TECHNICAL NOTE

Reactive route selection from pre-calculated trajectories – application to micro-UAV path planning

J. Hall
jah215@eng.cam.ac.uk
Division of Information Engineering, Engineering Department
Cambridge University
Cambridge, UK

D. Anderson
dave.anderson@glasgow.ac.uk
Aerospace Sciences Research Division, School of Engineering
Glasgow University
Glasgow, UK

ABSTRACT

Operating micro-UAVs autonomously in complex urban areas requires that the guidance algorithms on-board are robust to changes in the operating environment. Limited endurance capability demands an optimal guidance algorithm, which will change as the environment does. All optimal path-planning routines are computationally intensive, with processor load a function of the environmental complexity. This paper presents a new algorithm, the reactive route selection algorithm, for storing a bank of optimal trajectories computed off-line and blending between these optimal trajectories as the operating environment changes. An example is presented using a mixed-integer linear program to generate the optimal trajectories.

1.0 INTRODUCTION

Arguably one of the most mission-critical components in the guidance system of airborne autonomous vehicles is the path-planning algorithm. The operational viability of the platform is crucially interwoven with the efficacy of the chosen path planning algorithm to provide timely, reactive and robust guidance commands, especially in uncertain operational environments[1]. Such environmental considerations are increasingly becoming prevalent in the algorithm design and selection process as many military UAV operational requirements are now moving from long-range/standoff ISR (Intelligence, Surveillance and Reconnaissance) missions to local intelligence and surveillance gathering in tightly-constrained urban environments[2]. Practically, this requires that the guidance algorithm must be robust to changes in the local mapped environment. However, for many scenarios typical of UAV operations, it is still desirable to implement waypoint-driven guidance trajectories off-line to ensure optimality of the chosen flight profile. This is particularly true of lightweight micro-UAVs where optimal path planning is essential to maximise operational effectiveness against the low range and endurance induced by weight constraints on the powerplant.

Path planning for UAVs, both on-line and off-line, has been an active research area for well over the past decade or more[3-5]. Many approaches have been proposed using techniques developed from a variety of different engineering and computing disciplines, however all techniques can be classified as belonging to one of two types: either a globally optimal or locally adaptive process. Locally adaptive algorithms are very popular with the mobile robotics community forming an integral part of the wider activity of self-localisation and mapping (SLAM)[6]. Many such techniques are based upon graph theoretic algorithms originally developed within the artificial intelligence community such as Dijkstra, A*, D* etc.[7 8] and more recent graph or search-tree construction algorithms based on random searches such as the Rapidly-exploring Random Tree (RRT) algorithm[9]. However, tree construction and search can be a computationally intensive process if the desired path trajectories are of high dimension, for example when position, rate, attitude, orientation, time and other kinematic constraints are required. The resulting computational constraints mean that finding the optimal path by such methods is due more to luck than design.

Global, optimal waypoint-driven path-planning techniques are

Paper No. 3627. Manuscript received 7 September 2010, revised version received 31 January 2011, 2nd revised version received 22 July 2011, accepted 10 August 2011.

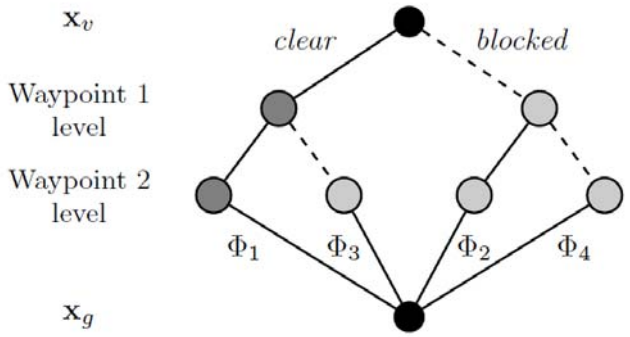

Figure 1. Trajectory graph Γ: each path to goal represents a candidate trajectory that is computed offline. A solid line denotes the optimal path from a given waypoint, dashed line denotes the alternative path. Path Φ_1 represents the original optimal trajectory.

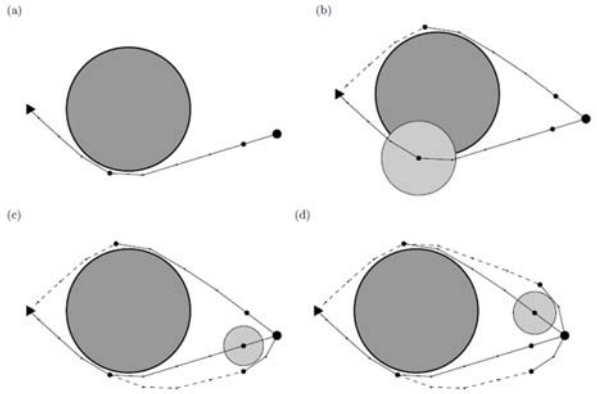

Figure 2. Simple scenario involving a UAV (triangle), a goal (large dot) and an obstacle (circle). Each trajectory is broken into time-steps (small dots) and waypoints (dots). Plots (a)–(d) show the generation of trajectories Φ_1-Φ_4 from Fig. 1 by placing potential obstacles along previous trajectories.

necessarily computed off-line because the entire world state-space needs to be represented in the optimisation algorithm: host platform, enemy (red), own (green) and allied (blue) forces, geographic obstacles and targets etc. Depending on the number of states, quantity (and type) of constraints and the cost function, a variety of nonlinear optimisation techniques have been employed. Some effective techniques include ad-hoc approaches using spatial/temporal segmentation with Voronoi diagrams and dynamic programming[3], nonlinear programming[5,10] and multi-modal techniques[11]. Another popular approach borrowed from robotics is the use of artificial potential fields[12], although this method requires some skill in defining the problem to obtain time-optimal trajectories. A technique that has gained some prominence as one of the most effective optimal waypoint generation techniques is that of Mixed-Integer Linear Programming (MILP)[13-15]. However, one of the main problems with the MILP planning algorithm is that finding the solution to the constrained optimisation problem is a computationally intensive process and one which must be repeated each time the environment changes. In modern control theory, real-time implementation of computationally intensive optimisations is a crucial research objective for Model Predictive Control (MPC) of fast systems[16,17]. Essentially, all of the techniques currently under development either simplify the optimisation problem or move large sections of the computation off-line. The resulting controller is then implemented on-line using a look-up table[18,19].

Therefore, our assertion is that a practical solution to the problem of implementing optimal path re-planning onboard micro-UAVs with limited processing capability is to store and search a bank of pre-calculated optimal trajectories corresponding to a finite, but highly probable, subset of all possible configuration spaces. The algorithm managing this process we have entitled the Reactive Route Selection (RRS) algorithm and is the main contribution of this paper. In RRS, the optimal trajectory that minimises mission completion time is calculated offline using information about known obstacles and pre-specified waypoints positioned well within the platforms primary sensor range. Potential obstacles are then sequentially introduced at each waypoint along the trajectory with the remaining optimal trajectory re-planned using the new obstacles. This process is repeated for all the trajectories in order to build up a bank that can be used along with sensor data by the UAV online. Although the path-planning method used in this paper was MILP, in theory any optimisation process could be used.

2.0 ONLINE RE-ROUTING

2.1 General concept

In most operational environments the UAV will have to react to changes in threat level, unforeseen blockages or other deviations from the assumed state-space map, requiring transition to an alternative optimal trajectory. As all optimal trajectory algorithms (such as MILP) require the specification of intermediate waypoints in their formulation, each of these locations can be used as local goals. Ideally, each waypoint is selected to lie within the range of the primary platform sensor, meaning that at the j^{th} waypoint on the i^{th} trajectory, $x_{wp}^{i,j}$, the platform can determine whether the next waypoint on the current trajectory is valid. A waypoint is declared valid if there are no obstacles detected on the path segment between $x_{wp}^{i,j}$ and $x_{wp}^{i,j+1}$. If the next waypoint is invalid, an alternative trajectory must be chosen. To do this, the RRS algorithm computes a database of trajectories offline by introducing potential obstacles at each waypoint along each individual trajectory and re-planning the mission from the previous waypoint. Used online, it also selects the most appropriate trajectory from the database in the aircraft guidance system. Assuming all trajectories have the same number of enumerated waypoints, the database architecture is that of a binary graph Γ with the nodes in each layer representing a candidate waypoint along each trajectory. All but the penultimate node (final waypoint) has therefore two branches as there are two choices for the next waypoint: clear or blocked. Figure 1 illustrates the graph for a complete set of trajectories corresponding to the scenario shown in Fig. 2.

2.2 RRS algorithm theory

Consider a global state-space, $\mathbb{E}$. The centres of the vehicle x_v, the k^{th} obstacle x_{os}^k, the goal x_g and waypoint positions on the i^{th} trajectory Φ_i (where the j^{th} column of Φ is $x_{wp}^{i,j}$ can be defined in $\mathbb{E}$. Next consider Fig. 3. There are four principal distances shown in Fig. 3 that require explanation. The first, δ_r is the desired turn radius of the vehicle and is a function of the aerodynamics, current flight state and desired operational strategy. This last variable requires further explanation. For a system where the search time should be minimised for example, more aggressive manoeuvring is warranted and therefore the desired turn radius may be the minimum achievable turn radius for that flight state. However for a mission where endurance or fuel consumption is more important, the desired turn radius will be that which maximises lift/drag

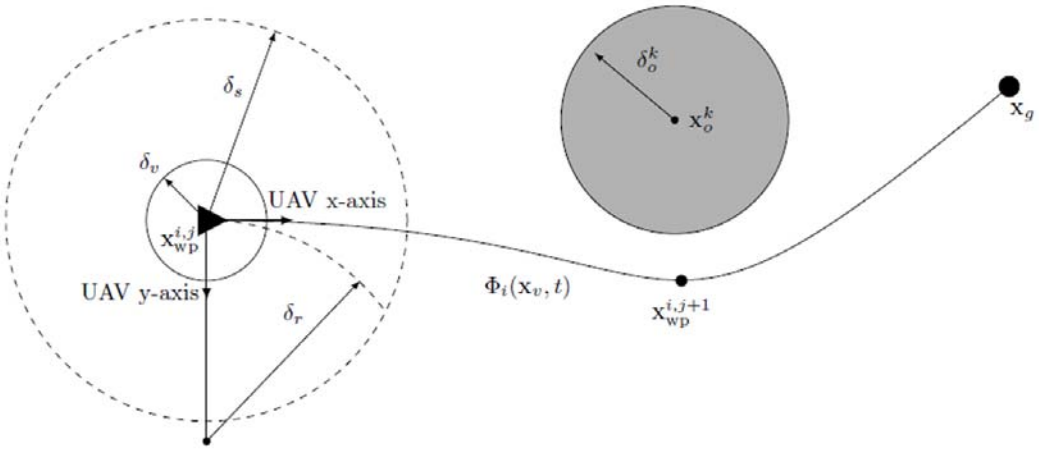

Figure 3. UAV obstacle avoidance geometry.

ratio. We shall return to this point later. The second, δ_v, is the maximum spatial extent of the vehicle. δ_s is the sensor range and finally δ_o^k is the size of the k^{th} obstacle. The physical extent of the vehicle and obstacles can now be defined as the sets;

$$\mathbf{V} = \{\forall x \in \mathbb{E} \,|\, \delta_v > \| x - x_v \|\} \quad \ldots (1)$$

$$\mathrm{O}^k = \{\forall x \in \mathbb{E} \,|\, \delta_0^k > \| x - x_o^k \|, k \in \{1 \ldots K\}\} \quad \ldots (2)$$

respectively, where K is the number of obstacles and $\|.\|$ denotes the Euclidean norm. With these sets the miss criteria, which also defines all feasible locations in $\mathbb{E}$ can be defined as;

$$\mathbb{E}_{feas} = \{\forall x \in \mathbb{E} \,|\, (\mathbf{V} \cup \mathbf{O}^k) = \varnothing, k \in \{1 \ldots K\}\} \quad \ldots (3)$$

Using this constraint it is possible to define an optimal trajectory $\Phi_i^*(x_v,t)$ as the flow from x_v to x_g using any of the global, optimal path planning algorithms discussed earlier, provided they solve the problem;

$$\begin{aligned} &(\min_{\Phi} J(\Phi(x_v,t), x_g) \\ &s.t.\ \ \Phi(x_v,t) \in \mathbb{E}_{feas}) \end{aligned} \quad \ldots (4)$$

according to some cost function J and also respecting vehicle dynamics and constraints. In the examples to follow, a MILP solver was used exclusively, but any equivalent technique could be substituted. Each trajectory has n waypoints spaced evenly along $\Phi_i^*(x_v,t)$ such that the distance between waypoints is always within the range of the vehicle sensor i.e.

$$\left\| x_{wp}^{i,j+1} - x_{wp}^{i,j} \right\| \leq \delta_s \quad \ldots (5)$$

The RRS algorithm populates its trajectory tree by considering what would happen if the next waypoint in $\Phi_i^*(x_v,t)$ were blocked by some change in $\mathbb{E}$. This can be simply achieved by re-classifying $x_{wp}^{i,j}$ (each waypoint in turn, along the ith trajectory) as an obstacle and computing b new trajectories $\Phi_{i+a}^*(x_v,t), a \in \{1 \ldots b\}$ using the modified state-space $\mathbb{E}_m$. The value of $b = n - 1$ for all trajectories except the initial optimal trajectory where $b = n$ i.e. potential obstacles are not placed on the first waypoint of a trajectory except the initial optimal one. However, the question remains as to the appropriate size of the new obstacle δ_o^{j+1}. To illustrate the concept the following simplifying assumptions can be made without loss of generality:

- The UAV is flying with zero sideslip (flight vector is aligned with the body x-axis),
- The evasive manoeuvre most likely to maximise survivability is the maximum constant turn rate achievable for the current flight state.
- The most benign evasive manoeuvre is the turn rate which maximises the lift/drag ratio of the vehicle and is known a priori.

These conditions mean that the centre of the turning circle is a distance δ_r along a vector orthogonal to the flight velocity, the UAV y-axis in this case. Then, the closest the vehicle comes to the obstacle in normal RRS operation yields the maximum new obstacle size;

$$\delta_{o,max}^i = max\left(\left\|x_{wp}^{i,j+1} - \left(x_{wp}^{i,j} + \delta_r \hat{y}\right)\right\|, \left\|x_{wp}^{i,j+1} - \left(x_{wp}^{i,j} - \delta_r \hat{y}\right)\right\|\right) - (\delta_v + \delta_r) \quad \ldots (6)$$

where the max(·,·) function encodes the choice between a right or left turn evasive manoeuvre and $\hat{y}$ is the unit vector along the UAV y-axis. This gives the condition that $\delta_o^i \leq \delta_{o,max}^i$. An additional constraint that the new obstacle does not cover the goal and therefore render the problem infeasible must also be included. In some cases it may be more desirable to increase δ_r to reduce induced drag and so increase likely endurance. Such an approach may be appropriate if, for example, the UAV sensor range δ_s far exceeds the distance between waypoints or the operating state space has few obstacles.

One problem with the algorithm as it stands is that if an obstacle was detected just beyond waypoint $x_{wp}^{i,j+1}$ (referring to Fig. 3) the vehicle would have to wait until it reached that waypoint before altering trajectory, and by that time the branching trajectory may not avoid the obstacle either. However, it would still be possible for the ***RRS*** algorithm to guide the UAV to the final goal using a trajectory blending technique as shown in Fig. 4 (provided that the avoidance manoeuvre is within the vehicle constraints). A general blending strategy for non-overlapping trajectories is;

$$\Phi_b(\tau) = (1 - \alpha)\,\Phi_1(\tau) + \alpha\Phi_2(\tau) \quad \ldots (7)$$

where $\tau \coloneqq t - t_c$ and $\alpha \in [0,1]$ (note that here time t is used as the dependant variable to conform with standard practice when solving the flow of a differential equation). Equation (7) is used to blend from the (possibly) nonlinear flow $\Phi_1(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^n$, onto the nonlinear flow $\Phi_2(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^n$ across a finite blending horizon τ of length Δ_t. Instantaneous switching between trajectories is wholly unrealistic due to the finite acceleration and velocity capabilities of the UAV in flight, irrespective of which translational or angular state

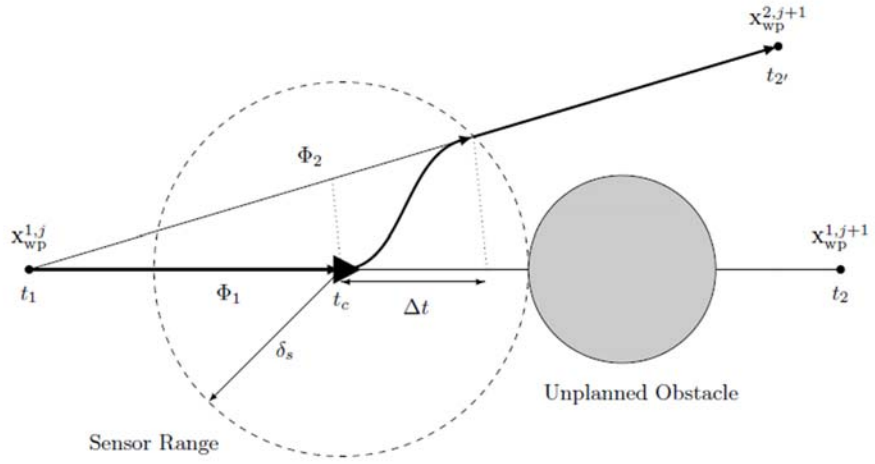

Figure 4. Trajectory blending using linear (or nonlinear) interpolation.

trajectory is blended. The type of blending is controlled via the parameter α. Linear blending is shown by Equation (8).

$$\alpha \coloneqq \begin{cases} 0 & \tau \leq 0 \\ \tau/\Delta t & 0 < \tau \leq \Delta t \\ 1 & \tau > \Delta t \end{cases} \quad \ldots (8)$$

The slope of the blending trajectory, which is also the local rate of the state variable, is given by the difference between Φ_2 ($t_c + \Delta t$) and $\Phi_1(t_c)$ divided by Δt. It is then simple to incorporate known velocity constraints into the blend. However, the accelerations at the beginning and end of each blend can be quite high. Following the approach suggested in Dever[20] for example, a static nonlinearity such as a boundary-value polynomial or the hyperbolic tangent function may be substituted in the definition of α or perhaps even a Dubins curve used to smooth the initial and terminal accelerations. One of the benefits of the simple nonlinear functions however is the existence of continuous derivatives which can be easily compared to δ_r to ensure feasibility of the blending trajectory. Figure 5 shows a simple example of linear and hyperbolic tangent blending between two flows with $t_c = 4$s and $\Delta t = 2$s.

Finally, all that is left is to select the best trajectory to blend into. Assume that the vehicle sensors detect an obstacle centred on $x_{wp}^{i,j+1}$ when traveling between $x_{wp}^{i,j-1}$ and $x_{wp}^{i,j}$ and that $\delta_o^{i+1} > \delta_{0,max}$ such that all trajectories emanating from $x_{wp}^{i,j}$ are infeasible. The improved algorithm then sets $x_{wp}^{i,j}$ to blocked and begins to traverse the tree Γ by going back up the tree to node $x_{wp}^{i,j-1}$ and traversing the blocked branch. With reference to Fig. 1, this corresponds to the situation where one discovers an obstacle between waypoint 1 and 2 of trajectory Φ_1 while traveling towards waypoint 1 rendering trajectories Φ_1 and Φ_3 infeasible. In this case the algorithm would blend into trajectory Φ_2. Ideally, both remaining candidate trajectories should be checked at all points in the neighbourhood of the obstacle, but this is time consuming and contradicts the purpose of the ***RRS*** algorithm. Consequently, only $x_{wp}^{2,j+1}$ and $x_{wp}^{4,j+1}$ are tested. If none of these trajectories is found feasible, a return-to-base option is triggered.

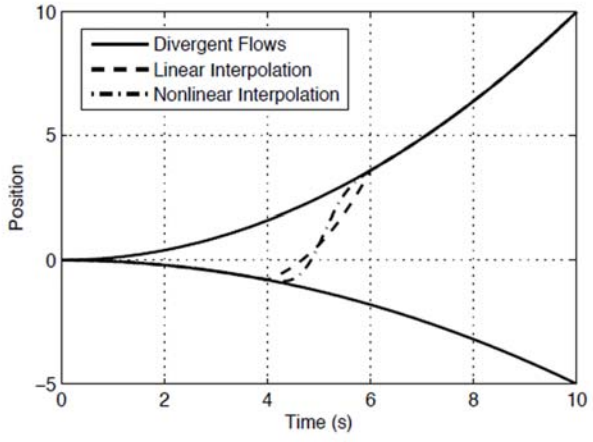

Figure 5. Linear and nonlinear interpolation of two divergent flows.

3.0 MILP FORMULATION

MILP was used to implement the optimisation procedure in Equation (4). The mathematical formulation, taken from Bellingham[14], will now be described. The full, nonlinear UAV dynamics were not implemented explicitly in the MILP formulation, rather we followed the standard approach of modelling the kinematics as a simple point mass and incorporating aerodynamic nonlinearities, couplings etc as constraints on the kinematic model[14].

$$s_{t+1} = \begin{bmatrix} I & \delta t I \\ 0 & I \end{bmatrix} s_t + \begin{bmatrix} \frac{1}{2}\delta t^2 I \\ \delta t I \end{bmatrix} u_t \quad \ldots (9)$$

where the state vector is $s = [x_v^T, \dot{x}_v^T]^T$ and Δt is the discrete time-step length.

The cost function was taken as the mission completion time with a small weighting on the force exerted by the UAV to penalise fuel usage and ensure a unique optimal solution. It is given over a horizon of N time-steps as follows:

$$\min_{u_{0,N-1}} J = \sum_{t=1}^{N} t a_t + \varepsilon \sum_{t=0}^{N-1} \|u_t\| \quad \ldots (10)$$

where $a_t \in [0,1]$ is 1 if the UAV reaches the target at time t and 0 otherwise. To enforce the condition that the UAV reached the target within N time-steps, the following constraints are added to the optimisation problem;

$$\left\| x_{v,t} - x_g \right\| \leq M(1 - a_t) \quad \ldots (11)$$

$$\sum_{t=1}^{N} a_t = 1 \quad \ldots (12)$$

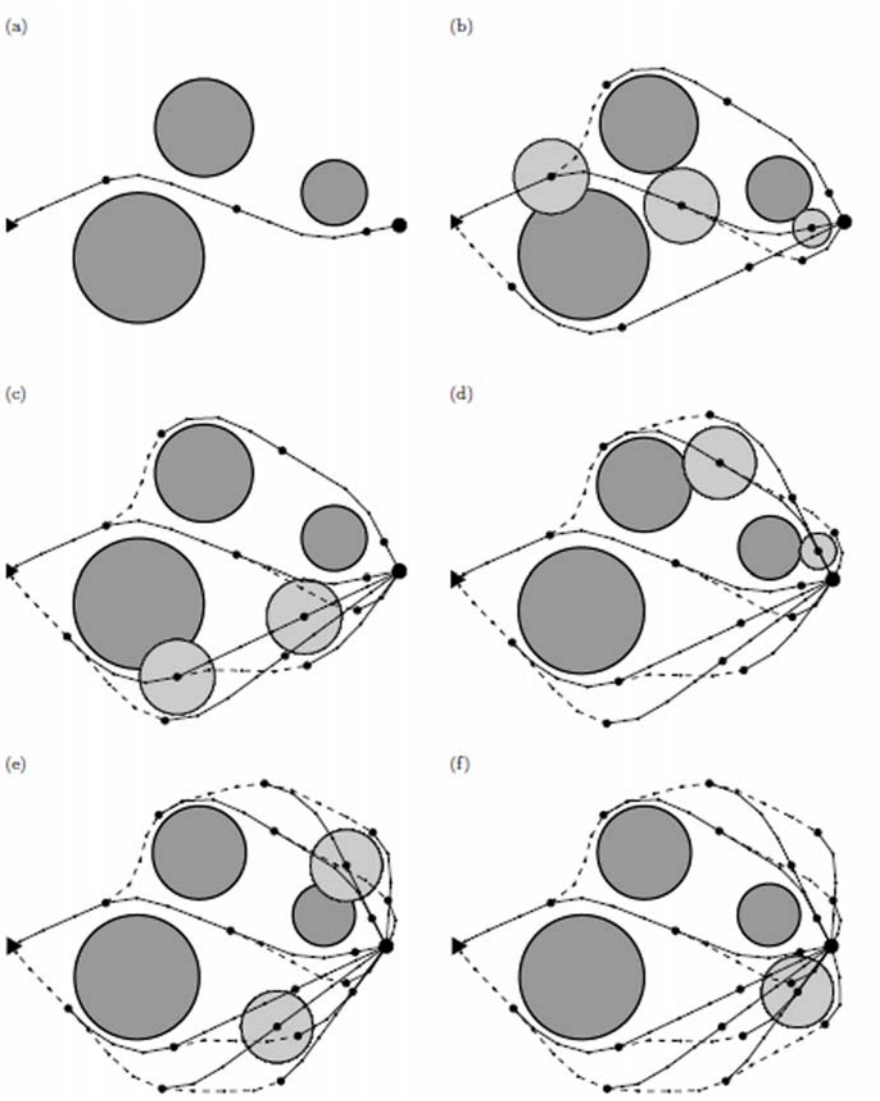

Figure 6. Trajectory bank generation for a three obstacle scenario where $\delta_s = 4$, $\|\dot{x}_v\| = 2$, $\|u\| = 2$, $\delta_t = 0{\cdot}5$ s, $N = 20$ and $\varepsilon = 0{\cdot}1$. UAV is represented by a triangle, target as a big dot and obstacles as circles. Each trajectory is broken into time-steps (small dots) and waypoints (dots).

where M is some large number that acts to relax the constraint for all instances of time except one, where x_{vt} must equal x_g. For details on how state, input and obstacle avoidance constraints were implemented the reader is referred to[14].

4.0 EXAMPLE

The trajectory bank generator was coded in MATLAB with the MILP computation carried out by CPLEX optimisation software[21]. A scenario involving three obstacles was passed through the code. The resulting build-up of trajectories is displayed in Fig. 6.

Figure 6(a) shows the optimal path from start location to target. Figure 6(b) shows the generation of trajectories branching from the optimal path. Figure 6(c), 6(d) and 6(e) show the generation from the first, second and third suboptimal trajectories respectively. The completed trajectory bank is then displayed in Fig. 6(f). The size of this bank slightly exceeds the size of the corresponding binary tree, which would contain 2^3 paths since the original trajectory contains three waypoints. This is simply because some of the suboptimal trajectories generated were long enough to warrant the inclusion of an extra waypoint.

The path planning method outlined here has the advantages of giving a low computational burden to the UAV computers as a full MILP does not have to be solved at each time-step (as in the case of MPC), and is able to provide a degree of robustness into the planner. This degree of robustness could easily be increased by giving the UAV m path options at each waypoint by introducing a potential obstacle at the start of the first suboptimal trajectory, therefore increasing the size of the trajectory bank to $m^n + \varepsilon$ cells. It can be seen that there will be a trade-off between size of on-board memory usage and robustness in the path.

5.0 CONCLUSIONS AND FURTHER WORK

This investigation has tackled the problem of optimal path re-planning for a single UAV in a cluttered and dynamic environment by proposing a novel approach to the computation, storage and selection of optimal, feasible trajectories. By fusing the standard, widely accepted approach of discretising the operating state space with the constraints imposed by the UAV sensors range limitations a workable, heuristic reduced-order dynamic state space was shown to exist. Furthermore it was also shown that with careful waypoint selection in the optimal trajectory generation program (in this case

mixed-integer linear programming), a practical solution exists for computing off-line and storing a finite bank of feasible, optimal trajectories corresponding to the most important dynamic changes in the environment i.e. those prohibiting flight along pre-computed waypoint legs.

Although more research is required in categorising the most likely changes in the configuration space, in moving the computationally intensive global trajectory optimisation calculation off-line, the RRS algorithm provides a potential framework for implementing optimal and robust path-planning on low-cost micro and nano-scale unmanned aerial vehicles. One interesting potential improvement is the possibility of blending trajectories between trees. As mentioned in Section 2.2, each tree is a function of the desired turn radius, which essentially governs the level of aggressiveness in the evasive manoeuvres. A future addition to the algorithm would therefore be the computation of multiple trees, each with different manoeuvre agility and through sensing the real-world obstacle select the most appropriate alternative trajectory for the mission. This will be the topic of a future paper.

REFERENCES

1. RATHBUN, D., KRAGELUND, S., PONGPUNWATTANA, A. and CAPOZZI, B. An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments, 21st Digital Avionics Systems Conference, 2002, pp 8D2-1-8D2-12.
2. MoD Grand Challenge Overview, 2009. http://www.science.mod.uk/engagement/grand_challenge/grand_challenge.aspx
3. BORTOFF, S.A. Path Planning for UAVs, IEEE American Control Conference, Chicago, Illinois, USA, 2000, pp 364-368.
4. RYAN, A., ZENNARO, M., HOWELL, A., SENGUPTA, R. and HEDRICK, J.K. An Overview of Emerging Results on Cooperative UAV Control, 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, 2004, pp 602-607.
5. GEIGER, B.R., HORN, J.F., DELULLO, A.M., LONG, L.N. and NIESSNER, A.F. Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming, AIAA Guidance, Navigation and Control Conference, Keystone, Colorado, USA, 2006, pp 801-805.
6. DURRANT-WHYTE, H. and BAILEY, T. Simultaneous localisation and mapping (SLAM): Part 1 The Essential Algorithms, *Robotics and Automation Magazine*, **13**, (2), pp 99-110.
7. NILSSON, N.J. *Principles of Artificial Intelligence*, Tioga Publishing Company, 1980.
8. FERGUSON, D. and STENTZ, A. The Delayed D* Algorithm for Efficient Path Replanning, IEEE International Conference on Robotics and Automation, 2005, pp 2045-2050.
9. LAVALLE, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*, Iowa State University, Iowa, USA, C. S. Dept, 98-11, 1998.
10. EELE, A. and RICHARDS, A. Path-planning with avoidance using nonlinear branch-and-bound optimization, *AIAA J Guidance, Control and Dynamics*, March-April, **32**, (2), pp 384-394.
11. GODBOLE, D., SAMAD, T. and GOPAL, V. Active Multi-Modal Control for Dynamic Maneuver Optimization of Unmanned Aerial Vehicles, IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 2000, pp 1257-1262.
12. PAUL, T., KROGSTAD, T.R. and GRAVDAHL, J.T. Modelling of UAV formation flight using 3D potential fields, *Simulation Modeling Practice and Theory*, **16**, (9), pp 1453-1462.
13. SCHOUWENAARS, T., DEMOOR, B., FERON, E. and HOW, J.P. Mixed-Integer Programming for Multi-Vehicle Path Planning, European Control Conference, Porto, Portugal, 2001, pp 2603-2608.
14. BELLINGHAM, J., TILLERSON, M., RICHARDS, A. and HOW, J.P. Coordination and Control of Multiple UAVs, AIAA Guidance, Navigation and Control Conference and Exhibit, Monteray, CA, USA, 2002, pp D12-D18.
15. KABAMBA, P.T., MEERKOV, S.M. and ZEITZ, F.H. Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking, *AIAA J Guidance, Control and Dynamics*, March-April, **29**, (2), pp 279-288.
16. WANG, Y. and BOYD, S. Fast Model Predictive Control Using Online Optimization, 17th World Congress, Seoul, S Korea, 2008, pp 6794-6979.
17. ANDERSON, D., LOO, M. and BRIGNALL, N. Fast model predictive control of the Nadar singularity in electro-optic systems, *AIAA J Guidance, Control and Dynamics*, March-April, **32**, (2), pp 626-632.
18. BEMPORAD, A., MORARI, M., DUA, V. and PISTIKOPOULOS, E.N. The explicit linear quadratic regulator for constrained systems, *Automatica*, **38**, pp 3-20.
19. DUA, V., BOZINIS, N.A. and PISTIKOPOULOS, E.N. A New Multiparametric Mixed-Integer Quadratic Programming Algorithm, 34th European Symposium of the Working Party on Computer Aided Process Engineering, 2001, pp 979-984.
20. DEVER, C., METTLER, B., FERON, E., POPOVIC, J. and MCCONLEY, M. Nonlinear trajectory generation for autonomous vehicles via parameterized maneuver classes, *AIAA J Guidance, Control and Dynamics*, March-April, **29**, (2), pp 289-302.
21. CPLEX, ILOG CPLEX 9.0 User's Manual, 2003.