Stratway: A Modular Approach to Strategic Conflict Resolution

George Hagen* Ricky Butler*
Jeffrey Maddalon*
NASA Langley Research Center, Hampton, Virginia, 23601, USA

In this paper we introduce Stratway, a modular approach to finding long-term strategic resolutions to conflicts between aircraft. The modular approach provides both advantages and disadvantages. Our primary concern is to investigate the implications on the verification of safety-critical properties of a strategic resolution algorithm. By partitioning the problem into verifiable modules much stronger verification claims can be established. Since strategic resolution involves searching for solutions over an enormous state space, Stratway, like most similar algorithms, searches these spaces by applying heuristics, which present especially difficult verification challenges. An advantage of a modular approach is that it makes a clear distinction between the resolution function and the trajectory generation function. This allows the resolution computation to be independent of any particular vehicle. The Stratway algorithm was developed in both Java and C++ and is available through a open source license. Additionally there is a visualization application that is helpful when analyzing and quickly creating conflict scenarios.

I. Introduction

Automated airspace separation concepts perform two key functions: first, that aircraft conflicts must be detected and, second, that safe and efficient resolutions to these conflicts must be found. The algorithms that implements these functions are called *conflict detection and resolution* (CD&R) algorithms. Resolution algorithms are further classified into two groups: tactical and strategic. A tactical algorithm determines a resolution that is conflict free, usually with a single guidance maneuver, whereas a strategic resolution produces not only a conflict free path, but one that also satisfies an ultimate objective such as reaching an initial approach fix at the given required time of arrival. This paper describes a particular strategic conflict resolution algorithm called Stratway.

A. Strategic Resolution

As a research effort, Stratway is designed to solve only part of the self-separation problem. Specifically, it is assumed to operate in class A airspace for distributed self-separation concepts.[3] In addition, Stratway works under several engineering assumptions such as that the data provided to it is current and error-free.

In most cases, en-route airspace aircraft are required to maintain a separation distance from each other of 5 nautical miles horizontally and 1000 feet vertically. An intrusion into this volume is a *loss of separation* (LoS). A loss of separation that is predicted to occur within a certain lookahead time is a *conflict.* An aircraft's maneuver to avoid a conflict is a *resolution.* Avoiding collisions is considered a different problem.

Critical to both conflict detection and resolution is a model of the aircraft's behavior. One such model only relies on an extrapolation of the aircraft's current position and velocity. This *state-based* model has several advantages. First, the information is currently available in the Automated Dependent Surveillance

*Research Engineer, Safety-Critical Avionics Systems Branch, Mail Stop 130

Broadcast (ADS-B) standard,[13] and, due to an aircraft's momentum, over a short time period this model is very accurate. The relative simplicity of this model allows for strong verification of safety properties. Finally, given typical traffic densities over short time horizons, an aircraft is rarely involved with more than one conflict, which greatly simplifies algorithm design. Algorithms that use state-based trajectories provide *tactical* resolutions, which involve a single guidance maneuver. However, tactical resolution algorithms do not necessarily use state-based trajectories.

A major limitation of the state-based model is that planned turns and altitude changes, usually called the "pilot's intent," are not accounted for and so these simple trajectories are not accurate over long time horizons. This inaccuracy can lead to both false warnings, where an aircraft unnecessarily maneuvers to avoid a conflict that does not exist, and missed alerts, where an aircraft does not recognize that there is a conflict. This latter situation can result in a *pop-up conflict* where an aircraft makes a change in vertical speed and does not realize that this maneuver will create a near-term conflict.

The desire for solutions for conflict detection and resolution over longer time horizons introduces new issues. Over long time periods an aircraft could be involved in conflicts with more than one aircraft. Resolving two or more conflicts requires a more complex maneuver. Furthermore, a pilot is not only interested in avoiding conflicts—reaching the destination at the contracted time is also of high importance. In reaching the desired destination, other constraints such as avoiding high traffic zones, adverse weather, and special use airspace must also be taken into account. To address these issues, there is a desire for *strategic* resolutions, where an aircraft does not make a single guidance maneuver, but rather can make complex changes to its flight plan.

Because of the long time horizons involved in strategic resolutions, more information is needed about the planned paths of each aircraft. This information is called the aircraft's *intent information.* At a first level of approximation, an aircraft's intended path can be a sequence of 4-D points (say, a flight plan's waypoints). With appropriately large separation buffers even this coarse information can be useful for CD&R with intent. However a suitably large separation buffer may introduce unnecessary capacity constraints. More precise information about the aircraft's intended path can come in the form of *trajectory change points* (TCPs) that indicate the position, time, and details of a planned velocity change. Instead of traditional waypoints, which may omit time or altitude information, TCPs include such information as the predicted beginning and end of a climb or turn. Such intent information is included in the ADS-B standard although many manufacturers do not implement this part of the standard. The use of intent information also introduces the possibility that an aircraft will not *conform* to its intended path. Conformance issues are not addressed in this paper.

B. Motivation

Numerous conflict detection[6] and resolution algorithms have been published in the literature.[7] Both tactical[15] and strategic[14] resolution algorithms have been published. A candidate comprehensive solution for automated airborne separation is the Autonomous Operations Planner (AOP),[1] which includes conflict detection, tactical resolution, and strategic resolution algorithms.

The goal of our research with Stratway is to investigate a particular modular approach to build a strategic conflict resolution system. The design philosophy of modules is to divide the functionality of the system into clearly identifiable units with clear interfaces. Initially many complex algorithms use modules as a means to divide the complexity of the problem into manageable units. However, strong interfaces tend to weaken over time due to the need for flexibility or pressures on the development schedule. Our motivation is to examine the modular approach from the standpoint of verification. Because the computer logic necessary for automated aircraft separation is both highly complex and safety critical, verification of this logic is essential.

Because the solution space is so large for strategic resolutions, algorithms to find resolutions typically employ heuristics to narrow the search space. Heuristics are generally incomplete, so potential solutions will be missed. Due to this behavior, verification of the correct operation of a heuristic algorithm is particularly challenging. Through a modular approach we aim to isolate the heuristic elements to ensure that their ill-defined behavior does not impact the system's safety argument.

Strategic conflict detection relies on accurate trajectories of the ownship and traffic aircraft. Intent based trajectories are typically developed from two sources of information: the flight plan and the information broadcast over ADS-B. An aircraft's flight management system (FMS) can build a trajectory from a flight plan and so similar functionality must be available to the CD&R system. Unfortunately this high-resolution trajectory will not be available to traffic aircraft, so in these cases the trajectory must be inferred from ADS-B information. To complicate matters further, there is not a universally accepted definition of what comprises a trajectory. For these and other reasons, modular versions of strategic CD&R systems have not been readily available.

In our work we are seeking to fill this gap by developing modules for key strategic CD&R functions. To ensure the modules are not specific to one programming environment we have implemented them in both Java and C++ with virtually identical behavior (i.e. up to floating point precision differences). Stratway has a well-defined Application Program Interface (API) that can be called from other programs. We provide this software in open source form.[10] To aid in concept exploration the Stratway algorithm can also be executed as a batch application or as an interactive visualization program that can display aircraft flight plans and trajectories, graphically develop scenarios, and apply selective execution of individual Stratway solution strategies.

Although the Stratway algorithm has been designed for airborne solutions for aircraft self-separation, it is highly configurable to enable the exploration of the solution space for strategic CD&R systems, and it has been used for centralized approaches.[2] This configuration ability has allowed Stratway to be used in studies of system-wide optimization[5] and traffic flow management, as well as a resolver for "background" traffic in human-in-the-loop simulations. The Stratway program has been integrated into several airspace system simulators, including ACES[11] and MACS.[12]

II. Reasons for Modularity

Verification is the process of ensuring that a system meets its specification. The conventional way to verify that a system meets its specification is to engage in an extensive testing effort. There is always the possibility, especially with complex systems, that problems exist that will not be revealed through testing, since testing must be incomplete. With safety-critical systems, such problems can have disastrous consequences. Mathematical verification is advantageous in that it forces the designer to both rigorously specify the system and the desired properties and to ensure that the system specification is satisfied for *all* input conditions. It is able to reveal unlikely errors that could easily be missed through testing or human inspection. The process of verification can also reveal a deeper understanding of both the system in question as well as the problem it is trying to address, possibly leading to new and better solutions. Two precursors to verification are a well-defined interface and a clear statement of required functionality. Both of these can be provided through a modular architecture.

From a verification standpoint, this makes a modular design highly desirable, but the verification of the overall system can greatly depend on how these modules are defined. Modules could instead be specified in such a way to emphasize performance or to address some other goal. Our decision with Stratway is to create modules that facilitate the verification of safety properties. In our case, the ability to limit analysis to relatively simple, internally complete units greatly enhances what can be proven about a system. In this paper we discuss some of the consequences of this design decision in addition to describing the Stratway application.

III. Stratway Solution Technique

A strategic CD&R solver takes aircraft intent information concerning the ownship and an arbitrary number of traffic aircraft as input, and produces a series of maneuvers that will keep the ownship conflict free, possibly constrained by such things as a final destination and arrival time.

Stratway works strictly with 4D points and the lines connecting them. We call this set of points for an aircraft a *plan*, distinguishing it from a (predicted) trajectory or traditional flight plan. A plan can be an abstraction of either of these, and the 4D points may have supplementary information that enables Stratway to produce better solutions. As an example, some 4D points can be designated as immutable in the horizontal, vertical, or time dimensions—e.g. a required time of arrival point would likely be fixed for all three attributes. Great circle trajectories are addressed by internally breaking longer segments into smaller ones.

By restricting the input plans to be piecewise linear functions, we are able to leverage the fast detection modules developed for state-based solutions[10] and extend them to include intent. Once a conflict has been detected, the solution methods manipulate the original plans into new, conflict-free candidate plans, which can then be either confirmed as feasible by the FMS, or converted back into an abstract flight plan to generate a new trajectory.

Stratway uses a heuristic approach to resolve conflicts. Specifically, it assumes the input plans are close to optimal and thus attempts to only make small changes to these plans. It also assumes that a plan with multiple conflicts can be solved one conflict at a time. This means that multiple conflicts are usually solved with multiple maneuvers. The advantage of this approach is that Stratway is always making progress towards a solution. Finally, Stratway's heuristic returns once the first solution is found—it does not find multiple candidate solutions then order them by some cost function. In theory, such an approach could be added to the Stratway algorithm. However, our concern is for safety, not optimization. The primary control the user has on the optimality of the resolutions is that the order the resolution strategies are applied can be changed.

In order to resolve conflicts, the Stratway program provides localized solution techniques, called *strategies*, such as stretching an earlier leg to avoid a conflict after a turn, moving to a different altitude to pass over or under a conflict, shortcutting turns, and several ways to maneuver around conflicts. Additionally there are strategies for avoiding conflicts by adjusting speed, and other variations involving vertical changes. The API to the program allows a calling program to select any subset of solution techniques or use the default set. There are also many parameters that can be set to customize the solver. These include limits on how sharp turns can be, the maximum percentage changes to the ground speed, minimum lead-in times, etc.

It is important to emphasize that the complexity of the problem can be arbitrarily high: there are many ways that solutions can be constructed, such as using path stretches (of different lengths), step climbs (at different altitudes), removal of waypoints, and time changes, among others. Because of this complexity it is possible that a solver such as Stratway may not be sophisticated enough to find a resolution within a reasonable amount of time. And, in the cases of extreme congestion, there is the possibility that an aircraft may simply have no feasible resolutions available—other functions as part of the self-separation concept must ensure that an airspace volume will never become this crowded.

A. Internal Architecture

Internally, Stratway consists of three major components: a main interface loop, an intent-based conflict detection module (Detector), and an expandable set of strategies that perform localized resolutions. These components are illustrated in Figure 1. As can be seen in that diagram, a set of ownship and traffic plans are input into Stratway, and the ownship plan is analyzed for conflicts. When one is detected, the set of plans, along with information about the conflict, are passed to the strategies.

Stratway's modular decomposition for verifiability is shown in Figure 1. The heuristic search is encapsulated in the strategies. Although these strategies have a well-defined interface, their behavior is much less well defined. Ill-defined behavior is difficult or impossible to verify. Therefore we side-step this issue through the modular decomposition. Since the strategies themselves cannot be verified, instead we place a verifiable "checker" on their output. This checker is the intent-based conflict detection module (Detector). Furthermore, since the main loop is routing data among the solutions, it must also be verified. Due to Stratway's modular nature, only the Detector and the main loop components are safety critical.

Stratway returns a plan and status. If the status is "conflict-free," then the returned plan will indeed be conflict free. Otherwise the returned plan represents a partial solution that may have resolved some (but not all) conflicts. Typically Stratway will return a single revised ownship plan representing the first

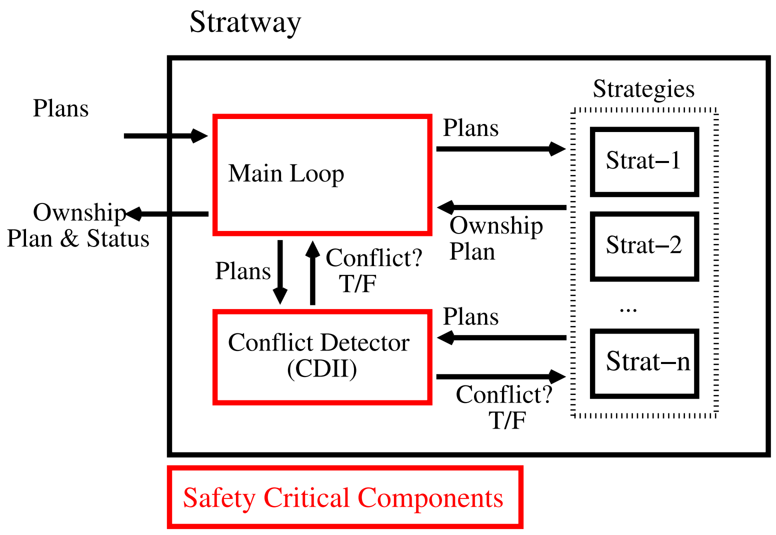


Figure 1: Data Flow in Stratway.

discovered solution. As we are primarily concerned with safety, considerations such as flight efficiency (fuel consumption, etc.) are not directly addressed.

Strategies are given a user-defined ordering and are applied against each conflict in turn. This allows the user, for example, to indicate a preference for track-based solutions over vertical ones, or only consider resolutions that make left-hand turns. If a strategy fails to resolve a conflict (as well as any secondary conflicts), the next strategy is attempted. Once a solution has been found for all conflicts, the detector is run again, against the entire revised ownship plan, and if it is truly conflict free, it is returned as a solution.

Stratway also allows a user to investigate several strategy orderings as a single query. This simply loops through the sets of orderings and returns a normal solution for each, allowing the user to then choose the resolution that was most favorable.

B. Detection

Detection in Stratway is based on the CDII algorithm from the ACCoRD framework.[10] CDII extends a verified state-based Euclidean pairwise conflict detection algorithm to allow for intent information for both the ownship and traffic aircraft. Stratway's Detector object itself extends CDII to simultaneously function on an arbitrary number of traffic aircraft plans. In addition, it transparently acts to mitigate inaccuracies in geodetic-to-Euclidean projections on longer flight legs.

Given a set of plans, the Detector object is able to return the number of conflicts that involve the ownship. For each of these conflicts, it is also able to provide both the time into and time out of conflict, as well as the time of closest approach and the projected horizontal and vertical separation.

C. Resolution Strategies

We have developed numerous resolution strategies for use in Stratway. As mentioned above, Strategies generally provide localized solutions, and are typically iterative in nature. For example, a climb to avoid a conflict might be gradually increased to higher and higher altitudes, until the immediate region is conflict free (a success) or until some pre-determined cutoff is reached (a failure). Each strategy applies a local search by introducing, deleting, or modifying one or more points in the plan around the conflict. In a few cases this manipulation is determined analytically, but usually iteration is used to make progressively larger adjustments to points to avoid secondary conflicts when re-capturing the plan. Several strategies are approximations of AOP's pattern-based resolutions.[14]

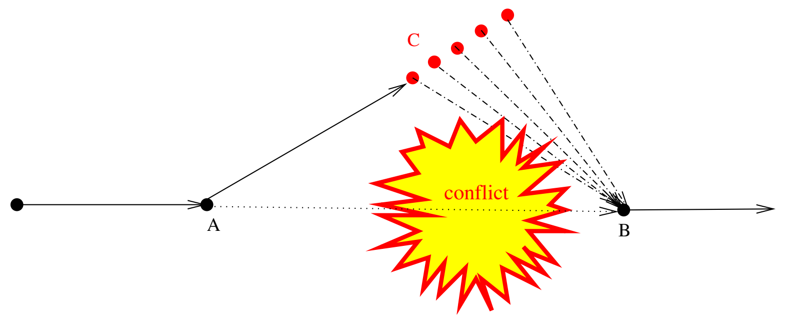


Figure 2: rrTrk Strategy

One simple example is a strategy we call rrTrk, shown in Figure 2. This strategy avoids a conflict by diverting the ownship to either the left or right, flying around the traffic aircraft through a new point. It first ensures that the conflict is isolated by two points (A and B in the figure). It then introduces a third point (C) between A and B. An analysis of the conflict is used to determine an initial direction and heading, and point C is gradually moved along the heading until both the avoidance and return segments are conflict-free. This, like most other strategies, assumes the original plan was near optimal by some measure, and attempts to minimize changes.

By default Stratway uses a backtracking search when generating a sequence of resolutions. If no strategy can solve a given conflict, there is the chance that using a different solution to an earlier conflict will change the current circumstances sufficiently that it becomes solvable. Stratway will return to an earlier (successful) resolution and apply untried strategies before re-attempting the current conflict. This allows all combinations of strategies to be attempted before returning a failed status.

IV. A Modular CD&R Architecture

As mentioned previously, Stratway has been designed specifically to be a modular unit, intended to be incorporated into larger systems, such as human-in-the-loop airspace simulations. This choice also affects our ability to verify safety properties about the system: it allows us to produce code that, by decoupling components, encourages simplicity of modules, and that facilitates eventual verification of safety properties of these modules. This philosophy also extends to Stratway's place in a larger system.

This decision, while it facilitates verification, does come at some cost. Not only has there been increased design effort in creating well-defined interfaces, but there are also trade-offs in performance.

A. Incorporation in a Airborne Application

For reference, a typical decomposition of the relevant current avionics would probably be similar to that in Figure 3. This includes transmission of ADS-B information, which in the case of most currently available systems is the ownship state information. Our concern with Stratway is in adding self-separation capabilities.

A natural approach might be to add self-separation as a capability of an already existing flight management system (FMS). In Figure 4 we illustrate a monolithic self-separation system and the context that it operates within, as may be typical in a research application. This design allows the use of the FMS' trajectory generation functionality to be directly linked to the separation assurance logic, which provided several advantages. Primarily, it can allow the constraints from both systems to be applied at once, granting the ability to generate a (possibly optimized) conflict-free trajectory in a single pass. There is also the potential benefit of utilizing accurate information that is privy to the ownships' local avionics.

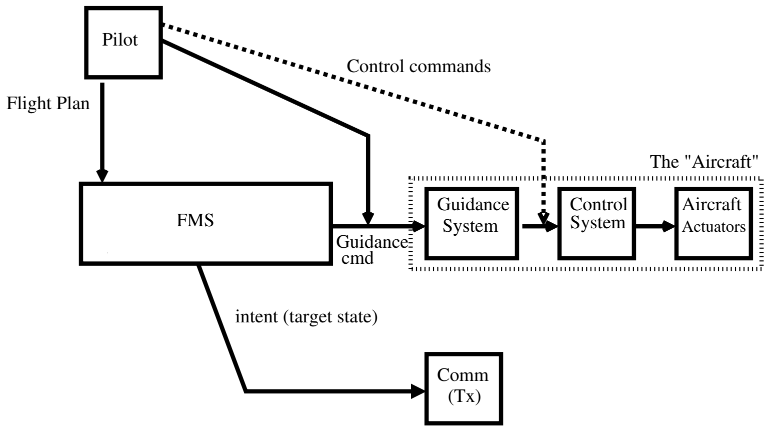


Figure 3: Current avionics architecture.

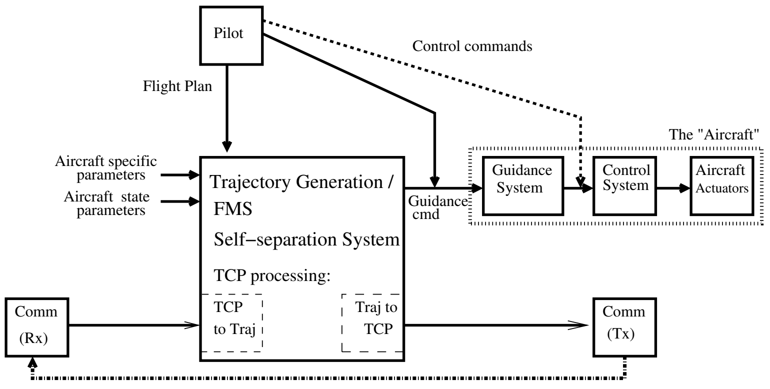


Figure 4: Monolithic Architecture Data Flow

This, however, makes it extremely easy to tightly tie the separation assurance logic to a particular FMS, and since separation assurance is a safety critical function, the entire FMS then becomes a safety-critical system and needs to be certified as such. Mathematically proving useful safety properties of such a complex system is often beyond the capabilities of current verification techniques. In addition to the increased difficulty of needing to deal with irrelevant data (and figuring out what data is irrelevant), important data may be scattered throughout the larger system, and the relation between two pieces of relevant data may be highly obscured. Specific to verifying separation properties, it is necessary to untangle FMS- and trajectory generation-specific functionality from that needed for aircraft separation.

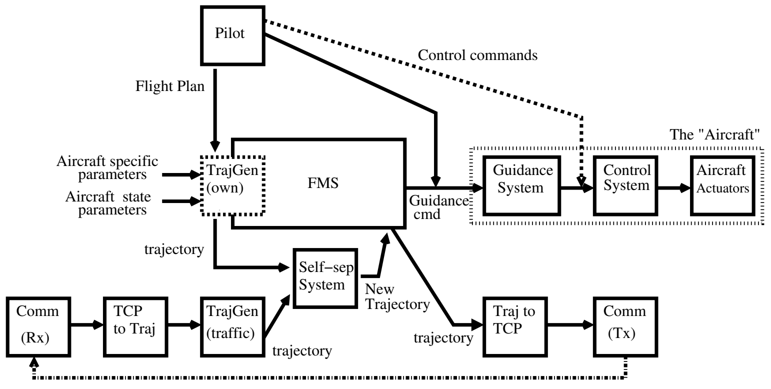


Figure 5: Modular Architecture Data Flow

In contrast, our approach with Stratway is a modular architecture where the trajectory generation function (or plan generation function, in our case) is decoupled from the separation assurance logic. Trajectory generation could still be based on FMS algorithms, but this would need to be made accessible via an interface, and could be treated as a separate component. An example of this architecture can be seen in Figure 5. A more detailed example of the separation assurance subsystem can be seen in Figure 6.

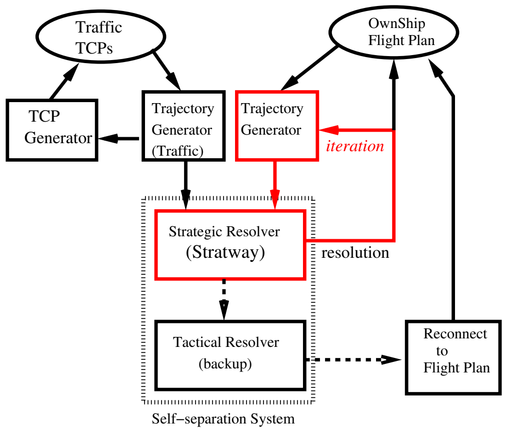


Figure 6: Important CD&R functions and their basic relationships

Such a modular approach also provides advantages. First, it provides improved testability and verifiability. The modules can be tested and verified separately, which is much simpler and far more effective than testing the full system. The verification of self-separation properties need not depend on the particulars of how a trajectory comes about, only its description of movement through space. Furthermore, the strategic and tactical components of such a system can be completely independent, allowing, say, a state-based back up tactical system that can be strongly verified.

Having a well-defined generic interface to modules also decreases system complexity and can reduce the long-term software maintenance cost. The overall reliability of software is also improved by decreasing complexity. Additionally, the modular structure and well-defined interfaces allow it to be reused in other places. Finally, this approach allows the CD&R functionality to work for many different aircraft.

Decoupling the CD&R functionality from the trajectory generation capability of the FMS, however, is also a source of a major disadvantage to this approach, depicted as the red loop in Figure 6. This means that a less accurate plan/trajectory will be generated within the CD&R system and, consequently, another call to the FMS may be necessary to refine the final trajectory to something that is feasible for the aircraft. This in turn may re-introduce conflicts, requiring another pass with the separation assurance system, and so on. Hopefully this would converge into a realistic conflict-free trajectory within a few iterations, but, if not, something would need to be done to prevent an endless loop. One possibility would be to keep a history of previous resolutions and compare them to the current one in order to ensure that each iteration is different. Another potential solution could be to simply let the strategic solver fail after a fixed number of attempts, either to see if traffic aircraft alter their plan sufficiently to allow it to succeed later (if the conflict is far enough in the future), or allow a tactical resolver produce a short-term solution.

Although there are advantages and disadvantages to both approaches, we believe that improved verifiability, increased reliability, and reusability are more important than increased performance, so we favor the modular approach. We also believe that there are ways of compensating for less accurate trajectories produced within the CD&R module. A simple solution we use in Stratway is to allow the user to specify the size of buffers used for detection and resolution separately. If the user sets a larger resolution buffer, this ensures that any resulting plan modifications include extra space for avoiding traffic aircraft. Some work has been done in determining the appropriate size for such buffers in the face of uncertainty.[4]

In an airborne context, there is also the constraint of an aircraft's forward momentum. If there is a nearby conflict, Stratway handles this by delaying any resolutions a short time. Instead of possibly returning an immediate (and physically unachievable) velocity change, Stratway will instead create a fixed point a short time in the future (30 seconds, by default) along the current path, ensuring that any resolutions allow the pilot some time to actually complete a suggested maneuver.

B. Aircraft Trajectories

For purposes of conflict detection and resolution, we want estimated trajectories that are accurate, especially with respect to turns, climbs and descents. But turn and climb dynamics can vary depending on the type of aircraft. This creates a basic dilemma: if a very accurate trajectory is pursued, then the strategic CD&R system can be locked into a particular aircraft and a particular flight management system. But if a generic approach is pursued, whereby a large class of aircraft can be accommodated, then the trajectory will inevitably be less accurate.

We have sought to make Stratway as independent of a particular trajectory generator as possible. Since it is not tightly coupled to any particular trajectory generation algorithm, Stratway takes a proposed trajectory (in the form of a plan) and produces a candidate conflict-free abstract trajectory (another plan). These can then be re-submitted to the external trajectory generator to generate a more accurate trajectory, which could itself be checked for conflicts. More complex trajectory generators could be developed as needed.

Regardless of the amount of information that one has about his or her own aircraft, the information about the traffic aircraft is necessarily limited. The strategic CD&R system must rely on information that is broadcast for the traffic aircraft. This in turn requires that a common trajectory specification language be developed. In the airspace arena this language is the language of Trajectory Change Points (TCPs).[13] These are broadcast within the ADS-B Trajectory Change Report (TCR). The information contained in the TCR is a subset of the information available to the ownship and so it is more inaccurate than the ownship trajectory models. Each TCP defines a leg of the trajectory, which is a path segment, possibly curved, that ends at a a specified TCP. Typical TCPs include Direct to Fix, Course to Fix to Fly-By-Turn, Target Altitude, Top of Climb, etc. A precise semantics for these TCPs will be necessary to ensure that the verification of the CD&R system is sound.

One important aspect to consider for real-world systems is that ADS-B information is, because of bandwidth restrictions, degraded in quality, and traffic trajectories need to be reconstructed from this information. This may produce an asymmetry in conflict calculations, where more precise ownship trajectories may lead one aircraft to detect a potential conflict, e.g. in a parabolic instead of circular turn, that is missed by its counterpart. A procedure that relies heavily on consistent world information could be negatively affected by such a situation.

Whether this is an important problem or a minor inconvenience deserves further study. One could easily argue that such a conflict would be borderline, and so at worst only be a negligible intrusion into a large protection zone. Another possible solution would be to implement flight rules such that any aircraft that detects a potential conflict must maneuver to avoid it. Another possibility is for all aircraft to use the same degraded ADS-B information (and reconstructed trajectories) for both traffic and ownship for such procedures.

C. Stratway in a Centralized Application

The primary focus of Stratway has been in an airborne based self-separation concept. However it may also be used for de-conflicting aircraft traffic in a centralized system approach, such as part of a traffic flow management system. Since trajectory generation in these systems is not likely to be tightly tied to specific aircraft flight management systems (and high-fidelity trajectories may not be a priority for centralized applications like traffic flow management), plans could be generated with little effort.

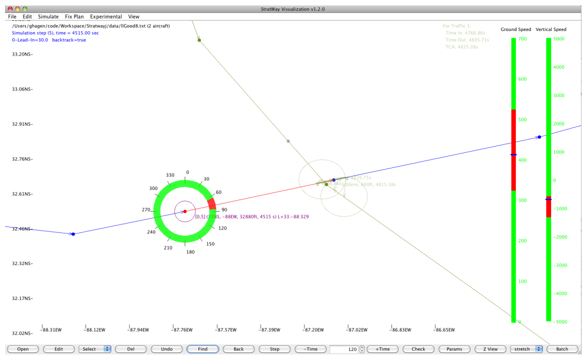
In this case a single copy of Stratway may simply be called multiple times, once for each aircraft in a set, assigning each as the ownship in sequence. By default Stratway attempts to deconflict entire plans, so little else need be here done except to ensure that the origin and destination locations are fixed as immutable.

Note, however, that Stratway does not take advantage of the fact that the larger system has (near) universal knowledge of the airspace—it is still based on de-conflicting the path of a single aircraft. Since only one aircraft maneuvers at a time, Stratway is not likely to discover any solutions that are globally optimal with respect to any cost function except safety.

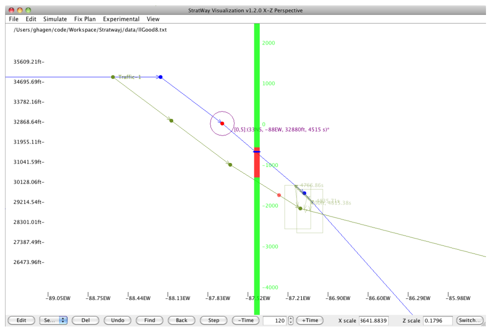
V. The Visualization Environment

Around the core Stratway module we have also created visualization software in Java. This provides a graphical user interface (GUI) to aid in developing strategies as well as easily generating and visualizing scenarios of interest. Once created, a scenario can then be examined at any point in time during the ownship's flight. This tool has been used in the development of HITL experiments, as the GUI allows for a simple means of generating 4D plans and then modifying them to produce conflicts. These scenarios can be developed completely by hand with a mouse or input as simple text files.

The Stratway visualization consists of a primary window that shows a top-down view of various plans (Figure 7a). Points in a plan are indicated by colored dots, with legs between points indicated by correspondingly colored arrows between sequential points. The user can open a secondary window that displays a vertical view of the scenario (Figure 7b), as well as polar and projection-centered views. These displays also visualize potential conflict information in the form of a reticule (around the ownship's current position) and vertical bars indicating the ownship's conflict prevention bands[9] as well as details of detected conflicts in the form of a heavier "dog bone" bar at the location of the conflict. Supplementary information can also be provided about the various plans and aircraft. Figure 7 shows an impending conflict between the ownship (blue and, for the current segment, red) and traffic aircraft (green). The ownship's current position is the red dot surrounded by a circle; it is flying in an easterly direction while descending. The traffic's position is the grey dot slightly above then center of the display; it is flying southeasterly while descending. In this situation, the ownship will pass over the traffic aircraft too closely, creating a conflict. The reticule (for track changes) and bars (for ground and vertical speed changes) indicate the consequences of potential tactical


StratWay Visualization v1.2.0
File Edit Simulate Fix Plan Experimental View
Ground Speed Vertical Speed
Open Edit Select Del Undo Find Back Step -Time 120 +Time Check Params Z View stretch Batch

(a) Primary Display


StratWay Visualization v1.2.0 X-Z Perspective
File Edit Simulate Fix Plan Experimental View
Edit Se... Del Undo Find Back Step -Time 120 +Time X scale Z scale Switch...

(b) Vertical View

Figure 7: Stratway Visualization

maneuvers: maneuvering into green regions will avoid conflict with any traffic, while red regions will lead to a conflict in the next few minutes.

There is also preliminary support for no-fly zones that may represent weather or special use airspace, as shown in Figure 8. This figure displays a scenario where several aircraft are flying to and from a central hub, with a series of disruptive weather cells, represented by moving polygons, passing through the area. Here the ownship will need to maneuver twice to avoid an area of inclement weather, once in the immediate future, as well as after it has passed through the hub.

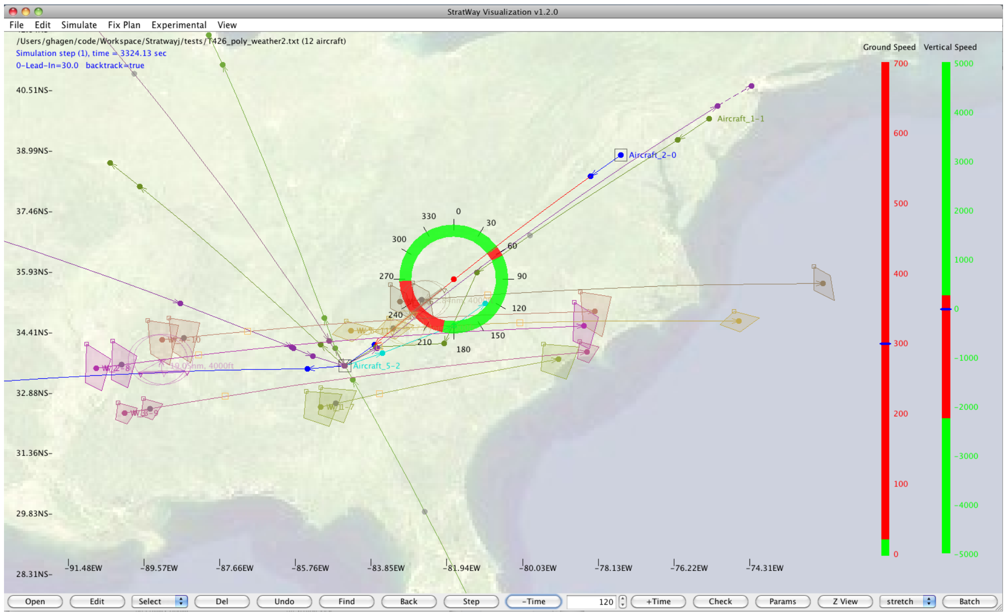

StratWay Visualization v1.2.0
File Edit Simulate Fix Plan Experimental View
/Users/ghagen/code/Workspace/Stratwayj/tests/T426_poly_weather2.txt (12 aircraft)
Simulation step (1), time = 3324.13 sec
0-Lead-In=30.0 backtrack=true
Ground Speed
Vertical Speed
Aircraft_1-1
Aircraft_2-0
Aircraft_5-2
40.51NS
38.99NS
37.46NS
35.93NS
34.41NS
32.88NS
31.36NS
29.83NS
28.31NS
-91.48EW
-89.57EW
-87.66EW
-85.76EW
-83.85EW
-81.94EW
-80.03EW
-78.13EW
-76.22EW
-74.31EW
Open Edit Select Del Undo Find Back Step -Time 120 +Time Check Params Z View stretch Batch

Figure 8: Weather Polygons in Stratway (background image credit NASA Visible Earth)

Strategies can be implemented individually or in a similar batch mode to the API interface, and all API parameters and functionality can be accessed through the GUI. For the example in Figure 7, the default batch resolution is to delete one of the intermediate points, effectively lowering the ground speed of the ownship and allowing the traffic to pass in front of it. In the case of Figure 8, the default solution for avoiding the initial weather zone is to climb over it, while the second (after passing through the hub) is to divert the ownship around it.

Scenario generation is a simple matter of sketching out plans with a mouse and then tweaking them to achieve the desired results. Points can be manipulated individually, or entire plans can be modified at once, and tools are available to help visualize the bounds of conflicts.

VI. Verification and Future Work

We have made efforts to design Stratway in such a way that it is verifiable. Central to this is our modular approach. Key components can be separated from the larger system and it will be easier to prove properties about those components. Verification of the system is a long-term goal, and has not been completed, but the architecture we have chosen will allow us to still make statements about the overall system with high degrees of assurance even if the eventual verification does not take into account every line of code.

Because of our heuristic approach to the resolution problem, in our future work we will not attempt to say that Stratway is *complete*—there is always the possibility that a resolution is missed. Our goal, instead, will be to show that the system is *correct*—if Stratway provides a solution, there will be a high degree of assurance that the resolutions provided will indeed produce conflict-free plans.

Key to achieving this goal is the simple idea of filtering. If we verify this property for the conflict detection module, then we are able to check any proposed solutions with this module as a final step and reject any that still contain conflicts. By doing this we will be able to say that any solution provided by Stratway is, indeed, conflict free. Although this approach ensures that Stratway will never produce an unsafe solution, it allows that Stratway may not produce any solution. In this case something outside of Stratway must ensure that a resolution is produced by another algorithm: perhaps a tactical intent-based resolution algorithm[15] or a state-based algorithm.[8] In the event that Stratway is unable to produce a conflict-free resolution, it is left to some other system (such as a tactical resolver) to solve the problem.

Another design choice driven by the goal of verifiability is to emphasize simplicity in our algorithms. The basic algorithms all use Euclidean geometry and are all, ultimately, based on simple vector arithmetic and a bit of set theory, avoiding transcendental functions whenever possible—trigonometric functions only come into play when a result needs to be output as a track angle.

Much work has already been done with this goal in mind, and much of the core of the detector algorithms has already been proven to be both complete and correct for Euclidean systems with perfect state knowledge. There is still work to do in this area, however. We hope to eventually also provide proofs involving multiple-aircraft detection, as well as the correctness of various ancillary computations (time of closest approach, etc). Further goals include proving properties of the projections used to transform geodetic coordinates to Euclidean, as well as proofs based on the actual code base as opposed to the algorithms it is based on.

VII. Summary

This paper presents a strategic conflict detection and resolution application called Stratway. We have chosen a modular design that is decomposed in such a way to facilitate verification of safety properties. A modular approach allows us to examine each simpler component, keep relevant data segregated, and compositionally prove properties for the larger system. Although the full system has not been fully verified, much work has been already done in verifying several key algorithms. An additional benefit is that, in decoupling the CD&R functionality from the trajectory generation functionality, we are able to create a generic system that is not tied to any particular flight model. In this paper we have detailed several of the ramifications of this design decision.

Stratway itself is a self-contained module not tied to any particular trajectory generator. It has an open source application programming interface, with both Java and C++ code available, allowing it to be used in other systems. It may also be used as a stand-alone system, and has a graphical user interface designed to quickly and easily sketch out air traffic scenarios for use in experimentation. Stratway has been used in several experiments both for scenario design and as a CD&R system, and has been integrated into several larger simulations, including ACES and MACS.

References

[1]M.G. Ballin, V. Sharma, R.A. Vivona, E.J. Johnson, and E. Ramiscal. A flight deck decision support tool for autonomous airborne operations. In *AIAA Guidance, Navigation and Control Conference*, Monterey, CA, 2002.

[2]Matthew S. Bigelow. Examining the relative costs and benefits of shifting the locus of control in a novel air traffic management environment via multi-agent dynamic analysis and simulation. Master's thesis, Georgia Institute of Technology, May 2011.

[3]Vincent Capezzuto. Application Integrated Work Plan. Technical report, FAA, June 2010.

[4]Heber Herencia-Zapana, Jean-Baptiste Jeannin, and César Muñoz. Formal verification of safety buffers for state-based conflict detection and resolution. In *Proceedings of 27th International Congress of the Aeronautical Sciences, ICAS 2010*, Nice, France, 2010.

[5]Carolyn Kaplan, Johann Dahm, Elaine Oran, Natalia Alexandrov, and Jay Boris. The monotonic lagrangian grid for rapid air-traffic evaluation. In *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, Fort Worth, Texas, Sept 2010. AIAA 2010-9336.

[6]D. Karr, D. Roscoe, and R. Vivona. Conflict detection using variable 4D uncertainty bounds to control missed alerts. In *AIAA Guidance, Navigation, and Control Conference*, Keystone, Colorado, Aug 2006.

[7]J. Kuchar and L. Yang. Survey of conflict detection and resolution modeling methods. In *AIAA Guidance, Navigation, and Control Conference*, pages 1388–1397, New Orleans, LA, August 1997. AIAA-97-3732.

[8]César Muñoz, Ricky Butler, Anthony Narkawicz, Jeffrey Maddalon, and George Hagen. A criteria standard for conflict resolution: A vision for guaranteeing the safety of self-separation in NextGen. Technical Memorandum NASA/TM-2010-216862, NASA, Langley Research Center, Hampton VA 23681-2199, USA, October 2010.

[9]Anthony Narkawicz, César Muñoz, and Gilles Dowek. Provably correct conflict prevention bands algorithms. *Science of Computer Programming*, 2011. In Press.

[10]NASA. Airborne coordinated conflict resolution and detection (ACCoRD), 2011.
http://shemesh.larc.nasa.gov/people/cam/ACCoRD/.

[11]NASA. Airspace concept evaluation system (ACES), 2011. http://www.vams.arc.nasa.gov/activities/aces.html.

[12]Thomas Prevot and Joey Mercer. MACS: a simulation platform for today's and tomorrow's air traffic operations. In *AIAA Modeling and Simulation Technologies Conference*, Hilton Head, South Carolina, Aug 2007. AIAA 2007-6556.

[13]RTCA SC-186. *Minimum Aviation System Performance Standards for Automatic Dependent Surveillence Broadcast (ADS-B)*. RTCA, 2002.

[14]R. Vivona, D. Karr, and D. Roscoe. Pattern-based algorithm for airborn conflict resolution. In *AIAA Guidance, Navigation, and Control Conference*, Keystone, Colorado, Aug 2006. AIAA-2006-6060.

[15]D.J. Wing, R.A. Vivona, and D.A. Roscoe. Airborne tactical intent-based conflict resolution capability. In *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, Sept 2009.