An Emergency Landing Planner for Damaged Aircraft

Nicolas Meuleau[*] and **Christian Plaunt** and **David E. Smith** and **Tristan Smith**[†]

Intelligent Systems Division
NASA Ames Research Center
Moffet Field, California 94035-1000
{nicolas.f.meuleau, christian.j.plaunt, david.smith, tristan.b.smith}@nasa.gov

Abstract

Considerable progress has been made over the last 15 years on building adaptive control systems to assist pilots in flying damaged aircraft. Once a pilot has regained control of a damaged aircraft, the next problem is to determine the best site for an emergency landing. In general, the decision depends on many factors including the actual control envelope of the aircraft, distance to the site, weather en route, characteristics of the approach path, characteristics of the runway or landing site, and emergency facilities at the site. All of these influence the *risk* to the aircraft, to the passengers and crew, and to people and property on the ground. We describe an emergency landing planner that takes these various factors into consideration, and proposes possible routes and landing sites to the pilot, ordering them according to estimated risk. We give an overview of the system architecture and input data, describe our modeling of risk, describe how we search the space of landing sites and routes, and give a preliminary performance assessment for characteristic emergency scenarios using the current research prototype.

1. Introduction

On July 19, 1989, United flight 232, a DC-10 enroute from Denver to Chicago, suffered an uncontained failure of the fan blades in the number two (rear) engine. The resulting debris severed hydraulic lines in the airplane's tail resulting in loss of all hydraulic fluid and consequent loss of all aircraft control surfaces. Miraculously, a DC-10 flight instructor on board the aircraft was able to regain some semblance of control using differential thrust from the two remaining engines. An emergency landing was subsequently attempted at Sioux City, IA. Because of the high landing speed, high descent rate, and limited control, the aircraft broke up on impact, but 10 of 11 crew members and 175 of the 285 passengers survived the accident (NTSB 1990).

This accident, and others involving structural damage to aircraft, motivated research on adaptive control systems aimed at allowing a pilot to continue to fly a damaged aircraft using stick inputs. The adaptive controller translates those inputs into novel combinations of thrust vectoring and control surface movements in order to achieve the pilots intent. Testing of these controllers in full motion simulation and in test aircraft has been very successful so far (see for example (Burcham *et al.* 1996; Burken & Burcham 1997; Gundy-Burlet *et al.* 2004)). As a result, such control systems are being seriously considered for next generation military and civil transport aircraft. This capability, while quite remarkable, only addresses the first piece of the problem – regaining control of the aircraft. Once this is achieved, the next problem is to determine the best site for an emergency landing. In general, the decision depends on many factors including the actual control envelope of the aircraft, distance to the site, weather en route, characteristics of the approach path, characteristics of the runway or landing site, and emergency facilities available at the site. All of these influence the *risk* to the aircraft, to the passengers and crew, and to people and property on the ground. A purely secondary consideration is airline and passenger convenience.

Although pilots are highly trained in emergency procedures, structural damage and the consequent changes in flight characteristics strain the limits of their intuition and ability to assess different possible options. It would therefore be very useful to have an automated system that could, in seconds, generate and evaluate different possible emergency landing plans, and present the best options to the pilot. Furthermore, as the flight progresses, it is necessary to continually update and re-evaluate the set of options to take into account the changing location, altitude and velocity of the aircraft, subsequent degradation or failures that change the predicted control envelope, and updated weather and airport information.

Fundamentally, this problem is a 3D path planning problem involving dynamics (aircraft speed and direction), with complex optimization criterion. It may, for example, be possible for the aircraft to fly through a region of moderate turbulence, but because of the limited control authority there is increased risk of loss of control. It might also be easier (as it was for United 232) for the aircraft to make right turns rather than left turns, or to handle a right crosswind rather than a left crosswind.

Traditionally, difficult path planning problems have been solved using either discretization of the space or by generating probabilistic road maps. As we will explain in more detail later, both of these approaches have proven problem-

---

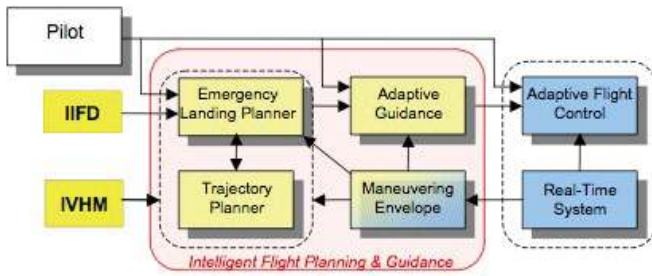[*]Carnegie Mellon University

[†]Mission Critical Technologies

Figure 1: An overview of the IFPG Architecture in IRAC, including the Emergency Landing Planner.

atic for this domain. Instead, we generate "roadmaps" by starting with a 2D visibility graph, and augmenting the edge set in the vertical dimension to allow paths above, below or through obstacles where possible. This systematic approach to roadmap construction is proving to be reasonably effective because of the columnar nature of obstacles in this domain. The planning search is then a hybrid discrete/continuous version of A* that searches for paths of low risk in this roadmap.

In the sections that follow, we give an overview of the system architecture, describe how obstacle information is obtained, describe how we assess risk, and describe the details of our prototype path planning algorithm.

2. Architecture

The Emergency Landing Planner is one component of the Integrated Flight Planning and Guidance (IFPG) subsystem of the Integrated Resilient Aircraft Control (IRAC) Project (see Figure 1). In the IFPG architecture, when some sort of damage or failure occurs that impairs the aircraft in some significant way, several things happen. First, the Adaptive Flight Control subsystem helps the pilot retain or regain control of the aircraft. While this is happening, the IFPG subsystem dynamically gathers: airport and obstacle data from the Integrated Intelligent Flight Deck (IIFD); aircraft health information from the Integrated Vehicle Health Management (IVHM) subsystem; and aircraft control limitations from the Maneuvering Envelope subsystem. It then uses this information to construct the 3D planning problem to be solved by the Emergency Landing Planner.

As the Emergency Landing Planner finds usable solutions that do not violate any of the obstacle or controllability constraints, it consults the Trajectory Planner to refine these solutions into more detailed flight plans. (The trajectory planner has a much more detailed but computationally more expensive model of aircraft performance and dynamics.) The pilot then chooses from among the proposed flight plans.

This IFPG emergency planning architecture allows for flexibility in the amount of autonomy delivered by the IFPG subsystem. The pilot can choose any of the solutions proposed by the IFPG subsystem based on experience, and on the information and predictions delivered by the IIFD,4 IVHM, and Adaptive Flight Control subsystems.

The planning problem to be solved consists of the follow-

ing:
1. The start state, consisting of the current position, speed, direction, and altitude of the aircraft.
2. The control envelope for the damaged aircraft, including airspeed range, allowable bank angles, descent range, and control responsiveness.
3. The potential landing sites within the estimated landing range of the aircraft, together with the characteristics of those landing sites, such as urban density, runway length and width, weather conditions, and emergency facilities.
4. All of the "obstacles" that must be considered while flying to any landing site. Some of these may be hard obstacles like terrain, but others may be regions with weather conditions that simply present increased risk.

Within these constraints, the Emergency Landing Planner searches, based on explicit modeling preferences, for the best solutions that can be found. These are then presented to the pilot (see Figure 5) as possible landing sites. Alternative landing sites are regularly re-evaluated to account for the evolution of the plane's situation.

3. Obstacles

To determine the best routes and landing sites, the planner has to consider a set of dynamic and static *obstacles*. These obstacles are derived from various on-line sources. There are five rough categories of obstacle data:

- Terrain
- Urban development
- Special Use Airspace (SUA) and Temporary Flight Restrictions (TFRs)
- Radar observations (rain, showers, thunderstorms)
- Icing and turbulence reports

Terrain, urban development, SUA, and TFRs are all relatively static, so the obstacles can be constructed on a daily basis and cached. Radar observations and icing and turbulence reports are much more dynamic, and must be constructed in real time for the region in the vicinity of the aircraft. Terrain represents hard obstacles that the aircraft cannot violate. The other areas are all soft obstacles that the aircraft can violate, but with increased risk. In general, that risk is a function of the controllability of the aircraft, and, in the case of weather obstacles, of the severity of the weather. All of these obstacles are *columner* in nature – that is, they have a 2D boundary, a floor, and a ceiling. We represent them using convex polygons together with their floor and ceiling altitudes, and the associated risk.

4. Assessing Risk

There is risk associated with various phases of an emergency landing – en route, approach, landing, and emergency response. We use *expected loss of life* as our measure of risk, because it allows us to take into consideration casualties on the ground, as well as passengers and crew. It is difficult to give precise estimates of risk, given all the uncertainty associated with aircraft capabilities, pilot performance, and weather. However, our purpose here is simply to provide a

rough means of ranking alternatives for presentation to the pilot, and to allow the pilot to focus on the most critical factors affecting the decision on landing site. We do not intend to display this numeric information to the pilot.

4.1 En Route Risk

The primary factors influencing risk en route to the landing site are: controllability of the aircraft, distance and time to the site, complexity of the flight path (e.g. number of turns), weather along the path (thunderstorms, icing, and turbulence), and risk of further deterioration in aircraft performance and handling. The en route portion of the flight is generally at sufficiently high altitude (in the US) that terrain is not an issue.

We represent controllability in terms of the probability of retaining control for different flight regimes. Let P_{stable} represents the probability of maintaining control at each infinitely small step of a traverse or, loosely speaking, the probability of succeeding *per nautical mile (nm)*. Then for a flight leg of length D, the probability of succeeding is:

$$P_{leg} = (P_{stable})^D$$

If weather is involved in a leg, we take the probability of loss of control to be:

$$P_{leg} = \left(P_{stable} * P_w^S\right)^D$$

where P_w is the probability of an undamaged aircraft successfully traversing light weather, and S is the severity of the weather (thunderstorm, icing, or turbulence) with 1 being light and 5 being extreme. Initially, we have chosen P_w to be 0.9. For a normally functioning transport aircraft, ($P_{stable} \approx 1$), this means that the chance of maintaining control when flying through extreme thunderstorms, icing or turbulence is $.9^5 \approx .59/nm$. This may prove to be too low, but for now, it biases the planner away from routes through serious weather obstacles.

Risk of further deterioration in aircraft capabilities is difficult to assess. Ultimately, we assume that this risk will be given to us by the control and diagnostic systems, and will depend on the nature of the damage or failures. For now, we assume that the probability of maintaining control in the face of further degradation is relatively high, but not one: $P_{degr} = .99/nm$. This introduces an additional factor inside the distance exponent in the calculation of P_{leg}, which introduces a mild bias for selecting shorter routes and closer landing sites.

There is additional risk associated with initiating and concluding turns, so if we let P_{turn} be the probability of maintaining control for a turn, the probability of success for a route with multiple legs and T turns is taken to be:

$$P_{route} = (P_{turn})^T * \prod_{l \in legs} P_l$$

Given the probability information for a route, the expected risk (number of fatalities) is the product of the number of persons onboard the aircraft and the probability of failure along the route:

$$Risk_{route} = B * (1 - P_{route})$$

where B is the number of persons onboard the aircraft.

4.2 Approach Risk

When the aircraft reaches the approach environment (low altitude in the vicinity of the landing site) several additional risk factors come into play, including urban development along the approach path, ceiling, and visibility. Assuming the aircraft navigation equipment is functioning normally, there is little additional risk associated with approaches where the ceiling is at least 1000 ft above the *Decision Altitude (DA)* for the approach (typically 200 ft for a standard ILS approach). However, if the ceiling is below the *DA*, the risk is high. We take the probability of success when a ceiling is present as:

$$P_{Ceil} = \begin{cases} 1 & \text{if } Ceil \geq 1000 + DA \\ \frac{Ceil - DA}{1000} & \text{if } 1000 + DA > Ceil \geq DA \\ 0 & \text{if } Ceil < DA \end{cases}$$

Similarly, there is little risk when the visibility is greater than 3 miles, and extreme risk when it is less than half a mile. We therefore take the probability of loss due to ceiling as:

$$P_{Vis} = \begin{cases} 1 & \text{if } Vis \geq 3 \\ \frac{Vis - .5}{2.5} & \text{if } 3 > Vis \geq .5 \\ 0 & \text{if } Vis < .5 \end{cases}$$

We assume that loss of control of a transport category aircraft over a densely populated area will cause loss of life within at least a .1 square mile area. Thus, if the approach path takes us a distance D over population density N the risk is:

$$Risk = (.1N + B) \left(1 - P_{Ceil} * P_{Vis} * P_{leg}\right)$$

where P_{leg} is the probability of success for the final approach leg, as calculated for en route flight. Thus, if controllability is still high (e.g. loss of one engine), and weather is good, there is little bias against approach paths over heavily populated areas. However, if controllability or weather is poor, there is additional bias against high population density approach paths.

4.3 Runway Risk

The primary risk factors associated with runway choice are runway length, width, and relative wind. In general, the length required is determined by landing speed, braking condition, and relative wind. Landing speed can be much higher than normal if controllability is low, but we assume this is given to us as part of the control envelope. As a general rule, an aircraft needs about 40ft of runway for each knot of speed at touchdown, thus:

Runway-reqd $\approx 40 * (Approach\text{-}speed - Headwind)$

This can go up by as much as a factor of two if braking quality is poor (water, snow, or ice on the runway). It can also go up considerably if pitch or speed controllability is poor, so these two factors need to be added to the above equation. If runway length is sufficient, as computed above, there is no additional risk; risk increases as the runway length is reduced below that. We therefore compute probability of success due to runway overrun as:

$$P_{length} = \frac{Rnwy\text{-}length}{Rnwy\text{-}reqd}$$

when *Rnwy-reqd* > *Rnwy-length*. Thus, if the runway length is zero, the result is considered equivalent to a crash. Similar factors are required if the runway width is low, if aircraft speed is high, or if the crosswind is too high, given aircraft controllability. So overall, the probability of a successful landing can be assessed as:

$$P_{rnwy} = P_{length} * P_{width} * P_{speed} * P_{xwind}$$

4.4 Airport Risk

Finally, the risk of landing at a particular airport is influenced by the availability of emergency facilities at the airport and in the surrounding community. If facilities are absent this can result in greater loss of life if the aircraft loses control on landing, or if it runs off the end of the runway. If we represent emergency facilities as being a number between zero (no facilities) and 1 (good facilities) we can estimate the facilities risk as follows

$$Risk_{ldg} = B * \left(1 - \left(P_{rnwy} \right)^{(2-facil)} \right)$$

Thus, if emergency facilities are good, no additional risk is incurred, but if they are poor, the risk increases somewhat.

Taken together, all of this risk information for route, approach, runway, and airport facilities allow us to evaluate different possible emergency landing plans.

5. Path Planning

The role of path planning is to determine the best path from the current position of the plane to any candidate landing site. There has been extensive previous work on path planning and obstacle avoidance. See (Choset *et al.* 2004) for a survey of this field. Our research in the IRAC project lead us to experiment with several approaches, including cell decomposition, roadmaps, and probabilistic algorithms.

Our first attempt was to do a 3D cell decomposition of the Euclidean space and search this decomposition using A* for the best paths from the current position to runways within range. There were two problems with this approach. First, even using cells that are one square mile and 1000 ft thick, the search space was very large (typically 2-3 million cells). Performance was not adequate when there were significant weather or terrain obstacles present. Second, the resulting paths had lots of artificial zig-zag turns in them, even if the actual travel could be in a straight line. Since turning increases risk, this artificially biased the search against paths that did not go 0, 45, or 90 degrees to the axes of the grid. To get around this second problem, we tried doing cell decomposition in "configuration space", which included aircraft descent rate, turn rate, and heading, in addition to location. This search space was so large that it proved completely impractical to do any kind of systematic search. We therefore also experimented with the construction and use of probabilistic roadmaps in the 6D configuration space.[1] While this approach typically finds a path in the space, the paths are often highly non-optimal as illustrated in Figure 2,

---

[1]To do this, we extended the Object-Oriented Programming System for Motion Planning (OOPSMP) software package developed at Rice University.
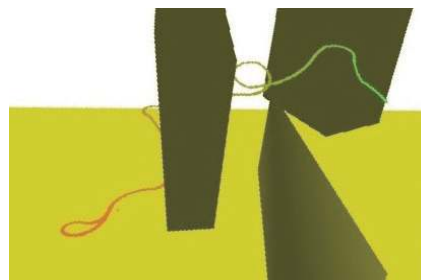

Figure 2: Meandering path generated using OOPSMP in the 6D space.

and the search time can vary wildly depending on the obstacles present, and the points chosen by the probabilistic algorithm. A further problem with this approach is that it is not clear how to adapt it to consider both paths that traverse through weather obstacles as well as paths that go around those obstacles.

As a result, we have adopted a somewhat different, more systematic method of generating long-range roadmaps that relies on the typical characteristics of obstacles in this domain.

5.1 Roadmaps

A roadmap is a topological representation of the environment that captures the connectivity of the free space. Several types of roadmaps have been proposed in the literature. In this work, we adapt one of the earliest and most common techniques: the *visibility graph*. This graph is defined in two-dimensional space. Remember that obstacles are represented as 2D polygons with an associated floor, ceiling and risk. For the purpose of building the initial visibility graph, we consider all obstacles above a certain risk level, ignore the floors and ceilings of the obstacles, and use only their two-dimensional polygonal representation. In order to account for aircraft dynamics and turn radius, we expand the size of each obstacle by an amount determined by aircraft controllability. The nodes $\mathcal{V}$ of the visibility graph include: the start location v_0, the possible destinations v_g, and all the obstacle vertices (corners between two edges of the polygons). The edges $\mathcal{E}$ are straight lines between vertices that do not traverse any obstacle (see Fig. 3). In 2D, the visibility graph is guaranteed to contain the shortest path from the start to the goal. Unfortunately, this property does not hold if the same approach is applied in higher dimensions, or if some of the obstacles are soft.

The *reduced visibility graph* or *tangent graph* is a subgraph of the visibility graph that is also guaranteed to contain the the shortest 2D path. Because it contains fewer edges, it is easier to solve. It is based on the observation that the shortest path traverses only edges that are tangent to obstacles. Therefore, non-tangent edges can be safely removed from $\mathcal{E}$ (see Fig. 3). Determining the set of tangent edges can be a difficult problem (Liu & Arimoto 2004). Instead of computing this set exactly, we eliminate edges whose extremities are not "locally tangent" to an obstacle.
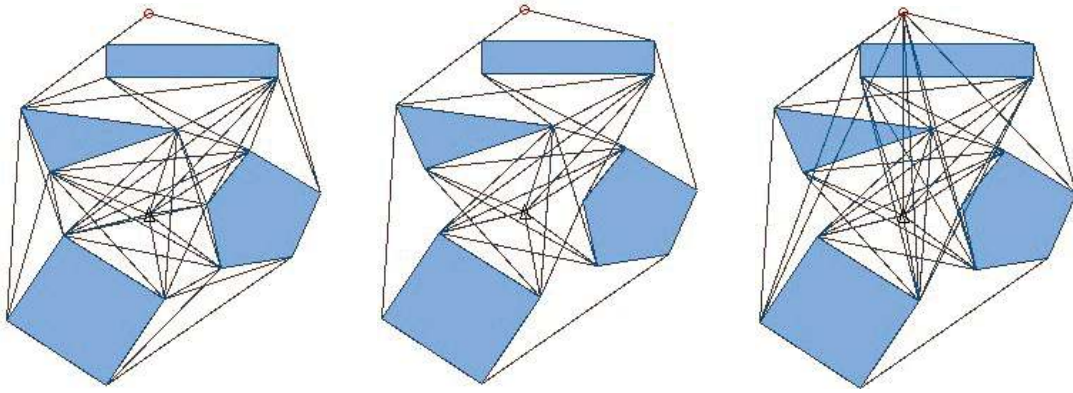
Figure 3: Three types of roadmaps (from left to right): visibility graph (58 edges), tangent graph (45 edges), and extended tangent graph (69 edges). The aircraft is represented by the small triangle in the center of the figure, and the targeted landing site is the small circle at the top of the figure.

Consider an edge e incident to vertex v, between two sides s and s' of the polygonal obstacle. Then, e is locally tangent in v if s and s' fall on the same side on the line passing through e. A tangent edge may not contain an extremity that is not locally tangent (but the converse is not true). Therefore, we can safely eliminate edges with an extremity that is not locally tangent. This eliminates fewer edges than the real tangent graph; however, since testing for local tangency is very cheap, it is a good overall compromise.

So far, we have limited the discussion to a 2D framework. In our emergency landing domain, obstacles have a floor and a ceiling, and it is sometimes possible to fly above, below or through some of them. To account for this possibility, the tangent graph is augmented with a set of *secondary* edges and vertices.

To build secondary edges, we first consider connecting v_0 to v_g. We enumerate all obstacles that intersect the segment between these two vertices and all ground-level variations along this segment. As shown in Fig. 4, the path from v_0 to v_g is divided into a series of *slices* inside of which the ground level is constant and the same set of obstacles is traversed. We represent this cut through the 3D space by a chain of (secondary) vertices and edges: there is one secondary edge for each slice in Fig. 4 and one secondary vertex between each two consecutive slices.

Similar operations are repeated for connecting different pairs of vertices in the 2D map:

- v_0 or v_g to any corner of an obstacle, if the segment intersects another obstacle and does not have any non-locally tangent extremity;

- Two corners of different obstacles, if the segment intersects a third obstacle and does not have any non-locally tangent extremity.

Next, *primary* edges (those originally present in the tangent graph) are replaced by chains of secondary edges to account for ground level variations along those edges (if the ground level is constant along the edge, it is left untouched).

We call the resulting graph an *extended tangent graph* (see Fig. 3). We can associate with each edge e of $\mathcal{E}$
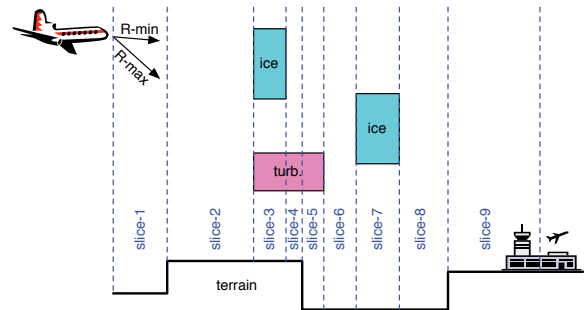

Figure 4: Cut of the 3D space along the segment from the plane (v_0) to the targeted landing site (v_g). The cut is divided into 9 slices where ground elevation and obstacles are constant. Each slice is represented by a secondary edge in the extended tangent graph. Secondary vertices's are added between each two consecutive secondary edges.

a set of altitudes with equivalent cost or risk. It is represented as a finite set of altitude intervals and denoted $\mathcal{U}_e = \{(f_i, c_i), i = 1, 2, \ldots, k_e\}$. Notations f and c stand for *floor* and *ceiling* as intervals (f_i, c_i) represent *tunnels* along e through which the plane can navigate. If e traverses no obstacle, then $\mathcal{U}_e$ contains a single interval ranging over all possible altitudes. For instance, slice 3 in Fig. 4 contains five tunnels (altitude intervals) and slice 8 only one tunnel.

The extended tangent graph contains more edges than the tangent graph, and often contains more edges than the visibility graph. Although it is not guaranteed to contain the shortest path in 3D space, it allows for some possibilities of movement that are not represented in the 2D graphs, such as going above, below, or through an obstacle. In the next subsection, we show how to determine, given the vertical maneuverability of the plane, what trajectories are possible in the 3D cut of Fig. 4.

5.2 Hybrid A*

The extended tangent graph $\mathcal{G}$ is a 2D topological map that accounts for some opportunities of movement in the 3D space, such as going above, below, or through an obstacle. In our prototype planner, this graph is exploited by a 3D path planning algorithm called Hybrid A* (HA*). HA* is an extension of the A* algorithm that can handle a form of continuous state variables. It can also be seen as a deterministic special case of the HAO* algorithm (Meuleau *et al.* 2009). Table 1 summarizes the relations between these algorithm.

	discrete states	**hybrid states**
deterministic	A*	HA*
stochastic	AO*	HAO*

Table 1: Relationship between algorithms.

The basic principle of HA* is to reason about hybrid states $s = (v, h)$, where v is a vertex of $\mathcal{G}$ and $h \in \mathbb{R}$. Being in state (v, h) represents being at location v and altitude h. States are called hybrid because they have a discrete component v and a continuous component h. The 3D path planning problem is formalized as the problem of finding a sequence of hybrid states leading from the plane to the targeted runway, which are all particular hybrid states. An important characteristic of this problem is that there are (uncountably) infinitely many states that are reachable from a given position, which must be accounted for during optimization.

HA* addresses this issue by computing finite partitions of the (infinite) hybrid-state space. Then it performs standard A* search in the space of partitions. The partitions are built on the fly, as the search progresses.

Formally, HA* associates with each vertex v of $\mathcal{G}$ a finite set of intervals $\mathcal{R}_v = \{(l_i, u_i), i = 1, 2, \ldots, k_v\}$ representing the altitudes at which v can be reached from the current position of the plane. For convenience, we denote an altitude interval by the triple $[v, l, u]$ where v is the vertex of $\mathcal{G}$ to which the interval is attached, and l and u are the bounds on altitude.

The sets $\mathcal{R}_v$, $v \in \mathcal{V}$, can be computed incrementally, by propagating altitude intervals into $\mathcal{G}$. An interval I_0 is created to represent the initial situation of the plane: if v_0 is the vertex representing the plane and h_0 is the current altitude, then $I_0 = [v_0, h_0 - \epsilon/2, h_0 + \epsilon/2]$, where ϵ is any very small positive number. This seed is added to $\mathcal{R}_{v_0}$ and then pushed through every edge starting in v_0, which creates new altitude intervals that are propagated through the graph in turn.

Given an edge $e = (v, v')$ of length $d(v, v')$ and a tunnel $(f, c) \in \mathcal{U}_e$, *pushing* an altitude interval $[v, l, u]$ through (f, c) consists of computing a new altitude interval $[v', l', u']$ such that:

$$u' = \min\{\min\{u, c\} - \rho_n d(v, v'), c\}$$
$$l' = \max\{l - \rho_x d(v, v'), f\} \tag{1}$$

where $\rho_x > 0$ is the maximum descent rate of the plane and ρ_n its minimum descent rate. (If the plane can still climb, then $\rho_n < 0$.) The first line of Eqn. 1 may be understood as follows: if the set of reachable altitudes in v is (l, u),

Algorithm 1 Hybrid-A* for 3D shortest path

1: *// Initialization*
2: Create $I_0 = [v_0, h_0 - \epsilon/2, h_0 + \epsilon/2]$ representing the plane. Set $g(I_0) = 0$. Compute $h(I_0)$ as the Euclidean distance from v_0 to v_g. Set $f(I_0) = g(I_0) + h(I_0)$. Add I_0 to OPEN.
3: *// Main Loop*
4: **while** OPEN $\neq \emptyset$ **do**
5:     Pick $I = \arg\max_{I' \in \text{OPEN}} [f(I')]$.
6:     **if** the target is included in I **then**
7:         **return**(success).
8:     **end if**
9:     Remove I from OPEN, add I to CLOSED.
10:     **for** all edges $e = (v, v') \in \mathcal{G}$ such that $g(I) + d(v, v')$ is lesser than the plane range **do**
11:         **for** all tunnels $(f_i, c_i) \in \mathcal{U}_e$ **do**
12:             Compute interval $I' = [v', l', u']$ obtained by pushing I through (f_i, c_i) using Eqn. 1.
13:             **if** I' is consistent ($l' <= u'$) **then**
14:                 Set $g(I') = g(I) + d(v, v')$. Compute $h(I')$ as the Euclidean distance from v' to v_g. Set $f(I') = g(I') + h(I')$.
15:                 MERGE(I').
16:             **end if**
17:         **end for**
18:     **end for**
19: **end while**
20: **return**(failure).

then when we enter the tunnel (f, c) leading from v to v', our maximum altitude is $\min\{u, c\}$. During the traversal of the tunnel, our altitude may increase by the amount $-\rho_n d$ at most (which may be positive or negative depending on ρ_n). Finally, our altitude when we exit the tunnel can not exceed c. The second line of Eqn. 1 is derived from a symmetric equation:

$$l' = \max\{\max\{l, f\} - \rho_x d, f\}$$

taking into account the fact $\rho_x > 0$. Note that if the new interval is inconsistent ($l' > u'$), then it is discarded.

Pushing an altitude interval (l, u) through an edge e of $\mathcal{G}$ consists of pushing it through every tunnel of $\mathcal{U}_e$, and then taking the union of the resulting intervals. It creates at most as many interval as there are tunnels in $\mathcal{U}_e$. This operation is the basis of the reachability analysis performed by HA*.

The HA* algorithm is presented in Alg. 1. It is a very similar to standard A*, the main difference being that vertices $v \in V$ are replaced by altitude intervals. It impacts the algorithm in the following ways:

- The OPEN and CLOSED lists contain altitude intervals instead of nodes. The algorithm stops when the target *is included in* the most promising interval picked from OPEN (line 6).

- Instead of listing all possible successors of a node, HA* pushes an interval through an edge to get all its successor intervals (lines 11 to 13).

- When a new node is created, regular A* goes through a series of tests to check whether this node is already present in OPEN or CLOSED. In HA*, the situation is

Algorithm 2 MERGE(I), where $I = [v, l, u]$

1: *// Intersection with intervals in* OPEN
2: **for** all intervals $I' = [v, l', u'] \in$ OPEN such that $(l, u) \cap (l', u') \neq \emptyset$ **do**
3:     Call I'' the interval of v defined by $(l, u) \cap (l', u')$.
4:     **if** $g(I) < g(I')$ **then**
5:         Replace I' with $I' \setminus I''$ in OPEN.
6:         Set $g(I'') = g(I)$ and $h(I'') = h(I)$.
7:         Add I'' to OPEN.
8:     **end if**
9: **end for**
10: *// Intersection with intervals in* CLOSED
11: **for** all intervals $I' = [v, l', u'] \in$ CLOSED such that $(l, u) \cap (l', u') \neq \emptyset$ **do**
12:     Call I'' the interval of v defined by $(l, u) \cap (l', u')$.
13:     **if** $g(I) < g(I')$ **then**
14:         Replace I' with $I' \setminus I''$ in CLOSED.
15:         Add I'' to OPEN.
16:         Set $g(I'') = g(I)$ and $h(I'') = h(I)$.
17:     **end if**
18: **end for**
19: *// Intervals intersecting with none of* OPEN *and* CLOSED
20: Compute the set difference of I and every interval in OPEN and CLOSED. The result is a set of altitude intervals called Δ.
21: **for** all intervals $I' = [v, l', u'] \in \Delta$ **do**
22:     Add I' to OPEN.
23:     Set $g(I') = g(I)$ and $h(I') = h(I)$.
24: **end for**
25: **delete**(I);

more complex because a newly created interval I may intersect partly with intervals in OPEN, partly with intervals in CLOSED, and partly with none of the two. Therefore, the test is replaced by a call to the function MERGE, described in Alg. 2. This procedure ensures that each hybrid state included in the new interval I receives the same treatment as discrete states in standard A*.

HA* exhibits the same convergence properties as A*, that is, it is guaranteed to be optimal if the heuristic is admissible. A proof of this statement is obtained by adapting the proof of convergence of HAO* (Meuleau *et al.* 2009) to the particular case of deterministic problems.

6. Results, Status and Issues

We have tested our initial prototype planner in standalone fashion on a number of scenarios generated using real and synthetic weather data. Figure 5 shows a screen shot of the results for an aircraft over northeastern Arizona. In the map on the left side, the aircraft position is represented by the small triangle in the center, and there are various terrain and weather obstacles depicted. The right hand side provides a ranked list of the best options for an emergency landing. For each option, the airport and runway are given, along with runway length, distance, and any significant risk factors. In this case, the highest ranking option is for the aircraft to proceed around the line of thunderstorms to its west and land on runway 03 at Flagstaff (KFLG), as depicted on the map. Runway 21 at Flagstaff is closer and has more favorable

winds, but would require flying through the thunderstorm line to get to the approach point. Likewise, Grand Canyon (KGCN) runway 21 is also slightly closer and longer, but would again necessitate flying through a line of thunderstorms. Figure 6 shows what happens to the path to Flagstaff runway 03 when the range of the aircraft is reduced. In this case, the only way to reach this runway involves flying through the thunderstorm line, which increases risk. As a result, the best options are now the runways at Winslow (KINW), as shown in the list in Figure 7. A similar result occurs when the controllability of the aircraft is reduced; in that case Winslow is preferred because the paths are more direct and involve less turning.

Our performance objectives for the Emergency Landing Planner are to provide these option lists within ten seconds of any significant change to the state and controllability of the aircraft, and to continually update the lists as the aircraft position and state evolve. Our preliminary testing indicates that we are within range of these objectives and expect that some minor optimization will allow us to meet them.

The Emergency Landing Planner and the other components shown in Figure 1 are being integrated into a full motion 767 simulator for testing with a number of different damage models and scenarios. In early 2010, a small group of airline pilots will be subjected to several different damage scenarios with and without the assistance of the adaptive controller and the Emergency Landing Planner. We intend to record objective data on pilot performance, and also ask the pilots to provide subjective assessments of the systems and interfaces. There are a number of user interface issues associated with presenting alternatives to the pilot. For example, how many alternatives should we present at a time? Should we give pilots the ability to modify the ranking criteria, and if so how? In an emergency, pilots have little attention to spare, and user interfaces need to be simple and intuitive. We do not have ready answers to these questions, and ultimately this will require input from pilots and other experts on human factors in the cockpit.

While our path planning algorithm performs well, it is not guaranteed to find the optimal path in all cases. In particular, the augmented visibility graph contains only a subset of the possible ways of going above, below, and through obstacles. In addition, the algorithm largely ignores aircraft dynamics. It gets away with this by expanding the size of obstacles to allow for the turning and control limitations of the aircraft. In most cases, this works fine, but if the aircraft is only able to turn in one direction, a large looping path may be necessary to get to one or more landing sites. In these situations, probabilistic road map (PRM) techniques seem to be more effective. We are considering a hybrid approach in which the initial roadmap is constructed using the augmented visibility graph, probabilistic techniques are used to further augment this roadmap, and hybrid A* is used to search the resulting graph. Ideally this would allow us to construct the roadmap quickly, take aircraft dynamics into account, but do a better job of searching for good solutions than is possible using only PRM techniques
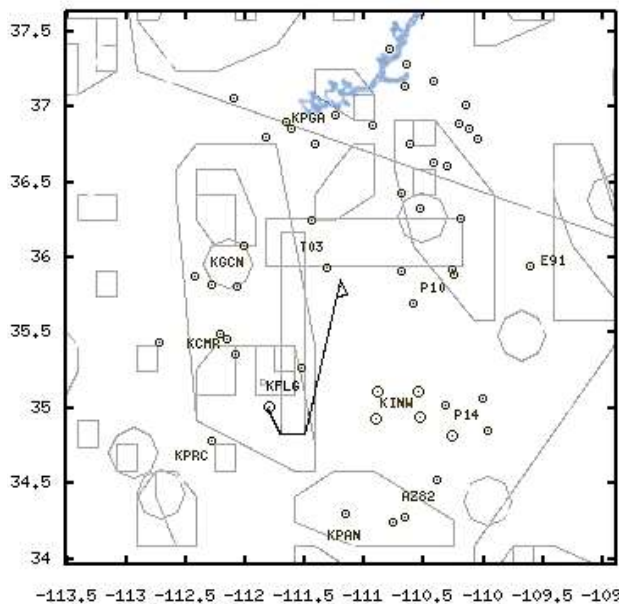
RUNWAY	LEN.	DIST.	SCORE	RISK		
KFLG-03	8800	87.488	0.916			
KINW-04	7484	59.454	0.895			
KINW-22	7484	59.869	0.895			
KINW-29	7086	70.019	0.886			
KINW-11	7086	48.238	0.839			
P14-03	6697	84.909	0.643	RWi	Fcl	
KFLG-21	8800	38.935	0.621	Enr		
P14-21	6697	86.272	0.596	Fcl	RWi	
T03-33	6247	14.646	0.512	Fcl	RWi	RLg
KPAN-24	5487	91.511	0.438	Fcl	RLg	RWi
E91-36	6909	95.994	0.382	Fcl	RWi	
P14-29	3256	98.585	0.245	RLg	Fcl	
P14-11	3256	72.141	0.242	RLg	Fcl	
KGCN-21	8992	74.181	0.240	Enr		
KGCN-03	8992	67.014	0.236	Enr		
KPRC-21L	7546	95.311	0.212	Enr		
KCMR-18	6001	93.366	0.197	Enr	RLg	
KPGA-33	5953	96.508	0.185	Enr	RLg	Fcl
AZ82-21	3419	92.931	0.109	Fcl	RLg	RWi
P10-04	4224	38.298	0.106	Fcl	RWi	RLg
P10-22	4224	60.119	0.104	Fcl	RWi	RLg
AZ82-03	3419	99.343	0.091	Fcl	RLg	RWi

Figure 5: A display of possible emergency landing sites for a damaged aircraft (small center triangle) over central Arizona. Sites are ranked according to decreasing risk, with the primary risk factors for each possibility listed. Risks depicted above include runway length (RLg), runway width (RWi), facilities (Fcl), and en route weather (Enr).

Figure 6: Path to Flagstaff when range is decreased
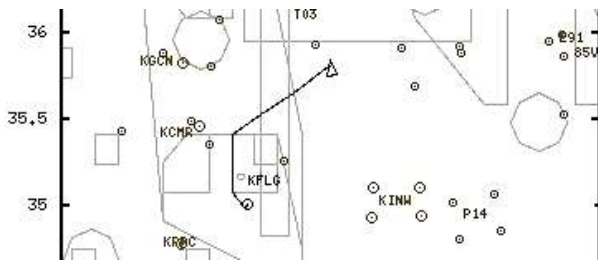
RUNWAY	LEN.	DIST.	SCORE	RISK		
KINW-04	7484	59.454	0.895			
KINW-22	7484	59.869	0.895			
KINW-29	7086	70.019	0.886			
KINW-11	7086	48.238	0.839			
P14-03	6697	84.909	0.643	RWi	Fcl	
KFLG-21	8800	38.935	0.621	Enr		
T03-33	6247	14.646	0.512	Fcl	RWi	RLg
P14-11	3256	72.141	0.242	RLg	Fcl	
KGCN-21	8992	74.181	0.240	Enr		

Figure 7: Top options when range is decreased.

Acknowledgments

References

Burcham, F. W.; Maine, T. A.; Fullerton, C. G.; and Webb, L. D. 1996. Development and flight evaluation of an emergency digital flight control system using only engine thrust on an F-15 airplane. Technical Report TP-3627, NASA.

Burken, J. J., and Burcham, F. W. 1997. Flight-test results of propulsion-only emergency control system on MD-11 airplane. *J. Guidance, Controls and Dynamics* 20(5):980–987.

Choset, H.; Lynch, K.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.; and Thrun, S. 2004. *Principles of Robotic Motion: Theory, Algorithms, and Implementation*. Cambridge, MA: MIT Press.

Gundy-Burlet, K.; Krishnakumar, K.; Limes, G.; and Bryant, D. 2004. Augmentation of an intelligent flight control system for a simulated C-17 aircraft. *JACIC* 1(12):526–542.

Liu, Y., and Arimoto, S. 2004. Computation of the tangent graph of polygonal obstacles by moving-line processing. *IEEE Trans. on Robotics and Automation* 823–830.

Meuleau, N.; Benazera, E.; Brafman, R.; Hansen, E.; and Mausam. 2009. A heuristic approach to planning with continuous resources in stochastic domains. *JAIR* 34:27–59.

NTSB. 1990. Aircraft accident report – United Airlines flight 232. Technical Report NTSB/AAR-90/06, National Transportation Safety Board.