Multi-Objective 4D Vehicle Motion Planning in Large Dynamic Environments

Paul P.-Y. Wu, *Member, IEEE,* Duncan Campbell, *Member, IEEE,* Torsten Merz, *Member, IEEE*

***Abstract*—This paper presents Multi-Step A* (MSA*), a search algorithm based on A* for multi-objective 4D vehicle motion planning (three spatial and one time dimension). The research is principally motivated by the need for offline and online motion planning for autonomous Unmanned Aerial Vehicles (UAVs). For UAVs operating in large, dynamic and uncertain 4D environments, the motion plan consists of a sequence of connected linear tracks (or trajectory segments). The track angle and velocity are important parameters that are often restricted by assumptions and grid geometry in conventional motion planners. Many existing planners also fail to incorporate multiple decision criteria and constraints such as wind, fuel, dynamic obstacles and the rules of the air. It is shown that MSA* finds a cost optimal solution using variable length, angle and velocity trajectory segments. These segments are approximated with a grid based cell sequence that provides an inherent tolerance to uncertainty. Computational efficiency is achieved by using variable successor operators to create a multi-resolution, memory efficient lattice sampling structure. Simulation studies on the UAV flight planning problem show that MSA* meets the time constraints of online replanning and finds paths of equivalent cost but in a quarter of the time (on average) of vector neighbourhood based A*.**

***Index Terms*—path planning, heuristic algorithms, multi-objective decision making, unmanned aerial vehicles.**

I. Introduction

An important component in the operation of vehicles in large, four dimensional (three spatial and one time dimension) dynamic environments is motion planning. This involves finding an optimal (least cost) or near-optimal sequence of 4D states that connect the initial vehicle state to a desired goal state [1]. For many applications, it is necessary to plan offline (e.g. prepare a mission plan for regulatory approval) and replan online when planning assumptions are invalidated by in-mission changes. The research is principally motivated by the operation of robotic vehicles, namely autonomous Unmanned Aerial Vehicles (UAVs) and equivalently, Autonomous Underwater Vehicles (AUVs).

UAVs and AUVs are characterised by: (i) operation in large, outdoor environments, (ii) movement in three dimensions (x, y, z), (iii) uncertain and dynamic operating environment, (iv) presence of environmental forces that affect motion (winds or currents), and (v) differential constraints on movement [2], [3]. Because of (ii) and (iii), the planning space must be four dimensional. Note that a dynamic environment refers not only to moving obstacles, but also to changing weather conditions. The proposed work mitigates the uncertainty inherent in a dynamic environment through online replanning and incorporation of tolerances in the planning process. Online replanning in this paper refers to the execution of a search (i.e. replan) during a mission. It is assumed that the inputs to the planner are constant during a replan, hence there is significant time pressure on the replanning process.

The motion plan is constrained by vehicle dynamics (such as maximum climb/ascent rate), environmental constraints (e.g. static and dynamic obstacles and wind/current) and rules of the air/sea. In addition, the planned path must satisfy (and optimise for) multiple, possibly conflicting objectives such as fuel efficiency and flight time. Due to the "curse of dimensionality" [4], it is not computationally feasible to plan in a high dimensional search space consisting of all the aforementioned variables. It is common, instead, to plan the path in the world space (x, y, z, t) [1] by aggregating the decision variables into a single, non-binary cost term [4]. This planning problem is a type of weighted region path planning [5]. An optimal path search algorithm like A* [6] is needed as the shortest path is not necessarily the least cost path.

One of the unique UAV/AUV characteristics listed above is the presence of wind (or currents). These constrain vehicle movements and affect travel time and fuel consumption. In the presence of wind, it is especially important to have high track angle resolution as low track angle resolution can result in suboptimal paths that contain spurious turns [7], [8]. This is a shortcoming of conventional search grids as the track angle is in increments of $45°$.

Note that 4D motion planning as described here should not be confused with trajectory planning, which finds a path expressed in terms of the degrees of freedom of the vehicle and velocity/angle rates [1]. Instead, a 4D motion plan comprises a geo-referenced sequence of 3D waypoints and the desired track velocities between them. In this paper, such tracks are also equivalently referred to as trajectory segments.

This paper presents Multi-Step A* (MSA*), a method for 4D vehicle motion planning based on variable length, angle and velocity trajectory segments. Section II reviews existing path planning techniques. Based on A* [6], the proposed method is presented in Section III and shown to be cost optimal. To take advantage of variable trajectory segments, a memory and time efficient multi-resolution lattice structure

The work was supported in part by the Australian Research Centre for Aerospace Automation (ARCAA) and the Commonwealth Scientific and Industrial Organisation (CSIRO) Information and Communication Technologies (ICT) Centre top-up scholarship

Dr Paul Wu, is a researcher with ARCAA, Queensland University of Technology (QUT), GPO Box 2434, Brisbane, 4000, Australia. (e-mail: p.wu@qut.edu.au)

Associate Professor Duncan Campbell, is with the School of Engineering Systems and ARCAA, QUT, Brisbane, 4000 Australia. (e-mail: da.campbell@qut.edu.au)

Dr Torsten Merz is a senior research scientist with CSIRO, ICT Centre, PO Box 883, Kenmore QLD 4069, Australia. (e-mail: torsten.merz@csiro.au)

is proposed in Section IV. A simulation study of MSA* for the UAV flight planning task is discussed in Section V. Analysis of the simulation results and a comparison of MSA* with existing work is discussed in Section VI.

II. Existing Work

Much of the recent work in vehicle planning has focused on techniques in computational geometry using a grid [7]–[16]. However, a shortcoming of many grid-based approaches (e.g. [14], [16]) is that the resultant path is confined to track angles that are multiples of $45°$. As a result, the path can be sub-optimal and may contain spurious turns [7], [8]. A lack of regular high resolution track angles also affects methods based on Voronoi graphs (e.g. [11], [17]), methods that use probabilistic sampling (e.g. [18], [19]), and generally methods where path angles are not considered (e.g. [15], [20]). A review of motion planning algorithms is provided below for methods that address the requirements of vehicle motion planning vis a vis the track angle problem, wind/current effects and multi-objective optimisation.

A. Methods with High Resolution Track Angles

A number of grid based methods determine the track angle in continuous space instead of sampling from predefined, discrete track angles. However, geometry based methods in 2D or 3D (or even 4D) space, such as Theta* [8] and A_{3D} [15], do not find the optimal path. Field D* [7] and 3D Field D* [13] find the optimal path but both assume a priori knowledge of cell costs (which are used to derive the track angle and track cost). This approach is infeasible in a 4D search space as the cost is dependent on the track angle. Nevertheless, it is shown that a multi-resolution search space can be used to mitigate the memory and time complexity of motion planning [1], [7], [9], [13].

Pivtoraiko and Kelly [12] present an alternate method that provides regular, high resolution track angles by defining a successor operator (i.e. parent child cell relationships) that has a predetermined number of successors at selected track angles. Like Theta*, parent cells are not necessarily adjacent to child cells, hence the notion of a vector neighbourhood [1]. However, the method is formulated for 2D vehicle planning with no consideration for winds/currents.

The framed quad/octree [9] enables high resolution track angles by placing sample nodes on the boundaries of each quad/octree decomposed cell. However, transitions between cells is again limited to increments of $45°$. Additionally, neither [9] nor [12] consider wind effects.

A number of existing planners model wind effects using weighted, polygonal (or polyhedral) shaped regions [21]–[24]. However, these methods do not consider multiple objectives and are not suited for planning in a dynamic environment. This is a similar shortcoming of AUV [14] and UAV [10] motion planners that incorporate wind.

Finally, there are motion planners based on artificial evolution (e.g. [10], [18], [19]) that plan in continuous space. A shortcoming of evolutionary algorithms is the inability to specify bounds on computation time or solution optimality [25]. This can be problematic for online replanning (due to real-time constraints), and for applications where determinism is a regulatory requirement (e.g. DO178-B [26] for aviation software).

B. Multi-Objective Planning Algorithms

None of the previously quoted methods explicitly address the requirement of optimising for multiple decision objectives, although many incorporate multiple path constraints (e.g. water currents and vehicle dynamics in [14]). Examples of explicit multi-objective planning algorithms can be found in the study of HAZardous MATerials (HAZMAT) transportation [27]. These algorithms combine a multi-objective decision function (typically a weighted sum) with a graph search algorithm (such as A* or Dijkstra's algorithm) on a grid [27]–[30] (refer to [31] for a description). This methodology is also used by Gu [11] for a bi-objective (risk and fuel objectives) UAV motion planner. These multi-objective planning algorithms (e.g. [11], [27]–[30]) almost universally adopt a global planning approach where the track cost is calculated at search time (much like a lazy probabilistic roadmap [1]).

An alternative approach to multi-objective path planning is to use a multi-objective search algorithm like [17], [32]. However, these algorithms are computationally expensive and in the case of [32], restricted to acyclic graphs; note that graphs derived from grids are cyclic.

A similar, direct approach to multi-objective path planning is logic based planning. Three candidate approaches include the Hierarchical Task Network (HTN) [33], Temporal Action Logic (TAL) [34] and MFSAT (Multi-Flip SATisfiability solver - which evaluates non-adjacent neighbours in logic space) [35]. An application of HTN to indoor robot navigation is described in [36]. However, logic based motion planning is generally computationally expensive and the resultant plan is typically non-optimal [1], [33].

It can be seen that existing methods do not fully address the multi-objective vehicle motion planning problem.

III. Multi-Step A* (MSA*)

The planning task is defined as finding a path P, through a roadmap S, starting at node s_0 and terminating at node s_G. Each node $s \in S$ is located at the centre of a 4D rectangloid cell defined in the world space $W(x, y, z, t)$. Assuming a regular grid sampling of the search space, each node s maps uniquely to a cell in W. Thus, s refers simultaneously to both the cell and the node located in the centre of each cell. The global planning approach described in Section II is adopted whereby tracks are evaluated online and the initial roadmap is not explicitly represented (like with [20]). Instead, the roadmap is defined implicitly through a successor (or neighbourhood) operator Γ where for a given source (or parent) node s, $\Gamma(s)$ denotes a set of cell sequences $\gamma_{s'} \in \Gamma(s)$ which begin at s and terminate at the successor (or child) node $s' \in S'$.

Consider the modeling of a vector neighbourhood like [12] where s' does not necessarily lie adjacent to s. The successor operator is assumed to denote a linear trajectory/track connecting the centre of cell s to the centre of cell s' (refer to

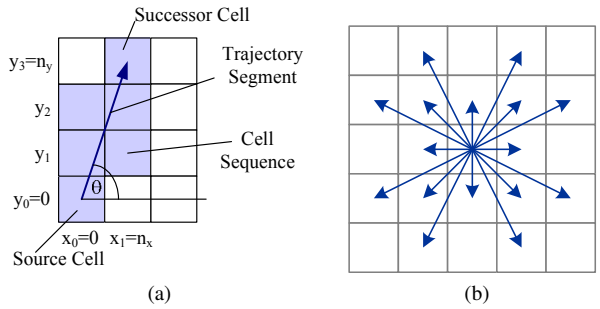

Fig. 1. Successor operator illustration, (a) a single 2D trajectory segment and the corresponding cell sequence, and (b) an example 2D successor operator showing individual trajectory segments. Note that the trajectory segment terminates in the centre of the successor cell, which is not necessarily adjacent to the source cell.

Fig. 1). It is assumed that turns (possibly required between tracks) have negligible impact on the overall path in terms of travel time and fuel consumption. For each successor s', the trajectory intersects a sequence of cells between s and s'

$$\gamma_{s'} = \{s_{j+m-1} = s' \| s_j = s, \ldots, s_{j+m-2}\} \tag{1}$$

where $s_j, \ldots, s_{j+m-1}$ are a sequence of m cells and $\|$ denotes a conditional dependence, i.e. $s_{j+m-1} \| s_j, \ldots, s_{j+m-2}$ is interpreted as cell s_{j+m-1} via a sequence of cells $s_j, \ldots, s_{j+m-2}$. In the ensuing sections, Γ is derived for a three dimensional world ($\Gamma_{3D} \in \Gamma$, Section III-A) and then extended to four dimensions (Section III-B). This is possible as the search dimensions are orthogonal.

A. Multi-Step 3D Successor Operator

As illustrated in Fig. 1a, it is possible to determine the horizontal track angle θ and slope from the endpoints of each trajectory segment. The vertical track angle ϕ and slope can be similarly determined for a 3D trajectory segment. In aviation, the horizontal and vertical track angles are referred to as the ground track angle and flight path angle respectively. Note that the vertical slope is given by the climb/descent rate and the ground speed. The angles can then be used to determine the cells that intersect the track using a variant of Bresenham's pixel algorithm [37]. Note, because the vehicle controller has non-zero trajectory tracking error, it is necessary to include cells whose edges/corners touch the trajectory line. This prevents the possibility of the vehicle squeezing through an infinitesimally small gap or "brushing past" an obstacle. Doing so provides an intrinsic tolerance to navigational and controller uncertainty as there is always a safety margin between the trajectory segment and the boundaries of a cell.

Using Bresenham's [37] line drawing concept, determination of a 3D cell sequence is based on the displacement (n_x, n_y, n_z) of the successor cell from the source in terms of the number of cells in the x, y, and z dimensions respectively. This sequence is invariant to the physical dimensions of each cell but assumes regular cells. The 3D line equations are

$$\begin{cases} y = \frac{n_y}{n_x}x, z = \frac{n_z}{n_x}x & \text{if } |n_x| \geq |n_y|, |n_x| \geq |n_z| \\ x = \frac{n_x}{n_y}y, z = \frac{n_z}{n_y}y & \text{if } |n_x| < |n_y|, |n_y| \geq |n_z| \\ x = \frac{n_x}{n_z}z, y = \frac{n_y}{n_x}x & \text{if } |n_x| \geq |n_y|, |n_x| < |n_z| \\ y = \frac{n_y}{n_z}z, x = \frac{n_x}{n_y}y & \text{if } |n_x| < |n_y|, |n_y| < |n_z| \end{cases} \tag{2}$$

Note that, as is done in [37], line symmetry properties are exploited to avoid slopes greater than one. The cells in the sequence are determined by selecting and then applying the appropriate equation in (2) for each successor (n_x, n_y, n_z). The equation is evaluated at the midpoints between cells, i.e. $0.5, 1.5, \ldots, n - 0.5$ cell widths. If the midpoint lies on an edge, cells that share that edge are included in the cell sequence. If the midpoint intersects a corner point, all cells that share that corner point are included. This produces a cell sequence that has Manhattan stepping with non-zero spacing between the trajectory segment and cell boundaries. The horizontal and vertical track angles, θ and ϕ respectively, can be calculated from the displacement as shown in (3) and (4) respectively.

$$\theta = \arctan\left(\frac{n_x \delta x}{n_y \delta y}\right) \tag{3}$$

$$\phi = \arctan\left(\frac{n_z \delta z}{\sqrt{(n_x \delta x)^2 + (n_y \delta y)^2}}\right) \tag{4}$$

where δx, δy and δz correspond to the x, y and z dimensions of each cell respectively. Note that as Γ is specified a priori, there is no need to optimise the cell sequence generation algorithm.

Consider the design of Γ_{3D}. From (3) and (4), it can be seen that arbitrary track angles are possible, however this can result in successors that are displaced by a large cell distance (n_x, n_y, n_z) from s. For example, a horizontal track angle resolution (i.e. maximum angular distance between sample points) of $45°$ is achieved with a maximum cell displacement of 1 ($\max(n_x, n_y) = 1$ assuming square shaped cells). However, for a resolution of $26.6°$, a maximum cell displacement of 2 is required (refer to Fig. 1b). It is possible to reduce the physical track distance by increasing the grid resolution (i.e. make each cell smaller), however this also increases the computation time due to a larger search space. Thus, the design of Γ_{3D} is dependent on the available computation time, desired track length and angle resolution for a specific application.

B. Extending the Successor Operator to 4D

Consider the extension of Γ_{3D} to four dimensions where each cell has dimensions $(\delta x, \delta y, \delta z, \delta t)$; δt specifies a duration of time spent inside a 3D cell. The vector neighbourhood concept extends to the time dimension in that s' lies at a discretised time level s'_{t_l} that is displaced from s_{t_l} by n_{t_l} time levels; n_{t_l} corresponds to the track traversal time.

The cost of traversal of a particular track (needed in many search algorithms [1]) in vehicle motion planning is dependent on the track velocity. Due to the presence of wind, it is not possible to predefine a set cruise velocity for each successor in

the four dimensional successor operator Γ. However, given a 3D successor with displacement (n_x, n_y, n_z), it is possible to generate multiple 4D successors s' with displacement (n_x, n_y, n_z, n_{t_l}) where $n_{t_l} \in N_{t_l}$. The choice of successor time level displacements N_{t_l} is application specific as it is dependent on track lengths, knowledge of expected wind magnitudes and the minimum and maximum cruise velocity of the vehicle. For a given Γ_{3D}, the minimum and maximum cruise velocities can be used to determine an initial estimate of the lower and upper bounds respectively for N_{t_l} using (6) (assuming zero wind). Further refinement of N_{t_l} can be achieved through Monte Carlo simulation over expected wind conditions by inspecting resultant cruise and track velocities for a given choice of N_{t_l}.

Note that for each successor, the cell sequence generated using (2) can be extended to 4D by simply calculating the time level displacement $n^s_{t_l}$ for each cell s on the sequence

$$\begin{cases} n^s_{t_l} = n_{t_l} \frac{n^s_x}{n_x} & \text{if } |n_x| \geq |n_y|, |n_x| \geq |n_z| \\ n^s_{t_l} = n_{t_l} \frac{n^s_y}{n_y} & \text{if } |n_x| < |n_y|, |n_y| \geq |n_z| \\ n^s_{t_l} = n_{t_l} \frac{n^s_z}{n_z} & \text{otherwise} \end{cases} \quad (5)$$

where n^s is the displacement of cell s from the source node. Note that in (5), the spatial dimension with the maximum displacement is used to calculate the cell sequence quantised time level displacements, as this gives the maximum sampling resolution.

Given n_{t_l}, the vehicle cruise velocity can be derived from the track length via the track velocity $\vec{v_t}$.

$$|\vec{v_t}| = \frac{\sqrt{(s'_x - s_x)^2 + (s'_y - s_y)^2 + (s'_z - s_z)^2}}{n_{t_l} \delta_t} \quad (6)$$

This track velocity is itself a sum of the cruise and wind velocity vectors

$$|\vec{v_t}| \cos\phi \begin{pmatrix} \sin\theta \\ \cos\theta \end{pmatrix} = |\vec{v_c}| \cos\phi \begin{pmatrix} \sin\alpha \\ \cos\alpha \end{pmatrix} + \begin{pmatrix} v_{w_x} \\ v_{w_y} \end{pmatrix} \quad (7)$$

where θ and ϕ are the horizontal and vertical track angles respectively, $\vec{v_c}$ is the cruise velocity (e.g. the airspeed of an aircraft), $\vec{v_t}$ the track velocity (e.g. the groundspeed of the aircraft), α is the vehicle heading angle and (v_{w_x}, v_{w_y}) are the horizontal wind magnitudes. All angles are measured from true north in navigational tasks [38]. Note that (7) is formulated in two dimensions instead of three as the horizontal component of the track and track velocity is far greater than the vertical for both UAVs [38] and AUVs [14]. The vertical component of $\vec{v_c}$ and wind are treated as constraints instead. By separating the x and y components from (7) and then solving the resultant simultaneous equations, it is possible to get an expression for $\vec{v_c}$ given $|\vec{v_t}|$ and (v_{w_x}, v_{w_y}).

$$|\vec{v_c}| = \sqrt{|\vec{v_t}|^2 - 2|\vec{v_t}|(v_{w_y}\cos\theta + v_{w_x}\sin\theta) + v^2_{w_x} + v^2_{w_y}} \quad (8)$$

Note that we are not interested in the negative root which corresponds to traversal in the opposite direction.

The preceding section describes in effect a formulation of a vector neighbourhood (obtained using a successor operator) like the one in [12] but for (x, y, z, t). Unlike [12], the ensuing sections describe a multi-step variant of A* [6] (Section III-C) that enables the use of a variable successor operator Γ_s for each node s. This variable operator enables the implementation of multi-resolution search and also enables the imposition of a structure on the search space. These are further discussed in Section IV.

1: **for all** $s \in S$ **do**
2: $g(s) \leftarrow \infty$
3: **end for**
4: $Queue \leftarrow \emptyset$
5: $s \leftarrow s_0$
6: $Queue.Insert(s)$
7: $g(s) \leftarrow 0$
8: **while** $s \neq s_g$ **do**
9: $s = Queue.Pop()$
10: **if** $s = s_g$ **then return**
11: **end if**
12: $S' \leftarrow Succ(\Gamma_s(s))$
13: **for all** $s' \in S'$ **do**
14: $\hat{g}(s') = g(s) + c(s' \| \gamma_{s'})$
15: $\hat{f}(s') = g(s') + \hat{h}(s', s_g)$
16: **if** $\hat{g}(s') < g(s')$ **then**
17: $g(s') \leftarrow \hat{g}(s')$
18: $Queue.Insert(s')$
19: **end if**
20: **end for**
21: **end while**

Fig. 2. MSA* Pseudocode. Note that at line 12, $Succ(\Gamma_s(s))$ extracts the set of successor nodes $s' \in S'$ where s' is the last cell in each cell sequence $\gamma_{s'} \in \Gamma_s(s)$.

C. Search Algorithm

The pseudocode of the MSA* algorithm is listed in Fig. 2. Note that s_0 and s_g refer to the start and goal nodes respectively. Like A*, nodes are placed on a priority queue sorted according to the evaluation function f which is itself the sum of the cost to come, g, and estimated cost to go, $\hat{h}$. In Fig. 2, $Queue.Insert$ refers to the addition of a node s' to the queue such that $f(s^*) \leq f(s)\ \forall s \in Queue$, where s^* is the topmost element in $Queue$. $Queue.Pop$ is the removal of this topmost element.

The key distinction between MSA* and A* lies in the cost function c. A* computes the cost (a scalar value) as a function of the cells s and s', whereas the MSA* cost is a function of multiple cells as defined in the cell sequence (equation (1)).

$$g(s') = g(s) + c(s' = s_{j+m-1} \| s = s_j, \ldots, s_{j+m-2}) \quad (9)$$

The cost c is calculated using a two step process. Firstly, the decision variables x_i (e.g. fuel, risk) are uniquely mapped or calculated from the cell sequence such that $x_i = \rho_i(s, \ldots, s')$ where ρ_i is the mapping function. For example, the total risk probability is the sum of the risk probability density value for each cell on the cell sequence. Note that constraints can be imposed by setting $c = \infty$ if a particular decision variable exceeds a specified limit (e.g. maximum risk). Otherwise,

a Multi-Criteria Decision Making (MCDM) cost function is used to transform the decision variables into a single cost term, c, where c is non-zero and monotonic (i.e. $c > 0$). This second step could be a weighted sum aggregation (like that used in [27]–[30]), or a fuzzy mapping (e.g. [31]). Using a weighted sum approach, each decision variable is mapped onto a commensurate scale on the interval $[0, 1]$ using a value function $u_i(x_i)$. The final cost is $c = w_0u_0(x_0) + \ldots + w_{n-1}u_{n-1}(x_{n-1}) + \delta_c$ where n is the number of decision criteria and δ_c is a small positive value to ensure $c > 0$. A comprehensive evaluation of the decision objectives and decision variables for UAV flight planning is provided in [39] (refer to Section V-A for a brief summary).

D. Cost Optimality of MSA*

It can be shown that MSA* will find the least cost path given a predefined set of successor operators Γ_s for each node $s \in S$ and MCDM cost function c. As MSA* is derived from A* [6], cost optimality can be shown in a similar manner as well.

Lemma 1: Consider any globally optimal path $P^* = (s_0, \ldots, s_n)$. It can be shown that P^* is itself composed of optimal paths.

Proof: The optimal path for a given node s_{i_k} is a path $P^* = (s_0, \ldots, s_{i_k})$, such that for all possible paths $P \in \Pi$, $g(s_{i_k}\|P^*) < g(s_{i_k}\|P)$. Recall that any path is made up of an integer number of trajectory segments K and each segment is represented by a cell sequence of length m_j for segment j. Thus, the index to a node (or cell) in P^* at the k^{th} trajectory segment is

$$i_k = -k + \sum_{j=0}^{k-1} m_j \tag{10}$$

For the case where $k = K$, the lemma is trivially true by definition of $P^*_{i_K}$ as $i_K = n$, $P^*_n = (s_0, \ldots, s_n)$. Consider the case for $k = K - 1$ trajectory segments, whose cell sequence $P_{i_{K-1}} = (s_0, s_1, \ldots, s_{i_{K-1}})$ is a subset of the optimal path P^*. If $P_{i_{K-1}}$ is not a least cost path, then there exists another path $P'_{i_{K-1}} = s_0, s'_1, \ldots, s'_{i_{K-2}}, s_{i_{K-1}}$ such that $g(P'_{i_{K-1}}) < g(P_{i_{K-1}})$. But, as the $K - 1^{th}$ cell sequence $s_{i_{K-1}}, \ldots, s_{i_K}$ is unchanged, then given (9), the cost term c is unchanged. This implies that there exists a path $P'_{i_K} = (s_0, s'_1, \ldots, s'_{i_{K-2}}, s_{i_{K-1}}, \ldots, s_{i_K})$ such that $g(P_{i_K}) < g(P^*_{i_K})$, contradicting the definition of $P^*_{i_K}$. Therefore, $P_{i_{K-1}}$ must also be an optimal path. By mathematical induction, any optimal path P^* must itself be composed of optimal paths. ■

Theorem 1: If the heuristic is admissible [6], then MSA* will find the optimal path if one exists. An admissible heuristic $\hat{h}$ is an estimate of the cost to go that is always less than the actual cost to go, $\hat{h}(s_j, s_g) \leq h(s_j, s_g)$.

Proof: Consider an optimal path $P^* = (s_0, \ldots, s_g)$ which contains K trajectory segments. On an optimal path, $g(s_j) = g^*(s_j)$ for all $j = (0, 1, \ldots, i_K)$. From line 15, because $\hat{h}(s_j, s_g) \leq h(s_j, s_g)$, therefore, $f(s_j) \leq f^*(s_j)$.

In the trivial case where $K = 0$ (i.e. $s_0 = s_g$), MSA* discovers the solution in one iteration. During initialisation (lines 1-7 in Fig. 2), s_0 is placed on the queue with cost $g(s_0) = g^*(s_0) = 0$. Upon expansion of s_0, MSA* terminates.

Consider the case where $K = 1$, i.e. $s_g \in S'_0$ where $S'_0 = \Gamma_{s_0}(s_0)$. Let S^* denote the set of nodes on the queue that lie on an optimal path. After one iteration (i.e. expansion of s_0), at least one of the successors s'_0 is a member of S^*. Consider the contrary where none of the nodes $s'_0 \in S'_0$ lie on an optimal path and/or are not on the queue. There are two possibilities, one is that none of the successors are reachable (in which case no path exists) or at least one of them lies on a least cost path. Note that if a path exists, then an optimal path also exists.

For this latter case, given a node s'_0, an optimal path $(s_0, \ldots, s'_0)$ must exist that does not contain any nodes in S'_0 because by Lemma 1, an optimal path $(s_0, \ldots, s_n)$ comprises optimal paths $(s_0, \ldots, s_{n-1})$, $(s_0, \ldots, s_{n-2}), \ldots$ and none of the nodes in S'_0 lie on an optimal path (by assumption). This is not possible, hence, at least one node s'_0 must lie on an optimal path in which case, by line 16, s'_0 would be added to the queue at the expansion of s_0. Therefore, where a path exists, $S^* \neq \emptyset$ and $s_g \in S^*_0$ for the scenario $K = 1$.

The preceding argument can be extended to show that up until algorithm termination, $S^* \neq \emptyset$. Let S'_k denote the set of nodes generated by $\Gamma_{s_{i_{k-1}} \in S'_{k-1}}(\Gamma_{s_{i_{k-2}} \in S'_{k-2}}(\ldots \Gamma_{s_0}(s_0)))$ - i.e. *all* nodes that can be reached in k trajectory segments. From above, all nodes $s : s \in S^*, s \in S_k$ $(k = 1)$ are placed on the queue after the first iteration. If optimal paths of length $k + 1$ exist, expansion of $s : s \in S^*, s \in S_k$ must yield nodes $s' : s' \in S^*, s' \in S_{k+1}$. Otherwise, as before, there would exist nodes $s' \notin S_k$ that result in optimal paths of length $k+1$ trajectory segments which is not possible. Hence, $S^* \neq \emptyset$.

Because Γ_s is a finite set for all $s \in S$ and because trajectory segments incur a non-zero and non-negative cost c, there are only a finite number of nodes such that $f(s) \leq f^*(s_g) = f^*(s_0)$. Therefore, as $S^* \neq \emptyset$, the nodes on the optimal path $P^* = (s_0, s_{i_1}, \ldots, s_{i_k} = s_g)$ are expanded (in sequence) in a finite number of iterations, terminating with the expansion of s_g.

It is not possible to terminate without finding the optimal path if one exists. Consider the scenario where MSA* terminates such that $f(s_g) = g(s_g) > f^*(s_0)$. But, by the analysis above, there exists a node $s \in S^*$ just before termination such that $f(s) \leq f^*(s_0) < f(s_g)$. Hence, s would be expanded instead of s_g, contradicting the assumption that MSA* would have terminated. Therefore, MSA* will find an optimal path $(s_0, \ldots, s_g)$ in finite time where such a path exists.

■

IV. Multi-Resolution Lattice Structure

A class of variable successor operators is presented that can be used to implement a lattice based multi-resolution search space for the purposes of reducing computation time. The use of variable successor operators Γ_s is made possible by the MSA* search algorithm. As before, the three dimensional lattice structure is presented first followed by a conceptual extension to 4D. It is shown that lattice based MSA* reduces the size of the search space without sacrificing track angle resolution or soundness [40].

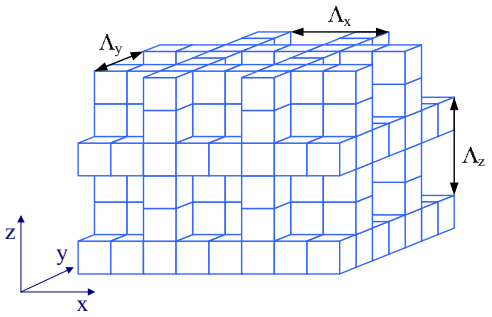

Fig. 3. General lattice structure, (x, y, z) dimensions shown.

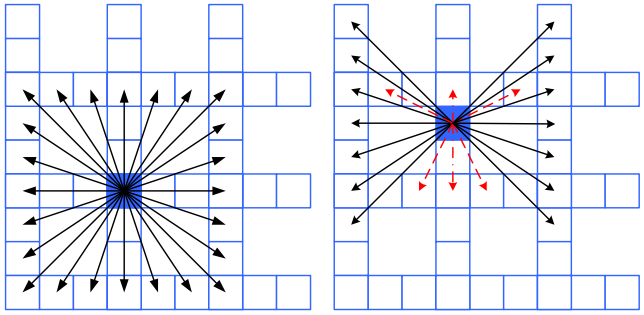

Fig. 4. Top (x-y) view of lattice showing trajectory segments for (a) lattice position $(0, 0)$, (b) $(0, 2)$.

A. 3D Lattice

A 3D illustration of the lattice structure is presented in Fig. 3 with a 2D cut-away (x-y) view showing the source-successor trajectory segments in Fig. 4. The lattice comprises a series of planes parallel to the x-y, y-z and x-z Cartesian planes at regular intervals of Λ_x, Λ_y and Λ_z (defined in terms of the number of cells) in the x, y and z planes respectively.

The lattice design methodology is a three step process. Firstly, a series of base 3D successor operators Γ_0 are chosen, one for each search space resolution. The choice of Γ_0 takes into account the required track angle resolution and track length, both of which are related to the sampling density. Γ_0 must be such that, in any plane x-y, x-z or y-z, all successors lie on the border of a rectangle centred at the source node (as in Fig. 4a).

Using Γ_0, it is then possible to define the spacing between planes in the lattice, Λ_x, Λ_y, Λ_z using

$$\begin{aligned}\Lambda_x &= \sup_{\gamma_{n_x}} \left\{\gamma \in \Gamma_0 : \gamma_{n_x} \geq \gamma_{n_y}, \gamma_{n_z}\right\} \\ \Lambda_y &= \sup_{\gamma_{n_y}} \left\{\gamma \in \Gamma_0 : \gamma_{n_y} \geq \gamma_{n_x}, \gamma_{n_z}\right\} \\ \Lambda_z &= \sup_{\gamma_{n_z}} \left\{\gamma \in \Gamma_0 : \gamma_{n_z} \geq \gamma_{n_x}, \gamma_{n_y}\right\}\end{aligned} \tag{11}$$

where sup denotes the supremum operator (least upper bound). Note that if the start or goal nodes do not lie on the lattice, it is a simple matter to connect those nodes to one that is on the lattice using a local search technique (refer to [1]).

Using this lattice structure, it is then possible to define individual Γ_s operators for each node on the lattice. This is shown for a 2D lattice for the sake of clarity in Fig. 4. Due to the regularity of the lattice, nodes located at equivalent positions on the lattice share the same successor operator $\Gamma_{\vec{p}}$ for position $\vec{p}$. Two positions are equivalent if and only if they are separated by integer multiples of Λ_x, Λ_y and Λ_z cells. A lattice position $\vec{p}$ can be uniquely defined based on the modulus

$$\vec{p} = (mod(x, \Lambda_x), mod(y, \Lambda_y), mod(z, \Lambda_z)) \tag{12}$$

Referring to Fig. 3, the total number of unique lattice positions n_p is

$$n_p = \Lambda_x\Lambda_y + \Lambda_x(\Lambda_z - 1) + (\Lambda_y - 1)(\Lambda_z - 1) \tag{13}$$

Consider the case where the source node is at lattice position $(0, 0, 0)$, i.e. at the intersection of lattice planes (Fig. 4a). The trajectory segments are chosen to terminate at successors that lie on the border of a rectangle centred on the source node with dimensions of $(2\Lambda_x, 2\Lambda_y, 2\Lambda_z)$ cells. Therefore, $\Gamma_{\vec{p}=(0,0,0)} = \Gamma_0$. For cells located at different positions on the lattice (refer to Fig. 4b), the successors are chosen to maximise the number of identical successors to $\Gamma_{(0,0,0)}$. Additionally, where possible, the successors are chosen to terminate at lattice position $(0, 0, 0)$. This ensures that the same track angle can be maintained over consecutive trajectory segments to avert unnecessary turns.

B. 3D Multi-Resolution Lattice

The purpose of multi-resolution sampling is to reduce the total number of nodes and thus reduce computation time. In many applications, it is possible to divide the search space into regions of fine sampling resolution and regions of coarse sampling resolution. In UAV flight planning for example, fine sampling resolution is required at lower altitudes but a coarse sampling resolution can be used for high altitude en-route airspace. It is easy to implement multi-resolution search by using multiple base successor operators Γ_0^i.

Consider the division of the search space into a series of N rectangular prism shaped regions each of which has a lattice resolution of $(\Lambda_x^i, \Lambda_y^i, \Lambda_z^i)$, $i = 1, 2, \ldots, N$ and a base successor operator Γ_0^i. Note that N is typically a small number as it is necessary to check the bounds of each region at every iteration. Each region must have dimensions that are a multiple of $\Lambda_x^i, \Lambda_y^i, \Lambda_z^i$. This ensures that all trajectories must terminate on and originate from a lattice plane separating the two regions (refer to Fig. 5).

Furthermore, assume that for any two adjacent regions i and j, $\Lambda_x^i, \Lambda_y^i, \Lambda_z^i$ is an integer multiple of $\Lambda_x^j, \Lambda_y^j, \Lambda_z^j$. This way, successors in Γ_0^i are displaced at integer multiples of those in Γ_0^j, which ensures that all horizontal track angles (calculated using (3)) in region j also exist in region i (see Fig. 5). Unfortunately, this is not the case for the vertical track angle even though angles in the x-z and y-z planes are replicated in region i. To avoid a large increase in the number of successors, it is also possible to filter out successors in Γ_0^i that are on track angles not represented in Γ_0^j.

The aforementioned multi-resolution lattice provides a means for fine and coarse sampling corresponding to smaller and larger values of Λ respectively. There is no reduction

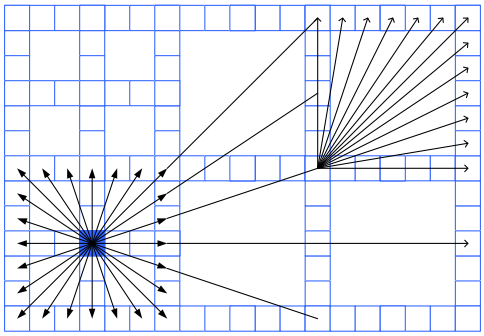

Fig. 5. Multi-resolution lattice with $\Lambda_x = \Lambda_y = 3$ on the left and $\Lambda_x = \Lambda_y = 6$ on the right. Selected trajectory segments are shown for the sake of clarity.

in track angle resolution for coarsely sampled regions as larger values of Λ enable higher track angle resolutions. Additionally, each track is evaluated at the cell resolution using cell sequences defined in Section III-A. This guarantees path soundness [40, Ch. 25.6] by avoiding the problem of mixed cells (which contains free space and obstacles [1]).

C. 4D Multi-Resolution Lattice

Extension of the previous 3D lattice to four dimensions involves selection of a suitable set of values $n_{t_l} \in N_{t_l}$ for each 3D successor in $\Gamma_{\vec{p}}$ using the methodology described in Section III-B. The full 4D multi-resolution lattice structure is implicitly defined through the variable successor operators where, at each iteration of MSA* search, $\Gamma_{\vec{p}}$ is selected based on (12) and the boundaries of each Γ_0^i region. This approach enables consistent track angle resolution across fine and coarse resolution regions without sacrificing soundness.

In addition, the lattice structure reduces memory usage. The underlying cell grid of MSA* contains $N_x N_y N_z N_{t_l}$ nodes where N_x, N_y, N_z and N_{t_l} are the total number of sample points in the x, y, z and t dimensions respectively. The total memory requirement (i.e. number of nodes) N for a lattice can easily be derived by counting the number of nodes in the x, y and z planes (refer to Fig. 3) and then subtracting overlap regions.

$$N = N_{t_l}\left(N_y N_z \alpha_x + N_x N_z \alpha_y + N_x N_y \alpha_z - \alpha_x \alpha_y N_z - \alpha_x \alpha_z N_y - \alpha_y \alpha_z N_x + \alpha_x \alpha_y \alpha_z\right) \quad (14)$$

where $\alpha_x = \left\lfloor \frac{N_x - 1}{\Lambda_x} + 1 \right\rfloor$ (and similarly for y and z) and all division operations are integer divisions. Note that the $N_y N_z \alpha_x$ term in (14) counts the number of nodes in the x plane for all x planes, where α_x is the number of x planes (similarly for y and z). The term $-\alpha_x \alpha_y N_z$ subtracts overlaps between the x and y planes (similarly for overlaps between the x and z, and y and z planes), and $\alpha_x \alpha_y \alpha_z$ represents the overlap between the $\alpha_x \alpha_y N_z$, $\alpha_x \alpha_z N_y$ and $\alpha_y \alpha_z N_x$ terms.

V. Experimental Analysis

This section discusses some of the practical aspects of implementing the proposed algorithm including evaluation of MSA* against an existing vector neighbour based search algorithm (like [12]) in simulation. Such a comparison is used to evaluate the computational efficiency of MSA*.

A. UAV Flight Planning Application

The UAV mission flight planning problem was chosen to provide a practical context for evaluation of MSA*. This is an important application as onboard mission flight planning (especially online replanning) has been shown to be a key enabler in the operation of UAVs in the National Airspace System (NAS) [39]. The mission being undertaken is the delivery of a medical package to a remote location using a small UAV. This mission is operated under Visual Flight Rules (VFR) using Australian Civil Aviation Regulations (CAR) [41]. The medical delivery task is ideal for evaluating a 4D search algorithm due to the presence of multiple decision criteria, dynamic elements in the operating environment and the significant effect of wind on a small UAV. The three major decision objectives for the medical package delivery mission are safety, the rules of the air and mission efficiency [39].

The safety objective is modeled with the aircraft separation management, storm cell avoidance and population risk criteria. For simulation purposes, the aircraft separation requirement (5NM horizontal and 1000ft vertical) for en-route airspace is adopted. The cylindrical shaped separation region is represented with an approximate, probabilistic model [39]. This model is similarly used to describe storm cells. Finally, the population risk criterion refers to the minimisation of the risk presented to people and property on the ground in the event of a crash. For the purposes of simulation, this risk value is approximated with a Normalised Population Density (NPD) value.

The flight plan must also conform to the rules of the air, such as the cruising levels rule, low flying restrictions (minimum altitude above ground level) and segregated airspace (avoiding no-fly zones). For aircraft flying on headings from $0°$ to $179°$, the permissible flight levels are at odd multiples of 1000ft plus 500ft Above Mean Sea Level (AMSL) (e.g. 1500ft, 3500ft, 5500ft AMSL). For headings between $180°$ and $359°$, the cruise levels are at even multiples of 1000ft AMSL plus 500ft (e.g. 2500, 4500, 6500ft AMSL). The cruising levels rule is intended to minimise the risk of a head-on collision, and is mandatory above 5000ft.

The flight plan also needs to optimise for the objectives of the mission itself (i.e. the delivery task). These objectives include the delivery time (i.e. the time of arrival at the goal node) and fuel consumption. With a 4D search, it is possible to not only find a path that minimises the delivery time, but also to designate a specific delivery time or acceptable time window (like with [16]). The flight time, along with the cruise velocity, altitude and rate of climb are parameters used to optimise fuel consumption.

These decision variables, in combination with the dynamic constraints of the aircraft, are used to calculate the cost term c in (9). Note that for the purposes of planning, it is assumed that all situational awareness information (e.g. wind information and information about other aircraft) is available. For further details and candidate data sources for each decision variable,

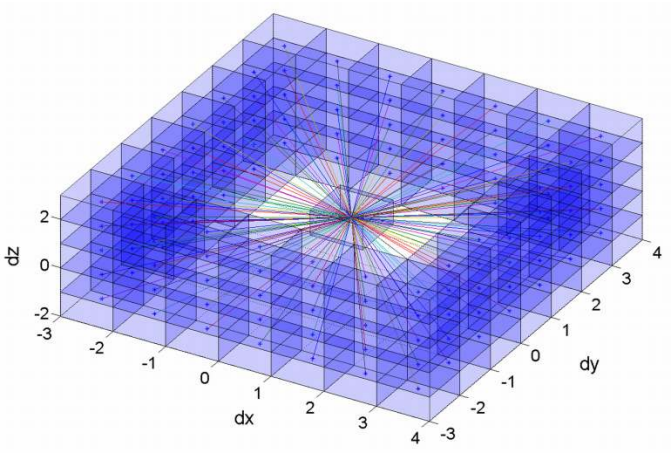

Fig. 6. Chosen Γ_0 successor operator, showing source cell (centre) and successor cells joined by trajectory segments. $\Lambda_x = 3, \Lambda_y = 3, \Lambda_z = -\infty$, $N_{t_l} = \{2, 3, 4\}$min. Note that $\Lambda_z = -\infty$ corresponds to a lattice where there are no z-planes.

refer to [39]. Note that as the dynamic constraints are closely linked with the chosen successor trajectory segments in Γ, these are discussed together in the following subsection.

B. Experimental Setup

The primary purpose of this experimental analysis is to compare the computational efficiency and solution path of the proposed algorithm with that of existing algorithms. In these comparisons, each test algorithm uses a different (set of) successor operator(s) but the same cost function c (a weighted sum of utility values) and heuristic function h.

1) Test Algorithms: Two different variants of MSA* are compared against a benchmark algorithm, Vector A* on 1000 randomly generated planning scenarios. Vector A* is a direct extension of A* using a vector neighbourhood (like that used in [12]). The successor operator is chosen to reflect the base successor operator used in the other test algorithms and is shown in Fig. 6. Using this successor operator, there are 360 successors for each source node. Vector A* is in effect a special case of MSA* where Γ_s is constant throughout the entire search space. Due to the popularity of A* and related algorithms in robotics [42], Vector A* serves as an ideal benchmark of calculation time and path cost for a deterministic 4D planning algorithm with selectable track angle resolution. Two variants of MSA*, MSA*1 and MSA*2, are tested against the benchmark. MSA*1 uses a single, fine resolution lattice based on Γ_0 as shown in Fig. 6. MSA*2, on the other hand, uses a multi-resolution lattice where $N = 2$. A fine resolution lattice (based on Γ_0 in Fig. 6) is used for altitudes below 7000ft, and a coarse resolution lattice ($\Lambda_x = 6, \Lambda_y = 6, \Lambda_z = -\infty$, $N_{t_l} = \{4, 6, 8\}$min) is used for altitudes above 7000ft.

All experiments were performed on a 3.3GHz Intel Core 2 Duo QX6850 CPU with 4GB of physical RAM running 32-bit Microsoft Windows XP.

2) Dynamic Constraints: The dynamic constraints of the aircraft were considered in the selection of Γ_0 in Fig. 6. Oftentimes, these dynamic constraints are modeled with a minimum turn radius [1]. Assuming a maximum airspeed of 126 knots, a bank angle of 30° and a force of one g ($32.2ft/s^2$), the worst case turn radius is approximately 0.4NM (refer to [38, (3.9.10)]). As the turn radius is less than half the cell size, it is possible to execute a 180° turn within the bounds of a single cell. However, it is still desirable to minimise the turn angle as it is difficult for the flight controller to execute such a turn with accuracy under strong wind conditions. Turn angles are further discussed in Section V-C.

For UAV operations, it is also necessary to incorporate climb/descent rate constraints and a maximum airspeed constraint (a constant value). The maximum climb rate, however, decreases with altitude and is zero at the aircraft ceiling [38]. At sea level, the maximum climb rate for a small UAV is limited to approximately 1000ft/min [43]. This matches the maximum climb rate achieved using Γ_0 under no wind conditions:

$$\frac{\max\limits_{n_z}(n_z \delta_z)}{\min\limits_{n_{t_l} \in N_{t_l}}(n_{t_l} \delta_t)} = 1000\text{ft/min} \tag{15}$$

3) Simulation World: The simulation worlds were generated randomly to enable a Monte-Carlo evaluation of the test algorithms. Each simulation world comprises a terrain map, no-fly zones, other aircraft, storm cells, wind map and population density map (as a simple model of risk presented to people and property on the ground). For each world, a number of start and goal pairs were randomly chosen. The mission area for each world was arbitrarily chosen to be 50NM×50NM×15000ft×90min with a cell resolution of 1NM×1NM×1000ft×1min (1NM=1852m, 1ft=0.3048m). Note that the maximum distance of the search area approximately matches the maximum operating range of the RQ-7A Shadow UAV [43].

An artificial terrain map is randomly generated through summation of bivariate Gaussian functions with randomly chosen parameters (A, b, c, σ, n). Population density is also generated using this equation.

$$z(x, y) = \sum_{i=0}^{n} A_i e^{-\frac{(x-b_i)^2 + (y-c_i)^2}{\sigma_i^2}} \tag{16}$$

Maps for the other decision variables can also be randomly generated through random selection of parameter values. For example, the parameters for a cylindrical aircraft separation zone are position, velocity, standard deviation, radius and height. The velocity is assumed to range between 50 knots and a speed limit (for flight below 10000ft) of 250 knots [41]. The radius and height are specified in aviation regulations. Similarly, storm cells are randomly generated with an average radius of 13.5NM and height of 8NM [44, Fig. 5]. The rate of movement of a storm cell is assumed to be between 10 to 40 knots for altitudes between 0 and 15000ft [44, Fig. 1]. A method for modeling each of the remaining decision variables is provided in [39].

Finally, a simple algorithm was used to generate wind maps that mimic real world winds. Firstly, a number of seed nodes are randomly generated at different positions (x, y); each seed is characterised by a position $\vec{p}_i$, a direction ϕ_i and a vector of wind magnitudes $\vec{m}_z$ for each altitude level z. For each

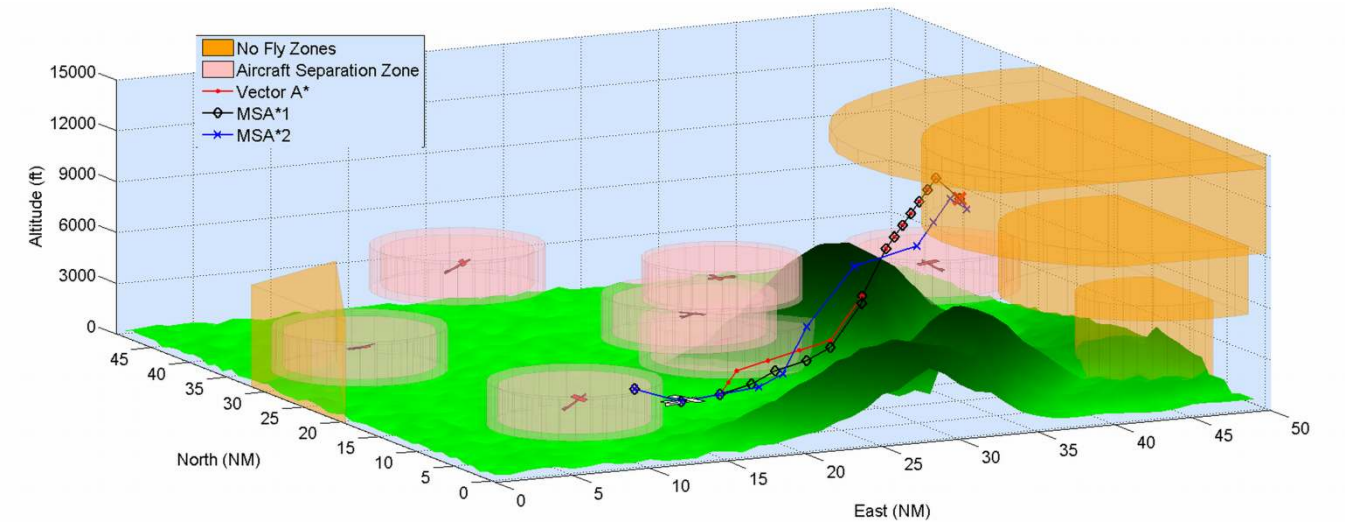

Fig. 7. Example planning scenario showing no-fly zones and other aircraft at $t = 120s$. Note for aircraft and weather, the inner cylinder represents the separation zone/storm cell extents (around the expected position) and the outer cylinder is the 2σ uncertainty boundary (which grows with time). Note also that a red X marks the goal position.

node $s = (x, y, z)$ in the world space, a weighting vector $\vec{u}$ is calculated

$$u_i = a_0 \left|\vec{d_i}\right| + a_1 \left|\angle\vec{d_i} - \phi_i\right| \tag{17}$$

where $\vec{d_i} = \vec{s} - \vec{p_i}$ and a_0, a_1 are weights. The largest element in $\vec{u}$ is then scaled by a^*; this gives the "winning seed" more weighting. For a given node s, the wind magnitude is $f_m(s) = \vec{u} \cdot \vec{m}_z$. The wind magnitude for each altitude level is randomly chosen based on average wind speeds (refer to [39]). The direction f_d is calculated in a similar manner, $f_d(s) = \vec{u} \cdot \vec{\phi} + \sigma_z$ where σ_z is a small, random perturbation added to simulate wind shear.

C. Results

A Monte-Carlo simulation of the three tests algorithms is performed on 1000 randomly generated planning scenarios. The results of these simulations are presented below and evaluated with respect to computation time and path cost. In addition, the algorithms are also evaluated on three special case tests scenarios. These were constructed to determine the effect of local minima and to test the adaptability of the planner to situations where the vertical wind velocity exceeds aircraft performance.

An illustration of a typical multi-objective planning scenario is provided in Fig. 7 (showing other aircraft and no-fly zones) and Fig. 8 (risk map). The solution path using Vector A*, MSA*1 and MSA*2 are also shown on each of these figures. In Fig. 7, all three planners select a path that avoids an aircraft on a converging course by descending and heading in an easterly direction. Once a risk of collision is averted, the paths continue in a shortest path fashion towards the goal (marked by a red cross). There are deviations only to avoid terrain (where the paths hop over a mountain in Fig. 7) and route around high risk (population density) areas (Fig. 8). Note that, as shown in Section III-D, each algorithm finds a path that satisfies all given constraints whilst minimising the overall path cost (which is a multi-objective cost function).

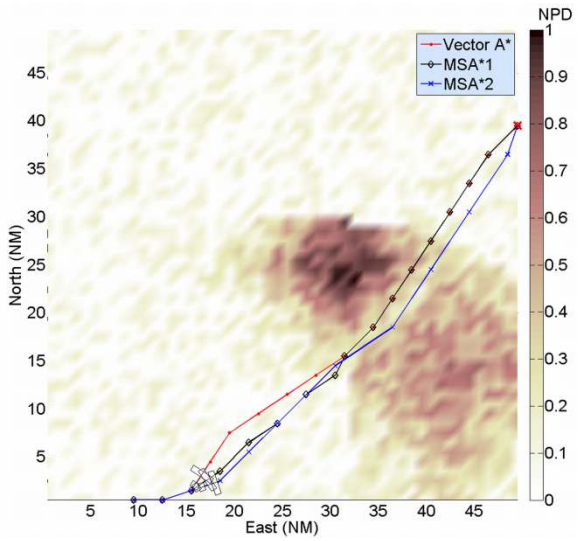

Fig. 8. Example planning scenario risk represented by a Normalised Population Density (NPD) map.

1) Computation time: The mean and standard deviation for the computation time (μ_t and σ_t respectively), along with the minimum and maximum computation time, and the loop count (μ_n and σ_n respectively) are presented in Table I for each test algorithm. From the results, it can be deduced that a lattice based successor operator can significantly reduce total computation time and that further time savings can be achieved with a multi-resolution lattice. A cumulative histogram of the speed increase is provided in Fig. 9 along with a statistical summary of the speed increase in Table II. For the test resolution level, Vector A*, MSA*1 and MSA*2 are all suitable for onboard replanning as the computation time is well within the minimum track traversal time of 2min. as specified in N_{t_l}.

2) Total path cost: The mean μ_c and standard deviation σ_c in the ratio of the path cost between MSA*1, MSA*2 and Vector A* is presented in Table III. A cumulative histogram

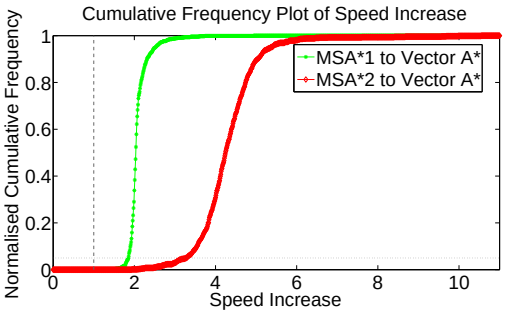

Fig. 9. Normalised frequency histogram of speed increase of MSA*1 and MSA*2 to Vector A*.

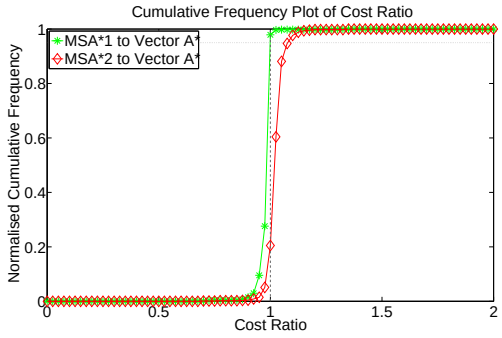

Fig. 10. Cumulative histogram of relative path cost.

of the ratios is illustrated in Fig. 10. As the successor operator $\Gamma_{\vec{p}}$ in MSA*1 is largely similar to Γ in Vector A*, it is not surprising to find that both return paths of approximately equivalent cost. Of particular note however, is the fact that, on average, MSA*2 finds paths that are only 3.3% costlier than Vector A*. Therefore, it can be seen that MSA* finds paths of equivalent cost but with significantly less computation time.

It is observed that each of the three test algorithms return a solution path that tends to follow the profile of a straight line (shortest path). This is attributable to the minimisation process of A* and the fact that all trajectory segments have a non-zero and non-negative cost value. As a result, the turn angles are typically small even without explicit optimisation of turn angles. The mean turn angle (and standard deviation) for Vector A*, MSA*1, MSA*2 is 11.7° (19.0°), 12.7° (22.4°) and 16.4° (21.9°) respectively.

TABLE I
COMPUTATION TIME AND LOOP COUNT

	μ_t	σ_t	min	max	μ_n	σ_n
MSA*2	4.46s	2.36s	0.31s	16.6s	65501	34250
MSA*1	9.23s	4.93s	0.53s	35.2s	161571	88386
Vector A*	19.25s	10.41s	0.81s	77.9s	289015	160575

TABLE II
SPEED INCREASE OVER VECTOR A*

	Percentile (%)						
	≤0	≤1	≤25	≤50	≤75	≤100	Mean
MSA*1	0.98	1.69	1.98	2.04	2.12	4.94	2.09
MSA*2	1.94	2.52	3.91	4.25	4.66	10.55	4.30

TABLE III
PATH COST RATIO

	μ_c	σ_c
MSA*1 to Vector A*	0.9891	0.0307
MSA*2 to Vector A*	1.0334	0.0376

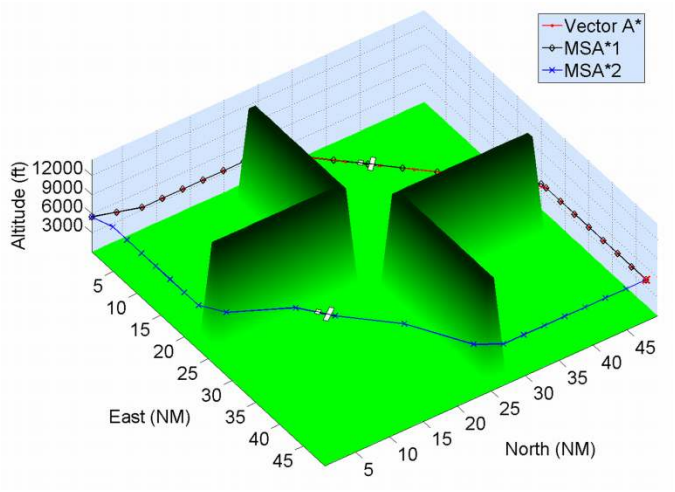

Fig. 11. Double bug trap case.

D. Special Cases

It is widely acknowledged that A*, and best-first search algorithms in general require significantly more computation time in the presence of local minima [1]. This was tested for the single and double bug trap case as recorded in Table IV; the double bug trap case is illustrated in Fig. 11. It can be seen that even though the absolute computation time is approximately double to 2.5 times the mean obtained in the previous Monte Carlo simulations, the relative computation time between Vector A*, MSA*1 and MSA*2 remains approximately the same as before.

A simulation scenario which mimics the presence of strong up/downdrafts in mountainous regions (where the vertical wind velocity can exceed the vehicle's climb rate) is depicted in Fig. 12. Even though a variety of wind conditions were simulated in the previous Monte Carlo experiment, this experiment specifically studies the effect of wind by setting other decision variables (e.g. no-fly zones, other aircraft, storm cells, risk) to zero. As shown in Fig. 12, only MSA*2 successfully finds a traversable path that satisfies aircraft climb constraints. The chosen path climbs in a switchback pattern before 'hopping over' the mountain. Recall that, as the aircraft's maximum

TABLE IV
COMPUTATION TIME

	Single Bug Trap		**Double Bug Trap**	
	Time	Loop Count	Time	Loop Count
MSA*2	9.59s	201197	9.17s	194981
MSA*1	23.45s	515657	22.61s	503871
Vector A*	51.14s	922544	50.65s	926276

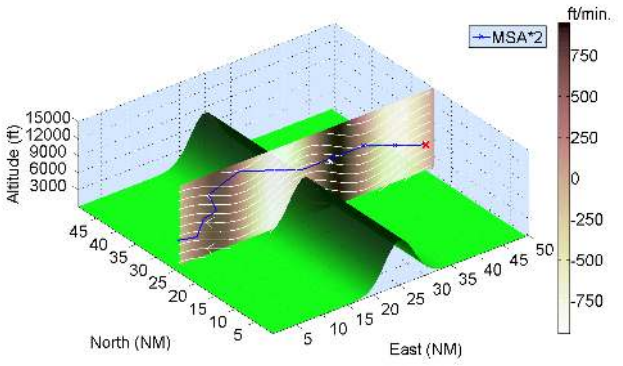

Fig. 12. Mountain wind simulation. The solution is found in 5.14s.

climb rate decreases with altitude, it is necessary to climb to 13500ft to accommodate loss of altitude in the downdrafts region. The reason that MSA*2 successfully finds a path while Vector A* and MSA*1 fail is because of the higher climb and descent rate required by the fine resolution successor operator. The same vertical distance is covered in a shorter time using the fine resolution operator (e.g. 1000ft in 2, 3 or 4min.) compared to the coarse resolution operator (e.g. 1000ft in 4, 6 or 8min.).

VI. Discussion

This section reviews the proposed algorithm with respect to existing work in light of the practical simulation results and theoretical findings.

A. Online Replanning

The primary objective of the simulation experiments presented in Section V is to determine whether the algorithm is fast enough for online replanning. Online replanning is needed to mitigate the uncertainty and unpredictably of an outdoor operating environment. Consider the practical implementation of a planner (such as MSA*) on a UAV. A replan is triggered when the environment changes or when the vehicle deviates beyond tolerance bounds on the originally planned path. It is assumed that a predictor module provides the planner (e.g. MSA*) with a start node such that the time required to reach the start node (from the current state) is a conservative estimate of the planning time. Thus, the planner completes planning (whilst the UAV is flying under reactive local control) prior to reaching the predicted start node – if this is not the case, the process is repeated. In practice, if the planning time is short relative to the dynamics of the mission (i.e. any changes to the predicted operating environment are within tolerance bounds during planning), this would not introduce instability into the overall control and planning loop.

An indicator of the available planning time can be derived from the minimum track traversal time as the motion plan is made up of discrete trajectory segments (i.e. tracks). For the successor operator selected in Fig. 6, the mean planning time for MSA*2, MSA*1 and Vector A* (of 4.46s, 9.23s and 19.25s respectively) was found to be much smaller than the minimum track traversal time (2min). All three tests algorithms were also shown to be capable of finding a path within the time constraints of online replanning for environments containing deep local minima and narrow escape passages (as per Fig. 11).

In these experiments, it was found that the proposed algorithm offered an approximately two-fold reduction in computation time. Further reductions were obtained by using a multi-resolution lattice. This increased speed is significant when planning over a larger search space or on a more difficult search space. For a 100NM$\times$100NM$\times$15000ft$\times$180min. search space, a similar Monte Carlo experiment found that MSA*2 is still able to meet the requirements for online replanning with a mean planning time of 48.25s and standard deviation of 20.62s (which is less than the minimum track traversal time of 2min.). However, Vector A* does not as it takes approximately four times the computation time. Similarly, when presented with local minima, which are known to be difficult to solve for best first search algorithms like A*, the computational efficiency of MSA* over Vector A* is significant in meeting online replanning constraints [1].

The previous discussion of algorithm computation time assumes that in each case, we plan from scratch. This approach of always discarding previous planning information was adopted because, in applications such as UAV package delivery, online changes can occur anywhere in the search map and affect large swaths of the search space. If a large number of nodes are changed and/or changes occur close to the goal node, replanning algorithms like D* and D* Lite are *less* efficient than one that plans from scratch [42]. The presence of fast moving aircraft and storm cells for example can affect large areas of the search space that are not necessarily localised around the vehicle's current position. Hence, it is more efficient to plan from scratch each time.

Due to the time critical nature of online replanning, it is preferable to use a fast and near-optimal planner rather than an optimal planner which may be too slow. Under these conditions, MSA*2 is the best candidate for online replanning out of the three test algorithms.

B. Lattice Structure

The computational efficiency of MSA* compared to Vector A* can be attributed to a smaller, lattice structure based search space compared to a full 4D grid. With the exception of the successor operator and cell sequence based sampling, Vector A* is virtually identical to MSA*. Using (14), a plot of the number of nodes for a search space of dimensions $50\times50\times15\times90$ given different values of Λ assuming $\Lambda_x = \Lambda_y$ is shown in Fig. 13. Note that the memory required in a full grid corresponds to the case where $\Lambda_x = 1$, or $\Lambda_y = 1$, or $\Lambda_z = 1$.

From Fig. 13, it is evident that larger values of Λ produce a lattice structure with fewer sample nodes. However, the corresponding successor operator Γ_s has potentially more successors per node with greater track angle resolution and greater track length. As a result, fewer search iterations are required but each iteration incurs more computation time. For example, it is possible to evaluate 17400 nodes per second

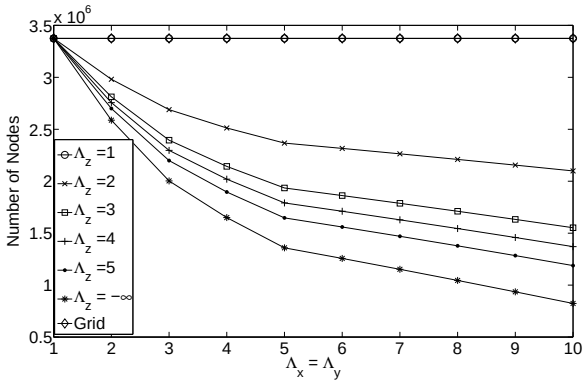

Fig. 13. Number of nodes in the search space for different values of Λ. Note that $\Lambda_z = -\infty$ corresponds to the constrained vertical track angle case described in Section IV-A.

(on average) using MSA*1 whereas only 14750 nodes per second are possible (on average) using MSA*2. It can be seen that the variable successor operator Γ_s is a crucial, application-specific design parameter that influences the path cost, the traversability of the path and computation time. For the demonstration UAV application, it has been shown that the selected fine resolution and especially the multi-resolution successor operators are effective at delivering a solution of comparable cost with significant savings in computation time. The use of a coarse resolution successor operator Γ_s for high altitudes in MSA*2 is especially suited to UAV planning because of the scarcity of obstacles and reduced climb rate at high altitudes (refer to Section V-D).

The lattice structure presented here is similar to the framed quad/octree presented by [9]. A key improvement in terms of 3D sampling is that the proposed lattice comprises sample planes that are one cell wide (Fig. 4) whereas that used in framed octree is two cells wide ([9, Fig. 4]). This results in fewer nodes in the search space and hence reduced memory and computation time requirements. Additionally, the track angle in a framed octree is constrained to intervals of 45° when transitioning between quadtree nodes. Finally, the proposed method guarantees path soundness by sampling each trajectory segment at the same high resolution cell size thus avoiding the problem of mixed cells when using cell decomposition based methods (such as quad/octree based methods like [9]).

C. Uncertainty

MSA* returns a path comprising a sequence of cells which form a corridor in 4D space around the planned trajectory. This differs from existing vector neighbour based methods like [12] which do not explicitly associate cells or a volume of space with each trajectory segment. Such a cell sequence provides an inherent tolerance to uncertainty. This approach avoids the intractability of directly incorporating uncertainty into the search space (using methods such as Markov Decision Processes) for a large, high dimensional search space [1].

The level of tolerance can be determined by finding the minimum perpendicular distance d between the track and the cell boundaries for each cell on the trajectory segment. This is shown in Fig. 14 where $\overrightarrow{AB}$ is the trajectory segment and

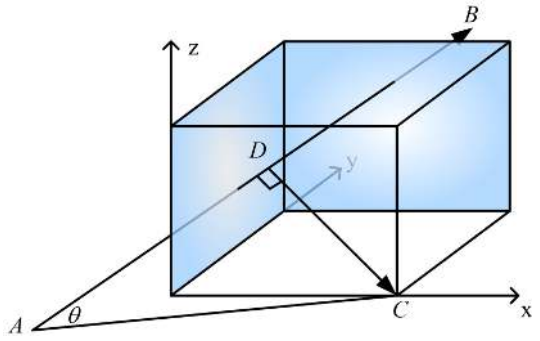

Fig. 14. Illustration of a cell in the cell sequence for a given trajectory segment AB.

$\overrightarrow{DC}$ is the perpendicular distance to an exterior corner of a cell on the sequence $\gamma_{s'}$. An exterior corner is one that is not completely enclosed by adjacent cells. The angle θ can be determined by the dot product of vector AB and AC,

$$\overrightarrow{AB} \cdot \overrightarrow{AC} = \left|\overrightarrow{AB}\right| \left|\overrightarrow{AC}\right| cos\theta \tag{18}$$

In $\triangle ADC$, as $\angle ADC = 90^\circ$ (by definition), then $AD = AC\cos\theta$. The position of D can then be determined using a simple vector line equation and AD

$$D = \vec{A} + \frac{\overrightarrow{AB}}{\left|\overrightarrow{AB}\right|} \left|\overrightarrow{AC}\right| \cos\theta \tag{19}$$

Because the cell sequence generated using (2) only includes cells that intersect the track, if D does not lie within the cell, then that particular corner is ignored. Otherwise, the perpendicular distance DC is

$$\left|\overrightarrow{DC}\right| = \left|\overrightarrow{AC}\right| \sin\theta \tag{20}$$

The value of d is determined by taking the minimum DC value for all corners of all cells in $\gamma_{s'}$. Note that $\theta = 0$ implies that the trajectory intersects a cell edge/corner in which case all adjacent cells were already included in the cell sequence. Hence, this implies a non-exterior cell corner. For the successor operator depicted in Fig. 6 and a 3D cell size of 1NM×1NM×1000ft, the minimum 3D tolerance is 166ft, and the minimum horizontal (x-y) tolerance is 0.14NM. Note that all transition manoeuvres (i.e. turns) needed to transition between tracks are assumed to be of negligible cost compared to the tracks themselves. These manoeuvres are assumed to stay well within the boundaries of the cell sequence.

It is possible to modify the cell sequence returned by the method described in Section III-A to enforce a minimum tolerance constraint (3D or horizontal only). For each exterior corner of each cell that does not satisfy the distance constraint, it is a simple matter to include *all* cells adjacent to that corner to increase the minimum tolerance. This procedure is repeated until all exterior corner points satisfy the minimum tolerance. The time level of each added cell can be determined in the same manner as in (5).

VII. Conclusion

This paper presented MSA*, a method for motion planning using a variable successor operator that finds least cost paths. A variable successor operator enables variable track length, angle and velocity trajectory segments that are modeled using a computer graphics inspired cell sequence. This provides an inherent tolerance to uncertainty based on the minimum distance between the track and cell sequence boundaries.

Additionally, a variable successor operator enables the imposition of a multi-resolution lattice structure on the search space which drastically reduces the number of search nodes and search time. Extensive simulations for a UAV flight planning task reveal that multi-resolution MSA* is approximately four times faster (on average) than vector neighbourhood based A* (Vector A*) but returns paths of approximately the same cost (average path cost ratio of 1.033). Even with a uniform, fine resolution lattice, MSA* is still twice as fast as Vector A* with an average path cost ratio of 0.99. It is shown that MSA* is suited to online replanning with an average computation time (4.46s for multi-resolution MSA*) that is a fraction of the minimum track traversal time (2min.).

Future work primarily revolves around the implementation and real world testing of the proposed algorithm in a closed loop intelligent control system. Such an implementation includes the predictor and scheduling elements discussed in Section VI-A and a study of the stability of the overall system. Additional avenues for study include the use of heuristic inflation (such as with anytime replanning A* [42]) and multi-objective heuristics to further reduce computation time.

Acknowledgment

The authors would like to express their sincere gratitude to Dr Oliver Obst for his insightful feedback on the paper.

References

[1] S. M. LaValle, *Planning Algorithms*. New York: Cambridge University Press, 2006.

[2] C. C. Eriksen, T. J. Osse, R. D. Light, T. Wen, T. W. Lehman, P. L. Sabin, J. W. Ballard, and A. M. Chiodi, "Seaglider: a long-range autonomous underwater vehicle for oceanographic research," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 424–436, 2001.

[3] S. S. Wegener, S. S. Schoenung, J. Totah, D. Sullivan, J. Frank, F. Enomoto, C. Frost, and C. Theodore, "UAV autonomous operations for airborne science missions," in *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, Chicago, Illinois, 2004.

[4] W. B. Powell, *Approximate dynamic programming*. New Jersey: Wiley-Interscience, 2008.

[5] J. S. Mitchell and C. H. Papadimitriou, "Weighted region problem. finding shortest paths through a weighted planar subdivision," *Journal of the Association for Computing Machinery*, vol. 38, no. 1, pp. 18–73, 1991.

[6] P. Hart, N. Nilsson, and B. Rafael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968.

[7] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: the Field D* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, 2006.

[8] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," in *AAAI Conf. on Artificial Intelligence*, 22-26 July 2007.

[9] A. Yahja, S. Singh, and A. Stentz, "An efficient on-line path planner for mobile robots operating in vast environments," *Robotics and Autonomous Systems*, vol. 33, no. 2&3, 2000.

[10] J. Rubio and S. Kragelund, "The trans-pacific crossing: long range adaptive path planning for UAVs through variable wind fields," in *22nd Digital Avionics Systems Conference*, vol. 2, 2003, pp. 8.B.4–81–12 vol.2.

[11] D.-W. Gu, W. Kamal, and I. Postlethwaite, "A UAV waypoint generator," 20-22 Sep 2004.

[12] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2005, pp. 3231–3237.

[13] J. Carsten, D. Ferguson, and A. Stentz, "3D Field D*: Improved path planning and replanning in three dimensions," in *IEEE Int. Conf. Intelligent Robots and Systems*, 2006, pp. 3381–3386.

[14] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Trans. Robotics*, vol. 23, no. 2, pp. 331–341, 2007.

[15] Y. Kim, D.-W. Gu, and I. Postlethwaite, "Real-time path planning with limited information for autonomous unmanned air vehicles," *Automatica*, vol. 44, no. 3, pp. 696–712, 2008.

[16] P. Tompkins, A. T. Stentz, and W. R. L. Whittaker, "Mission-level path planning for rover exploration," in *8th Conf. on Intelligent Autonomous Systems (IAS-8)*, March 2004.

[17] K. Fujimura, "Path planning with multiple objectives," *IEEE Robotics & Automation Magazine*, vol. 3, no. 1, pp. 33–38, 1996.

[18] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi, "An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments," in *Proceedings 21st Digital Avionics Systems Conference*, vol. 2, 2002, pp. 8D2–1–8D2–12 vol.2.

[19] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary algorithm based offline/online path planner for UAV navigation," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 33, no. 6, pp. 898–912, 2003.

[20] M. S. Branicky, S. M. LaValle, K. Olson, and Y. Libo, "Quasi-randomized path planning," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, vol. 2, 2001, pp. 1481–1487 vol.2.

[21] J. Reif and Z. Sun, "Movement planning in the presence of flows," *Algorithmica*, vol. 39, no. 2, pp. 127–153, June 2004.

[22] Z. Sun and J. Reif, "On robotic optimal path planning in polygonal regions with pseudo-euclidean metrics," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 37, no. 4, pp. 925–936, Aug. 2007.

[23] N. A. Papadakis and A. N. Perakis, "Deterministic minimal time vessel routing," *Operations Research*, vol. 38, no. 3, pp. 426–438, 1990.

[24] J. Sellen, "Direction weighted shortest path planning," *IEEE Int. Conf. Robotics and Automation*, vol. 2, pp. 1970–1975 vol.2, May 1995.

[25] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. San Francisco, CA: Morgan Kaufmann, 1995.

[26] RTCA, "Software considerations in airborne systems and equipment certification," 1992.

[27] G. F. List, P. B. Mirchandani, M. A. Turnquist, and K. G. Zografos, "Modeling and analysis for hazardous materials transportation - risk analysis, routing scheduling and facility location," *Transportation Science*, vol. 25, no. 2, pp. 100–114, 1991.

[28] P. B. Mirchandani and H. Soroush, "Optimal paths in probabilistic networks: A case with temporary preferences," *Computers & Operations Research*, vol. 12, no. 4, pp. 365–381, 1985.

[29] P. Leonelli, S. Bonvicini, and G. Spadoni, "Hazardous materials transportation: a risk-analysis-based routing methodology," *Journal of Hazardous Materials*, vol. 71, no. 1-3, pp. 283–300, 2000.

[30] M. Zhang, Y. Ma, and K. Weng, "Location-routing model of hazardous materials distribution system based on risk bottleneck," in *Int. Conf. on Services Systems and Services Management*, vol. 1, 2005, pp. 362–368.

[31] P. Wu, R. Clothier, D. Campbell, and R. Walker, "Fuzzy multi-objective mission flight planning in unmanned aerial systems," in *IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, Honolulu, Hawaii, 2007.

[32] B. S. Stewart and I. Chelsea C. White, "Multiobjective A*," *J. ACM*, vol. 38, no. 4, pp. 775–814, 1991.

[33] K. Erol, J. Hendler, and D. S. Nau, "HTN planning: Complexity and expressivity," in *AAAI National Conf. on Artificial Intelligence*, 1994, p. 1123.

[34] P. Doherty, J. Gustafsson, L. Karlsson, and J. Kvarnstrom, "(TAL) Temporal action logics: Language specification and tutorial," *Electronic Transactions on Artificial Intelligence*, vol. 2, no. 3-4, pp. 273–306, 1998.

[35] A. Mali and Y. Lipen, "MFSAT: a SAT solver using multi-flip local search," in *Proc. IEEE Int. Conf. Tools with Artificial Intelligence*, Nov. 2003, pp. 84–93.

[36] T. Belker, M. Hammel, and J. Hertzberg, "Learning to optimize mobile robot navigation based on HTN plans," vol. 3, Sept. 2003, pp. 4136–4141.
[37] J. Bresenham, "Pixel-processing fundamentals," *IEEE Computer Graphics and Applications*, vol. 16, no. 1, pp. 74–82, 1996.
[38] W. F. Phillips, *Mechanics of Flight*. Hoboken, New Jersey, USA: John Wiley & Sons, 2004.
[39] P. Wu, D. Campbell, and T. Merz, "On-board multi-objective mission planning for unmanned aerial vehicles," in *IEEE Aerospace Conference*, Big Sky, Montana, 7-14 March 2009.
[40] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2003.
[41] Civil Aviation Safety Authority (CASA), "Civil aviation regulations 1988 (CAR 1988)," August 2003.
[42] D. Ferguson, M. Likhachev, and A. T. Stentz, "A guide to heuristic-based path planning," in *Int. Conf. on Automated Planning and Scheduling (ICAPS)*, Monterey, CA, 2005.
[43] Office of the Secretary of Defense, "Unmanned aircraft systems roadmap: 2005-2030," Tech. Rep., 2005.
[44] NOAA National Weather Service, "Structure and dynamics of supercell thunderstorms," 2004, http://www.crh.noaa.gov/lmk/soo/docu/supercell.php.

Dr Paul Wu is a researcher at the Australian Research Centre for Aerospace Automation (ARCAA), Queensland University of Technology (QUT). In 2009, he received his PhD from QUT for his thesis on Multi-Objective Mission Flight Planning in Civil Unmanned Aerial Systems (UAS). Some projects he has worked on include the modelling of risk presented to population centres from overflight of aircraft, and delivery of multimedia content to mobile phones, for which he was awarded the Engineers Australia Queensland Division J H Curtis Award. He is currently undertaking research on algorithms and architectures for UAV path planning and risk analysis.

Dr Duncan Campbell is an Associate Professor at the Queensland University of Technology (QUT), in Brisbane, Australia. He is the Alternate Head of the School of Engineering Systems. Dr Campbell has over 65 internationally peer reviewed papers and researches in the areas of robotics and automation, embedded systems, computational intelligence, intelligent control and decision support. He leads a group in the Australian Research Centre for Aerospace Automation and has collaborations with a number of universities around the world including Massachusetts Institute of Technology, USA and TELECOM-Bretagne, France. Dr Campbell is currently the IEEE Queensland Section chapter chair of the Control Systems / Robotics and Automation Society Joint Chapter.

Dr Torsten Merz is a senior research scientist at CSIRO's Autonomous Systems Laboratory in Brisbane, Australia. His research interests include dependable autonomous systems, unmanned aircraft systems, robot architectures, robot perception, and real-time systems. Previously, he held a position as assistant professor at the Department of Computer and Information Science at Linköping University, Sweden. He was part of the WITAS UAV project team and involved in the development of flight modes and mission planners for unmanned helicopters. He earned a Diploma in Informatics from the University of Bielefeld and received a Doctor of Engineering from the University of Erlangen-Nuremberg, Germany.