AIAA Guidance, Navigation, and Control Conference and Exhibit
21 - 24 August 2006, Keystone, Colorado

AIAA 2006-6057

Conflict Detection Using Variable Four-Dimensional Uncertainty Bounds to Control Missed Alerts

David A. Karr* and Robert A. Vivona †

L-3 Communications Corporation, Titan Group, 300 Concord Road, Billerica, MA 01821 USA

Decision-support tools for maintaining pairwise aircraft separation rely on conflict detection to alert the operator when the predicted trajectories of aircraft will result in a loss of separation. But aircraft frequently do not follow their predicted trajectories exactly. This can cause missed alerts and the failure of strategic separation procedures. We present a technique for modeling a bounded region of uncertainty around a four-dimensional predicted trajectory and an algorithm for detecting conflicts between trajectories modeled in this way that avoids missed alerts as long as the aircraft remain within the specified regions of uncertainty. In addition, we present an algorithm for detecting the intrusion of a trajectory modeled in this way into an area hazard modeled as a polygonal region. The size of the region of uncertainty can vary along the trajectory continually and independently in the along-path, cross-track, and vertical dimensions, providing an opportunity to reduce the likelihood of false alerts while protecting against typical prediction errors. The algorithm has been implemented in the Autonomous Operations Planner, a NASA Langley prototype decision support tool for airborne self-separation.

I. Principles of Conflict Detection

Conflict detection (CD) is an essential part of a decision support tool (DST) that supports separation of aircraft. The DST assigns a protection zone to each aircraft that it models with two parameters, a *required lateral separation* R and a *required vertical separation* H. A *loss of separation* (LOS) occurs when the following two conditions are true simultaneously: the lateral distance between two aircraft is less than R (a *lateral LOS* has occurred), and the difference between the altitudes of the two aircraft is less than H (a *vertical LOS* has occurred). If we define the protection zone around each aircraft as a cylinder of radius R and height $2H$, with the aircraft in the center of the cylinder, then LOS occurs when one aircraft is inside the other aircraft's protection zone.

In order to maintain separation of aircraft that are not already in LOS, the DST must compute, for each aircraft of interest, a predicted *trajectory* indicating the lateral and vertical positions of the aircraft at future times. The DST *detects a conflict* between two aircraft if the predicted position of one aircraft at some future time is within the protected zone around the predicted position of the other aircraft at that same future time. In other words, a conflict is a predicted future LOS. The DST may be required to display a *conflict alert* to an operator when the DST detects a conflict between the predicted trajectories of two aircraft so that the operator can initiate *conflict resolution* (that is, find a new flight plan for one or both aircraft that avoids the conflict).

A DST can be ground-based or airborne. An airborne DST is responsible only for the separation of the aircraft on which it is installed (the *ownship*) from other aircraft (*traffic*), and need only detect conflicts between the ownship and each known traffic aircraft. This comprises N pairs of aircraft if there are N known traffic aircraft. A ground-based system observing N aircraft in its airspace might need to consider $N(N-1)/2$ different pairs of aircraft (each possible subset of 2 aircraft chosen from among N). But in either case, we can express the CD problem for the entire DST as simply the problem of determining whether there is a future LOS between the first aircraft and the second aircraft in each pair in some list of pairs of aircraft

*Senior ATC Software Developer, AIAA Member.
†Research Engineer, AIAA Associate Fellow.

that are of interest. Usually the DST is responsible only for finding conflicts that will occur during a time interval of a certain duration (the *lookahead time* or time horizon) starting at the current time.

Naïvely, we might expect that a conflict exists between two trajectories if and only if there is some time at which the predicted positions of the two aircraft are in LOS. Experience shows, however, that discrepancies often arise between any feasible prediction of future aircraft positions and the positions that occur after time has passed.[1–4] A CD algorithm based on the naïve criterion might predict that two aircraft will pass each other without LOS, even though a LOS would occur before the lookahead time had elapsed if the aircraft were actually to continue their current flight plans without intervention. This is a *missed alert*, a false negative result of conflict detection; the LOS occurs (if it is allowed to occur) because at the closest approach one or both aircraft are not exactly where they were predicted to be. A missed alert does not inevitably lead to actual LOS, because the DST (or some other system) may detect the conflict when it is significantly closer than the lookahead time (enabling a more accurate prediction) but still far enough to plan and execute a *resolution* (a maneuver that avoids the LOS). But in any case, a missed alert reduces the usefulness of a DST for maintaining separation, because it can prevent the operator from receiving a conflict alert at the time when the operator should act upon the alert.

In order to prevent missed alerts, we would like conflict detection to account for the uncertainty in the predicted trajectory of each aircraft. In the approach described in this paper, in addition to the aircraft's position at any given time, the predicted trajectory used by the CD algorithm specifies *uncertainty bounds* within which the actual future aircraft positions might vary from the predicted positions. Taking this uncertainty into account, we no longer say that two trajectories are in conflict only if there *will* be a LOS between the two aircraft; instead, the trajectories are in conflict if there *can* be a LOS between the two aircraft when each aircraft is flying within the limits of its predicted trajectory.

Performing CD for this new definition of a conflict may increase the frequency of *false alarms*, that is, false positive results. A false alarm occurs when the algorithm predicts a possible LOS between two aircraft even though the aircraft would not actually have a LOS if they were allowed to follow their flight plans without intervention. A false alarm may incur the cost of unnecessary conflict resolution. But the cost and frequency of false alarms should be weighed against the cost and frequency of missed alerts.[4] Under certain reasonable assumptions, the cost of a missed alert is likely to be much higher than the cost of a single false alarm.

In the following sections we define the ways in which the actual four-dimensional trajectories of aircraft can deviate from predicted trajectories, that is, the types of prediction error. We describe a model of uncertainty, Prediction Uncertainty Bounds (PUBs), that is designed to enable CD to prevent missed alerts to the extent that is feasible without causing an excessive number of false alarms. The PUBs model does this by allowing the bounds of uncertainty around a predicted trajectory to vary in a suitable fashion in each dimension along different parts of the trajectory. We describe an algorithm, Prediction Uncertainty Bounds-based Conflict Detection (PUB-CD), designed to perform CD efficiently for two aircraft whose trajectories have been modeled with PUBs. In addition, we describe an efficient algorithm to detect if and when an aircraft whose trajectory is modeled with PUBs has a conflict with (that is, penetrates) an area hazard.

II. Prediction Errors

There are several types of discrepancy that can occur between predicted and actual states of aircraft. In general, we will not consider all the discrepancies that could occur due to changes in intent, that is, because a pilot initiated a new maneuver after a trajectory had been predicted; accounting for all such possibilities is outside the scope of this paper.

In the following paragraphs, we review the usual dimensions in which prediction error is measured. When measuring prediction error along an actual trajectory, the error at each point would be measured relative to one or more reference points, or corresponding points on the predicted trajectory. There are various ways of choosing these reference points; we postpone specifying our choice of reference points until we are able to justify it, that is, until we describe how multiple types of error may apply at a single point.

In addition to the nature and possible source of each type of error, we depict some effects the errors can have on conflict detection. In the figures, the protection zone appears as a circle in the horizontal plane and as a rectangle in any vertical plane.

A. Altitude Errors

An aircraft may be higher or lower at a point on the actual trajectory than at the corresponding point on the predicted trajectory. We refer to this type of discrepancy as an *altitude error*. It is measured in feet above or below the predicted altitude.

There are various causes for altitude errors. For example, the aircraft could start to lose altitude shortly before the predicted top of descent (TOD); the descent could occur at a non-uniform vertical rate that cannot be exactly computed from data available to the DST; or the aircraft could have departed significantly from its planned descent profile due to pilot or FMS control. A similar list of causes could affect altitude predictions during climb. Some variation in altitude is also to be expected during level flight. Figure 1 shows an example in which an altitude error in the vertical profile of aircraft B results in a loss of vertical separation from aircraft A.

B. Cross-Track Errors

The actual position of an aircraft at any time could be to the right or left of the corresponding point on the predicted trajectory. We refer to this type of discrepancy as a *cross-track error*. It is measured in nautical miles perpendicularly from the predicted path to the actual aircraft position.

Cross-track error could be caused by errors in predicting the effect of winds, by errors in modeling the radius of a turn, by turn overshoots, or by various other causes. Figure 2 shows a cross-track error in the path of aircraft B that results in a loss of lateral separation from aircraft A.

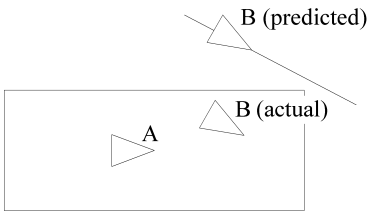


Figure 1. Altitude Error

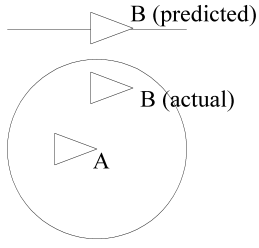


Figure 2. Cross-Track Error

C. Along-Path Errors

The actual position of the aircraft at any time might be ahead of or behind the position that had been predicted for that time. In general this corresponds to whether the aircraft had flown at a larger or smaller mean groundspeed (averaged over time) than was implied by the predicted trajectory. We call this type of error an *along-path error*.

Along-path error could be measured in nautical miles from the predicted position forward or backward to the actual aircraft position. Along-path error can also be measured in seconds from the estimated time of arrival (ETA) to the actual time of arrival at a given point. For example, the nominal trajectory for an aircraft starting at a certain point might predict that the aircraft would be 7 nautical miles from its starting point after 60 seconds. Suppose the aircraft flies faster than this prediction implies, so that after 60 seconds it has actually flown 8 nautical miles from its starting point. We could say that the aircraft is 1 nautical mile ahead of the predicted state at 60 seconds. Alternatively, we could say that the aircraft arrived 7.5 seconds early at the point 7 nautical miles from the starting point, since it took only 52.5 seconds to fly 7 nautical miles at the aircraft's actual ground speed.

The causes of along-path errors can include erroneous or incomplete predictions of wind and guidance system effects not modeled by the DST. In general, speed will be difficult to model accurately during climb or descent because the aircraft is maintaining constant computed airspeed or a constant Mach number, and groundspeed is therefore a function of altitude as well as wind.

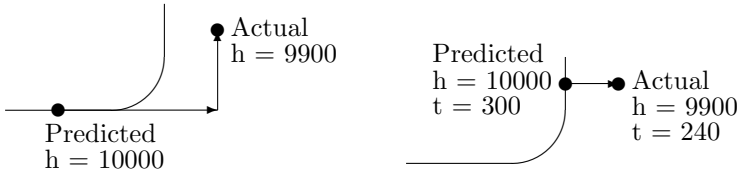
Along-path errors can result in the loss of either lateral or vertical separation, or both. Figure 3(a) shows an along-path error of aircraft B (actual is faster than predicted) that causes a lateral LOS with aircraft A. Figure 3(b) shows an along-path error of aircraft B that causes a vertical LOS with aircraft A.

(a) Error Affecting Lateral Separation (b) Error Affecting Vertical Separation

Figure 3. Along-Path Errors

D. Errors in Multiple Dimensions

It is of course possible for two of the errors we have described, or all three, to manifest themselves at a single point along the aircraft's actual trajectory. For example, at some time in the future the aircraft might be 0.5 nautical mile past its predicted position (an along-path error), 0.1 nautical mile to the right of the predicted lateral path (a cross-track error), and 500 feet above the predicted vertical path. But lateral turns result in ambiguities in these measurements. Consider the case in which an aircraft intending to make a 90-degree left-hand turn flies faster than predicted, but also flies outside the predicted turn (that is, to the right of the predicted path) and 100 feet lower than predicted. To quantify these errors, one could project a straight line forward from the predicted point to measure the along-path error in nautical miles and another line perpendicular to the first to measure the cross-track error, as in Figure 4(a). In this admittedly exaggerated example, this results in a "cross-track error" to the *left*. This counter-intuitive result arises because this technique did not account for the turn that the aircraft should have made after the predicted point. This problem is resolved by measuring the error at the actual point relative to the point that was predicted to occur at the minimum lateral distance from the actual point, not at the same elapsed time. Instead of an along-path error measurement that says the aircraft has flown farther than predicted, we have a measurement that says the aircraft passed this point sooner than predicted,[a] as shown in Figure 4(b).



(a) Using Straight-Line Distance (b) Relative to Abeam Point

Figure 4. Measuring Multiple Errors at One Point

In this paper, we define the multiple dimensions of uncertainty according to the error model shown in Figure 4(b), that is, in order to model simultaneous errors in multiple dimensions we measure the cross-track error by taking the lateral distance from the actual position to the *abeam point* on the predicted path (such that the direction from the abeam point to the actual position is perpendicular to the lateral path at the abeam point), and the altitude error by taking the vertical distance from the actual position to the abeam point. The along-path error is expressed by the difference in time between the actual time at which the aircraft reaches this position (that is, when it actually passes the abeam point) and the time when it was predicted to pass the abeam point. Using these definitions enables us to model all dimensions of the error at an actual trajectory point in a way that reflects actual error behavior with reference to just one point on the predicted trajectory (including the location, direction, and time of arrival at that point).

These definitions are based on the assumption that the actual trajectory will follow the predicted trajectory with sufficient accuracy, including turns, TOD, and other features in approximately the same locations, so that the abeam point will always be well-defined. For example, we assume that if an aircraft's intent is to

[a] In principle, one could still measure the along-path error in terms of distance rather than time, provided that the distance is measured along the predicted trajectory. For the purposes of the PUB-CD algorithm, however, it is convenient to measure this error in the time dimension.

turn at a waypoint, it will execute the turn at approximately the same place whether it arrives there much earlier than expected or much later.

III. Modeling Prediction Error

The PUB-CD algorithm requires *prediction uncertainty bounds* (PUBs) to be determined along all parts of the trajectories of all aircraft. It is assumed that the trajectory contains a sequence of specified *trajectory points* representing the predicted states of the aircraft, starting with the state from which the trajectory was predicted. Each trajectory point stores the latitude, longitude, and altitude of a point in the airspace, along with the estimated time of arrival and other predicted properties of the aircraft (such as groundspeed). The trajectory points should be spaced closely enough that it may be assumed that the aircraft moves linearly at a constant groundspeed between trajectory points. (The distance between trajectory points should be allowed to vary so that the points can be spaced closely when direction or speed are changing rapidly, and farther apart when direction or speed are constant or more slowly changing.)

The trajectory also stores data describing the *trajectory segment* between each consecutive pair of trajectory points, in particular, data describing the size and shape of the PUBs around that segment, represented in such a way that the bounds in each dimension can be specified independently and are either constant along the length of the segment or grow or shrink linearly from an initial value to a final value. The PUBs may be thought of as surrounding the trajectory of an aircraft in such a way as to form a four-dimensional "tube" (whose "cross-section" may vary continually in both size and shape) within which the aircraft is thought to be very likely to remain as it actually flies according to the intent on which the predicted trajectory was based.

In the following sections we describe three dimensions of prediction uncertainty bounds that we believe are sufficient to model aircraft trajectory uncertainty over a wide range of circumstances.

A. Altitude Bounds

The bounds of altitude error on each trajectory segment are described by the maximum altitude errors above and below the predicted trajectory, which are linear functions of the path distance along the segment. These functions are defined by the maximum possible error above the predicted altitude and the maximum possible error below the predicted altitude (in feet) at the first point of the segment, and the rates of growth of the bounds, one rate for the growth of the maximum error above the predicted altitude and one rate for the growth of the maximum error below (in units of feet per nautical mile of path traveled) as we traverse the segment from the start to the end. The altitude bounds do not need to be symmetric around the predicted trajectory.

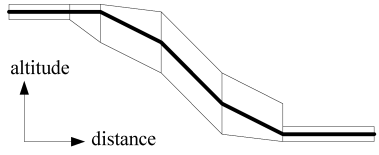


Figure 5. Altitude Bounds during Descent

Figure 5 shows how the altitude bounds might account for errors in modeling the vertical profile of a descent. In this example we assume that the accuracy of predictions is much better during level flight than during the descent. The vertical bounds are therefore shown as relatively tight during the initial level flight segment. Approaching the top of descent (TOD), the maximum error below the path grows in order to protect against the loss of altitude prior to the predicted TOD that is likely to occur in such circumstances.[b] The upper altitude bound remains close to the path just before the TOD; if the maximum above-path error were to grow along that segment, we might "detect" spurious conflicts with aircraft flying level above this path. As the aircraft descends, we gradually allow the maximum error above the path to grow until it reaches a maximum value. Finally, at the bottom of descent (BOD), the maximum error below the path shrinks to reflect the assumption that the aircraft will level off at the target altitude.

B. Cross-Track Bounds

The bounds of cross-track error on each trajectory segment are described by the maximum cross-track error at each point, which is a linear function of the path distance along the segment. This function is defined by

[b]This example assumes that the trajectory predictor modeled the TOD as an instantaneous change in vertical rate, as many systems do, and that the aircraft starts the maneuver soon enough to make a gradual transition onto the desired descent path.

the maximum cross-track error (in nautical miles) at the first point of the segment and by the rate of growth of the maximum cross-track error (in nautical miles per nautical mile of path traveled) as we traverse the segment from the start to the end. The cross-track bounds are symmetric to the left and right of the segment, and the displacement of the aircraft position due to cross-track error is assumed always to be perpendicular to the trajectory segment.

Figure 6 shows an example of how these bounds can be applied, in this case to account for errors in modeling the radius of a turn at a waypoint. The predicted path is shown as a heavy line, with a thinner line outlining the region enclosed between the cross-track bounds along each segment. In this example, the straight-flight segments before and after the turn have relatively narrow cross-track bounds; the region between the bounds grows during the first segment of the turn, remains wide during the segment in the middle of the turn, and shrinks again in the last segment of the turn.

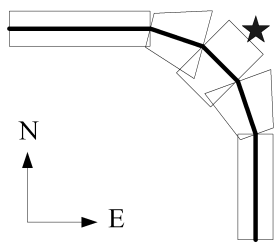


Figure 6. Cross-Track Bounds in a Turn

The gaps and overlaps in the regions covered by the cross-track bounds in Figure 6 are due to the perpendicular projection of cross-track error combined with the division of the trajectory into linear segments. At each trajectory point in a turn, the direction of the modeled cross-track error changes instantaneously. (The features in the figure are exaggerated to make them clearly visible.) This means that along a hypothetical actual path on the outside of the predicted turn, there would be short segments along which the cross-track error was undefined, so that LOS could not be detected at those points. Unless the LOS somehow occurred only during this short gap, however, it would be detected on the segments just before and/or after. As long as the required lateral separation is much greater than the width of the gaps, the gaps do not appear to have significant impact on conflict detection. (If the algorithm had to be applied in an environment where the gaps were significant, relatively straightforward extensions of the model and the algorithm would allow the gaps to be closed.)

C. Along-Path Bounds

The bounds for along-path error are the hardest to visualize, because they are modeled in a dimension that does not "sweep" physical space around the path as the cross-track and altitude bounds do. They can be represented graphically on a time-distance profile of the trajectory as shown in Figure 7. The heavy line in the figure plots the predicted elapsed time at each point along the path against the distance from the start of the trajectory. This function is linearly interpolated between the computed trajectory points, and each change in slope corresponds to a change in mean groundspeed; trajectory segments on which the aircraft flies faster are plotted with shallower slopes than segments on which the aircraft flies slower.

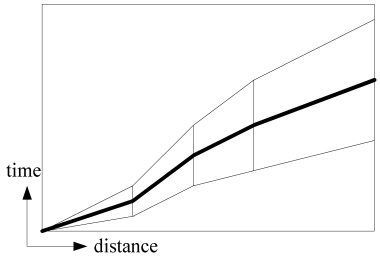


Figure 7. Along-Path or ETA Bounds

In a computed trajectory, along-path error is modeled by an uncertainty in the time of arrival at each point. Within any single segment of the trajectory, the maximum error is a linear function of the distance along the segment. In the example shown, the uncertainty of the time of arrival gradually increases along each segment, as indicated by the lighter lines above and below the predicted profile, which show (respectively) the slowest the aircraft might fly and the fastest it might fly. The along-path error is really a *speed* error, that is, it is the result of incorrectly estimating the slope of the time-distance profile. The bounds of this error need not grow uniformly on all segments, however; for example, the maximum error may grow much faster during a descent segment than during level flight, and the bounds might shrink as we approach a meter fix with a required time of arrival (RTA), on the assumption that the aircraft will somehow or other meet the RTA.

The bounds shown in Figure 7 are symmetric in the time dimension (above and below the predicted time-distance profile as shown) but are not symmetric in the spatial dimension. This is because the rate of growth of the maximum error changes not when a particular time passes but when the aircraft reaches a particular distance from its starting point. This is a consequence of the fact that we define all our uncertainty bounds (including this one) as functions of the distance traveled along the path. The justification for this is that the sources for along-path error are dependent primarily on such things as whether the aircraft is

level or descending, which are more closely tied to where the aircraft is along its path than to exactly how long it took the aircraft to get there. For example, if we believe that the uncertainty in speed is larger during descent than during level flight, then the effects of this increased uncertainty should start to appear as soon as the aircraft might be descending, that is, when the aircraft is sufficiently close to the TOD, and not sooner.

IV. Effects of Uncertainty on Conflict Detection

Before developing the algorithm for CD in the presence of uncertainty in the predicted trajectories, we review two previously implemented CD algorithms. We then examine some complications that arise when attempting to extend such algorithms to account for prediction uncertainty.

A. Conflict Detection in the Absence of Uncertainty

In the absence of prediction errors, or if we choose to ignore them or account for them simply by enlarging the protection zone of each aircraft, there are at least two fairly obvious and relatively simple ways to approach the problem of performing CD on two trajectories.

One approach is to find the lateral and vertical position of each aircraft at each time in a regular series of timesteps. For each simultaneous pair of positions, it is a simple matter to measure the lateral and vertical separation between them. The algorithm can simply continue stepping through both trajectories in parallel until it finds a single time when the separation between the aircraft is less than the required amount both laterally and vertically, that is, when the positions of the two aircraft are in LOS. If such an event occurs, the trajectories are in conflict; if the algorithm examines all the time steps from the initial state up to a sufficient lookahead time without finding a time when the aircraft are in LOS, the trajectories are considered not to be in conflict.

A slightly more sophisticated algorithm examines each of the two trajectories one segment at a time.[4] Each segment of the first aircraft's trajectory must be compared against any segment of the second aircraft's trajectory that overlaps its start and end times; only events during the time overlap between the segments (that is, the time period that they have in common) and before the expiration of the lookahead time need to be considered. The algorithm must then determine when the lateral and vertical separation requirements are violated. The trajectory segments must be short enough so that the motion of the aircraft may be modeled as linear between them (straight-line travel at constant flight path angle and groundspeed).

Two level-flight trajectory segments are easily determined either to satisfy or violate the vertical separation requirement during their entire time overlap. If either segment is climbing or descending, vertical separation may be lost and/or regained during the time overlap, but a simple set of linear equations with time as the only variable will tell when these events occur, that is, during what time period (if any) the aircraft violate the vertical separation requirement.

The handling of the lateral separation requirement is only slightly more complex. Since the distance between two points in a plane can be expressed as a quadratic function of Cartesian coordinates of the two points (and even on the surface of a spherical or spheroidal Earth, a quadratic function is an excellent approximation of the distance when the distance is not much greater than the required separation), and since the aircraft are assumed to move at constant speed along straight lines, the times when lateral separation is lost and regained are found by solving a quadratic equation with time as the variable; if the equation has no real solution, then separation is never lost.

If there are violations of each class of separation—lateral and vertical—between the two segments, it is then straightforward to determine (by comparing start and end times of the various events) whether the two aircraft are ever simultaneously in violation of both the lateral and vertical separation requirements. If so, the trajectories are in conflict. If the algorithm examines all segments up through the lookahead time without finding a conflict, it reports no conflict.

One difference between the two algorithms is that the time-step-based algorithm can "miss" a LOS that begins and ends between two consecutive timesteps; the larger the time step, the more likely it is that this might occur. The ability of the segment-by-segment algorithm to detect conflicts, in contrast, is not affected by the choice of a time step. The main attractiveness of the segment-by-segment algorithm, however, is that for trajectories that are likely to be encountered—that is, without excessive turns or changes in speed—this algorithm tends to require much less processing than the time-step-based algorithm.[4]

B. Complexity of Conflicts in the Presence of Uncertainty

The conflict detection algorithms described above can exploit the following convenient properties:

1. Lateral LOS can be detected completely independently of vertical LOS. If a lateral LOS and a vertical LOS are both found at time t, there is a conflict at time t.

2. The time and aircraft locations at first LOS (when the aircraft first enter into LOS) and at last LOS (when the LOS finally ends) are unambiguous and are easily found.

Neither of these features can be maintained once prediction errors, in particular along-path errors, are taken into account.

Figure 8 shows an example of why lateral and vertical LOS cannot be treated independently in the presence of along-path errors. In this example, which is a side view showing aircraft A flying level and aircraft B descending, we see that if aircraft B flies faster than predicted it will violate vertical separation with aircraft A, whereas if it flies slower than predicted it will violate lateral separation, and both of these violations occur with the same state of aircraft A. But it may be that the possible displacements of aircraft B relative to aircraft A throughout this encounter occur only along the line between these two extreme predictions, or to the right of that line. That is, when aircraft A reaches this position, it has a potential lateral LOS with aircraft B and a potential vertical LOS with aircraft B, but it cannot actually have complete LOS. In other words, it is not sufficient to detect when or where aircraft A can get into lateral LOS and when or where it can get into vertical LOS; we must examine the nature of the lateral and vertical behaviors in more detail to determine whether both types of LOS can occur for the same combination of prediction errors.

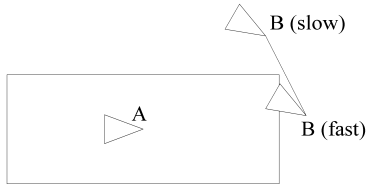


Figure 8. Interdependence of Lateral and Vertical LOS

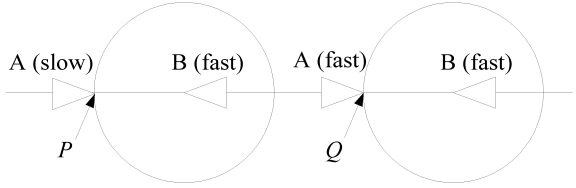


Figure 9. Ambiguity of First LOS

Figure 9 shows how the location and time of first LOS can be ambiguous. In this example, aircraft A and aircraft B are flying head-on along the same course at the same altitude. The figure shows two possibilities for how the first LOS might occur. In both cases, aircraft B is flying faster than predicted. If aircraft A flies slower than predicted, it violates lateral separation with aircraft B at the point labeled P, but if aircraft A flies faster than predicted, it closes the gap with aircraft B more quickly and violates lateral separation at the point labeled Q. The LOS at Q occurs not just in a different location from the LOS at P, but it is earlier in time as well due to the faster closing speed.

Each of these outcomes represents a first LOS for a particular combination of prediction errors, but a DST that indicates the times and/or locations of conflicts would typically be required to report a single time and/or location at which the first LOS of a conflict would occur. When prediction errors were not taken into account, LOS could occur in only one way. Among all the predicted states in which aircraft A could be in LOS, there was one unique state that was both the earliest possible state (fewest seconds elapsed since the starting state) and the first such state along the path (fewest nautical miles from the starting state). Now that we have accounted for prediction errors, we find that the LOS at Q occurs earlier in time than the LOS at P, but the LOS at P is fewer nautical miles from the starting state of aircraft A. A third possible LOS (not shown in the figure), occurring in a case where aircraft B flies slower than predicted, minimizes the distance of the LOS from the starting state of aircraft B. This leads to an ambiguity as to which of these possible LOS events is "first." If a "first LOS" is to be reported to the operator, the CD algorithm must somehow resolve or handle this ambiguity.

As we will show, however, these complications can be addressed by a variation of the segment-by-segment algorithm.

V. Detecting Conflicts between Uncertain Trajectories

This section outlines the PUB-CD algorithm for detecting conflicts between a pair of predicted aircraft trajectories in the presence of uncertainty

As explained at the start of this paper, the fundamental problem is to determine whether the trajectories of a given pair of aircraft are in conflict. We assume that predicted trajectories for both aircraft have been created prior to the start of the PUB-CD algorithm. We assume that the data of these trajectories conforms to the previously-stated model of trajectories and prediction uncertainty, that is, that each trajectory is a four-dimensional path passing through a finite sequence of specified trajectory points; that we can reconstruct predicted states along the trajectory segment between each pair of trajectory points with sufficient accuracy by means of linear interpolation;[c] and that each segment has been surrounded by prediction uncertainty bounds, where the size of the PUBs in each dimension at each predicted point is either a constant or linear function of distance along the segment. The trajectory must already be subdivided into segments that justify these assumptions; for example, segments of 50 or 100 nautical miles may be suitable during straight level flight, but a turn may need to be divided into several much shorter segments, and a segment in climb or descent may be limited to a few thousand or even a few hundred feet gain or loss of altitude. If the motion of the aircraft is expected to differ somewhat from true linear motion over a segment (for example, to allow a turn to be modeled without requiring an excessive number of trajectory points along the turn arc), the PUBs should be made at least large enough to contain the expected curved path in four-dimensional space.

Each trajectory can therefore be considered as simply a sequence of linear trajectory segments. A conflict between these two trajectories can be detected only if it is possible to identify at least one point on one segment of the first aircraft's trajectory, and one point on one segment of the second aircraft's trajectory, such that the aircraft as they fly past those points can be in LOS, taking into account all possible prediction errors within the uncertainty bounds. If such a pair of points exists, we say the two trajectory segments have a LOS. That is, any conflict between the trajectories can be described in terms of LOS between individual pairs of trajectory segments.

The CD algorithm must iterate through the pairs of segments where a conflict might occur. Figure 10 illustrates a simplified version of the control structure used by PUB-CD while searching for a conflict between two trajectories. For the purpose of determining time overlap, the earliest possible starting time and latest ending time, as defined by the along-path bounds, are used instead of the predicted start and end times. (The use of the time overlap as a condition of the inner loop is not strictly necessary for correctness of the algorithm, but is a useful optimization.)

```
For each segment ℓ1 of trajectory 1, while no LOS
is found:
    For each segment ℓ2 of trajectory 2 that can
    overlap the time interval of segment ℓ1, while
    no LOS is found:
        Test for LOS between segments ℓ1 and ℓ2.
If LOS was found:
    Search additional segment pairs to find first
    and last LOS.
```

Figure 10. Probing Trajectories for Conflict

Once a LOS is found between two segments, the inner and outer loops end. The next two sections of this paper describe how the PUB-CD algorithm determines whether there is a LOS between two trajectory segments and why the algorithm searches for the first and last LOS on additional pairs of segments if it finds a LOS.

VI. Detecting Conflicts between Trajectory Segments

We now describe the algorithm for detecting conflicts between a single ownship trajectory segment and a traffic trajectory segment.

In principle, we might approach the question of prediction uncertainty as follows. Consider any point along a segment of a predicted trajectory. We expect the aircraft to fly past that point during its actual flight, but not necessarily through that exact predicted location at the predicted altitude at the predicted time. That is, the aircraft will eventually reach a state that is laterally at or abeam the predicted point, but

[c] This is equivalent to the assumption that the trajectory points are spaced closely enough so that we can assume, without excessive loss of accuracy, that the aircraft travels at constant ground speed in a straight line, either in level flight or at a constant vertical path angle, between any two consecutive trajectory points.

it may be displaced to one side of the predicted location (cross-track error), above or below the predicted location (altitude error), and it may reach that state earlier or later than the predicted time (along-path error). In this sense the predicted state is surrounded by a three-dimensional region of uncertainty spanned by the cross-track, altitude, and time dimensions.

As the predicted positions of the aircraft travel along a trajectory segment from one end to the other, the regions of uncertainty around the predicted positions sweep out a four-dimensional shape that is the region of uncertainty around the entire segment. We must test the separation between this region and the corresponding uncertainty region around a segment of another aircraft's trajectory as follows: if there are any two points, one in each segment's uncertainty region, for which the time difference is zero, the vertical difference is less than the required altitude separation, and the lateral difference is less than the required lateral separation, then the two aircraft could potentially be in LOS while they are each within the bounds of their predicted trajectories, and therefore there is a conflict.

This four-dimensional model can be very complex and difficult to visualize. Fortunately, we can apply a suitable abstraction to reduce the problem to a simpler two-dimensional problem for which we can implement a correct and efficient analytical solution.

A. Reducing Conflict Detection from Four Dimensions to Two

In order to simplify conflict detection, when examining any possible state of an aircraft within the uncertainty region around a four-dimensional trajectory segment we will always refer to the state's *along-path coordinate*. The along-path coordinate is simply the lateral path length s from the start of the segment to the abeam point of the actual state, that is, to the point on the predicted path closest to the actual position. Path length is the independent variable on which all other variables depend (including predicted values and error bounds). For the purpose of identifying points along a single trajectory segment, the variable s ranges from 0 to the length of the segment.

To detect a conflict between the two trajectory segments, then, we are looking for predicted states at along-path coordinate s_1 in the first aircraft's trajectory segment and along-path coordinate s_2 in the second aircraft's trajectory segment, such that it is possible for the two aircraft to fly past those two along-path coordinates

- at the same time;
- at altitudes that violate vertical separation; and
- at lateral positions that violate lateral separation.

If all three of these conditions are satisfied, we say that the along-path coordinates (s_1, s_2) are in LOS.

We can therefore reduce the conflict-detection problem to the problem of finding a region in the s_1–s_2 *plane*, a two-dimensional Cartesian plane with coordinate axes for s_1 and s_2. The set of all along-path coordinates (s_1, s_2) that are in LOS is a set of points in the s_1–s_2 plane; this set is the *conflict region* of the two trajectory segments. Since the s_1–s_2 plane is an abstract space, in contrast to the four-dimensional physical space in which the trajectories and uncertainties are modeled, the conflict region in this plane does not necessarily have an exact analogue in this physical space. The usefulness of the abstract conflict region is that if two trajectory segments are not in conflict, their conflict region is the empty set.

In order to determine whether the two trajectory segments are in conflict, we apply a sequence of *conflict bounds* to the s_1–s_2 plane. Each conflict bound describes a part of this abstract plane where no LOS can possibly occur due to geometry or some other relation between the trajectory segments. Once the regions defined by all possible conflict bounds have been eliminated (that is, once we have ruled out the points (s_1, s_2) defined by all known conditions that imply separation between the aircraft at those along-path coordinates), whatever part of the s_1–s_2 plane that remains (if any) is the conflict region, that is, there is no way to rule out the possibility of a LOS at any pair of along-path coordinates within this region.

B. Conflict Bounds Imposed by the Ends of the Segments

During the calculation of the various bounds of the conflict region, the CD algorithm applies various mathematical formulas that were derived on the assumption that s_1 and s_2 can be any real numbers, that is, the s_1–s_2 planeis infinite. That is, we use a mathematical model in which each trajectory segment is extended to an infinite line. By definition, however, we can have a LOS between the two trajectory segments at

the positions (s_1, s_2) only when each aircraft is actually on its proper trajectory segment, that is, when $0 \leq s_1 \leq L_1$ and $0 \leq s_2 \leq L_2$ where L_1 and L_2 are the lengths of the corresponding segments. These inequalities specify four linear bounds in the s_1–s_2 plane that any points in LOS must satisfy. There may be many pairs of coordinates (s_1, s_2) that are in LOS, but they can occur only within the rectangular region shown in Figure 11.

C. Conflict Bounds Imposed by Time

Some pairs of coordinates (s_1, s_2) may be ruled out by considering times of arrival at those coordinates, taking into account estimated time of arrival and the maximum possible prediction error. For each s_1 and for each s_2 there is an earliest and latest possible time of arrival, defining a time interval during which the relevant aircraft can be at that along-path coordinate; the two aircraft can be at (s_1, s_2) at the same time only if the time intervals of s_1 and s_2 overlap. An example of conflict bounds that can result from this condition is illustrated by Figure 11.

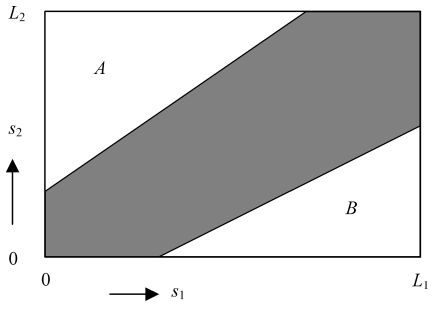


Figure 11. Conflict Bounds Imposed by Segment Length and by Times of Arrival.

At any (s_1, s_2) in the unshaded region A in Figure 11, aircraft 1 will have flown past s_1 before aircraft 2 reaches s_2, even if aircraft 1 is flying as slow as possible (within the uncertainty bounds) and aircraft 2 is flying as fast as possible. Since the aircraft cannot be at these locations at the same time (a precondition for LOS) no LOS can occur there. The boundary between these points and the shaded region consists of the points (s_1, s_2) for which aircraft 2 (at its fastest) just reaches s_2 at the same time that aircraft 1 (at its slowest) is passing s_1. Conflict is ruled out in the unshaded region B for similar reasons, except that in this case it is aircraft 1 that must fly its fastest to reach s_1 just as aircraft 2 (at its slowest) is passing s_2. Any combination of (s_1, s_2) in LOS can occur only in the shaded region between the two triangles.

In general, let a_1 denote the latest possible time when aircraft 1 can arrive at coordinate s_1, and let b_2 denote the earliest possible time when aircraft 2 can arrive at coordinate s_2. Then one precondition of LOS is that $a_1 \geq b_2$, that is, aircraft 1 can arrive at s_1 before aircraft 2 passes s_2. By assumption, a_1 is a linear increasing function of s_1 and b_2 is a linear increasing function of s_2. It follows mathematically that the coordinates (s_1, s_2) for which $a_1 = b_2$ lie along a straight line with positive slope, and that LOS can occur only on or below this line. A similar derivation shows that the locations reached when aircraft 1 is flying its fastest and aircraft 2 is flying its slowest also lie along a straight line with positive slope, and that LOS can occur only above this line.

Where these lines enter and exit the rectangle, and even whether they intersect it at all, depends on the relative predicted times at which the two aircraft enter their segments as well as on the bounds of uncertainty of the time at each point along each segment. In any case, due to well-known facts about linear inequalities over the coordinates of a Cartesian plane, if any points in the s_1–s_2 plane satisfy all six of the linear bounds found so far, these points form the interior of a convex polygon. This polygon contains the conflict region. If no points satisfy all six linear bounds, no LOS is possible and the algorithm on this pair of segments can terminate.

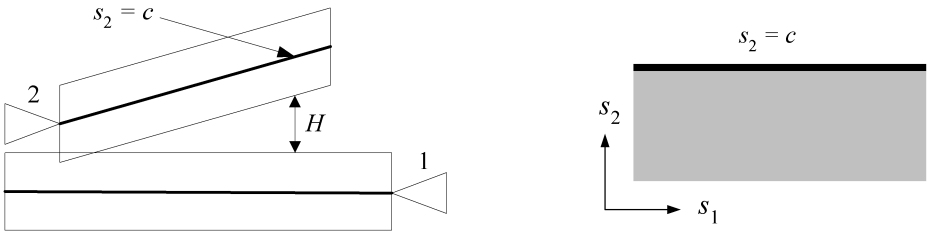
D. Conflict Bounds Imposed by Vertical Separation

A vertical LOS may not occur at all combinations of (s_1, s_2) on the two trajectory segments. If both aircraft are level, then of course either all (s_1, s_2) are in vertical LOS or none are. If either aircraft is climbing or descending, then vertical LOS may occur at some coordinates (s_1, s_2) and not at others.

Let A_1 be the highest possible altitude of the first aircraft when it is at along-path coordinate s_1. This can be computed by adding the maximum above-path altitude error at s_1 to the predicted altitude. Let B_2 be the lowest possible altitude of the second aircraft when it is at s_2. This can be computed by subtracting the maximum below-path altitude error from the predicted altitude. Each of these altitudes can be increasing, decreasing, or unchanging from one end of the aircraft's trajectory segment to the other end, but we have assumed the rate of change is constant. That is, A_1 has a constant derivative with respect to s_1, and B_2 has a constant derivative with respect to s_2. As a result, the condition $B_2 = A_1 + H$, where H is the required

vertical separation, defines a relation between s_1 and s_2 that can be plotted as an infinite straight line in the s_1–s_2 plane. This line divides the plane into two half-planes; the half-plane consisting of all the points for which $B_2 < A_1 + H$ is the set of points "inside" the bound defined by A_1 and B_2. At any point "outside" the bound, $B_2 \geq A_1 + H$, which implies that no conflict can occur at that combination of (s_1, s_2).

For example, consider the trajectory segments depicted in Figure 12(a). Aircraft 1 is on a level-flight segment, and aircraft 2 is on a climbing segment, at the beginning of which aircraft 2 has lost vertical separation with aircraft 1. For this example we assume the altitude bound above aircraft 1 is constant over the length of its segment, so that the position of aircraft 1 along its segment has no effect on whether there is a vertical LOS. The vertical LOS therefore persists until aircraft 2 reaches the along-path coordinate at which $B_2 = A_1 + H$. Let $s_2 = c$ at this point. Then the aircraft must be out of LOS whenever $s_2 > c$, that is, at any point in the unshaded region above the line $s_2 = c$ in Figure 12(b), whereas at any point in the shaded region below that line, the aircraft would be in vertical LOS.



(a) Physical Model (b) Conflict Bound

Figure 12. A Conflict Bound Determined by Altitudes and Vertical Rates

Similar reasoning applies to the relationship between the highest possible altitude A_2 of the second aircraft and the lowest possible altitude B_1 of the first aircraft. This can produce another bound of the conflict region.

Any bounds derived from these altitude comparisons are superimposed on the linear bounds determined by the segment ends and by time of arrival, resulting in a region that is the intersection of the regions "inside" each of the bounds. If this intersection region is empty, no conflict is possible, and the algorithm terminates. If the intersection region is not empty, it consists of a convex polygon in the s_1–s_2 planeand all the points inside the polygon. At any point (s_1, s_2) inside this polygon, s_1 and s_2 are the positions of predicted points along the trajectory segments of aircraft 1 and 2, respectively, which the two aircraft can reach at the same time (within the bounds of along-path error) and at which the vertical separation between the aircraft (within the bounds of altitude error) can be less than the required separation. The algorithm must still determine whether the aircraft can be in lateral LOS at any of these points.

E. Linear Conflict Bounds Imposed by Lateral Separation

We now begin to look at the conditions of lateral separation. The set of possible lateral position of the first aircraft, for the along-path position s_1, is a line segment that lies perpendicularly across the predicted trajectory segment. This "cross-track" line segment includes all locations that result from combining that along-path coordinate with any cross-track error from the maximum error on the left side of the path to the maximum error on the right side. The possible lateral positions of the second aircraft for the along-path position s_2 define another line segment in a similar fashion. A lateral LOS occurs at (s_1, s_2) only if the distance between the closest points on the two "cross-track" segments is less than the required separation.

In the first stage of bounding conflicts by lateral separation conditions, we allow either aircraft to have a cross-track displacement from its predicted path, but we place bounds on this cross-track error for only one aircraft. The other aircraft is (for now) assumed to be capable of an arbitrarily large cross-track error. This enables us to enforce some of the restrictions on the conflict region due to lateral LOS; the rest will come later, when we limit the cross-track errors of both aircraft simultaneously.

Figure 13 shows the geometry of lateral conflicts of this type. The predicted trajectory segments for the two aircraft are shown by heavy lines. In this example we bound the cross-track error only for aircraft 1. These bounds are indicated by a pair of thin lines bracketing the predicted path of aircraft 1. We assume that all parts of this figure lie in a single geometric plane. In applications in which trajectories are defined

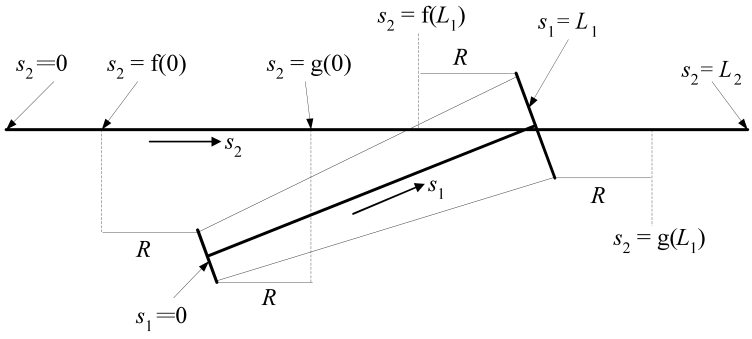


Figure 13. Limits of LOS at the extreme cross-track displacement of one aircraft.

along the surface of a spherical model of the Earth, this assumption can be enforced by projecting both trajectory segments onto a common plane.[d]

Now consider the cross-track segment across the trajectory segment of aircraft 1 at an arbitrary along-path position s_1. One such segment is shown in Figure 13 at $s_1 = 0$ and another at $s_1 = L_1$. The ends of such a segment define two limiting points along the trajectory segment of aircraft 2: these points, at $s_2 = f(s_1)$ and $s_2 = g(s_1)$, are respectively the first and last along-path coordinates of aircraft 2 that can be in lateral LOS when aircraft 1 is at s_1, assuming aircraft 2 can have an arbitrarily large cross-track error. These values of s_2 are found by adding or subtracting the required lateral separation, R, from the along-path coordinates of the two ends of the cross-track segment of aircraft 1 as illustrated in the figure.

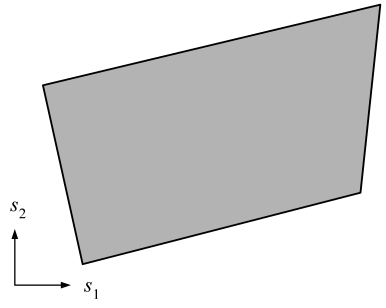


Figure 14. Linear bounds of lateral LOS

It can be shown mathematically that $f(s_1)$ is a certain easily-computed linear function of s_1, and similarly for $g(s_1)$. These functions therefore define two linear bounds of the conflict region in the s_1–s_2 plane. One of these bounds states that lateral LOS cannot occur if $s_2 < f(s_1)$, and the other states that lateral LOS cannot occur if $s_2 > g(s_1)$. Two more linear conflict bounds are defined by bounding the cross-track error of the second aircraft while allowing the cross-track error of the first aircraft to be arbitrarily large.

The set of points that simultaneously satisfy all four linear bounds of lateral LOS typically form a quadrilateral region, for example the shaded region in Figure 14. This quadrilateral is not necessarily contained in the rectangle shown in Figure 11, that is, some or all of the values of (s_1, s_2) that satisfy these four new bounds may not correspond to points on both trajectory segments. In certain special cases, such as when the predicted paths are parallel, these bounds may instead define a band between two lines in the s_1–s_2 plane. But in any case, the result of combining these bounds with the other bounds of conflict is the region in the s_1–s_2 plane which is the intersection of the regions "inside" each set of bounds. This region is either a convex polygon or the empty set.

F. Quadratic Conflict Bounds Imposed by Lateral Separation

If the conflict bounds we have determined so far have not already eliminated any possible LOS, they are still insufficient to determine whether LOS can occur. For example, if the lower left corner of the shaded quadrilateral in Figure 14 is the point (s_1, s_2), we know that if we proceed far enough in the cross-track direction from the predicted position s_1 of aircraft 1, eventually we will come to a point P that is within distance R from the cross-track segment of aircraft 2 (that is, the segment within the cross-track bounds) at the predicted position s_2. In a typical crossing scenario, however, to reach such a point P we would have to go beyond cross-track bounds of aircraft 1; if we restrict the cross-track error at s_1 and s_2 to stay within the

[d]The resulting distortion of the geometry is negligible when the segments are close enough laterally for there to be any possibity of conflict; for example, if the segments are limited to a length of 50 or 100 nautical miles, the discrepancy in lateral distances between the spherical model and the planar model is on the order of inches in every case that matters.

bounds, the closest reachable points will be separated by a distance greater than R, and so the coordinates s_1 and s_2 cannot actually be in lateral LOS.

In order to account for all the conditions imposed by the cross-track bounds, we consider the lateral LOS that can occur when each aircraft follows an actual course at one extreme limit of cross-track error (leftmost or rightmost), that is, when each aircraft travels along one of its cross-track bounds.

Figure 15 illustrates one such way in which a lateral LOS can arise. In the figure, the predicted trajectory segments are shown by heavy lines; the range of cross-track errors at each end of each segment is also shown by a heavy line that crosses the end of the segment, that is, by line segments that show all possible starting and ending positions when cross-track error is taken into account. The thin lines connecting the ends of the cross-track segments show the cross-track bounds.

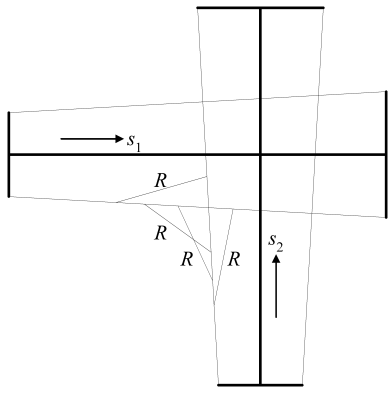


Figure 15. LOS at extreme cross-track displacement

Also shown in Figure 15 are several pairs of points along the right-hand cross-track bound of the first aircraft and the left-hand cross-track bound of the second aircraft, each pair at a distance of exactly R (the required lateral separation radius) from each other. If we set up Cartesian coordinates in the geometric plane in which this figure lies, then the distance between any two points is a quadratic function of the Cartesian coordinates of both points. But it can be shown that the Cartesian coordinates of the points along the cross-track bounds of each trajectory are linear functions of the along-path coordinates (s_1 or s_2) of those points, and that the distance between a point on the right or left bound of one segment and a point on the right or left bound of the other segment is a quadratic function of the two points' along-path coordinates, s_1 and s_2. Therefore, for each pair of points shown in Figure 15 (and for every other pair like them), the along-path coordinates (s_1, s_2) at those points satisfy a quadratic equation.[5]

In certain special cases the solution of the quadratic equation is the set of coordinates of two parallel lines, or there is no solution. These cases arise when a cross-track bound of one segment is parallel to a cross-track bound of the other segment. In such cases additional logic is required to determine whether the quadratic equation limits the possible lateral LOS; if so, it either imposes new linear bounds on the conflict region (handled the same way as previous linear bounds) or it eliminates all possibility of conflict between these two segments.

In every other case, for example in the case illustrated in Figure 15, the quadratic equation describes an ellipse in the s_1–s_2 plane. It can be shown that this ellipse lies entirely within the four linear bounds of lateral LOS in the s_1–s_2 plane and that it is tangent to two of these linear bounds.

The ellipse corresponding to the pairs of points shown in Figure 15 is in the lower left corner of the quadrilateral defined by the four linear bounds of conflict. If the first aircraft were crossing the other's trajectory from right to left instead of from left to right, the ellipse would have been in the upper right corner of the quadrilateral. But in either case, there is an ellipse for each of the four possible combinations of leftmost or rightmost cross-track positions of each aircraft, one ellipse in each corner of the quadrilateral determined by the linear bounds of lateral conflict, as shown in Figure 16.

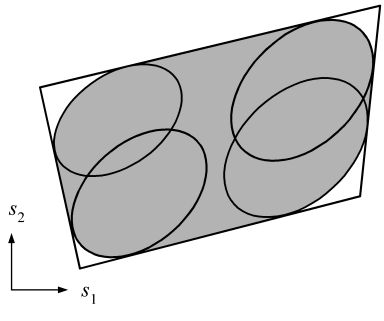


Figure 16. Bounds of all possible lateral LOS

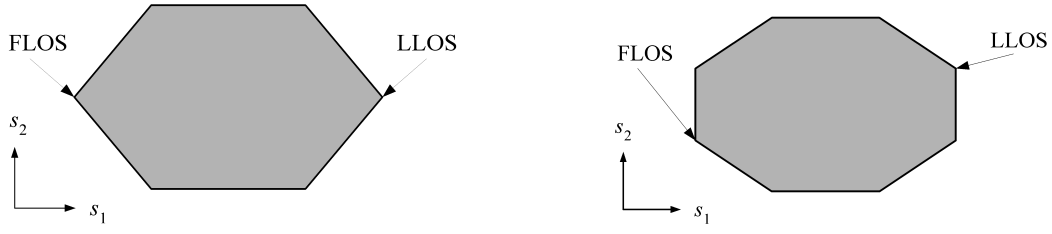
These four ellipses, in fact, are simply limiting cases of the sets of LOS points that occur when we restrict each aircraft to travel along a straight line somewhere in the region between the cross-track bounds of its trajectory segment. In any such scenario, the points in the s_1–s_2 plane that are in LOS form a region bounded by an ellipse (except in the special cases already described). Since every LOS must necessarily satisfy at least the linear bounds already found, each such ellipse must be contained within the quadrilateral in Figure 15. By varying the tracks of the aircraft left or right between their cross-track bounds, it is possible to reach points in lateral LOS anywhere in the shaded region in Figure 16, but it is not possible to have a LOS in any of the unshaded regions in the corners. The lateral bounds of LOS therefore restrict the conflict region to lie within the shaded area of the figure. The calculations determining the ellipses, and the various

special cases that apply to the bounds of lateral conflict, have been described in more detail in an earlier report.[5]

The actual conflict region for the two trajectory segments is the intersection of the shaded region in Figure 16 with the convex polygon determined by the linear bounds previously determined. The result is the conflict region for the two trajectory segments: either the empty set, indicating that all LOS has been ruled out, or a convex figure, typically with some straight edges and possibly with some elliptical arcs along its boundary.

G. Finding First and Last LOS

It is impractical to represent every detail of the conflict boundary in the s_1–s_2 plane when quadratic bounds are taken into account, because any of the elliptical arcs may intersect the linear bounds in several places, so that the boundary of the conflict region includes multiple segments of a single ellipse. Ultimately, however, we need only determine whether the region contains any points at all, and then determine the first and last loss of separation, and we are now able to do so.



(a) Position of Aircraft 1 Dominates

(b) Position of Aircraft 2 Serves as Tiebreaker

Figure 17. First and last LOS on linear bounds from the perspective of aircraft 1

In the case of an DST intended for airborne use, such as the AOP, it is natural to favor definitions that are made from the perspective of the ownship, since the results will be viewed by the crew of that aircraft. This leaves only the question of whether "first" means "earlier" or "closer."

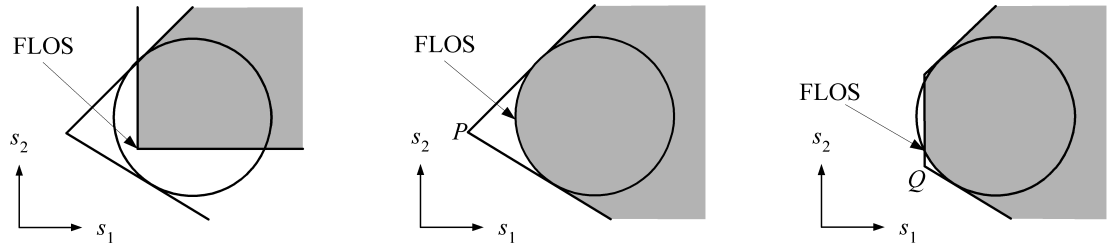
In the following analysis, the first LOS from the perspective of an ownship is defined as the location of the ownship that is closest to its starting point among all the points in LOS. (This definition is intended to allow a worst-case conflict "dogbone" to be displayed in a view of the airspace around the ownship, indicating the maximum extent of LOS along the length of the trajectory.) When the conflict region is a polygon as in Figure 17(a), we simply examine the s_1-coordinate of each vertex of the polygon. The vertex with the least value of s_1 is the first LOS, and the vertex with the greatest value of s_1 is the last LOS.

When there are two vertices tied for the least value of s_1, the first LOS may be defined to occur at whichever of these two vertices has the lesser value of s_2, as shown in Figure 17(b). The idea here is that it is most important to show the worst-case size of the ownship conflict for display, but traffic data may eventually be examined as well, so the traffic aircraft provides the tiebreaker between two equally bad cases for the ownship. The AOP follows a similar logic with the last LOS: given two vertices at the maximum value of s_1, the last LOS is the vertex with the greater value of s_2.

For a ground-based system, in which it does not make sense to favor one aircraft over another when determining first and last LOS, the preceding definitions must be modified. For example, the first LOS might be defined to occur at a "compromise" position such as the point that minimizes the sum of s_1 and s_2, or the position of first LOS might be computed separately for each aircraft in such a way as to minimize that aircraft's along-path distance. In the latter case, the locations of first LOS for two aircraft in conflict would not necessarily represent states that both aircraft can reach at the same time, and the separation between the two aircraft at first LOS might be strictly greater than required in one or both dimensions.

In general, however, the conflict region may not be a polygon; part of its boundary may be determined by quadratic bounds that define elliptical arcs. The algorithm then proceeds in two phases: first, find the first LOS and last LOS on the conflict polygon, that is, the first and last LOS assuming that only the linear conflict bounds need be considered. Second, examine each of the quadratic bounds and determine whether the estimate of the first or last LOS needs to be revised.

Consider the first loss of separation based solely on the conflict polygon, and compare it with one of the quadratic conflict bounds. Figure 18(a), which shows some of the bounds of a conflict between two trajectory



(a) Linear Bounds (b) A Quadratic Bound (c) Linear and Quadratic Bounds

Figure 18. First LOS Determined by a Combination of Linear and/or Quadratic Bounds

segments, shows one possible outcome of this comparison. In this figure, linear bounds of the conflict are shown by heavy lines, and the equation of one of the quadratic bounds is shown by an ellipse. The relevant portion of the conflict polygon in this example is the shaded region in the figure. The portion of the ellipse that is on the boundary of lateral LOS (that is, the arc that cuts off the corner on the left) is completely outside the linear bounds of the conflict region, and so does not actually bound that region. The first LOS can be determined from the linear bounds alone, that is, it is the leftmost point of the polygonal region that remains after all linear bounds have been applied. The location of this point is determined by simultaneously solving the linear equations of the two edges that meet there.

A second case is shown in Figure 18(b). In this case, the first LOS according to the linear bounds alone is at one of the corners of the quadrilateral determined by the linear bounds of lateral LOS (the point labeled P in this example). In this figure, the leftmost point of the conflict region is on the ellipse that "cuts off" that corner of the quadrilateral, as indicated in the figure. More generally, this case applies when the leftmost vertex of the polygon determined by the linear bounds falls somewhere in or on the "cut off" corner, and the leftmost point of the ellipse falls inside the polygon determined by the linear bounds. The s_1 coordinate of the first LOS is then simply the minimum s_1 on the ellipse, which can be determined analytically.[5]

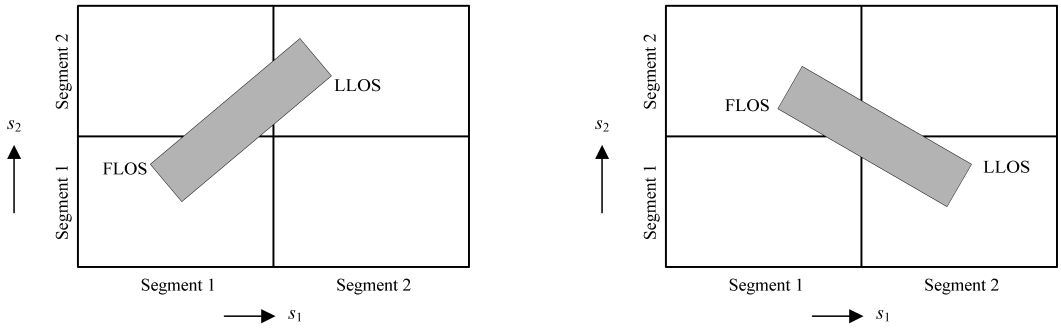
If neither of the preceding two cases is true, then the first LOS occurs neither at the intersection of the lines for two linear bounds (since then it would be a vertex of the conflict polygon) nor is it in the middle of an elliptical arc along the boundary of the conflict region. Instead, the first LOS must occur at a point along the conflict region where the ellipse intersects the conflict polygon, for example as shown in Figure 18(c). These points of intersection can be found by solving the equation for the ellipse and the equation of the relevant line simultaneously. The leftmost such point found is the first LOS in this case.

If there is a possible LOS at the start of the segment, this is the first LOS on the segment. Similarly, any LOS at the end of the segment is the last LOS on the segment. These are not necessarily the first or last LOS over the entire trajectory, however. We examine this question in the next section.

VII. Multiple-Segment Trajectory Conflicts

It will not always be the case that an aircraft flies into conflict and out of conflict again within a single trajectory segment. It may be that the first LOS for an aircraft is encountered along one segment of the aircraft's predicted trajectory but the last LOS occurs on a later segment (that is, the aircraft passes one of the discrete trajectory points specified in the trajectory data structure before it gets out of LOS). It is also possible that multiple trajectory segments of the other aircraft's trajectory may be involved in the same conflict. In any of these cases, we may have to examine more than one pair of segments in order to determine the first and last LOS of the conflict for either aircraft.

In order for a conflict to extend across multiple segments, it must be possible for either or both of the aircraft to be in LOS at one end or the other of their respective trajectory segments. Figure 19(a) shows one example. In this figure, each of the rectangular panels represents a different combination of segments selected from the two trajectories; the region inside a single panel corresponds to all possible combinations of along-path coordinates of the aircraft when each is on its designated trajectory segment. In other words, the shaded region in this figure is the union of the conflict regions for several different pairs of trajectory segments. (The shape of this region has been greatly simplified from what it would likely be in an actual case derived from predicted trajectories.) In this example, the first LOS occurs when each aircraft is in its



(a) Same-Direction Progression

(b) Reverse Progression

Figure 19. Multiple-Segment Conflicts

own first segment, and the last LOS occurs when each aircraft is in its own second segment. In order to remain in LOS between these two states, in this example the second aircraft first enters its second segment, and later the first aircraft enters its own second segment.

Figure 19(b) is another example of a multiple-segment conflict. This figure illustrates an example of the ambiguity of first LOS that we found earlier, due to aircraft approaching each other at uncertain speeds. In this case, the uncertainty in the speeds of the two aircraft is so great that distance from the starting location of aircraft 2 to the last LOS from the perspective of aircraft 1 (which occurs when aircraft 1 flies very fast and aircraft 2 flies very slowly) is actually less than the distance from aircraft 2 to the first LOS from the perspective of aircraft 1 (which occurs when aircraft 1 flies very slowly and aircraft 2 flies very fast). On the other hand, since these first and last LOS locations were determined from the perspective of aircraft 1, the first LOS will always be closer to aircraft 1 than the last LOS is, by definition.

In general, the effects on the first and last LOS due to the along-path uncertainty can make it difficult if not impossible to determine which pair of trajectory segments yields the overall first LOS for aircraft 1 unless multiple pairs of trajectory segments in LOS are examined to determine which pair's first LOS is "first." An algorithm to perform this search has been derived and implemented.[5]

VIII. Conflicts with Area Hazards

In the PUB-CD algorithm, it is assumed that the lateral extent of each area hazard is modeled by a polygon of three or more sides, and bounded by minimum and maximum altitudes. The polygon is described by its vertices, which are points on the surface of the Earth. An aircraft is within the area hazard if its lateral position is within the polygon and its altitude is between the minimum and maximum altitudes.

A trajectory is considered in conflict with an area hazard if the aircraft could be within the area hazard at any time during that trajectory. The conflict begins at the first possible predicted position for which the aircraft could be inside the hazard, taking cross-track and altitude errors into account, and ends at the last such predicted position. Because a conflict can include predicted positions where the aircraft can be inside the conflict only if it is off-course, it is possible for a conflict to start before the predicted path actually enters the conflict; in fact, it is possible that a trajectory whose predicted path passes very close by (but does not enter) an area hazard could be in conflict with that hazard only because the cross-track error at some points extends into the area hazard.

We assume that the polygon describing the hazard is a simple polygon, that is, its sides never cross and only two sides meet at any vertex; that is, it divides the sphere into exactly two regions, an "inside" and an "outside," each region fully connected to itself. We do not assume that the polygon is convex nor do we assume whether the vertices are traversed in clockwise or counterclockwise order when the polygon is received as input.

Conflicts between a trajectory and an area hazard are somewhat simpler to detect than conflicts between trajectories, because the area hazard has no prediction uncertainty (or if it does, the boundaries of the hazard have been enlarged to include the regions of uncertainty). Also, the PUB-CD algorithm, as currently implemented in the AOP, assumes that area hazards do not move. Moving area hazards would require an

algorithm more sophisticated than the one described here.

A. Conflict with a Trajectory Segment

In order to detect any conflict between a trajectory and an area hazard, we divide the trajectory into its individual segments. For each segment, we can use the following algorithm to determine whether that segment is in conflict with the area hazard.

An example of a trajectory segment is shown in Figure 20. The diagram shows the plane in which the segment and its cross-track error lie, with the along-path coordinate s plotted on the vertical axis and the cross-track position u plotted on the horizontal axis. That is, the predicted path along this segment is the line segment from $(0, 0)$ to $(0, L)$, where L is the length of the trajectory segment. The predicted path is shown by a heavy line. The actual position of the aircraft starts somewhere along the line segment between $(-w_0, 0)$ and $(w_0, 0)$ (the "start" of the trajectory segment) where $u = w_0$ is the maximum cross-track error at the beginning of the segment, and ends somewhere along the line segment between $(-w_L, L)$ and (w_L, L) (the "end" of the trajectory segment) where $u = w_L$ is the maximum cross-track error at the end of the segment. The line segment from $(w_0, 0)$ to (w_L, L) and the line segment from $(-w_0, 0)$ to $(-w_L, L)$ are, respectively, the right-hand and left-hand cross-track bounds of the trajectory segment.

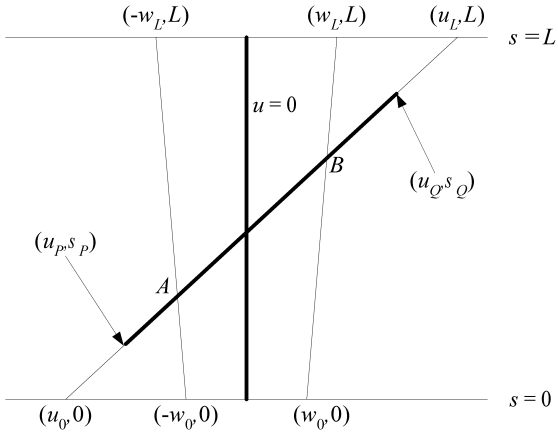


Figure 20. Conflict with a Side of a Polygon

Also shown in Figure 20 is one side of the polygon defining the area hazard, projected onto the plane of the trajectory segment. In this example, the two vertices of this side of the polygon, P and Q, have cross-track and along-path coordinates (relative to this trajectory segment) of (u_P, s_P) and (u_Q, s_Q). The *actual polygon side* for our purposes is the line segment between these two points, shown as a heavy diagonal segment in the diagram, and the *extended polygon side* is the infinite line on which the actual side lies.

In practice there are many possible configurations other than the example shown here. The extended polygon side may not cross both edges of the trajectory segment: it can enter or leave at the start or end of the segment instead, or it may not intersect the trajectory segment at all. The actual polygon side may intersect only one edge of the trajectory segment, or it might not intersect the trajectory segment at all even though the extended polygon side does.

In any case, as long as the polygon side is not perpendicular to the trajectory segment, the *overlap* between the trajectory segment and this side of the polygon is taken to be the range of along-path coordinates s of the intersection of the actual polygon side with the trajectory segment. In Figure 20, the overlap ranges from the along-path coordinate of point A to the along-path coordinate of point B.[e]

The various points of intersection of the extended polygon side with other lines in the figure can be determined by simple linear formulas. The CD algorithm first solves linear equations for u_0 and u_L. Depending on whether u_0 is less than, greater than, or between $-w_0$ and w_0 and whether u_L is less than, greater than, or between $-w_L$ and w_L, the algorithm determines whether the extended side of the polygon intersects the region between the cross-track bounds of the trajectory segment. If it does, as in Figure 20, the CD algorithm solves linear equations for s_A and s_B, the along-path coordinates of points A and B respectively. The trajectory segment is considered to intersect the polygon side if the range of values between s_A and s_B overlaps the range of values between s_P and s_Q (that is, if the segment $\overline{AB}$ and the actual polygon side are not disjoint), then the range between s_A and s_B is considered the overlap of this trajectory segment with this side of the polygon.

To determine the overlap of the trajectory segment with the entire polygon, we examine the overlap with each side of the polygon. The overlap with the polygon includes at least the union of all these individual

[e] In addition, it is possible to determine whether the trajectory segment is "entering" or "leaving" the polygon during its interaction with this side. In Figure 20, assuming that the "inside" region of the polygon lies along the right-hand side of the line as we travel from the vertex at (u_P, s_P) to the vertex at (u_Q, s_Q) (a fact that can be determined in a preprocessing stage when the polygon is first read in), the trajectory segment is exiting the polygon during this crossing.

overlaps. The overlap should also include any portions of the trajectory segment that are inside the polygon although they do not overlap an individual side of the polygon. The beginning (or end) of the segment overlaps the polygon without overlapping any individual side if and only if the entire start (or end) of the segment, including the cross-track buffer, is inside the polygon. Therefore the segment overlaps the polygon at the start (or end) if and only if it overlaps at least one side there *or* all points at the start (or end), in particular the predicted position, are inside the polygon. So a simplistic algorithm for determining the overlap with the polygon is that the overlap starts at the first overlap with any individual side, or at the start of the segment if the predicted start position is inside the polygon, and the overlap ends at the last overlap with any individual side, or at the end of the segment if the predicted end position is inside the polygon.[f]

The information about the lateral conflict must be combined with information about the vertical conflict. The segment conflicts with the area hazard if and only if there are positions along the segment that intersect the polygon both laterally and vertically. We find the points of lateral intersection using the algorithm just described; this gives a range from the first lateral LOS to the last lateral LOS on this segment. A relatively simple algorithm (which is not necessary to detail here) finds the first and last vertical LOS. Each of these LOS points is described by an along-path coordinate. The overlap between these two ranges of along-path coordinates is the portion of the trajectory segment that is in conflict with the area hazard.

B. Conflict with Multiple Trajectory Segments

In some cases a single trajectory segment may enter and exit an area hazard so that the first LOS and last LOS between the trajectory and that area hazard both occur on the same segment. In cases where the single-segment algorithm finds the first overlap after the start of the segment and the last overlap before the end of the segment, we know the conflict is entirely contained in that segment.

To determine the conflict in the more general case, where the trajectory could enter the area hazard during one segment and leave during a later segment, the AOP iterates through the segments of the trajectory starting at the beginning. The first segment that is found to have an overlap therefore contains the true first LOS for the entire trajectory. But as long as the last segment examined has an overlap with the area hazard at the very end of the segment, it is possible for the conflict to be continued in the next segment, and so AOP must advance to the next segment. AOP continues to iterate through the trajectory until it either finds a segment that exits the area hazard prior to the end of the segment (in which case that is the last LOS for the entire trajectory), or it finds a segment with no conflict at all (which can happen in the case where the last LOS for the entire trajectory occurs exactly at the end of the previous segment), or it reaches the end of the trajectory (which can happen when the trajectory ends inside the area hazard).

It is possible that the trajectory may enter, exit, and re-enter the area hazard. Therefore, even when a last LOS with this area hazard is found, the algorithm should continue to probe the trajectory up to the look-ahead limit. If the trajectory re-enters the area hazard, a new conflict is detected. It is possible for the trajectory to have multiple conflicts with the area hazard in this way, each reported separately.[g]

IX. Computational Performance of the Algorithms

The PUB-CD algorithm has been implemented in the Autonomous Operations Planner, an experimental DST designed to support delegation of separation authority to an aircraft flight deck.[6] This implementation has been used in numerous multi-aircraft simulations under various conditions, including the presence of special-use airspace and weather hazards.

The current implementation of the algorithm finds conflicts (when they might plausibly exist) between trajectories interacting under many different circumstances, including large-angle crossings, small-angle cross-

[f] If a single trajectory segment exits and then re-enters the polygon, the simplistic algorithm will deem it to be overlapping the polygon along the path between that exit and re-entry even though the portion of the segment between those points, including the cross-track buffer around that part of the segment, is completely outside the polygon. A more sophisticated algorithm that kept track of whether the overlap with each side is "entering" or "leaving" should be able to determine that that intermediate portion of the segment is actually outside the polygon, but this requires separately tracking an arbitrarily large number of overlaps along each trajectory segment and requires logic to handle all the different special cases of interactions with multiple sides, including simultaneous overlap of multiple sides.

[g] Using the simplistic algorithm for detecting the end of the overlap with the polygon, multiple conflicts that are not separated by a trajectory point may be reported as a single conflict. But cases such as where the second conflict starts because the trajectory turned back toward the area hazard will always be reported as multiple conflicts.

ings, head-on approaches, and aircraft in descent. Most importantly, the application of relatively small amounts of uncertainty within its predicted trajectories prevented missed alerts that had tended to occur in AOP during several experimental simulations.[7–9]

Running on a single-CPU Dell Precision Workstation 340 with a 2.4 GHz Intel Pentium-4 processor and 1 GB memory, a selection of eight different aircraft pairs (taken from an actual conflict resolution during a typical simulated scenario, and excluding pairs of trajectories whose highest and lowest altitudes ruled out any conflict), the execution time of the AOP implementation of PUB-CD on each pair of aircraft ranged from 0.345 to 0.845 millisecond, not including the time to create the predicted trajectories.[h] The average execution time of these eight test cases was 0.59 millisecond. In other words, the aircraft pairs in these examples were processed at rates between 1180 and 2900 aircraft pairs per second (depending on the complexity of the trajectories) at a mean rate (averaged over all aircraft pairs) of 1700 aircraft pairs per second. This agrees with informal observations of the behavior of the AOP during simulations in which the AOP performed CD on several hundred aircraft in a fraction of a second.

X. Comparisons with Previous Work

In the taxonomy defined by Kuchar and Yang,[10] the trajectory predictions used by CD algorithms are classified as probabilistic (in which relative probabilities of various deviations from the predicted trajectory are estimated), nominal, or worst-case. The latter two categories are regarded as special cases in which the probability distribution is forced to zero everywhere except along the predicted path (in the "nominal" method) or everywhere outside of the worst-case bounds. In this context, strictly speaking, PUB-CD uses worst-case prediction, but the prediction uncertainty bounds (PUBs) are not intended to cover all possible procedural outcomes; instead they are designed to be fitted relatively tightly around the predicted trajectory within "worst-case" bounds that are determined by threshold values of a probability distribution. For this reason it may be expected that the false alarm rate for PUB-CD will be similar to the rates for probabilistic algorithms rather than the rates for more conservative worst-case algorithms.

Several of the CD algorithms in the taxonomy provide conflict resolution as an integral component of CD process; others (described as having "manual" conflict resolution) merely accept proposed conflict-resolution maneuvers for evaluation, that is, they determine whether the maneuvers have acceptable results. PUB-CD neither resolves conflicts directly nor provides hints as to how the conflicts might be resolved. In this sense, PUB-CD provides only manual conflict resolution, but it has been integrated into a genetic algorithm for conflict resolution in the AOP; PUB-CD distinguishes conflict-free resolutions from failed resolutions proposed by the genetic algorithm and thereby enables the genetic algorithm to generate a conflict-free resolution automatically.[11]

Unlike some algorithms in the taxonomy, PUB-CD accounts for separation and maneuvering in both the lateral and vertical planes of motion; moreover, it specifies explicit criteria for when a conflict alert should or should not be raised. There are no explicit limitations on the resolution maneuvers that can be evaluated by PUB-CD as long as the trajectory can be predicted for each maneuver, unlike some algorithms that allow only certain types of maneuver. Unlike some other CD algorithms, PUB-CD considers all hazards simultaneously (including area hazards as well as traffic) when determining when to raise an alert and when conflicts have been resolved for an aircraft, and therefore lends itself well to a conflict resolution algorithm such as the AOP's, which attempts to resolve a conflict globally rather than resolving pairwise conflicts one at a time until none remain.

The PUB-CD algorithm depends critically on the quality of the PUBs along each trajectory, that is, the size and shape of the bounds in each phase of flight must be adjusted so that they are not too optimistic (which would lead to missed alerts due to aircraft that flew outside the estimated error bounds without any change in intent) and at the same time not too conservative (which would lead to excessive false alarms). Recent work has been done to improve our knowledge of the probability distributions of prediction errors around different segments of the trajectories of aircraft with known intents.[1]

[h] In order to obtain accurate estimates, execution time was measured from the start to the end of 10,000 consecutive complete executions of the algorithm on each pair of aircraft and then divided by the number of executions.

XI. Prospects for Future Work

A number of improvements or modifications could be made to the PUB-CD algorithm, to its practical implementation, or to the quality of the data used by it. The need for these modifications ranges from the clearly essential to the speculative.

A. Improved Estimates of Error Bounds

The examples of prediction errors and bounds described in this paper are provided in order to give an idea of the types of uncertainty that can be modeled by PUBs. This document does not attempt to define which prediction-error effects are actually to be modeled by a trajectory predictor. A software system using this CD algorithm might not actually model all the details provided here, and it might model additional prediction-error effects not shown here.

In order to effectively apply the variable uncertainty regions around predicted trajectories so as to provide the desired level of protection against prediction error without consuming an excessive amount of redundant airspace (that is, to achieve the desired protection against missed alerts without causing too many false alerts), an extensive analysis of the regions of prediction uncertainty is required for all parts of all the modes of flight in typical operations. This requires a sufficient body of statistics from a representative sample of flight plans and the actual trajectories resulting from them, and an analysis that fits the observed prediction errors within uncertainty bounds of appropriate size and shape. Both the selection of the data sample and the method of fitting bounds around the data depend on the environment in which CD is to be performed and the application for which the results are to be used. A new environment or a new application may require the derivation of new values for the bounds.

B. More Detailed Models of Error Bounds

The bounds of altitude error are already specified in a way that allows asymmetry around the predicted path. The definitions of the bounds in the cross-track and time errors, however, assumed that the bounds would be symmetric in those dimensions around the predicted four-dimensional path. Specifically, aircraft can deviate equally far to the left or right of their predicted trajectories, and can arrive at a point equally early or late in comparison to the ETA. This assumption allowed some simplification of the implementation, for example after testing the maximum error in one direction it was often sufficient to simply reverse the sign of a variable in order to examine the maximum error in the opposite direction. For some purposes it may be useful to relax either or both of these assumptions, requiring new derivations and implementations of some of the formulas governing conflict detection. The general principles by which the four-dimensional problem is reduced to an examination of lines and elliptical arcs in a plane, however, would remain valid in such an approach.

A study of the gaps exhibited in the cross-track buffers around a turn in Figure 6 could more rigorously answer the obvious questions about the effect on CD accuracy. If necessary, the gaps might be eliminated in either of two ways. If the assumed direction of cross-track error were allowed to vary left or right of the perpendicular direction along a trajectory segment, the "cross-track" direction at the end of one segment could be matched to the direction at the start of the next segment. The effect on the shape of the conflict region would be an increase in the number of linear bounds of lateral LOS to be considered. Alternatively, if turns were modeled by circular arcs, the cross-track error would be radial and would not exhibit gaps; but this model might require additional non-linear bounds of the conflict region.

XII. Conclusion

The conflict detection algorithm presented here allows the use of prediction uncertainty bounds that promise to allow a reasonably sharp definition of the region of uncertainty around a trajectory. This algorithm avoids a significant class of missed alerts, while providing opportunities to reduce the number of false alerts that might otherwise have to be generated in order to avoid the missed alerts. This algorithm was implemented and tested in an experimental system under various scenarios of traffic and area hazards.

Acknowledgments

This effort was conducted for NASA's Langley Research Center under subcontract to Raytheon Corporation. The authors would like to thank the NASA researchers and the Raytheon contract staff for their continued support.

References

[1]Gong, C. and McNally, D., "A Methodology for Automated Trajectory Prediction Analysis," *AIAA Guidance, Navigation, and Control Conference*, AIAA-2004-4788, Providence, Rhode Island, Aug. 16–19, 2004.

[2]Paielli, R. A. and Erzberger, H., "Conflict Probability Estimation for Free Flight," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 3, May–June 1997, also published as NASA TM-110411.

[3]Paielli, R. A. and Erzberger, H., "Conflict Probability Estimation Generalized to Non-Level Flight," *Air Traffic Control Quarterly*, Vol. 7, No. 3, 1999, pp. 195–222.

[4]Yang, L. and Kuchar, J., "Using Intent Information in Probabilistic Conflict Analysis," *AIAA Guidance, Navigation, and Control Conference*, Boston, MA, Aug. 10–12, 1998.

[5]Karr, D. A., "Conflict Detection with Dynamic Buffers," Jan. 2006, prepared for NASA Langley under the Raytheon Consolidated Information Technology Services (ConITS) contract.

[6]Ballin, M., Sharma, V., Vivona, R., Johnson, E., and Ramiscal, E., "A Flight Deck Decision Support Tool for Autonomous Airborne Operations," *AIAA Conference on Guidance, Navigation, and Control*, AIAA-2002-4554, Monterey, CA.

[7]Barhydt, R., Kopardekar, P., Battiste, V., Doble, N., Johnson, W., Lee, P., Prevot, T., and Smith, N., "Joint NASA Ames/Langley Experimental Evaluation of Integrated Air/Ground Operations for En Route Free Maneuvering," *USA/Europe ATM Seminar*, 2005.

[8]Doble, N., Barhydt, R., and Hitt, J., "Distributed Conflict Management in En Route Airspace: Human-in-the-Loop Results," *24th Digital Avionics Systems Conference*, Washington, DC, 2005.

[9]Doble, N., Barhydt, R., and Krishnamurthy, K., "Airborne Management of Traffic Conflicts in Descent with Arrival Constraints," *24th Digital Avionics Systems Conference*, Washington, DC, 2005.

[10]Kuchar, J. K. and Yang, L. C., "A Review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, No. 4, Dec. 2000, pp. 179–189.

[11]Vivona, R. A., Karr, D. A., and Roscoe, D. A., "Pattern-Based Genetic Algorithm for Airborne Conflict Resolution," *AIAA Guidance, Navigation, and Control Conference*, Keystone, Colorado, Aug. 21–24, 2006.