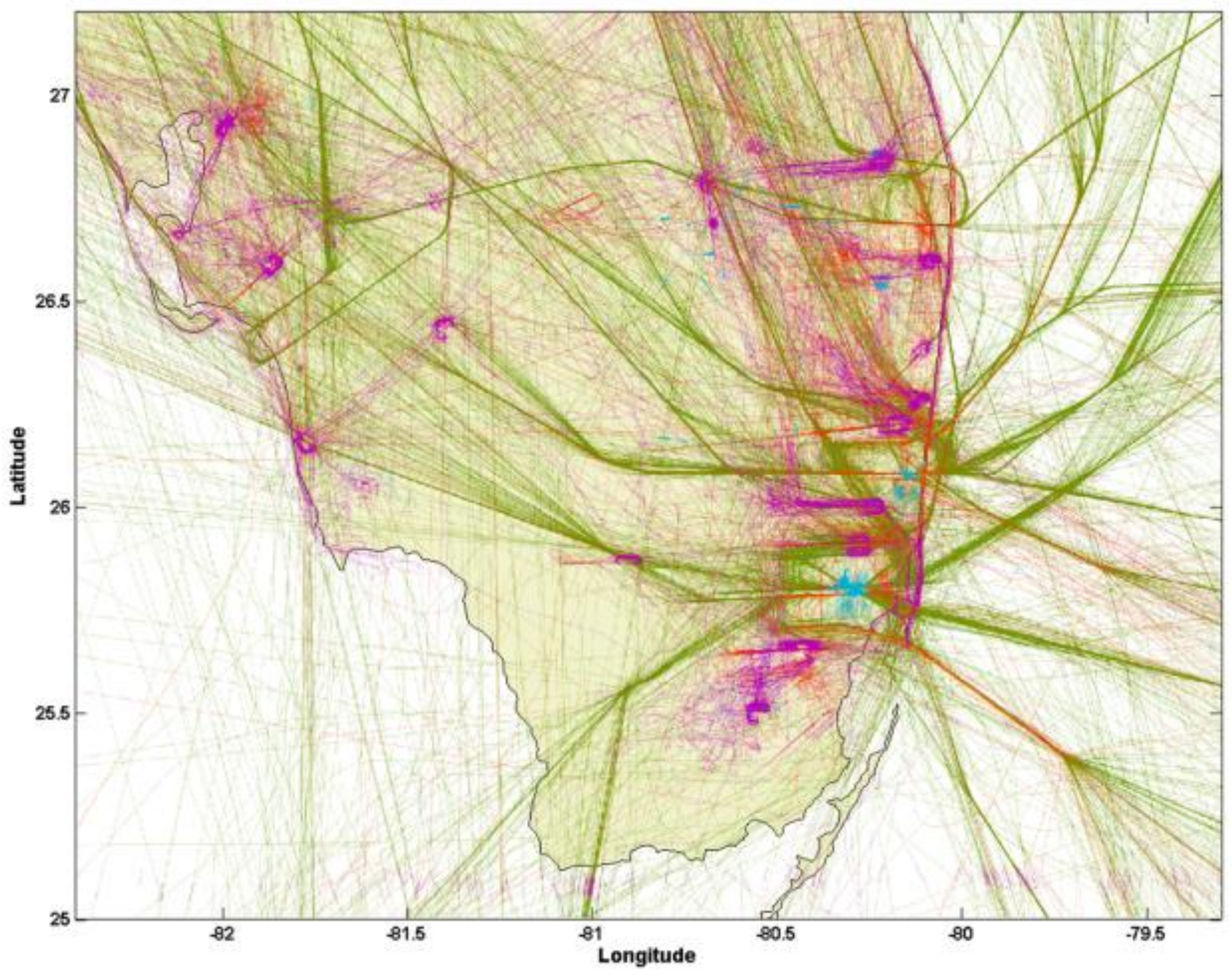

Figure 17. Lateral plot of track groups, colored by data set.

In addition to the groups which were not previously included in the Threaded Track, there is a substantial amount of unassociated data which has been fused into associated flights, creating a large amount of redundant coverage. Figure 18 shows a plot of the number of associated versus unassociated segments fused in the *fused ETMS* set. Shorter flights (with less segments) contain a higher count of associated segments, but for longer flights, which pass through more regions, can contain from 2x to 4x the number of unassociated segments. These unassociated segments are typically Tracon coverage while the flight is in cruise (over flights).

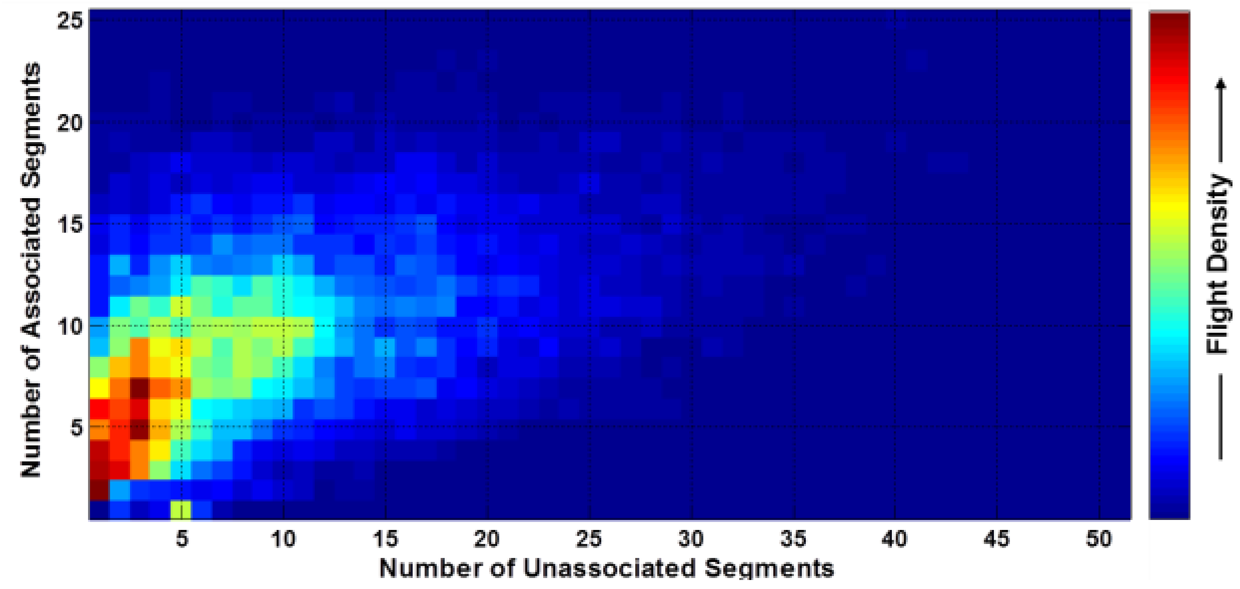

Figure 18. Comparison of associated and unassociated data for the fused ETMS set.

Figure 19 shows an example flight where unassociated and associated data have been fused together. On the left, we see there are only 6 associated radar segments, whereas there are 26 unassociated radar segments. Furthermore, while unassociated data provides only 2-3 overlapping sensors at any given time, with associated data the overlapping coverage can be in excess of 10 overlapping sensors. This level of redundancy substantially increases the fidelity of the data, moving from *a priori* modeled sensor accuracies to greater statistical confidence.

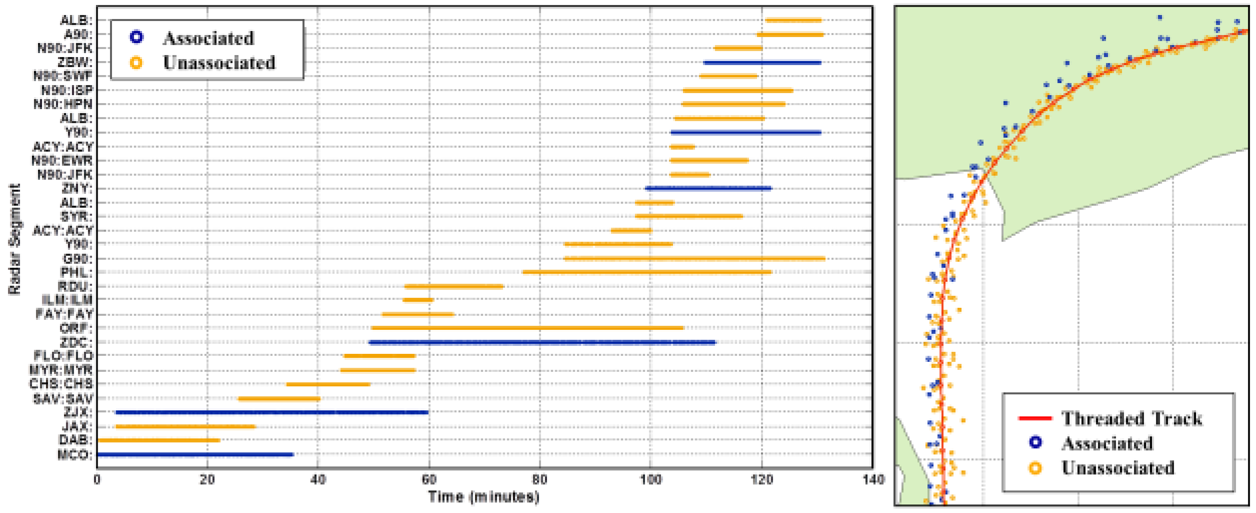

Figure 19. Sample flight showing its sensor coverage (left) and the fused trajectory (right).

6 Future Improvements and Recommendations

This report demonstrates a novel methodology for fusing flight trajectory surveillance data into a single synthetic track estimate. Compared with previous efforts, this method is based on spatial correlation without metadata requirements, enabling association of a larger portion of the source data. However, there are still many obstacles to incorporating this methodology into a production system. Several of these factors are outlined below.

6.1 Missing Features

6.1.1 Segment fusion across gaps

For the majority of flights there was enough time overlap between distinct surveillance sources so that joining segments across time gaps was not necessary. However, there are several instances of VFR flights which are tracked by a single facility (no redundant coverage) but results in sparse track points which are split into multiple segments. Similarly, touch-and-go flights at smaller airports without ground coverage often result in similar splits every time the aircraft lands.

There are two potential approaches to handle these cases within our current methodology. The first is to apply a more complex *segmentation* algorithm, which would function similar to a tracker and join across these gaps within a facility. The second approach would rely on creating special correlation scores in *geospatial joining* to extrapolate trajectory differences between segments across a temporal gap. Both methods are difficult since the fidelity of the scoring is substantially lower, and likely to produce false positives (linking two segments which are not actually related). Each of these methods requires further examination and experimentation, but will have to be addressed in order to provide an optimal trajectory fusion.

6.1.2 Addition of new data sources

The report demonstrated the fusion of NOP, ASDE-X, ETMS, and ADS-B [2], but there are additional trajectory sources which can help improve the capabilities and fidelity of the Threaded Track. One example is the Common Message Set (CMS) data, which is actually the source data for the NOP ARTCC data described earlier. CMS also provides additional flight metadata information (e.g. hand-offs and flight plans) which can be associated with the Threaded Track.

The methodology described in this report enables the integration of new data sources provided they meet the minimal requirements for *geospatial joining* and *trajectory synthesis*. This implies that each source must provide a set of latitude/longitude position reports, a *segmentation* algorithm for its trajectory points, and required pre-filtering steps in *coasting and outlier detection*, and a sensor model. Any source specific data which are not integral to the data fusion process must either provide a linkable ID (e.g. ETMS flight ID) to its data, or be supported within the Threaded Track data model (e.g. mode-S code).

6.1.3 Clock error compensation

A methodology was described above to compensate for various types of spatial radar registration errors, but always assumed an accurate report time for each data source. An additional feature is required to estimate the clock offset between facilities pairs. This clock error may vary with time and may also vary between position and altitude measurements. Without this feature, flights will be split across facilities with a large bias, and it may corrupt the measurements even in the presence of small biases.

6.1.4 ADS-B altitudes

.The ADS-B surveillance system reports uncorrected pressure altitudes and geometric altitudes. However, all radar data sources report pressure altitude below FL180 corrected with the local reference pressure of an appropriate reporting station, thus, creating a discrepancy between reported altitudes. This issue provides an opportunity to derive the pressure correction applied to the radar data by using the difference between ADS-B and radar altitudes. In addition to unifying the altitude sources, deriving local pressure term is a critical part of fusing the surveillance data with Rapid Update Cycle (RUC) weather data [7].

6.1.5 Phases of Flight

While the methodology for geospatial data fusion can be used to ensure that flight data is not merged between two flights which are in separate locations at the same time, it is possible that false positives may link the arrival end of a flight to another departure (or other types of linking errors), creating a merged flight.

Detecting these types of merged flights is possible by examining the phases of flight within a given group of trajectory data. For instance, if we were to identify multiple takeoff points for a given group of data, we would be able to detect the presence of a merged flight (which could then be split apart using a similar process to the *community detection* algorithm above). Development of this feature would require developing the components that define the “phases of flight” and the heuristics that can be used to measure them.

6.2 Performance

There are several opportunities to refine the source code to produce a more computationally efficient process. This includes both algorithmic design as well as changes to the map-reduce implementation to optimize the use of Hadoop’s computing power. In addition to the source code, the Hadoop cluster itself can be configured to help improve the performance of this process. These improvements may include rebalancing the map to reduce slot ratio, tuning the available memory, or creating job pools designed to reserve resources for the process.

6.3 Verification and Validation

While the methodology described in this report has been prototyped, this system requires extensive validation and verification (V&V). This V&V can be broken into two main components: grouping the trajectory segments, and synthesizing the optimal trajectory. The first component should ensure the optimal balance between split and merged flights, while the second component should provide the most accurate trajectory estimate given the source data.

While there is no "truth" data set to define which trajectory segments should be joined together, there are several approaches which can be applied to ensure the optimal balance between split and merged flights. First, the use of simple metrics (e.g.max speed or flight distances) can identify outlier cases, which can then be individually examined to ensure the methodology is being correctly applied on a case-by-case basis.

Other methods may compare the full solution with a solution by masking out specific components. For instance, how does the data fusion perform without the use of metadata versus with the metadata? How often are more trusted end-to-end segments such as ADS-B and ETMS split and joined (and are these cases attributed to errors in the methodology, implementation, or legitimate errors in the source data)? This approach can be used to isolate the effect of each component within the large complex process.

Another approach is to examine the variations in sensor coverage for each flight (e.g. were all the sources that were expected fused into the trajectory?). In addition to verifying the implementation of the methodology, this exercise will enable developing heuristics for the many tuning parameters in the algorithms instead of relying on data source assumptions.

The specific nature of the problem also offers another method of verification - through its algorithmic redundancy (consistency checks). There are several places in the process where data may be linked or split. For instance, all of the synthetic trajectories from Phase 1 (see Figure 1) should represent distinct groups which are not linkable in Phase 2 (unless through an ETMS segment). By allowing these trajectories to link using the same heuristics in Phase 2, we can verify that segments have been properly linked in Phase 1.

The synthetic trajectory can be validated against other sources such as Flight Operations Quality Assurance (FOQA), flight check data, etc. These sources are recorded directly from the onboard aircraft systems and capable of providing highly accurate measurements. However, these sources are typically sparse and only cover a small set of flights, and would therefore not be suitable for data fusion into the Threaded Track, but serve as a good consistency check where they exist. For instance, the radar position derived airspeed can be validated against the Pitot-tube measurements from the aircraft to provide an upper error bound.

6.4 Obstacles to Production

Provided all of the issues above are addressed, there are still several obstacles to creating a production system. These features are listed as follows:

1. **Data capture**

   Ensuring the full and timely data capture is the first bottleneck to the Threaded Track process. Since this is a data fusion of all trajectory sources, the algorithms can only be run once all sources have been acquired, making the process as fast as the slowest data source acquisition (or providing a limited solution).

2. **Computational Resources**

After data capture, there is a limit on the amount of computational resources available to complete the processing. Current estimates place this process capable of running at four to eight times faster than real-time. However, this means processing a backlog of data to be very intensive and will ultimately limit bug fixes and future developments.

3. **Automation**

   The Threaded Track must also be integrated into a process capable of automatically executing the processes and transferring data as new data becomes available. This will require extensive testing on large amounts of data to catch infrequent errors which can kill the process.

4. **Integrity monitoring**

   Finally, an integrity monitoring system would be required to ensure the data capture, processing, and output data meet the expectations of the end user.

7 References

[1] *Threaded Track Algorithm Development and Architecture*, The MITRE Corporation, Mclean, VA, March 2011, PBWP 10-2.C.1-3.

[2] Eckstein, Adric and Kurcz, Chris, *Aircraft Performance Metrics Using ADS-B Surveillance*, The MITRE Corporation, Mclean, VA, August 2012, MP 120429.

[3] *Spatial Keys – Memory Efficient Geohashes*, May 23, 2012, http://karussell.wordpress.com/2012/05/23/spatial-keys-memory-efficient-geohashes/.

[4] Girvan M and Newman MEJ, *Community structure in social and biological networks*, 2002, Proc. Natl. Acad. Sci. USA 99 (12): 7821–782.

[5] Dodonov YS and Dodonova YA, *Robust measures of central tendency: weighting as a possible alternative to trimming in response-time data analysis*, 2011, Psikhologicheskie Issledovaniya, 5(19).

[6] Douglas D and Peucker T, *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, 1973, The Canadian Cartographer 10(2), 112-122.

[7] Eckstein, Adric, *Threaded Track and Rapid Update Cycle (RUC) Data Fusion*, The MITRE Corporation, Mclean, VA, August 2012, PBWP 10-2.2-5.

Appendix A Radar Registration Equations

These algorithms provide a basis for deriving sensor biases from a set of correlated (overlapping) radars tracking multiple targets. Specifically, there is a set of least squares equations based on physical models of radar behavior which is used to empirically derive the radial, angular, and vertical biases for a given set of radar data at a given instance in time.

A.1 Radar Registration Correction:

$$\Delta x = \left(\varepsilon_{r,A} \sin(\theta_A) + r_A \varepsilon_{\theta,A} \cos(\theta_A)\right) - \left(\varepsilon_{r,B} \sin(\theta_B) + r_B \varepsilon_{\theta,B} \cos(\theta_B)\right) \tag{A.1}$$

$$\Delta y = \left(\varepsilon_{r,A} \cos(\theta_A) - r_A \varepsilon_{\theta,A} \sin(\theta_A)\right) - \left(\varepsilon_{r,B} \cos(\theta_B) - r_B \varepsilon_{\theta,B} \sin(\theta_B)\right) \tag{A.2}$$

Δx , Δy	Position difference between radars A and B for target
r_A , θ_A	Radial relative coordinates of target from radar A
r_B , θ_B	Radial relative coordinates of target from radar B
$\varepsilon_{r,A}$, $\varepsilon_{r,B}$	Radial error in radars A and B respectively
$\varepsilon_{\theta,A}$, $\varepsilon_{\theta,B}$	Angular error in radars A and B respectively

This pair of equations can be used to provide a least squares to solution to the radar registration error terms using multiple radars and multiple targets with redundant coverage areas.

A.2 Slant Range Correction:

$$r_c = r_e \cos^{-1}\left(\frac{z_s^2 + z_t^2 - r_t^2}{2 z_s z_t}\right) \tag{A.3}$$

r_e	Spherical radius of the earth
z_t	Altitude of the target
z_s	Altitude of the radar
r_t	Physical range of the target from the radar
r_c	Corrected range (lateral) of the target from the radar

The slant range correction provides a basic correction to compute the lateral range of a target given an external measurement of altitude. In the case of civilian radars, this altitude measurement is encoded in the transponder return and comes from the pressure altimeter of an aircraft.

A.3 Slant Range Error Propagation:

$$\varepsilon_{r_c} = \eta \cdot \varepsilon_{r_t} + \gamma \cdot \varepsilon_{z_a} \tag{A.4}$$

$$\eta = \frac{r_e}{\sqrt{1-\cos\left(\frac{r_t}{r_e}\right)^2}} \cdot \left(\frac{-r_t}{z_s z_t}\right) \tag{A.5}$$

$$\gamma = \frac{r_e}{\sqrt{1-\cos\left(\frac{r_t}{r_e}\right)^2}} \cdot \left(\frac{1+\left(\frac{r_t^2 - z_s^2}{z_t^2}\right)}{2z_s}\right) \tag{A.6}$$

ε_{r_c}	Error in the lateral target range
ε_{r_t}	Error in the physical target range
ε_{z_a}	Error in the target altitude

The slant range error terms can be derived using a propagation of error from the slant range correction equation. Equation A.4 then provides an expansion of the radial error terms in Equations A.1 and A.2 to solve for the radar registration corrections.

A.4 Vertical Error Model:

$$\varepsilon_{z_a} = \lambda \cdot (z_t - z_s) \tag{A.7}$$

λ	Solution parameter for the vertical error rate

The target altitude error term in Equation A.4 can also be expanded using a vertical error model to better fit the residuals in the least squares equations. In this instance the vertical error is represented as a linear function of altitude from the radar source.

Appendix B Cross Track Models

These algorithms provide the basis for the cross track smoothing. Specifically, this is one example of a set of models used in a multi-model least squares filtering solution given an input set of lateral trajectory measurements. The result is a mixed-model solution which will provide the location, direction, and curvature for a given set of input data. This process is iterated over blocks of trajectory measurements to build up the flights path over time.

B.1 Straight Least Squares Model:

$$ax + by = 1 \qquad \textbf{(B.1)}$$

$$E = \sum_i \left(ax_i + by_i - 1 \right) \qquad \textbf{(B.2)}$$

(x, y)	local orthogonal coordinates
(x_i, y_i)	data samples in local coordinates
(A, B)	linear solution parameters
E	least squares error for straight model

The straight model provides a least squares solution to the straight path of aircraft flight given a set of lateral trajectory measurements.

B.2 Turn Least Squares Model:

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \qquad \textbf{(B.3)}$$

$$E = \sum_i \left((x_i - x_c)^2 + (y_i - y_c)^2 - r^2 \right) \qquad \textbf{(B.4)}$$

(x, y)	local orthogonal coordinates
(x_i, y_i)	data samples in local coordinates
(x_c, y_c)	turn center solution parameters
r_c	turn radius solution parameter
E	least squares error for turn model

The turn model provides a least squares solution to the constant radius turn path of aircraft flight given a set of lateral trajectory measurements.

B.3 Mixed Model Solution:

$$\vec{x}_i = \frac{S_{turn}\vec{x}_{turn} + S_{straight}\vec{x}_{straight}}{S_{turn} + S_{straight}} \qquad \textbf{(B.5)}$$

$\vec{x}_{turn}$	Position solution provided by equation 3
$\vec{x}_{straight}$	Position solution provided by equation 1
S_{turn}	Inverse R-squared value of least squares turn model
$S_{straight}$	Inverse R-squared value of least squares straight model

The mixed model provides a weighted solution between the straight and turn models using the inverse R-squared values from each of the least squares residuals.

Appendix C Acronyms

ADS-B	Automatic Dependent Surveillance Broadcast
ARTCC	Air Route Traffic Control Center
ARTS	Automated Radar Terminal System
ASDE-X	Airport Surface Detection Equipment, Model X
ASR	Airport Surveillance Radar
ATC	Air Traffic Control
ATL	Hartsfield-Jackson Atlanta International Airport
CAASD	Center for Advanced Aviation System Development
CMS	Common Message Set
CPU	Central Processing Unit
CRS	CAASD Repository System
csv	comma separated value
DAG	Directed-Acyclic Graph
ETMS	Enhanced Traffic Management System
FL	Flight Level
FOQA	Flight Operations Quality Assurance
GA	General Aviation
I/O	Input/Output
ID	Identifier
IFR	Instrument Flight Rules
MSE	Mean Square Error
NAS	National Airspace System
NKX	Miramar Marine Corps Air Station
NM	Nautical Miles
NOP	National Offload Program
RA	Resolution Advisory
RUC	Rapid Update Cycle
SCT	Southern California Tracon

SMR	Surface Movement Radar
STARS	Standard Terminal Automation Replacement System
TCAS	Traffic Alert and Collision Avoidance System
TRACON	Terminal Radar Approach Control
V&V	Verification and Validation
VFR	Visual Flight Rules