

Determination of Horizontal and Vertical Phase of Flight in Recorded Air Traffic Data

Mike Paglione*

*Federal Aviation Administration, Simulation and Analysis Group
William J. Hughes Technical Center, Atlantic City International Airport, NJ, 08405*

Robert Oaks†

*General Dynamics Network Systems
William J. Hughes Technical Center, Atlantic City International Airport, NJ, 08405*

This paper presents two algorithms that determine an aircraft's phase of flight state in post-analysis using tracked radar surveillance data. These algorithms determine the horizontal phase of flight (whether the aircraft was flying straight or turning) and the vertical phase of flight (whether the aircraft was flying level or transitioning; i.e., ascending or descending). The paper presents the algorithms and identifies their input parameters. It then discusses the design of experiment analysis that was used to determine the algorithms' optimum performance using Global Positional Satellite position data. Examples clarifying the algorithms' application are provided along with a comparison of the new algorithms and the legacy version that it replaces.

I. Introduction

Most Air Traffic Service Providers in the United States and Europe anticipate significant growth in air traffic that is expected to out pace the capacity limits of our air traffic control systems, resulting in greater congestion and inefficiency. Broad advances in ground-based and airborne automation, such as Decision Support Tools (DSTs), are envisioned to help mitigate the problem. These tools have many purposes and typically serve to lower the complexity of airspace problems faced by the current human decision makers operating the system. They include tools that serve to predict future conflicts between aircraft, both for ground based controllers or airborne pilots, allowing more strategic separation management of aircraft. Air traffic management DSTs include capabilities that forecast where and when traffic workload will stress the system, allowing air traffic supervisors to make more efficient adjustments to either avoid the condition or alter staff and/or airspace accordingly. Such tools also include air traffic metering tools to efficiently sequence aircraft into en route and arrival flows, maximizing the capacity of the system. A common thread in all these DSTs is the accurate and timely modeling of an aircraft's current state and anticipated future path. This function is referred to as the Trajectory Predictor (TP) process.

The trajectory is the actual or future four-dimensional path of the aircraft. TP accuracy can be measured by post flight comparisons of predicted and observed aircraft trajectories. Many factors influence the accuracy of the predicted aircraft trajectories, since most, if not all, the data used by the TP to make the prediction is itself imprecise. The atmospheric forecasts, the radar surveillance data, the lateral, vertical, and speed intent data are all examples of the input sources used by the TP. The aircraft's horizontal and vertical state is also a significant factor in the TP's accuracy. It is common for researchers to segregate the flight into different states (e.g. turning, straight, level, climbing, and descending), since the performance is so dissimilar between them. In Ref. 1 researchers at the National Aeronautics and Space Administration Ames Research Center presented an automated method that partitioned trajectory statistics by phase of flight and look-ahead time. In Ref. 2 researchers at MITRE's Center for Advanced System Development evaluated a strategic conflict probe's accuracy by partitioning the input trajectory data by flight phase (e.g. climbing and descending). In Ref. 3 and Ref. 4 researchers at the Federal Aviation

*FAA Conflict Probe Assessment Team Lead, Simulation and Analysis Group; mike.paglione@faa.gov. AIAA Member.

†Principal Staff Member - Information Management, General Dynamics Network Systems; robert.ctr.oaks@faa.gov. AIAA Member.

Administration's (FAA) William J. Hughes Technical Center (WJHTC) presented a generic sampling technique to measure the trajectory accuracy of TPs and in Ref. 3 grouped horizontal and vertical measurements statistically.

Besides flight state or phase of flight being an important consideration in TP performance, tactical conflict probes utilize the flight state to make their near-term conflict predictions. In the FAA's current development of the next generation of Surveillance Data Processing (SDP), which is a part of the En Route Automation Modernization (ERAM) Program, the safety alert function is being evaluated by utilizing the aircraft state from post-analysis. In Ref. 5 researchers at the WJHTC present a plan implementing the results of a series of studies developing metrics for SDP and other ERAM sub-systems.

The tools developed by the researchers at the FAA's WJHTC require an algorithm to determine an aircraft's horizontal and vertical state in post-analysis using radar surveillance or track data. The legacy algorithm used to date is based on three consecutive smoothed data points. Through observation it has been known that the output from this algorithm is noisy and inaccurate. This paper discusses a replacement for this legacy algorithm. The paper presents the new algorithm and its components in Section II, provides an analysis to determine its optimal input parameters in Section III, shows the improvement provided by this new algorithm in Section IV, and presents concluding remarks in Section V.

II. Phase of Flight Detection Algorithms

The input to the Phase of Flight Detection Algorithms is four dimensional aircraft positional data provided by an En Route Host Computer System (HCS). For each aircraft, this data consisting of time, an x-position, a y-position, and an altitude, which as an aggregate defines the track of the aircraft.

A. Horizontal Phase of Flight Detection Algorithm

Figure 1 presents the pseudocode for the Horizontal Phase of Flight Detection Algorithm. This pseudocode is at a high level and does not explicitly identify its parameters or specify the functions that are calculated as a part of the algorithm, which are described in the following subsections. For performance, the algorithm is partitioned into three parts: data collection in line 1, Tier 1 testing between lines 2 – 8, and Tier 2 testing between lines 9 – 18. This will be clarified in the following paragraphs.

```

1 Collect xy track data in time window around track point ;
2 Calculate Pearson R and slope for selected xy track data ;
3 If (|Pearson R| > horzPearsonThreshHi) {
4     Declare track point to be in a flying straight state;
5 }
6 Else if (|Pearson R| < horzPearsonThreshLo) {
7     Declare track point to be in a turn state ;
8 }
9 Else {
10    Translate and rotate selected xy track data ;
11    Calculate R2 and Flatness for selected xy track data ;
12    If ((R2 > horzRSqrThresh) and (Flatness > horzFlatnessThresh)) {
13        Declare track point to be in a turn state ;
14    }
15    Else {
16        Declare track point to be in a flying straight state ;
17    }
18 }

```

Figure 1. Horizontal Phase of Flight Detection Algorithm

Figure 1, line 1 specifies the first step for determining the horizontal phase of flight for an aircraft at a specific time point, which is to collect positional data (xy-data) in a time window surrounding the time point. This time window is specified by the input parameters defined in Table 1. This data is collected for linear regression calculations and for quadratic regression calculations that are used in the Tier 1 and Tier 2 testing.

Figure 1, lines 2 through 8 represent the Tier 1 testing. Figure 1, line 2 specifies two statistical parameters that are calculated: the *Pearson R* and the *Slope*. Both of which are described in Section II.C.2. The *Pearson R* is used

for the Tier 1 testing, while the *Slope* is used during the later Tier 2 testing. This testing is based on a linear regression calculation described in Section II.C.2.

Figure 1, lines 9 through 18 represent the Tier 2 testing. This testing is necessary whenever the Tier 1 *Pearson R* test is inconclusive. The Tier 2 testing requires the calculation of R^2 , which is a quadratic regression statistic described in Section II.C.4, and *Flatness*, which is the measure of the flatness a quadratic curve and is also described in Section II.C.4. But first, as specified in Fig. 1, line 10, the data must be translated and rotated to ensure that the quadratic curve is aligned with the y-axis; this is described in Section II.C.3.

All of this algorithms input parameters are summarized in Table 1.

Table 1. Horizontal Phase of Flight Detection Algorithm Input Parameters

Input Parameter	Description
<i>horzTimeWindowBack</i> <i>horzTimeWindowAhead</i>	These input parameters, which are in seconds, specify a time window for collecting the xy-data. <i>horzTimeWindowBack</i> specifies the time interval for collecting the xy-data preceding the time point and <i>horzTimeWindowAhead</i> specifies the time interval for collecting the xy-data succeeding the time point.
<i>horzTier1TrackType</i> <i>horzTier2TrackType</i>	These input parameters are used to specify whether actual (i.e., recorded) or smoothed xy-data is to be used in the Tier 1 and Tier 2 testing. These parameters can assume the ASCII values: “actual” or “smooth”.
<i>horzPearsonThreshHi</i> <i>horzPearsonThreshLo</i>	These input parameters are used in the Tier 1 testing. <i>horzPearsonThreshHi</i> specifies the <i>Pearson R</i> value at which it is assumed that the aircraft is flying straight. <i>horzPearsonThreshLo</i> specifies the <i>Pearson R</i> value at which it is assumed that the aircraft is turning.
<i>horzRSqrThresh</i>	This input parameter is used in the Tier 2 testing and represents the minimum threshold of the R^2 value indicating the aircraft may be turning.
<i>horzFlatnessThresh</i>	This input parameter is used in the Tier 2 testing and represents the minimum threshold of the <i>Flatness</i> value indicating the aircraft may be turning.

B. Vertical Phase of Flight Detection Algorithm

Figure 2 presents the pseudocode for the Vertical Phase of Flight Detection Algorithm. This pseudocode is at a high level and does not explicitly identify its parameters or specify the functions that are calculated as a part of the algorithm, which are described in the following paragraphs.

1	Collect tz track data in time window around track point ;
2	Calculate slope for selected altitude track data ;
3	If (slope < -vertSlopeThresh) {
4	Declare track point to be in a descending state ;
5	}
6	Else if (slope > vertSlopeThresh) {
7	Declare track point to be in an ascending state ;
8	}
9	Else {
10	Declare track point to be in a level state ;
11	}

Figure 2. Vertical Phase of Flight Detection Algorithm

Figure 2, line 1 specifies the first step for determining the vertical phase of flight for an aircraft at a specific time point, which is to collect time and altitude data (tz-data) in a time window surrounding the time point. This time window is specified by the input parameters defined in Table 2. This data is collected for linear regression calculations used to calculate the *Slope* statistic.

Figure 2, line 2 specifies that the *Slope* statistic is calculated. This is described in Section II.C.2, but in this case directly represents the ascent or descent rate of the aircraft.

Figure 2, lines 3 through 11 represent the testing required to determine the vertical phase of flight. In this case, this is simply a test to see if the *Slope* is less than the negative value of the threshold, *vertSlopeThresh*, which

represents a descending state, or if the *Slope* is greater than the positive value of the same threshold, which represents an ascending state; otherwise the track point represents level flight.

All of this algorithms input parameters are summarized in Table 2.

Table 2. Vertical Phase of Flight Detection Algorithm Input Parameters

Input Parameter	Description
<i>vertTimeWindowBack</i> <i>vertTimeWindowAhead</i>	These input parameters, which are in seconds, specify a time window for collecting the altitude data. <i>vertTimeWindowBack</i> specifies the time interval for collecting the altitude data preceding the time point and <i>vertTimeWindowAhead</i> specifies the time interval for collecting the altitude data succeeding the time point.
<i>vertTrackType</i>	This input parameters is used to specify whether actual (i.e., recorded) or smoothed altitude data is to be used in the testing. This parameter can assume the ASCII values: “actual” or “smooth”.
<i>vertSlopeThresh</i>	This input parameter, which is in feet/second, specifies the threshold at which an aircraft is declared to be ascending or descending.

C. Algorithm Functional Support

1. Data Collection Functionality

The time windows for collecting the horizontal xy-data and vertical tz-data are defined by specifying a time parameter for data preceding the time point and another time parameter for data succeeding the time point. For the Horizontal Phase of Flight Detection Algorithm these are known as *horzTimeWindowBack* and *horzTimeWindowAhead* and for the Vertical Phase of Flight Detection Algorithm these are known as *vertTimeWindowBack* and *vertTimeWindowAhead*. These parameters, as pairs, form windows with the specific time point lying within this window for both the Phase of Flight algorithms. The values for these parameters are in seconds and are not constrained; therefore the time point may lie anywhere within the window.

The HCS track data used by both Phase of Flight Detection Algorithms is based on data recorded from Air Route Traffic Control Centers. This data is obtained from surveillance radars and processed by the HCS tracking algorithm before it is recorded. Even after this processing the data is noisy and has anomalies such as time gaps and erroneous data points.⁶ As such, it is again “cleaned” before it is used by the Phase of Flight Detection Algorithms. As a part of this cleaning process, smoothed data is also calculated using an 11-point smoothing algorithm. The parameters *horzTier1TrackType*, *horzTier2TrackType*, and *vertTrackType* indicate whether actual (i.e., recorded) or smoothed data is to be used in the calculations.

2. Linear Regression Functionality

The Tier 1 testing in the Horizontal Phase of Flight Detection Algorithm uses linear regression functionality to calculate the *Pearson R* correlation coefficient of the selected data. The *Pearson R* statistic is a commonly used coefficient that measures how well data fits a straight line.[‡] The *Pearson R* is calculated using Eq. (1).⁷

$$Pearson R = \frac{S_{xy}}{\sqrt{S_{xx}} \sqrt{S_{yy}}} \quad (1)$$

where x and y are the independent and dependent variables, $S_{xx} = \sum x_i^2 - \frac{(\sum x_i)^2}{n}$,

$$S_{yy} = \sum y_i^2 - \frac{(\sum y_i)^2}{n}, \text{ and } S_{xy} = \sum x_i y_i - \frac{(\sum x_i)(\sum y_i)}{n}.$$

[‡] The *Pearson R* is short for *Karl Pearson’s product-moment correlation coefficient* and is described in numerous books that discuss data correlation. In Ref. 7, which is used as a reference in this paper, this statistic is called *The Sample Correlation Coefficient*.

A *Pearson R* value of +1.0 indicates that there is a perfect positive linear correlation between the variables that are being evaluated, a value of -1.0 indicates a perfect negative linear correlation, and a value of 0.0 indicates that there is no linear correlation. For the Horizontal Phase of Flight Detection algorithm it is assumed that if the absolute value of the *Pearson R* is close to 1.0, the aircraft must be flying straight, and if the statistic is close to 0.0, the aircraft must be turning. These bounds are represented by the parameters *horzPearsonThreshHi* and *horzPearsonThreshLo*.

Another statistic that is available through the Linear Regression Functionality is the *Slope* statistic, which is the slope of the least squares regression line fit to the xy-pairs. It is calculated using Eq. (2).⁷ The *Slope* is used in both horizontal and vertical algorithms. In the horizontal algorithm, the *Slope* is utilized for the coordinate rotation and translation, described in Section II.C.3. In the vertical algorithm, the *Slope* is used to estimate the ascent/descent rate of the aircraft. For the Vertical Phase of Flight Detection Algorithm this bound is represented by *vertSlopeThresh*. Unlike the horizontal algorithm that is input with paired x-y coordinates in nautical miles, the vertical input includes paired time stamp and altitude data in units of seconds and feet, respectively.

$$Slope = \frac{S_{xy}}{S_{xx}} \quad (2)$$

3. Coordinate Translation and Rotation Functionality

If the *Pearson R* used in the Tier 1 testing of the Horizontal Phase of Flight Detection Algorithm does not conclusively determine if the aircraft is flying straight or turning, Tier 2 testing is necessary. This requires the use of the quadratic regression functionality that is discussed in Section II.C.4. Both functionalities require horizontal xy-data, for which the x-position is assumed to be the independent variable, and the y-position is assumed to be the dependent variable. For the linear regression functionality this assumption is unimportant; however for the quadratic regression functionality, which fits a quadratic function, this assumption has an impact because the quadratic function assumes that the independent variable is monotonically increasing in either the positive or negative direction. This requires that the xy-data be rotated. For convenience the xy-data is not only rotated, but it is also translated to the point of interest. This is done using the standard equation presented in matrix form as Eq. (3).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix} \quad (3)$$

where x and y represent a data point collected for the point of interest, x_r and y_r represent the point of interest, and Θ represents the angle of rotation (which is determined by the *Slope* calculated by the linear regression functionality).

4. Quadratic Regression Functionality

If Tier 1 testing is not conclusive, then Tier 2 testing is required, which uses quadratic regression functionality. The Horizontal Phase of Flight Detection Algorithm assumes that an aircraft flies on an arc during a turn, which can be approximated by the quadratic polynomial shown in Eq. (4).

$$y = a + bx + cx^2 \quad (4)$$

where x and y are the translated and rotated coordinates described in Section II.C.3 and a, b, and c are constant coefficients.

The constant coefficients in Eq. (4) can be determined using least squares techniques, which requires solving the matrix equation presented in Eq. (5).⁷

$$\begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix} = \begin{bmatrix} \sum 1 & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (5)$$

After the coefficients in Eq. (5) are solved for a given set of aircraft track positions, the quadratic regression functionality determines if the aircraft track positions reasonably fit this quadratic curve using a coefficient of multiple determination statistic called R^2 . This statistic indicates how well the aircraft is following the modeled curve representing the potential turn. R^2 is expressed in Eq. (6).⁷

$$R^2 = 1 - \left(\frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \right) \quad (6)$$

where y_i is the i^{th} coordinate on the y-axis, \hat{y}_i is the i^{th} quadratic predicted coordinate on the y-axis based on the polynomial presented in Eq. (4) using the coefficients that solve Eq. (5), and \bar{y} is the average of all the supplied y coordinate positions.

A R^2 that is greater than *horzRSqrThresh* indicates an acceptable fit to the modeled quadratic polynomial curve, otherwise the algorithm assumes the positions are not modeling the curve and are probably not turning. Since the algorithm would have already failed the Tier 1 linear regression test described in Section II.C.2, the aircraft positions may be exhibiting random error and are assumed straight.

Besides having fit the curve well, a curve may be modeled that is very flat or almost linear. A quadratic polynomial as expressed in Eq. (4) will be closely linear with a small coefficient c. However, the objective of this algorithm was to determine turns that were significant (about 10 degrees heading change or larger). As a result, even with a good fit from the R^2 , the modeled curve is also required to be above a certain *Flatness*.

Flatness is defined in Ref. 8 as something “having a relatively smooth or even surface.” In this case, the modeled quadratic polynomial curve is flat if it is very linear. This is calculated geometrically by first determining the peak position by solving for the derivative of Eq. (4). Next a straight line segment is calculated from the first aircraft position to the last supplied to the algorithm. The flatness is the perpendicular distance measure in nautical miles[§] formed by drawing a normal from the peak position to the line segment. An example is shown in Fig. 8 in Section II.D.1.

The quadratic regression functionality is used only after the linear fit is determined to be poor. It is supplied a parameter number of aircraft x and y coordinate positions, solves for the coefficients defined in Eq. (4) and Eq. (5) for the quadratic polynomial curve, determines the positions fitness to this curve from Eq. (6), and then checks the *Flatness* of the modeled curve. If both the fitness is above the input parameter *horzRSqrThresh* and the *Flatness* is above the input parameter *horzFlatnessThresh*, the aircraft position supplied is declared to be in a turn.

[§] Since all the positions supplied horizontally are supplied in the x-y stereographic plane in nautical miles, it is natural that this distance is also in nautical miles.

D. Descriptive Examples

1. Horizontal

Figure 3 presents a graph containing track point xy-data for a single flight. This aircraft flew in an easterly direction, and then turned right to a southwesterly direction. The track points marked with an X represent the actual data. The track points marked with a + represent the associated smooth data. This short flight segment will be used to illustrate the Horizontal Phase of Flight Detection algorithm.

The input parameters used for this description are:

- $horzTimeWindowBack = 55$
- $horzTimeWindowAhead = 55$
- $horzTier1TrackType = smooth$
- $horzTier2TrackType = actual$
- $horzPearsonThreshHi = 0.998$
- $horzPearsonThreshLo = 0.01$
- $horzRSqrThresh = 0.4$
- $horzFlatnessThresh = 0.25$

The rationale for using these values is presented in Section III.

Figure 4 presents a sample time point for which the Horizontal Phase of Flight Detection algorithm declared the aircraft to be flying straight because, during the Tier 1 testing, the *Pearson R* value was greater than the input parameter $horzThreshHi$. The actual and smoothed track point at this time point (66840 seconds) is circled. Since the $horzTier1TrackType$ was input as smooth and since the $horzTimeWindowBack$ was 55 seconds and the $horzTimeWindowAhead$ was 55 seconds, 11 smoothed track points were used in the linear regression functionality to compute the *Pearson R*. These 11 points are represented as circles overlaying the +. The *Pearson R* for this point was equal to 0.998850, which is greater than the input value for the $horzPearsonThreshHi$, which was 0.998; therefore the aircraft was declared to be flying straight at this time point.

Figure 5 presents a sample time point for which the Horizontal Phase of Flight Detection algorithm declared the aircraft to be turning because, during the Tier 1 testing, the *Pearson R* value was less than the input parameter $horzPearsonThreshLo$. The actual and smoothed track point at this time point (66910 seconds) is circled. Again, since the $horzTier1TrackType$ was input as smooth and since the $horzTimeWindowBack$ was 55 seconds and the $horzTimeWindowAhead$ was 55 seconds, 11 smoothed track points were used in the linear regression functionality to compute the *Pearson R*. These 11 points are represented as circles overlaying the +. The *Pearson R* for this point was equal to 0.005937, which is less than the input value for the $horzPearsonThreshLo$, which was 0.01; therefore the aircraft was declared to be turning at this time point.

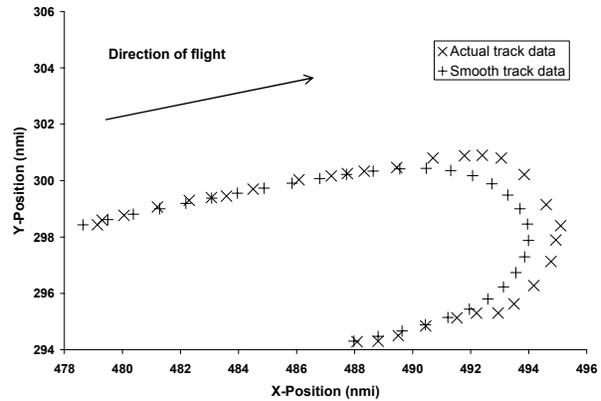


Figure 3. Horizontal Example Showing XY-Data

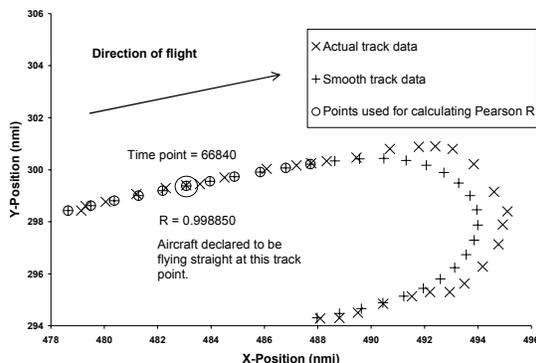


Figure 4. Horizontal Example Showing Track Point Where $R > horzThreshHi$

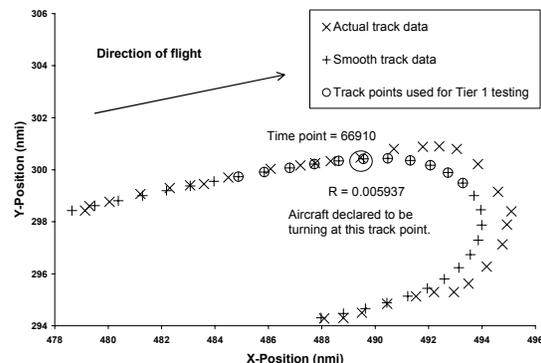


Figure 5. Horizontal Example Showing Track Point Where $R < horzThreshLo$

The Horizontal Phase of Flight Detection algorithm resolved the status of the time points presented in Fig. 4 and Fig. 5 with the Tier 1 testing. Fig. 6 and Fig. 7 present situations where the algorithm used the Tier 2 testing because the Tier 1 testing was inconclusive. In Fig. 6 the track data for the time point at 66900 seconds is circled. The *Pearson R*, which was based on 11 smoothed track data points, was calculated to be 0.616174, which lies between *horzPearsonThreshHi* (0.998) and *horzPearsonThreshLo* (0.10); therefore the Tier 2 testing was required. In this case the 11 actual track points used by the quadratic regression functionality are circled. This data resulted in an R^2 value of 0.946486 being calculated and a *Flatness* value of 0.032766 being calculated. Based on the values specified for the input parameters *horzRsqThresh* (0.4) and *horzFlatnessThresh* (0.25), the aircraft was declared to be flying straight at this time point. For the time point at 66902 seconds in Fig. 7, the linear regression functionality calculated a *Pearson R* value of 0.478350. This resulted in Tier 2 testing during which the quadratic regression functionality calculated a value of 0.645140 for R^2 and 1.220979 for the *Flatness*. Therefore, the aircraft was declared to be turning at this track point.

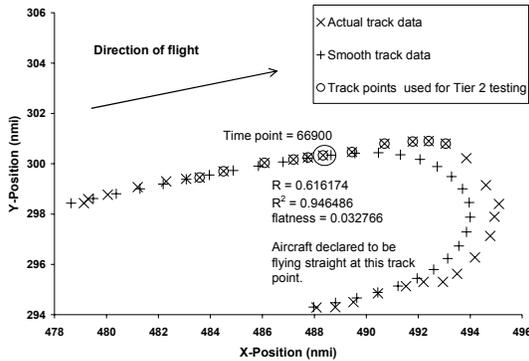


Figure 6. Horizontal Example Showing Track Point Where $horzThreshLo < R < horzThreshHi$ (1 of 2)

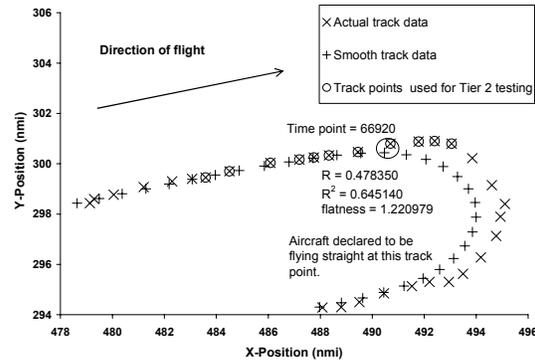


Figure 7. Horizontal Example Showing Track Point Where $horzThreshLo < R < horzThreshHi$ (2 of 2)

In order to better illustrate the coordinate translation and rotation that is used by the quadratic regression functionality to compute the flatness, the same 11 actual track points presented in Fig. 7 are shown translated and rotated in Fig. 8. The 2nd order quadratic function that is fitted to this data is shown as a solid line. The longer dashed line represents the line drawn between the first and last data points and the shorter dashed line represents the flatness value, as described in Section II.C.4.

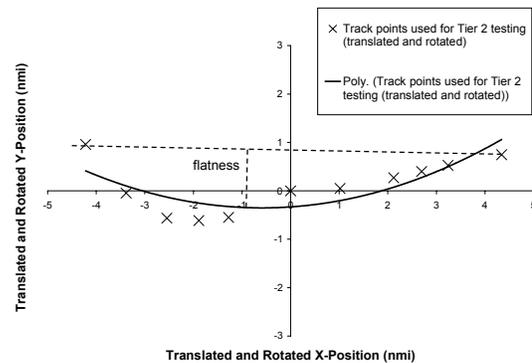


Figure 8. Horizontal Example Showing XY-Data After Translation and Rotation

2. Vertical

Figure 9 presents a graph containing track point altitude data for a single flight during its ascent phase. The track points marked with an X represent the actual data. The track points marked with a + represent the associated smooth data. Figure 10 shows the last track point that the Vertical Phase of Flight Detection algorithm declared to be ascending.

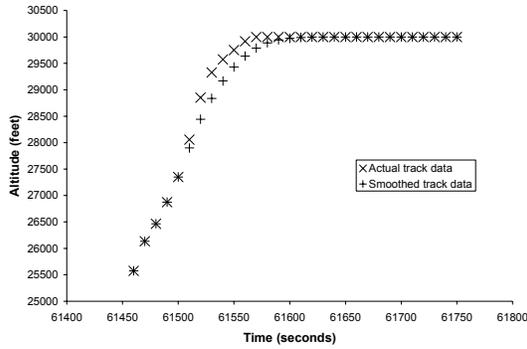


Figure 9. Vertical Example Showing Ascending Altitude Data

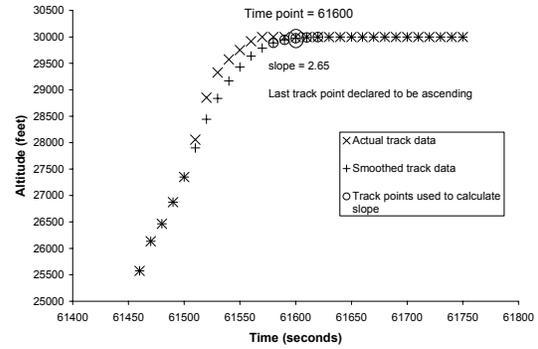


Figure 10. Vertical Example Showing Last Track Point Declared to be Ascending

Figure 11 presents a graph containing track point altitude data for a single flight during its descent phase. The track points marked with an X represent the actual data. The track points marked with a + represent the associated smooth data. Figure 12 shows the first track point that the Vertical Phase of Flight Detection algorithm declared to be descending.

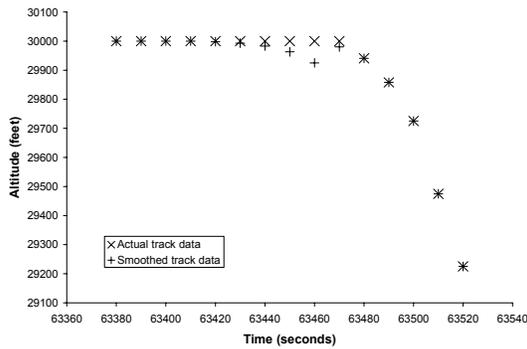


Figure 11. Vertical Example Showing Descending Altitude Data

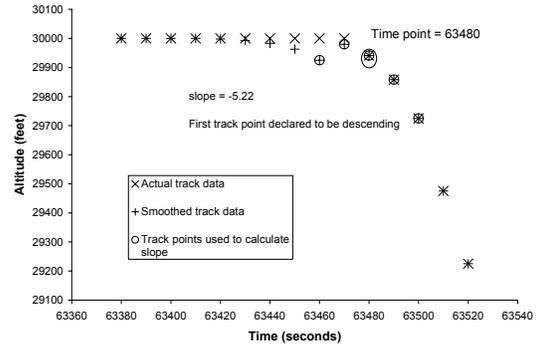


Figure 12. Vertical Example Showing First Track Point Declared to be Descending

III. Parameter Optimization

In Section II, both the horizontal and vertical detection algorithms were presented in detail and definitions of their various parameters were defined. The subject of this section is to determine the specific settings or levels in which these parameters should be set to optimize the algorithm's performance. For example, what value should be used for the *Pearson R* statistic in the linear regression calculation of the Horizontal Phase of Flight Detection Algorithm? Intuitively, the value should be close to 1.0, indicating the aircraft track positions are flying straight and thus linearly. However, the specific value ranging from 0.9 to 1.0 is difficult to define without additional information supplied by experimentation and analysis.

To address this challenge for all the various parameters, first the Section III.A, will present a set of metrics that reflect the primary errors a phase of flight detection algorithm would have. Next, Section III.B will discuss a reference set of truth data in which the metrics can be applied and algorithms can be compared against. Finally, a complete randomized block experiment will be presented that utilizes these metrics and reference data resulting in a compilation of optimal settings for the various parameters.

A. Metrics

Both algorithms, horizontal and vertical, are concerned with determining a surveillance radar track position's phase of flight state. For the horizontal dimension for example, either the algorithm detects a turn or does not. Simultaneously, the flight actually is either turning or not turning for the coincident track position. This is illustrated in Table 3 and results in four descriptive and mutually exclusive events. Both the actual event occurs and is detected to occur, referred to as a Valid Call (VC). In the second case, both the actual event does not occur (is not in a turn)

and is correctly not detected, referred to as a Valid No Call (NC). In the third case, the actual event does not occur (not in a turn) and is incorrectly detected to occur. This case is referred to as a False Call (FC), since the detection was truly not warranted. Finally, the last case is when the algorithm incorrectly detects there is not a turn and indeed one actually occurs. This case is referred to as a Missed Call (MC). For the vertical algorithm, the errors are completely analogous, but instead of turns the true events are vertical transitions (either ascending or descending). This is illustrated in Table 4.

Table 3. Horizontal Phase Of Flight Events

		Algorithm – Detected Event	
		Turn	No turn
Actual Event	Turn	Valid Call (VC)	Missed Call (MC)
	No turn	False Call (FC)	Valid No Call (NC)

Table 4. Vertical Phase of Flight Events

		Algorithm – Detected Event	
		Ascending or descending	Level
Actual Event	Ascending or descending	Valid Call (VC)	Missed Call (MC)
	Level	False Call (FC)	Valid No Call (NC)

For each flight, the phase of flight state is calculated on each surveillance track position with a frequency of about 12 seconds, so each flight may have several hundred position reports quantified into the eight outcomes, four per algorithm, as listed in Tables 3 and 4. The estimates of the two error probabilities can be calculated on a given flight. The estimated probability of missing or not detecting an actual turn/vertical transition is the ratio of missed calls to total number of actual events, which is the sum of missed and valid calls. This is expressed in Eq. (7).

$$P(MC) = \frac{MC}{(MC + VC)} \quad (7)$$

where P(MC) is the estimated probability of missed calls, MC is the quantity of missed calls, and VC is the quantity of valid calls

The missed detection's companion metric is the estimated probability of falsely detecting a turn/vertical transition that actually does not occur. This is defined as the ratio of false calls to the total number of straight/level track positions, which is the sum of false calls and valid no calls from Tables 3 and 4, respectively. This is expressed in Eq. (8).

$$P(FC) = \frac{FC}{(FC + NC)} \quad (8)$$

where P(FC) is the estimated probability of false calls, FC is the quantity of false calls, and NC is the quantity of valid no calls

The perfect algorithm would have zero probabilities for both missing and falsely detecting a turn or vertical transition. Therefore, the objective of developing these algorithms is to minimize these to error probabilities for many flights. A standard statistic is to average the two probabilities in Eq. (7) and Eq. (8) for all the sample flights applied.

B. Truth Reference Data Set

To implement the metrics defined in Section III.A requires a collection of actual turns and vertical transitions that are definitively determined with associated aircraft track positions as input into the algorithms presented in Section II. In a separate study completed in July 2005, a large sample of about 300 flight segments were collected from Global Positioning Satellite (GPS) reports, originally used to certify aircraft for reduced vertical separation minima (see Ref. 9 for details on this study). A sub-set from these flight samples was selected for use in this study.

A total of 57 flights were extracted from Salt Lake City Air Route Traffic Control Center collected from several days in January and February 2005. One-second time stamped GPS position reports were available for all 57 flights. These position reports were manually inspected for all flights and times and phase of flight determined. The results were recorded in a database table containing over 100,000 records. For each record, the position was defined with the horizontal and vertical phase of flight state. This truth reference data set and the respective input aircraft track data were required to calculate the error probabilities in Eq. (7) and Eq. (8). The truth reference data set defines the actual events in Tables 3 and 4 and the respective track data are the primary input into the detection algorithms.

C. Design of Experiment Results

Experiments are performed by most researchers and scientists in practically all disciplines. An experiment is defined in Ref. 10 as “a test or series of tests in which purposeful changes are made to input variables of a process or system so that we may observe and identify the reasons for changes in the output response.” To illustrate this further Fig. 13 presents the general model of a process under study as adapted from Ref. 10. An input stimulus is entered into a process with a set of controllable factors. These are the factors or independent variables in the experiment that are manipulated to study the output or response variables. The uncontrollable factors are not easily manipulated but through experimental design techniques such as blocking and randomization can be removed from the experiment. The output response variables are the dependent variables of the experiment. They are often determined by application of a metric or measured by a sensor device.

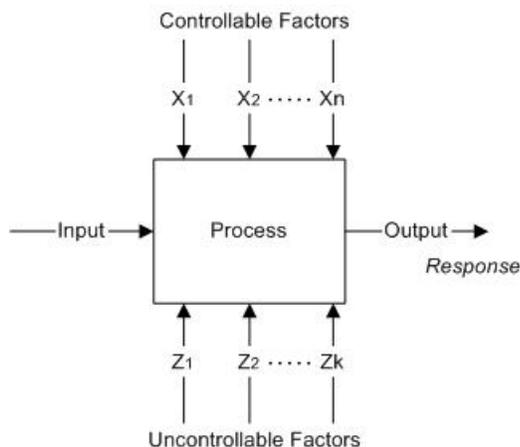


Figure 13. General Model of a Process (adapted from Ref. 10)

There are many purposes of performing an experiment. For this study, the objective was to determine the most favorable parameter settings for the Horizontal and Vertical Phase of Flight Detection Algorithms. The most favorable settings need to minimize the two error probabilities defined in the metrics in Section III.A. For the horizontal algorithm, there are a potential of eight controllable factors to consider, as listed in Table 1, while the vertical algorithm has at most four factors listed Table 2. Therefore, a set of experiments need to be designed utilizing the truth reference data set presented in Section III.B. The output response variables are the two statistics defined in Eq. (7) and Eq. (8). The following two sub-sections will present the design and analysis of experiments for the horizontal and vertical algorithms, respectively.

1. Experimental Design and Analysis of the Horizontal Algorithm

The Horizontal Phase of Flight Detection Algorithm has a potential of eight controllable factors listed in Table 1 as the parameters in the algorithm. The time window parameters, named *horzTimeWindowAhead* and *horzTimeWindowBack*, refer to how much time in seconds ahead and behind the current position, respectively, that the algorithm samples for its calculation of phase of flight. To simplify the experiment, these two parameters were set equal to each other and treated as one. For the rest of the paper, this factor will be referred to as the *horzTimeWindow*. Therefore, the experiment was reduced to an analysis of the seven factors, which are listed in Table 5.

There are many types of experimental designs in the literature. A factorial experiment is a very efficient experiment that evaluates a process under study with many factors.^{10, 11} The experiment is factorial since all the

combinations of the levels of these factors are examined. As a result, the main effects of the factors under study are estimated as well as all their interactions. For this study, not only are several factors involved, but there is little cost associated with running all the selected levels and their combinations. The selected levels are listed in Table 5. Most of the factors contain only two levels, covering an upper and lower bound. For factors *horzTier1TrackType* and *horzTier2TrackType*, there are only two nominal levels to choose from. Either the actual track report positions are used or the post-smoothed track positions are used. For the factor, *horzRSqrThresh*, three levels were evaluated. Unlike the other factors, it required a third value due to the sensitivity of the algorithm to this parameter. This results in 192 runs (2x2x2x2x2x3x2) for the full factorial experiment of the seven factors and all their level combinations.

Table 5. Horizontal Algorithm Experimental Factors

Factor	Levels Implemented	Units
<i>horzTimeWindow</i>	25, 55	Seconds
<i>horzTier1TrackType</i>	actual, smooth	-
<i>horzTier2TrackType</i>	actual, smooth	-
<i>horzPearsonThreshHi</i>	0.998, 0.995	-
<i>horzPearsonThreshLo</i>	0.6, 0.1	-
<i>horzRSqrThresh</i>	0.92, 0.82, 0.4	-
<i>horzFlatnessThresh</i>	0.25, 0.1	Nautical miles

Further examination of the algorithm involved in calculating the horizontal phase of flight and its input data reveals the need to address the various nuisance or uncontrollable factors. These include the aircraft type, aircraft speed, turn rate, navigation equipment, and others that may possibly impact the horizontal algorithm's detection of turns and straight events on the surveillance track positions. The study has no control over these nuisance factors and nor are they parameters in the algorithm. The objective is to determine the best levels to set the algorithm parameters to minimize the errors in detecting turns. The parameter settings and performance of the algorithm need to be robust over all these other nuisance factors. Ref. 10-12 all describe a special experimental design called the complete randomized block design. This experiment combines the previous discussion of a factorial experiment with a randomized block design. In this study, the blocks represent a flight, since all the nuisance factors may be grouped into this one special factor representing the flight. For all practical purposes, the full factorial (all 192 runs discussed above) will be performed completely on each flight. This focuses the experiment on the seven factors and their interactions and the large variability between flights is removed from the experiment. Each block of the experiment is a flight. The general model of this experiment is expressed in Eq. (9).

$$Y_{ijk} = \mu + \tau_i + \beta_j + \varepsilon_{k(ij)} \quad (9)$$

where Y_{ijk} is the dependent variable, μ is the overall mean, τ_i is the effect of the i^{th} treatment (effect of the controllable factors), β_j is the effect of the j^{th} block (in this study the flight effect), and $\varepsilon_{k(ij)}$ is the normally distributed random error term with an assumed 0 mean and equal variances

The treatment effect in the above model represents the full factorial of the seven factors under study. The full factorial yielding 192 runs contains all seven factors under study and all their interactions. For example, if the experiment included only three factors, $\tau_i = A_m + B_n + AB_{mn} + C_o + AC_{mo} + BC_{no} + ABC_{mno}$. You can see the main effects represented by single letters, two-way interactions by pairs, and the three-way interaction by three letters. For a seven factor factorial with one factor at 3-levels, the equation is too large to publish here, but has 192 terms. For simplification, Eq. (10) contains the seven factors and only their main and two-way interaction terms. For a full explanation of this model and the assumptions involved see Ref. 10 or 11.

$$\tau_i = A + B + C + D + E + F + G + AB + AC + AD + AE + AF + AG + BC + BD + BE + BF + BG + CD + CE + CF + CG + DE + DF + DG + EF + EG + FG + C' + AC' + BC' + DC' + EC' + FC' + GC' \quad (10)$$

where A is the first factor, B is the second factor up to G which is the last factor, C' represents the third factor at its third level

The model utilizes the truth reference data set described in Section III.B as input. The data set contained 57 aircraft flight segments; however seven of these flight segments had no turns at all. For the experiment, only the 50 remaining flight segments were used, referred to as 50 blocks for the experiment. This resulted in 9600 runs (50x192). Each run contains a flight segment and all its associated track positions, and the algorithm is implemented to calculate if the turns exist or not for each position. The parameter settings are determined by the levels defined by the particular run (e.g. $A = horzTimeWindow$ set to 55 seconds and all other factors at their lowest level). The metrics defined in Eq. (7) and Eq. (8) are applied to provide the response result, Y_{ijk} .

For this experiment, the algorithm was implemented in a Java program and a LINUX shell script was used to exercise the 9600 runs with results appended to a large ASCII file. After about three hours of run time on a dual 3.0 gigahertz processor computer, the results were then read by a commercial statistical software package** with two response variables, $P(MC)$ and $P(FC)$, using the model described in Eq. (9). The factor levels were explored at all their levels. The model matched 77% of the variability in the data for the $P(MC)$ response variable and 65% for $P(FC)$ response variable, where 100% would have completely modeled the variability between the response variables and the factors. The results for this model were determined to be acceptable.

The model results are illustrated in Fig. 14. The figure, referred to as the predictor profile by the software package used, shows the detected values of each factor over the interval of the experiment. The solid-black line in each chart in Fig. 14 shows the relationship the model predicts between the factor and the response variables. The dotted-red line represents the levels selected in the figure. The levels listed below the charts were selected to reduce the two response variables to their collective lowest value. For example, the $horzTimeWindow$ factor in the first set of charts in the figure exhibits a steep negative slope for $P(MC)$ and a much more shallow positive slope for $P(FC)$ response variable. The impact of the steep negative slope dominates, so the suggested level is 55 seconds, the high value for this factor. For the recommended levels shown in Fig. 14 and all 50 flights included in the experiment, the average $P(MC)$ and $P(FC)$ were respectively, 0.06 and 0.08.

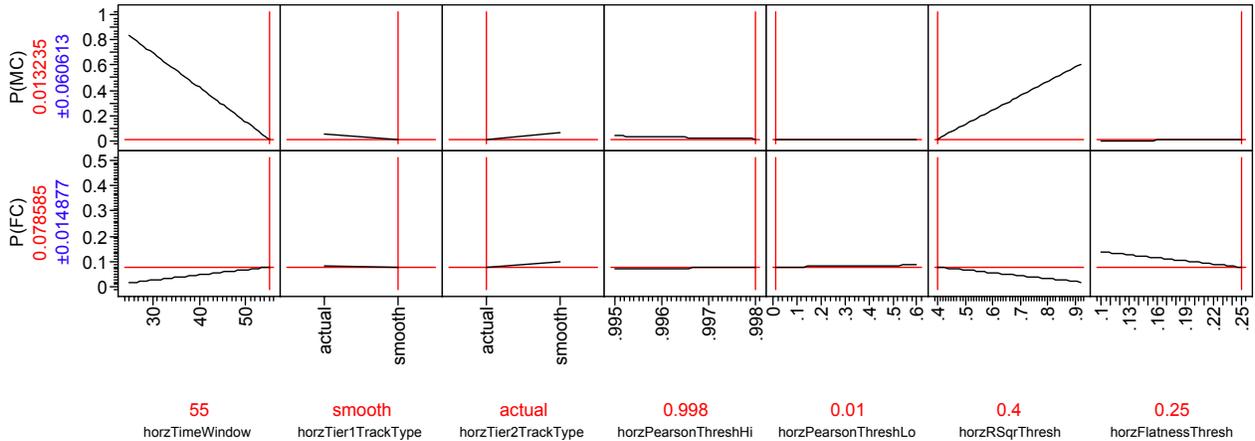


Figure 14: Predicted Model Results for the Horizontal Algorithm

** The authors used JMP developed by the SAS Institute for all statistical calculations, see www.jmp.com for details.

2. Experimental Design and Analysis of the Vertical Algorithm

The Vertical Phase of Flight Detection Algorithm has a potential of four controllable factors listed in Table 2 as the parameters in the algorithm. Like the horizontal algorithm, the time window parameters, named *vertTimeWindowAhead* and *vertTimeWindowBack*, refer to how much time in seconds ahead and behind the current position, respectively, that the algorithm samples for its calculation of phase of flight. To simplify the experiment, these two parameters were set equal to each other and treated as one. For the rest of the paper, this factor will be referred to as the *vertTimeWindow*. Therefore, the experiment was reduced to an analysis of the three factors, which are listed in Table 6.

Table 6. Vertical Algorithm Experimental Factors

Factor	Levels Implemented	Units
<i>vertTimeWindow</i>	25, 55, 120	Seconds
<i>vertTrackType</i>	actual, smooth	-
<i>vertSlopeThresh</i>	0.5, 1.5, 2.5	Feet per second

Using the lessons learned in the horizontal algorithm's analysis, the same model was applied as in Eq. (9). For the vertical algorithm only three factors are in the factorial of the model, but two have 3 levels as listed in Table 6. Therefore, the factorial part of the experiment results in 18 runs (3x2x3) per flight block. This is illustrated in Table 7. Table 7 lists all the combinations of factors and their levels. These 18 runs would then be performed on each block or flight in the data set. Similar to the horizontal algorithm experiment, the full 57 flights had 15 flights that exhibited no climbs or descents during the flight segments recorded. Therefore, the data set was reduced to 42 flights. From the 42 flights and 18 runs per flight, 756 runs were performed with the Vertical Phase of Flight Detection Algorithm. The model matched 89% of the variability in the data for the $P(MC)$ response variable and 66% for $P(FC)$ response variable.

Table 7: Vertical Algorithm Factor Levels

<i>vertTimeWindow</i>	<i>vertTrackType</i>	<i>vertSlopeThresh</i>
25	actual	0.5
25	actual	1.5
25	actual	2.5
55	actual	0.5
55	actual	1.5
55	actual	2.5
120	actual	0.5
120	actual	1.5
120	actual	2.5
25	smooth	0.5
25	smooth	1.5
25	smooth	2.5
55	smooth	0.5
55	smooth	1.5
55	smooth	2.5
120	smooth	0.5
120	smooth	1.5
120	smooth	2.5

Like the horizontal algorithm’s experiment, a Java program implemented the vertical algorithm and LINUX shell script ran the 756 runs. The results were input into the same commercial software package referred to in Section III.C.2 and all the levels ran were explored. The analysis produced the following Fig. 15. The selected levels illustrated provide the most favorable outcome with an average 0.03 for both $P(MC)$ and $P(FC)$ response metrics for the entire 42 flights selected for the experiment.

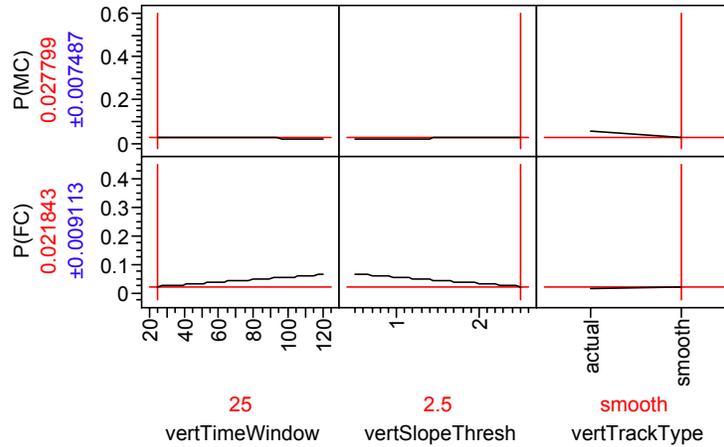


Figure 15. Predicted Model Results for the Vertical Algorithm

IV. Comparison to the Legacy Algorithm

As discussed in the Introduction in Section I, a legacy algorithm existed in the FAA prior to the work presented in this paper. It was first developed in 1999 and described in Ref. 4. The algorithm was simple in its construction utilizing only the post-smoothed track positions and one track position before and after the current point (i.e. about 12 seconds of a $horzTimeWindow$ or $vertTimeWindow$). Furthermore, the authors’ observations over the years using this legacy algorithm indicated it lacked the desired performance. Now with the availability of the truth reference data set presented in Section III.B and application of the metrics defined in Section III.A, the legacy algorithm can be evaluated and compared to the new algorithms presented in this paper. This analysis was performed on the full data set’s 57 flights, regardless of whether some flights had no turns or vertical transitions.

The comparative results are summarized in Table 8. The results are noteworthy. For the horizontal detection algorithm, a 0.134 reduction in the missed call detection errors, $P(MC)$, and a modest increase in the false call detection errors, $P(FC)$, of 0.024. For the vertical detection algorithm, a 0.033 reduction in the missed call detection errors, $P(MC)$, and a modest increase in the false call detection errors, $P(FC)$, of 0.012. Since both the missed and false detection errors have an inversely proportional relationship, it is not surprising that one is reduced and the other increased. However, the trade-off was very favorable in reducing the high missed detection probabilities of the legacy algorithms with a very modest gain on the false detection side. It is further hypothesized that improvements in the post-smoothing processing of the track positions may further improve this result. This will be left for future study.

Table 8. Comparison of legacy algorithm with new algorithm

	Legacy Algorithm		New Algorithm		Effect	
	Horizontal Turns	Vertical Transitions	Horizontal Turns	Vertical Transitions	Horizontal Turns	Vertical Transitions
$P(MC)$	0.203	0.081	0.069	0.048	-0.134	-0.033
$P(FC)$	0.055	0.038	0.079	0.050	0.024	0.012

V. Conclusion

FAA development of decision support tools with embedded trajectory predictors and improvements of many FAA automation systems require the analysis of aircraft trajectories. These trajectories include different phase of flight states of the aircraft in both the horizontal and vertical dimensions. The events include straight and turning states in the horizontal dimension and vertical transitions states, such as climbing, descending, and level, in the vertical dimension. A legacy algorithm was used since 1999 but lacked the desired performance. This paper presents the development of a set of algorithms for post analysis of aircraft surveillance trajectories or track that detect these events. The algorithms and their parameters are presented and incorporated into a set of designed experiments. To accomplish this, a set of metrics were first defined in Section III.A that measure the probability of missing a detection of a turn or vertical transition and the probability of falsely detecting these events. A truth reference data set was collected and analyzed for the aircraft's true turns and vertical transitions, utilizing highly precise GPS positional data with a frequency of one second. This allowed the application of the miss and false detection metrics defined in Eq. (7) and Eq. (8). It also provided the means to perform the designed experiments described in Section III.C, which in turn produced the optimal settings of the parameters for the new algorithms. The end result was the new algorithm detected both turns and vertical transitions with significantly less missed detection rates and very modest increases in the false detection rates. Overall, the new algorithms improved over the legacy methods, but even more importantly the methods, metrics, and truth reference data set presented can continue to be used as these algorithms are improved in the future.

One final note - these same methods and data may be shared with others to evaluate their algorithms in the same manner described in this paper. If the reader has this interest, please contact the FAA author with the request.

References

- ¹Gong, C. and McNally, D., "A Methodology for Automated Trajectory Prediction Analysis," American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference, Providence, RI, August, 2004.
- ²Arthur, W. and McLaughlin, M., "User Request Evaluation Tool (URET) Interfacility Conflict Probe Performance Assessment," 2nd USA/EUROPE Air Traffic Management R&D Seminar, Orlando, Florida, December 1-4, 1998.
- ³Cale, Mary Lee, Liu, Shurong, Oaks, Robert, Paglione, Mike, Ryan, Dr. Hollis F., Summerill, Scott, "A Generic Sampling Technique For Measuring Aircraft Trajectory Prediction Accuracy," 4th USA/EUROPE Air Traffic Management R&D Seminar, Santa Fe, New Mexico, December 3-7, 2001.
- ⁴Paglione, Mike M, Ryan, Dr. Hollis F., Oaks, Robert D., Summerill, J. Scott, Cale, Mary Lee, "Trajectory Prediction Accuracy Report User Request Evaluation Tool (URET) / Center-TRACON Automation System (CTAS)," DOT/FAA/CT-TN99/10, Federal Aviation Administration, William J. Hughes Technical Center, May, 1999.
- ⁵Automation Metrics Test Working Group, "En Route Automation Modernization Automation Metrics and Preliminary Test Implementation Plan," Version 2.7, Federal Aviation Administration, William J. Hughes Technical Center, NJ, June, 2005.
- ⁶Ryan, Hollis F. and Paglione, Mike M., "Heuristic Methods to Post Process Aircraft Radar Track Data," American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference, Austin, TX, August, 2003.
- ⁷Devore, Jay L., *Probability and Statistics for Engineering and the Sciences, Fifth Edition*, Pacific Grove, CA, Duxbury, 2000.
- ⁸Merriam-Webster Online, <http://www.m-w.com>, Merriam-Webster, Inc., (cited January 27, 2006).
- ⁹Paglione, Mike M., and Ryan, Dr. Hollis F., "Comparison of Host Radar Positions to Global Positioning Satellite Positions," Digital Avionics Systems Conference (DASC), Washington D.C, November 1-3, 2005.
- ¹⁰Montgomery, Douglas C., *Design and Analysis of Experiments, Fourth Edition*, New York, NY, John Wiley & Sons, 1997.
- ¹¹Hicks, Charles R., *Fundamental Concepts in the Design of Experiments, Fourth Edition*, New York, NY, Saunders College Publishing, 1993.
- ¹²Box, George E. P., Hunter, J. Stuart, Hunter, William G., *Statistics for Experimenters, Design, Innovation, and Discovery, Second Edition*, Hoboken, NJ, John Wiley & Sons, 2005.