Simulation of a Fixed-wing UAV Forced Landing with Dynamic Path Planning

Pillar Eng, Dr. Luis Mejias, Prof. Rodney Walker
Australian Research Centre for Aerospace Automation (ARCAA)
School of Engineering Systems,
Queensland University of Technology
p.eng@qut.edu.au, luis.mejiasalvarez@qut.edu.au, ra.walker@qut.edu.au
Dr. Daniel Fitzgerald
Australian Research Centre for Aerospace Automation (ARCAA)
Autonomous Systems Laboratory
CSIRO ICT Centre
daniel.fitzgerald@csiro.au

Abstract

This paper describes the current status of a program to develop automated forced landing techniques for a fixed-wing Unmanned Aerial Vehicle (UAV). The paper outlines two dynamic path planning algorithms that were developed based on processes used by human pilots in forced landings. To evaluate the performances of these algorithms, a simulation environment was created using a non-linear 6 degree-of-freedom aircraft model. The simulation also modelled prevailing wind conditions which are a major factor in the forced landing planning process. Results from Monte Carlo testing demonstrate that the second algorithm was able to land the UAV inside the designated landing area with a success rate of 52%. This is twice the success rate of the first algorithm. The results of the Monte Carlo tests will serve as a benchmark for further refinements to the second algorithm, such that it can be implemented in an autonomous UAV forced landing system.

1 Introduction

In recent years, interest has grown in using Unmanned Aerial Vehicles (UAVs) in civilian applications such as border patrol, search and rescue and highway traffic monitoring. One of the major hurdles to successfully integrating UAVs into routine civilian use is their inability to demonstrate a level of safety acceptable to the public. This is particularly important for engine failures that require the UAV to make a forced landing. While there has been extensive research into automated landing strategies for unmanned spacecraft and helicopter UAVs [Johnson et al., 2000; Serrano et al., 2005; Shakernia et al., 1999; Sharp et al., 2001], the problem of landing an unpowered, fixed-wing UAV autonomously in an unknown environment is still a relatively new area of investigation.

Simulated aircraft forced landings, as practiced by human pilots, involve visual estimation of wind conditions and recognition of appropriate landing sites. The pilot assesses this information and makes a descent plan in real-time based on his/her experience in the landing process. The problem is further complicated by the fact that the aircraft is in a continual descent and the wind conditions can vary at different altitudes. Finally, the pilot must avoid buildings, fences, powerlines and other obstacles in the flight path.

As an initial step in automating this process, a machine vision-based algorithm has been developed that selects candidate landing sites from the air [Fitzgerald, 2006], based on images obtained from a single camera. In addition, a team at the Australian Research Centre for Aerospace Automation (ARCAA) is currently working towards developing a prototype forced landing flight system [Fitzgerald et al., 2007]. A primary focus of this system is to provide a dynamic path planning capability for a UAV performing a forced landing in changing wind conditions. To investigate this problem, as well as to minimize the time, cost and risk associated with conducting experiments on a real UAV, a forced landing simulation was developed using MATLAB and FlightGear. This paper describes the algorithms and methodologies used in the simulation design, and provides an indication of the performance of the system as tested by Monte Carlo analyses. This simulation is intended to serve as a tool in the design and testing of a visual servoing and path planning system for automating a fixed-wing UAV forced landing. It will be further enhanced to model complex, uncooperative environments with hazards such as buildings, trees, light poles and undulating terrain, as well as machine vision for use in the feedback control loop.

2 Related Work

Autonomous landing systems for unmanned spacecraft and UAVs have been active research areas for several years. In the majority of the research, machine vision has been used as the primary sensor to perform the landings.

In work done at the NASA Joint Propulsion Laboratory [Johnson et al., 2000], machine vision algorithms were used to locate hazard-free landing areas and to provide estimates of the spacecraft position. This enabled the spacecraft to be guided to land safely. In addition, a technique combining available information from different visual sensors (camera, radar and lidar) is reported by [Serrano, et al., 2006]. This technique allowed an accurate hazard map to be constructed, which could then be used to aid in safe spacecraft landing.

In [Saripalli et al., 2003], the landing algorithm for an autonomous helicopter was integrated with algorithms for helipad acquisition and navigation from an arbitrary initial position and orientation. Machine vision was used for helipad detection and recognition, while a combination of vision and GPS was used for navigation. In [Johnson et al., 2005], structure from motion was used to generate a dense elevation map of the landing area, which then allowed hazards to be detected and a safe landing site to be selected. The experiment was conducted onboard an unmanned helicopter, and demonstrated that an autonomated landing in hazardous terrain without a structured landing aid was possible.

In [Fitzgerald, 2006], machine vision using a single camera was used to select candidate sites for a fixed-wing UAV forced landing. The site selection algorithm was tested on aerial images taken over south-east Queensland, and demonstrated a 99% confidence interval in locating obstacle-free sites suitable for landing. The use of computer vision for a simulated, hovering UAV forced landing is reported in [Mejias et al., 2006], where the UAV had to avoid powerlines and then seek a safe landing area amongst hazards on the ground, without the aid of a structured landmark such as a helipad. More recently, in work done at UC Berkeley [Templeton et al. 2007], vision-based terrain mapping was combined with a model predictive flight control system such that the helicopter UAV could identify hazards in unknown terrain, then generate plausible trajectories to a safe landing area before tracking the one that yields minimal cost from a reference trajectory.

In [Theodore et al., 2005], a full mission simulation of a helicopter UAV for landing in a non-cooperative environment is presented. The simulation modelled the helicopter dynamics, as well as winds, trees, buildings and other hazards on the ground. It served as an engineering tool for evaluating and testing invidual sensors and payloads prior to the actual flight tests.

The simulation described in this paper differs from those presented above in that it attempts to model human pilot reasoning in planning the forced landing flight path, while taking into account changing wind conditions. The path planning algorithm has been developed from consultation with general aviation (GA) pilots, and is based on the emergency landing procedure outlined in the Civil Aviation Safety Authority Australia (CASA) Visual Flight Rules (VFR) flight guide [CASA, 2001].

3 Forced Landing Simulation

3.1 Simulation Environment

The main component of the forced landing simulation was developed using the AeroSim blockset in MATLAB Simulink. The AeroSim blockset , which is a commonly used computer program in the aerospace industry, provides a comprehensive set of tools for the rapid development of non-linear 6-degree-of-freedom (6 DOF) aircraft models. Using this program, a basic model of an Aerosonde UAV was modified and expanded to include blocks for flight controls, path planning, GPS waypoint navigation, wind generation, wind correction and interfacing to FlightGear. By running MATLAB and FlightGear concurrently, the user is able to visualize the UAV flying in a manner as dictated by the Simulink model. Noise and sensor errors have not been introduced in order to reduce complexity, and a simplified MATLAB/Simulink model is shown in Figure 1.

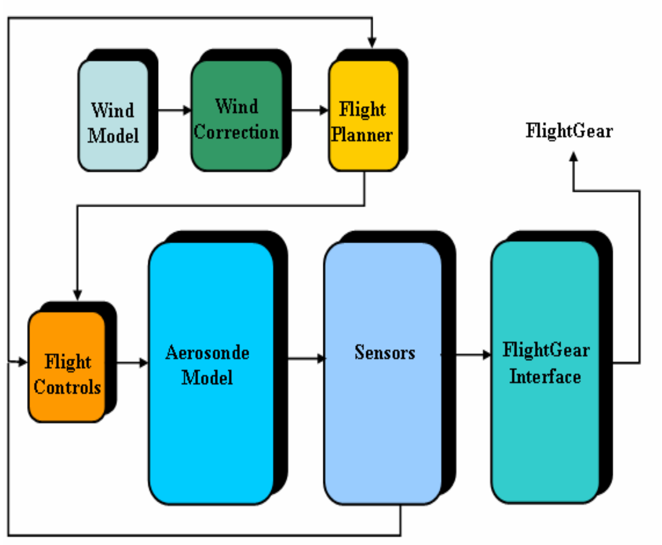

Figure 1 – Simplified Simulink Model

The following sections in this paper will describe the operations of the model in more detail.

3.2 World Model

The operating environment for the UAV is defined as a (3000m x 3000m x 3900ft; 1.62nmi x 1.62 nmi x 3900ft) volume of air space. The UAV can assume any initial bearing and position (latitude, longitude and altitude) that lies within this area. The initial altitude of the UAV Above Ground Level (AGL) is constrained between 1000ft to 3900ft, with the lower bound being the minimum altitude that an aircraft can assume while flying enroute over populated areas, as defined by CASA. The boundaries of the earth's surface are given by four coordinates that lie between 27˚23.4'S to 27˚25.8'S and 153˚6'E to 153˚7.8'E, with an altitude of 13ft above Mean Sea Level (MSL). These boundaries define an area of approximately (3000m x 3000m; 1.62nmi x 1.62nmi) around the location of Brisbane International airport, given by the landing site aim point coordinates as 27˚24.6'S, 153˚7.2'E. The landing site is assumed to be a (100m x 600m; 0.05nmi x 0.32nmi) rectangle. A diagram illustrating the world model used in the simulation is given in Figure 2.

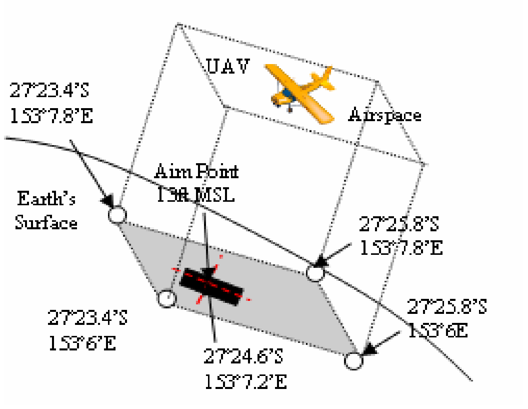

Figure 2 – Simulation World Model

The simulation uses a 6 DOF aircraft dynamic model provided in the AeroSim library. The equations of motion are implemented in body-axes and the model parameters, such as the initial position and bearing are read from a user-configurable MATLAB mat-file. The model outputs the aircraft state information, such as the aircraft velocities and angular velocities in 3-axes, which can be used as feedback to control the aircraft.

3.3 Wind Model

The wind model randomly generates the input wind velocity components in the navigation frame: W_N, W_E and W_D, giving the profiles shown in Figure 3. The wind components are used by the Aerosonde model (Figure 1) to calculate the effects of the resultant wind velocities incident on the aircraft, including wind shear and turbulence. Note that gusts have not been modelled in the simulation, instead, the input wind is assumed to blow with a constant magnitude and direction for 60 seconds, before changing magnitude and direction for the next 60 seconds. Whilst this does not necessarily represent the wind conditions found in an actual descent, it does present a challenging wind shift scenario for the simulation. The values of the W_N, W_E and W_D components were limited between ±14, ±8 and ±2kts respectively, with the coordinate system defined such that positive is towards North and East. These values were chosen based on the actual wind rose measurements for Brisbane, Australia, and correspond to a maximum wind speed of 60kts. A wind rose is a diagram that summarises the occurrence of winds at a location, showing their strength, direction and frequency. The wind rose used in the simulation represented wind measurements taken at 9 a.m. daily, from 1950 to 2000. More information on wind roses can be found on the Australian Government Bureau of Meteorology website.[1]

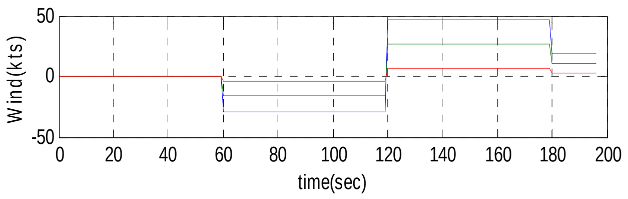

Figure 3 – Wind components (W_N: Blue, W_E: Green, W_D: Red). These components are used to compute the resultant wind vector incident on the UAV.

3.4 Wind Correction

The wind correction block (Figure 1) uses principles of the wind triangle [Robson et al., 2002] to calculate the wind correction angle, which is then compared with the current aircraft heading and passed as an input to the flight planner. The wind triangle is an analytical tool, commonly applied by GA pilots to compute the desired track to fly in the presence of winds. From Figure 4, suppose that waypoint B is 600m (0.32 nmi) north-east (045° true) of waypoint A and the UAV glides from A to B, maintaining a heading of 045° true and a constant true airspeed of 37kts. A wind velocity of 340°/9.7kts coming from the south-east will cause the UAV to drift to the left.

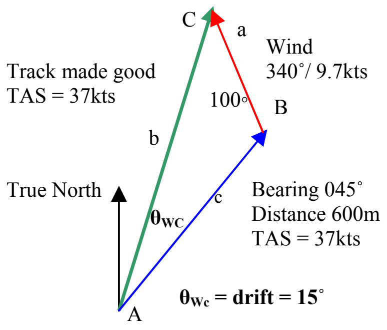

Figure 4 – Wind Triangle

The drift angle can be calculated using the law of sines as:

$$\frac{a}{\sin\theta_{Wc}} = \frac{b}{\sin B} \Rightarrow \theta_{Wc} = \sin^{-1}\left(\frac{9.7\sin 100}{37}\right) \approx 14.96^{\circ} \qquad (1)$$

This implies that the wind correction angle supplied to the flight planner must be 15° in the opposite direction, such that the "track made good" will converge on the required track to target.

3.5 Waypoint Navigation

Flat-Earth approximations are sufficient for the distances described in this simulation, with a range error of 6.67 x 10^4 m (36 nmi) at a range of D = 1.85 x 10^6 m (1000 nmi), an azimuth BT = 45 degrees and a latitude of Φ = 45 degrees [Kayton and Fried, 1997]. However, these approximations were not used as a basic navigational algorithm using the great-circle navigation method [Kayton and Fried, 1997] had already been designed by Duncan Greer and Troy Bruggemann of ARCAA [Bruggemann et al., 2005], and it was decided to expand on their algorithm to reduce the time taken for development. Using this method, the great circle track, which is the shortest distance between two points located on a sphere surrounding the earth, as well as the bearing between the two points can be calculated. Hence, a flight path consisting of a sequence of waypoints is traversed by flying a series of direct, curved paths to each successive point. From spherical trigonometry, the range D and bearing B_T to the target are given by [Kayton and Fried, 1997]:

1 www.bom.gov.au

$$D = R_G \cos^{-1}\left(\sin\Phi \sin\Phi_t + \cos\Phi \cos\Phi_t \cos(\lambda - \lambda_t)\right) \tag{2}$$

$$B_T = \sin^{-1}\left(\frac{\cos\Phi_t}{\sin(D/R_G)}\sin(\lambda - \lambda_t)\right)$$

Where R_G is the Gaussian radius of curvature at the current aircraft position, given by:

$$R_G = \sqrt{R_M \times R_P} \tag{3}$$

And R_M and R_P are the meridian and prime radii at the current aircraft location. The current aircraft latitude and longitude as measured by the onboard GPS are given by Φ and λ and the target latitude and longitude are given by Φ_t and λ_t. This implies that the calculated range to the target is actually the "distance-to-go" to the target, since the aircraft position is constantly changing. This characteristic of Equation 2 is particularly important when designing a tracking algorithm that will constantly adjust for the aircraft position (see §3.7). The UAV heading command was calculated using the following equation:

$$\varphi_{Cmd} = \varphi_T - \theta_{Wc} - B_T + k\theta_{TE} \tag{4}$$

Where φ_T is the aircraft's true heading, θ_{Wc} is the wind correction angle, B_T is the true bearing to the target waypoint and θ_{TE} is the angle of track error, multiplied by a constant, k. The value of k was determined empirically, such that the cross-track error was minimised. Both waypoint navigation and path planning (see §3.6) are handled by the flight planner (Figure 1).

3.6 Path Planning - Overview

The major function of the flight planner is to generate the flight path to the landing site. The forced landing pattern, as outlined in [CASA, 2001] was used to generate the initial target waypoints. In the forced landing configuration a cone, having the dimensions shown in Figure 5-A, is defined as the airspace to which the unpowered aircraft can fly in no wind. Assuming a general aviation aircraft lift-to-drag ratio of 9:1, the conservative angle of 10 degrees is calculated by:

$$\theta = \tan^{-1}\left(\frac{1}{9}\right) \approx 6.3^\circ \approx 10^\circ \tag{5}$$

The glide range is then the radius of a circle on the Earth's surface which contains all attainable landing sites (Figure 5-C). Note that in the current simulation, only one landing site has been assumed. With the wind taken into account, the cone is inverted as shown in Figure 5-B, and the waypoints for the flight planner are selected such that they lie within the cone, implying that they can be attained by the unpowered aircraft. By navigating to these waypoints in succession, the UAV is able to land on the selected site.

Two path planning strategies were developed for the forced landing simulation. The first attempts to guide the aircraft along a predetermined circuit to the landing site as shown in Figure 5-B. The waypoints for the circuit were chosen such that they lie within a fixed glide range, and the UAV then attempts to track towards the waypoints using the great-circle navigation method, while correcting for the wind to remain on course. The second strategy attempts to follow the standard landing circuit, however wind information is also continually assessed to determine its effect on the glide range, and the target waypoints are then adjusted accordingly to maximise the aircraft's ability to reach the aim point. As before, the great-circle navigation and wind correction methods are used while flying enroute. Both path planning strategies assume that the UAV is trimmed for a best glide speed of 37 kts, corresponding to a lift-to-drag ratio of 9:1 and a positive pitch attitude of approximately 3 degrees for landing. The waypoint coordinates for a standard right-hand circuit pattern are given in Table 1, and their relation to the landing site is depicted in Figure 6. In this figure, several possible flight paths using a combination of these waypoints are also shown, as indicated by the red, green and blue curves. A similar set of waypoint coordinates for a left-hand circuit pattern can also be generated.

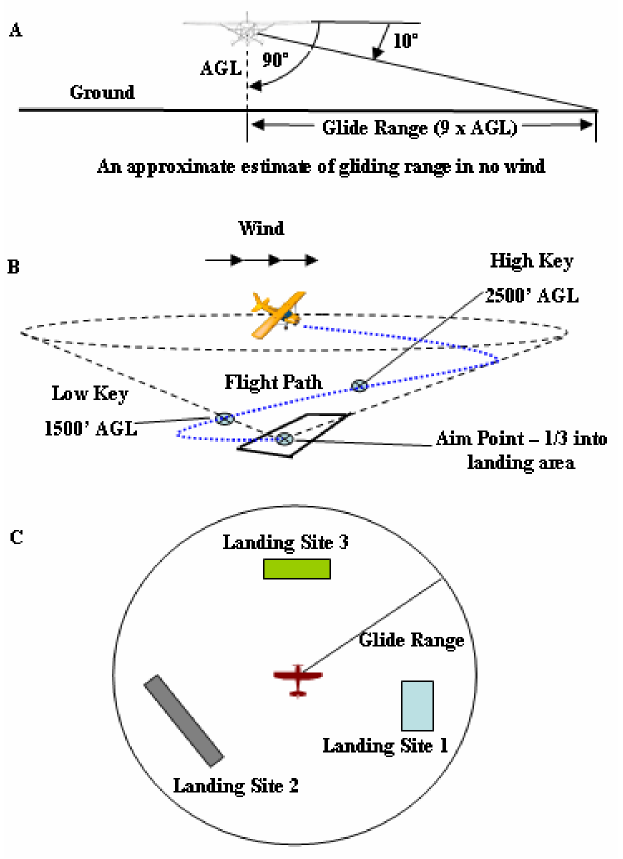

Figure 5 – Determination of Glide Range and Waypoints for a Forced Landing

Waypoint	Φ (rads)	λ (rads)	Alt (ft)
High Key (HK)	0.4782	2.6725	2500
Low Key (LK)	0.4783	2.6722	1700
End Base (EB)	0.4786	2.6721	1200
Decision Height(DH)	0.4786	2.6723	670
Overshoot (OS)	0.4787	2.6724	400
Aim Point (AP)	0.4784	2.6725	13

Table 1 – Waypoint Coordinates for Figure 9

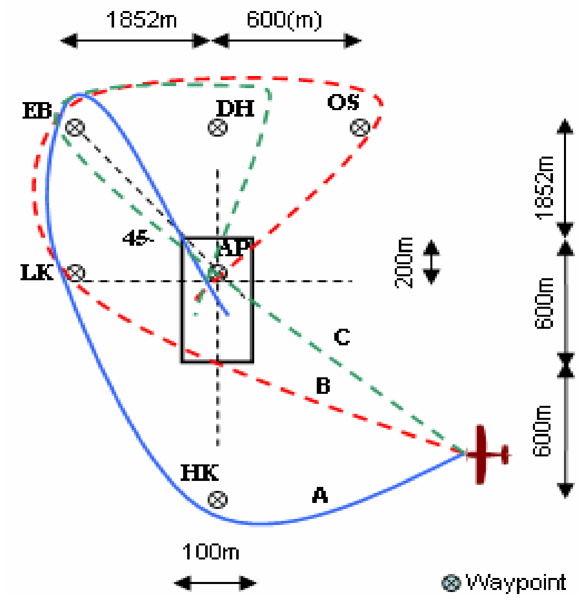

Figure 6 – Standard RHC Forced Landing Pattern

Path Planning – Algorithm 1

In this algorithm, a lift-to-drag ratio of 7:1 is chosen in favour of the GA ratio of 9:1 to account for strong winds, with a maximum head wind of 60kts. Should the aircraft encounter such a headwind, it will be blown off course. However, since the wind is commanded to change every minute, this may present an opportunity for the UAV to steer itself back on course. Of course, if the altitude of the UAV is insufficient to perform such a manoeuvre, or a sudden, strong head wind is encountered near the ground, the UAV will be unable to land at the designated landing site. This is an unavoidable limitation as a constant glide speed has been assumed.

The new ratio of 7:1 is used in calculating the first threshold slant range distance from the UAV to the aim point. Using the theorem of Pythagoras, the slant range distance, SR is given by:

$$SR = \sqrt{Alt^2 + 7.Alt^2} \qquad (6)$$

Where Alt is the initial altitude of the aircraft at the start of a forced landing. From the calculated SR value, seven other threshold values are created. These values in turn define a set of four threshold boundaries for the slant range, with each boundary corresponding to a subset of the set of waypoints similar to those listed in Table 1. The boundaries are: SR; SR to SR-1600'; SR-1600' to SR-2300'; SR-2300' to SR-3000'). These values were chosen based on past simulations with the Aerosonde model, and represent a "good approximation" of which waypoints the aircraft can reach, if its initial altitude produces a slant range distance that lies within these boundaries. The actual slant range from the UAV to the aim point is then calculated from Equation 2, using the initial UAV position and the position of the aim point. If this slant range value lies within a particular threshold boundary, then the UAV will navigate along the flight path described by the set of waypoints corresponding to that boundary. However, the UAV is not constrained to fly successively to each waypoint but can, depending on its current altitude above the ground, further define its flight path using all or several of these waypoints. For instance, once the UAV is at the end base point, if it is too low to continue to the overshoot waypoint, as determined by whether the current UAV altitude is within certain threshold limits, it will head for the aim point. In addition, while enroute between the decision height and overshoot waypoints, the UAV can also lose excess altitude by navigating back and forth between these points, meaning that the UAV should be at the right altitude for final approach to the landing site. If the first overshoot waypoint is not enough to lose the required amount of altitude, a second overshoot waypoint can be included in the flight path (compare this with Table 1, in which only one overshoot waypoint has been considered). This implies that a number of different flight paths can be generated from the one initial set of waypoints. In addition, the standard GA forced landing pattern, describing the distances between waypoints as shown in Figure 6 has been slightly reduced for this algorithm, giving the layout presented in Figure 7. This layout was chosen to improve the chances of the UAV being able to reach the aim point.

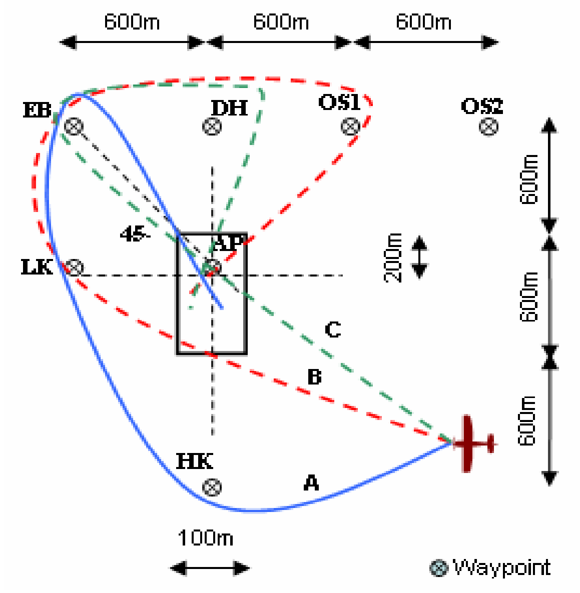

Figure 7 – Modified RHC Forced Landing Pattern

A further measure is built into the path planning algorithm such that if the UAV is initially much higher (>800ft) than the high key point, it will execute a circling descent to lose excess altitude. For this algorithm, only landings without the aid of flaperons have been considered. Thus, while tracking towards the aim point, should wind changes near the ground cause the UAV to have more lift such that it will overshoot the aim point, it is commanded to head for the far threshold of the landing site which is 400m (0.22nmi) away from the aim point. Once at this waypoint, if the UAV is still in the air, it is commanded to navigate back and forth between the aim point and the landing site threshold point until it is on the ground. The UAV can also perform both a right and left-hand circuit pattern. The path planning strategy described above is summarized in Figure 8.

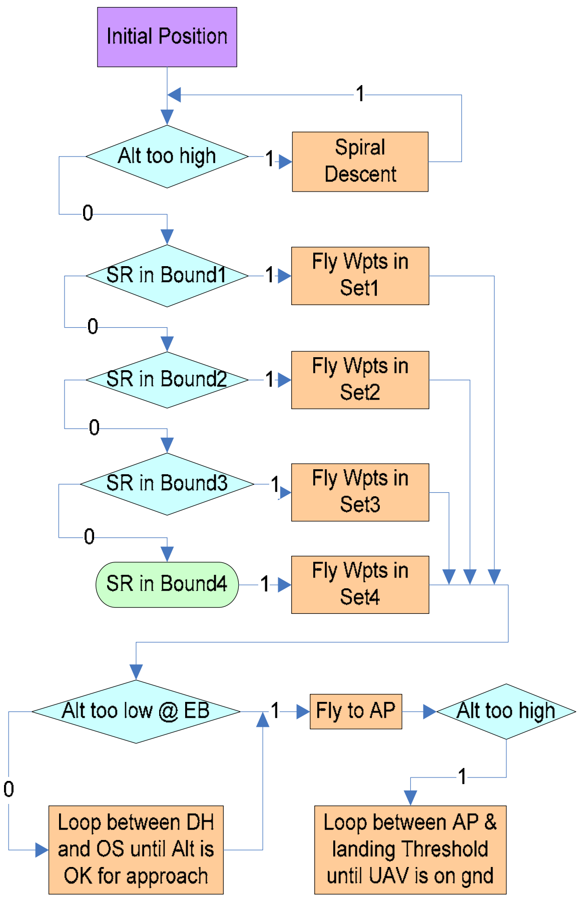

Figure 8 – Algorithm 1 State Transition Diagram

Path Planning – Algorithm 2

Consider the aircraft located at point A in Figure 9, enroute to point D. For argument, assume that a headwind is blowing with velocity V_w against the direction of travel. At time t_0 seconds, the aircraft is at altitude h_0, and it is desired to know whether the aircraft can glide to point D. That is, at time t_n seconds, will the aircraft altitude be greater than or equal to h_n? The gradient corresponding to the nominal lift-to-drag ratio of 9:1 is shown as the dashed, red line (Figure 9). This gradient is used to initialise the path planning algorithm. As the UAV moves through the air however, it will experience longitudinal phugoid motion (solid, blue line) that varies its glide slope, meaning that the nominal gradient is no longer valid for calculations. For simplicity, assume that this oscillatory motion can be linearized for small time epochs (≤10 seconds), as shown by the dashed, green line. Then, in the first epoch, the gradient from A to B is given by:

$$m = \Delta x / \Delta y$$

$$m = \frac{V_g(t_1 - t_0)}{h_0 - h_1} \tag{7}$$

Where V_g is the ground speed of the aircraft, calculated as the sum of the True Air Speed and the component of the wind velocity in the direction of travel:

$$V_g = TAS + V_w \tag{8}$$

If it is further assumed that the gradient is calculated every 10 seconds, using data collected from the previous 10 seconds, then the altitude after the next 10 seconds can be predicted using the equation of a straight line, expressed in general form as:

$$y = mx + c$$

$$Alt_{t_n+10} = \frac{V_g(t_n - t_{n-10})V_g(t_n + 10)}{h_{t_{n-10}} - h_{t_n}} + Alt_{t_n-10} \tag{9}$$

Here, n is an integer of the form 0,1,2...n, and $V_g(t + 10)$ gives the quantity Δx in Figure 9.

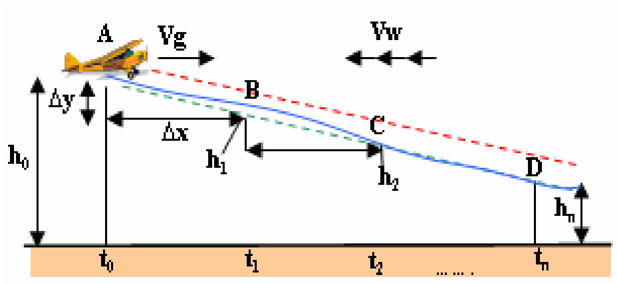

Figure 9 – Altitude Prediction

Should the predicted altitude at each t_{n+10} seconds still exceed that of the targeted waypoint, the UAV will continue to track towards this waypoint, otherwise, it will search for a new waypoint. However, once within a certain threshold distance (200m, 0.11nmi) of the target waypoint, the UAV will start searching for another waypoint. This is to avoid becoming trapped in a local minimum. If the target waypoint is the final waypoint and the UAV is so low over the ground that it cannot reach the other waypoints, then it will simply circle around this waypoint until it lands. By continually predicting the UAV altitude as a function of changes in the wind velocity, a flight path can be described such that it will always contain waypoints to which the UAV can navigate. Using the experience obtained from testing the first algorithm, the waypoints for this algorithm are selected from the standard waypoint locations for a right-hand circuit pattern (see Table 1 and Figure 6), as it is now known that the UAV can navigate to these waypoints and still be able to land on the landing site. A state diagram summarizing this algorithm is depicted in Figure 10. Note that to reduce complexity, waypoints corresponding to a left-hand circuit pattern were not used.

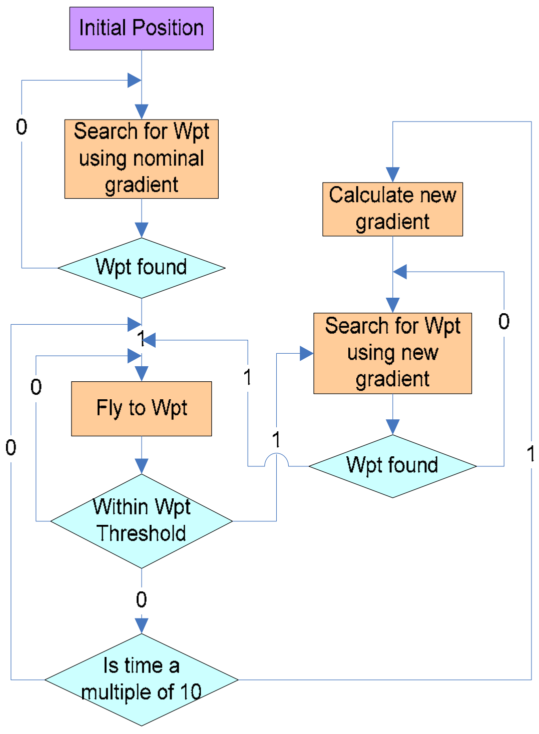

Figure 10 – Algorithm 2 State Transition Diagram

3.7 Waypoint Tracking

Another function of the flight planner is to provide tracking for the required flight path. Suppose that the flight path of the UAV as it is flying from waypoints A to B is given by the solid, black line in Figure 11, and the required track to fly is given by the dashed, red line. At point B' the aircraft is enroute to waypoint B and has a cross track error of XTE m (XTE/1852 nmi). The tracking algorithm calculates the angle of track error, θ_{TE} and multiplies it by a constant, k. The value of k was chosen such that the commanded heading, φ_{Cmd} will cause the UAV flight path to converge on the required track to fly, meaning that XTE would be minimised. The cross-track error is given by:

$$XTE = DTR \sin \theta_{TE} \qquad (10)$$

Where DTR is the track distance and the "distance-to-go" to the next waypoint.

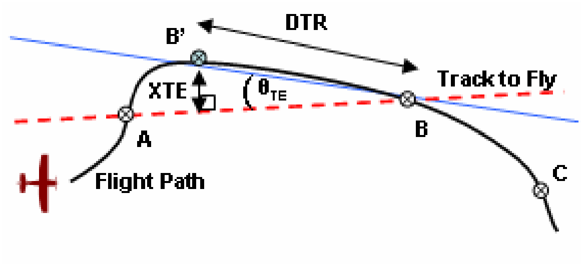

Figure 11 – Flight Path Tracking

3.8 Flight Control

Flight control for the Aerosonde UAV was achieved using a combination of Proportional-Integral (PI) and Proportional-Integral-Derivative (PID) controllers. A PI controller was used for the ailerons and rudder, while a PID controller was used for the elevators as per the recommendations of Nelson [1998]. The bank angle was used to control the ailerons, while the airspeed was used to control the elevators. Note that the aircraft is always constrained to glide at a constant airspeed of 37kts during the flight. The controller gains were tuned using a combination of the Ziegler-Nichols tuning method [Nelson, 1998] and by experimentation.

4 Results and Discussion

Of particular interest to this simulation is how well the two path planning algorithms will perform in varying wind conditions. To demonstrate this, a comparison between simulated forced landings using each algorithm was conducted. The UAV was initially located at (27°24", 153°7", 454') and faced due East.

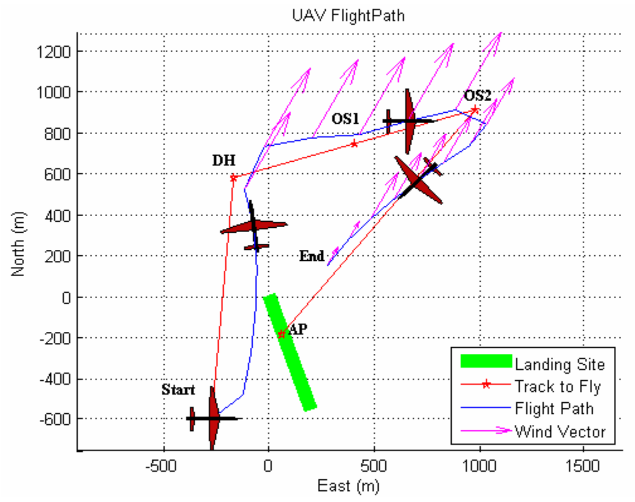

Figure 12 – Forced Landing with Algorithm 1

As can be seen from Figure 12, the UAV was able to navigate the required "track to fly" (red path) using Algorithm 1, while compensating for changes in the wind velocity. In this particular scenario, the wind direction was the same, however the wind magnitudes were different every minute, as can be seen by the varying length of the wind vectors (magenta). Notice also that the UAV could not reach the aim point (AP) due to the strong head winds it encountered while enroute from the second overshoot waypoint (OS 2). Using the same wind shifts for the second simulation, it can be seen from Figure 13 that Algorithm 2 produces a different flight path to that obtained with the first. The UAV initially heads for the end base (EB) while continually assessing the effects of the wind on its glide slope. After it has travelled a certain distance towards the EB point, it recalculates the glide slope gradient and realises that this waypoint cannot be attained. The UAV then seeks for a new waypoint that can be attained with its new glideslope, turning as it does so. While turning, the strong head winds have become tail winds, thus providing the UAV with greater lift and a shallower glide slope. This causes the UAV to initially head for the EB point once again, before finally heading for the aim point (AP). In both tests the UAV did not actually land witin the designated area, however it was able to navigate to a distance within 500m (0.27nmi) of the aim point. Note that the apparent “sharp” turns in both flight paths are due to the large sampling time of 10 seconds used for plotting.

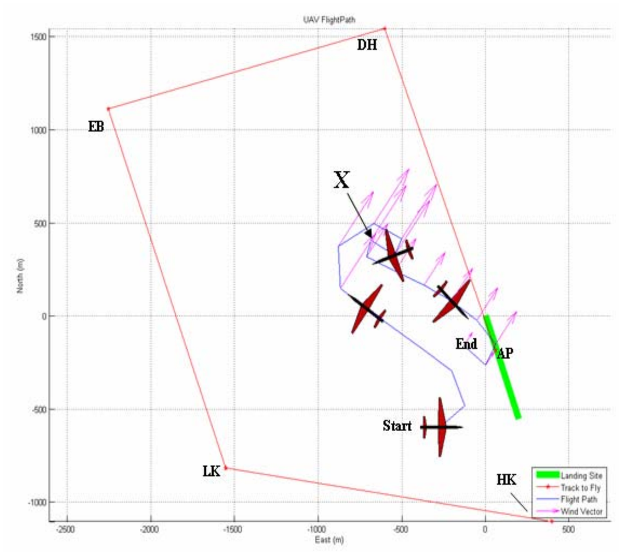

Figure 13 – Forced Landing with Algorithm 2

It should be noted that the performance of the different path planning algorithms cannot be ascertained based on only one set of tests, hence a further 100 simulations were run for each algorithm, with randomised initial aircraft positions, aircraft headings and wind velocities. From the first simulation using Algorithm 1, seven landings had a radial miss distance of greater than 1000m (0.54nmi) from the aim point, and these were excluded in further analysis as they were deemed to be spurious data points. Of the remaining 93 landings, 23 landings were within the site boundaries (confirmed by additional testing), giving a landing success rate of approximately 26%. For the second simulation using Algorithm 2, all landings were within 1000m (0.54nmi) of the aim point, and 52 landings were within the site boundaries. This corresponds to twice the landing success rate of the first algorithm. Although this result is not exemplary, it does represent a good baseline from which to evaluate further improvements to the path planning and control algorithms, such that the landing success rate can be improved. Results from the Monte Carlo simulations are summarized in Figure 14.

By comparing the frequency distributions for both Algorithms 1 and 2 (Figure 14), it can be seen that in general, Algorithm 2 outperforms Algorithm 1. The distribution for Algorithm 2 is skewed towards the 200m (0.11nmi) miss distance, while that for Algorithm 1 tends towards the 400m (0.22nmi) miss distance. The 200m miss distance could be due to the fact that that is the required threshold distance for successfully capturing a waypoint, as defined in Algorithm 2 (see Path Planning – Algorithm 2), while the 400m miss distance could be explained by the fact that the UAV is commanded to fly

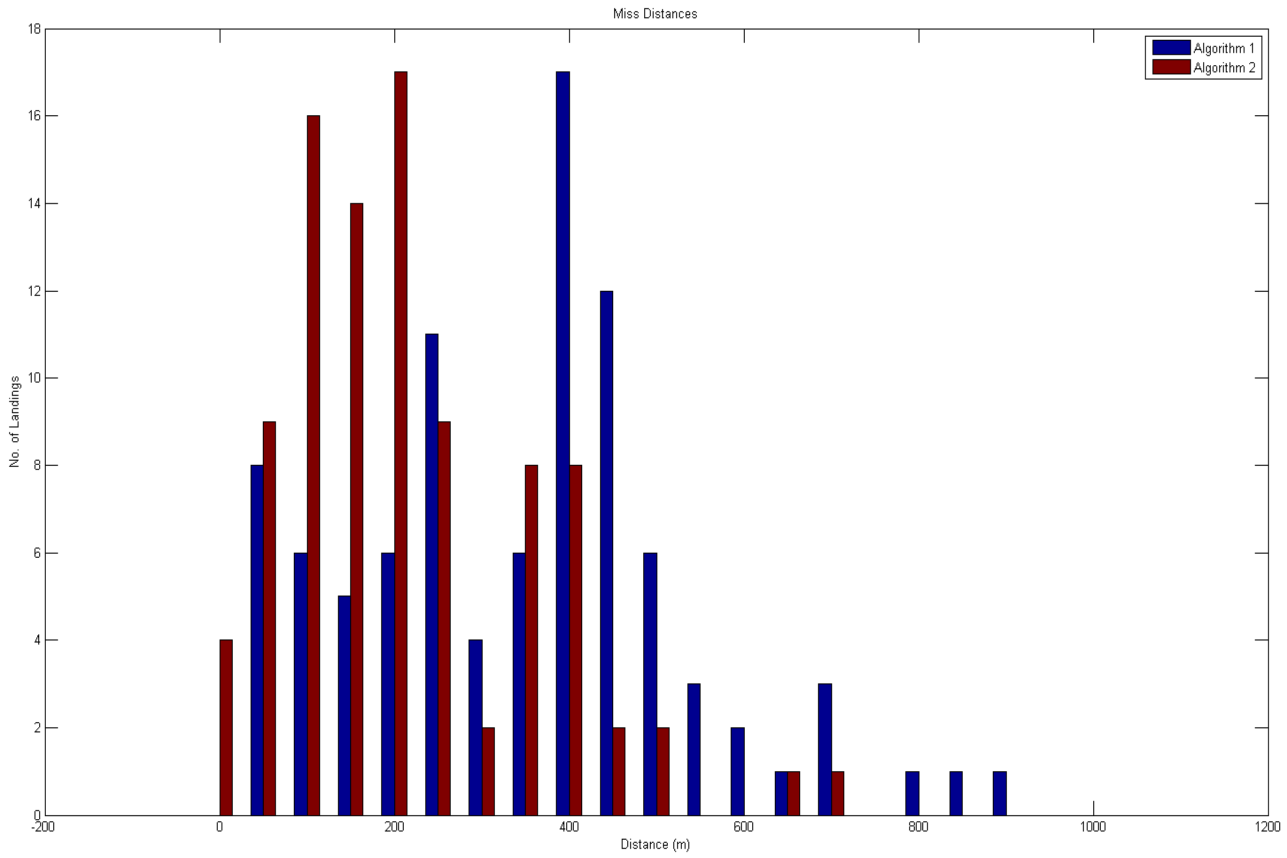

Figure 14 – Monte Carlo Simulation Results. Algorithm 2 produces more landings within the designated landing site and closer to the aim point (≤200m) than Algorithm 1 (≤400m) .

to the far threshold of the landing site (400m from the aim point), once the aim point is passed (see Path Planning – Algorithm 1). The differences in the assigned distances between the waypoints used for the algorithms could also have contributed to the differences in their respective distributions. Recall that Algorithm 1 followed a scaled implementation of the standard forced landing circuit pattern, while Algorithm 2 retained the standard circuit pattern. The fact that Algorithm 1 did not include a capacity for adjusting the flight path due to wind changes, as opposed to Algorithm 2, could also have resulted in the larger miss distances associated with that algorithm, although both algorithms could correct for changes to the aircraft heading due to wind, while enroute between two waypoints. In both cases, the aircraft pitch attitude was kept at approximately 3 degrees for landing and the aircraft flaperons were not used. Should the pitch attitude be adjustable, such as when on final approach to the aim point, and flaperons be used to change the aircraft lift-to-drag characteristics, perhaps more landings could be made within the designated area. The strong winds (maximum of 60kts headwind) modelled in the simulation could also have prevented a large number of landings from occuring within the designated area, as these winds were often greater than the UAV airspeed (37kts). The fact that the wind vector varied every minute could have also resulted in the larger miss distances, however this, as well as the former hypothesis will need to be further tested.

5 Concluding Remarks

This paper has disclosed the design and implementation of a fixed-wing UAV forced landing simulation, with modelling of winds, vehicle dynamics, control and path planning algorithms. These will be used to aid the design and testing of a visual servoing and path planning system for a fixed-wing UAV forced landing flight trial. Results comparing landing accuracies for two different path planning algorithms, as well as for the two corresponding Monte Carlo simulations were presented. These results show that Algorithm 2, which incorporated a reactive approach to changing wind conditions in planning a path to the landing site, outperformed Algorithm 1, which could only correct for wind while enroute between waypoints, but could not change the flight path due to adverse wind conditions. The 52% landing success rate of Algorithm 2 will serve as a good baseline from which to evaluate further improvements to the control and path planning algorithms, such that the landing success rate could be improved. For instance, gain-scheduling control could be combined with pitch attitude control to adjust the UAV glide path such that the aircraft will land as near to the aim point as possible. Future Monte Carlo tests will also make use of a larger data set in order to verify the success rate of Algorithm 2. In addition, future simulations will incorporate more realistic modelling of winds and gusts, as well as the modelling of buildings, trees and other hazards in the flight path. Camera images will also be simulated to test the visual servoing and path planning system to be designed, and the landing site boundaries will be reduced.

Acknowledgement

The authors wish to thank Duncan Greer and Troy Bruggemann from the Australian Research Centre for Aerospace Automation, for their assistance in developing the navigation and tracking algorithms. In addition, the authors wish to thank Flight Instructor Pravin Sen from the Royal Queensland Aero Club for the forced landing flight demonstrations.

References

[Bruggemann et al., 2005] Troy S. Bruggemann, Duncan G. Greer and Rodney A. Walker. GARDSIM – A GPS Receiver Simulation Environment for Integrated Navigation System Development and Analysis. In *Proceedings of the Smart Systems Postgraduate Research Conference*, pages 122--132, Brisbane, Australia, December 2005.

[CASA, 2001] Civil Aviation Safety Authority Australia – Aviation Safety Promotion Division. *VFR Flight Guide*. CASA, Canberra, 2001.

[Fitzgerald, 2006] Daniel L. Fitzgerald. *Candidate Landing Site Selection for UAV Forced Landings Using Machine Vision* (Ph.D. dissertation). Queensland University of Technology, Brisbane, 2006.

[Fitzgerald et al., 2007] Daniel L. Fitzgerald, Luis Mejias, Pillar Eng and Xi Liu. Towards Flight Trials for an Autonomous UAV Emergency Landing Using Machine Vision. *Accepted to the Australasian Conference on Robotics and Automation*, 2007.

[Johnson et al., 2000] Andrew E. Johnson, Yang Cheng and Larry H. Matthies. Machine Vision for Autonomous Small Body Navigation. In *Proceedings of the IEEE Aerospace Conference*, 7:661--671, Big Sky, Montana, March 2000.

[Johnson et al., 2005] Andrew Johnson, James Montgomery and Larry Matthies. Vision Guided Landing of an Autonomous Helicopter in Hazardous Terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3966--3971, April 2005.

[Kayton and Fried, 1997] Myron Kayton and Walter R. Fried. *Avionics Navigation Systems* (2 ed.). John Wiley & Sons, Inc., New York, 1997.

[Mejias et al., 2006] Luis Mejias, Pascual Campoy, Kane Usher, Jonathan Roberts and Peter Corke. Two Seconds to Touchdown – Vision-Based Controlled Forced Landing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3527--3532, October 2006.

[Nelson, 1998] Robert C. Nelson. *Flight Stability and Automatic Control* (2 ed.). McGraw-Hill Higher Education (A Division of The McGraw-Hill Companies, Inc.), 1998.

[Robson et al., 2002] David Robson, Melanie Sciberras and Peter Whellum, editors. *An Aviation Theory Centre Manual – Meteorology and Navigation for the Private and Commercial Pilot Licences*. Aviation Theory Centre Pty Ltd, Huntingdale, Australia, 2002.

[Saripalli et al., 2003] Srikanth Saripalli, James F. Montgomery and Gaurav S. Sukhatme. Visually-Guided Landing of an Unmanned Aerial Vehicle. In *IEEE Transactions on Robotics and Automation*, 19(3):371--380, June 2003.

[Serrano et al., 2006] Navid Serrano, Max Bajracharya, Ayanna Howard and Homayoun Seraji. A Novel Tiered Sensor Fusion Approach for Terrain Characterization and Safe Landing Assessment. In *Proceedings of the IEEE Aerospace Conference*, March 2006.

[Shakernia et al., 1999] Omid Shakernia, Yi Ma, T. John Koo, Joao Hespanha and S. Shankar Sastry. Vision Guided Landing of an Unmanned Air Vehicle. In *Proceedings of the Thirty-eighth Conference on Decsion & Control,* 4:4143--4148, Phoenix, Arizona, December 1999.

[Sharp et al., 2001] C. S. Sharp, O. Shakernia and S. S. Sastry. A Vision System for Landing an Unmanned Aerial Vehicle. In *Proceedings of the ICRA/IEEE International Conference on Robotics and Automation*, 2:1720--1727, Seoul, Korea, 2001.

[Templeton et al., 2007] Todd Templeton, David Hyunchul Shim, Christopher Geyer and S. Shankar Sastry. Autonomous Vision-based Landing and Terrain Mapping Using an MPC-controlled Unmanned Rotorcraft. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1349--1356, Roma, Italy, April 2007.

[Theodore et al., 2005] Colin Theodore, Steve Shelden, Dale Rowley, Tim McLain, Weiliang Dai and Marc Takahashi. Full Mission Simulation of a Rotorcraft Unmanned Aerial Vehicle for Landing in a Non-Cooperative Environment. In *Proceedings of the Sixty-first American Helicopter Society Annual Forum*, Grapevine, Texas, June 2005. American Helicopter Society, Inc.