One-Shot Strategically Deconflicted Route and Operational Volume Generation for Urban Air Mobility Operations

Ellis L. Thompson[1], Yan Xu[2] and Peng Wei[3]

Abstract— In the UAM space, strategic deconfliction provides an all-essential layer to airspace automation by providing safe, pre-emptive deconfliction or assignment of airspace resources to airspace users pre-flight. Strategic deconfliction approaches provide an elegant solution to pre-flight deconfliction operations. This overall creates safer and more efficient airspace and reduces the workload on controllers. In this research, we propose a method that constructs routes between start and end nodes in airspace, assigns a contract of operational volumes (OVs) and ensures that these OVs are sufficiently deconflicted against static no-fly zones and OVs of other airspace users. Our approach uses the A* optimal cost path algorithm to generate the shortest routes between the origin and destination. We present a method for generating OVs based on the distribution of aircraft positions from simulated flights; volumes are constructed such that this distribution is conservatively described.

I. INTRODUCTION

Safe and efficient flight planning is fundamental in air operations to ensure the safety of operators within the airspace. Additionally, efficient planning effectively uses the available airspace to provide optimal traffic throughput, reducing the burden of work for controllers in an environment with ever-increasing demand. Both the European Aviation Safety Authority and the Federal Aviation Administration predict a surge in Unmanned Aircraft Systems (UASs) operations with established numbers of both UASs and electric vertical takeoff and landing (eVTOL) vehicles by 2025 and 2024 respectively [1], [2].

To cope with this foreseen demand, the aviation sector is moving toward a more autonomous ecosystem. Where current air traffic control techniques rely on pilot-to-controller communication, the Urban Air Mobility (UAM) concepts favour intelligent and autonomous techniques. While tactical deconfliction is an obvious integration point en route, efficient strategic planning can ensure less congested airspace and reduce reliance on tactical methods.

In our previous work [3], we proposed a framework for route generation along with the generation of accompanying Operational Volumes (OVs). The results showed a valid candidate path could be generated through an airspace of both static and dynamic airspace constraints. Following the

[1]Ellis L. Thompson is with the School of Engineering and Applied Science, George Washington University, Washington, DC 20052, USA `thompson_e@gwu.edu`
[2]Yan Xu is with the Centre for Autonomous and Cyberphysical Systems, Cranfield University, Bedford, MK43 0AL, UK `yanxu@cranfield.ac.uk`
[3]Peng Wei is with the School of Engineering and Applied Science, George Washington University, Washington, DC 20052, USA `pwei@gwu.edu`

route generation, contracts of OVs could then be generated, with the addition of a black box simulator to simulate aircraft movements and leveraging the reachability analysis tool: DryVR [4].

In this paper, we build on our previous work by introducing a new OV construction technique and adapting an A* based algorithm for path planning. To ensure this new approach improved on the previous system, we generated the following desideratum:

1) Contracts of OVs, and by extension their accompanying routes, should be deconflicted with both static no-fly zones (NFZs) and OVs from proposed contracts belonging to other airspace agents.

2) There should be an improved synergy between the route generation and OV generation, with the former taking precautionary measures to ensure minimal potential overlap between OVs and, by extension, reduce route reconstruction frequency.

3) The route generation algorithm should prioritise optimal route generation, reducing the distance travelled. The existing Rapidly Exploring Random Trees (RRT*) solution should be reconsidered to do this.

4) The system should provide a candidate route and accompanying contract of OVs in a *one-shot* manner. A *one-shot* approach will aid in reducing the "plan, check, re-plan" reducing the time spent planning a flight pre-departure.

Additionally, we explore alternative approaches to generating the OV bounds. We aimed to improve the tightness of the generated OVs around the route while omitting any potential adverse side effects as a result of the wrapping effect [5], the volume grows exponentially in size becoming non-descriptive and overly conservative. We also refer to the candidate route and contract as being generated in a *one-shot* manner. In this project, we describe the system as *one-shot* when: exactly one candidate route and accompanying contract is generated, which is deconflicted with all NFZs and other contracts, without the need for several iterations. More precisely, the generated route is optimal and deconflicted with all NFZs and other contracts such that the resultant OVs are also deconflicted, so no adjustments are needed.

The structure of the remainder of this paper is as follows: Section II provides an overview of related work. Fundamental concepts and formal definitions are presented in Section III. Section IV covers the approach to route generation and the preliminary route-OV deconfliction. In Section V, the approach used to generate OVs is presented. Experimental

results and comparisons to our previous framework [3] are given in Section VI. Finally, a discussion and concluding remarks are presented in Section VII.

II. RELATED WORK

A. Route Generation

There are several approaches to route generation, for the movement of Unmanned Aerial Vehicles (UAVs). Our previous work examined techniques using RRT* as a conceptual baseline [6]. Our approach builds on the finite node concept imposed by [7] to improve computation time and memory storage space. A *rope pull* approach described in [8] optimises the generated route, removing excess turns or bends through simulating a *rope pull*. In essence, a route generated using RRT* could be optimised by directly connecting points that have a line of sight to another. Like RRT* based approaches Probabilistic Road Map (PRM*) [9], takes random samples from the configure space. Unlike RRT*, PRM* does not find a path but generates a practical graph for navigating through a region; a query phase then uses a path-finding algorithm to search for a valid route.

For RRT* and PRM*, the random nature of the node addition prevents these from having a guaranteed optimal solution. Classical approaches to optimal path planning include: Dijkstra's algorithm [10], A* [11] and the Bellman-Ford algorithm [12], [13], of which all produce minimum-cost paths. For generating optimal paths in a dynamic environment where the obstacles can change, D* and D*-Lite [14], [15] provide optimal and efficient solutions for these partially known environments.

B. Reachability Analysis

Reachability analysis is the computational process of computing reachable states from a given initial state. DryVR [4], is a data-driven approach to verification for automotive systems. Part of this approach is to perform reachability analysis and generate reachtubes using a black-box simulator. The bounded reach ability algorithm uses learned sensitivities from the trajectory data to overestimate the trajectories provided by the simulator.

Other approaches also focus on Lagrangian methods or an analytical solution [16]–[18], the latter of which can perform slower but is more robust than sensitivity and discrepancy approaches, and the former potentially provides a solution not as susceptible to the wrapping effect [5]. Another common approach is that of bloating techniques [19], [20]. The area is artificially increased in size until some parameter is met and exists in many reachability analysis approaches to extend the bounds of the generated volumes.

C. Strategic Planning

Strategic Deconfliction [21] approaches usually focus on pre-flight stratagem intending to either improve airspace utilisation or improve the likelihood of decreased en route conflict states between aircraft agents. Generating routes that have been strategically deconflicted with other proposed routes, as in [22]–[25], can make efficient use of the available airspace, reducing the number of en route conflict states through adapting routes around areas of high concentrated traffic flow.

III. FUNDAMENTALS

A. Contracts and Operational Volumes

An OV is a finite 4-dimensional region of airspace. The OV describes where an aircraft is expected to operate within a limited duration. More formally, we adapt the definition provided in [26]; For the airspace, $\mathcal{X} \subseteq \mathbb{R}^3$, an OV can be described as a tuple, seen in Equation 1, where $R_k \subseteq \mathcal{X}$ and T_k is monotonically increasing time.

$$\mathcal{C} = (R_1, T_1), (R_2, T_2), ..., (R_n, T_n) \qquad (1)$$

Each OV ($\mathcal{C}$) exists for a finite time $\mathcal{C}^{dur} = T_n - T_1$. To obtain the total volume over t_d, we can take the union of the OV over each interval:

$$\mathcal{C}^{vol} = \bigcup_{k=1}^{n} R_k \qquad (2)$$

A contract (Ξ), by extension, is the collection of OVs over a route. The duration of the contract (Ξ^{dur}) is therefore $\sum_{i=1}^{N} C^{dur}$ for all OVs in the contract.

In [3], we extended the definition of an OV to match that in Equation 3, adding an additional element D, which describes the distribution of the aircraft within the OV at time t. The approach to this previously represented the area of the OV as a set of grid squares and assigned a value, or probability, of an aircraft present in the square. This approach obfuscated the shape of the recently generated OV and was needlessly complex. In Section V-B, we provide a detailed description of how we have redefined D_n for this work.

$$\mathcal{C} = (R_1, T_1, D_1), (R_2, T_2, D_2), ...(R_n, T_n, D_n) \qquad (3)$$

B. A* Search Algorithm

The A* search algorithm [11] is a graph traversal and path search algorithm with a worst-case complexity of $O(b^d)$, where b is the branching factor and d is the depth of the desired goal. It is heuristically driven, prioritising exploring nodes that minimise $f(n) = g(n) + h(n)$, where $g(n)$ is the actual distance from the start node (Γ_{start}) to the current node (Γ_n) and $h(n)$ is the heuristic value, commonly the direct distance from Γ_n to the goal node Γ_{goal}.

IV. ROUTE GENERATION

For route generation, we employed the A* algorithm. Common in route planning tasks for its guaranteed optimality, we also have the added benefit of predicting where the aircraft could be when given a departure time and cruise speed. This benefit, especially over our previous approach using RRT*, allows us to perform preliminary route-OV deconfliction based on this predicted position. By performing this preliminary route-OV deconfliction, we can significantly increase the system's overall performance as we can expect fewer inter-OV conflicts when generating them.

A. Point Expansion

The environment in which we apply our A* algorithm is not constrained to a grid. We sacrifice the improved efficiency that comes with a predetermined grid of nodes in favour of an open environment, reducing the overall memory taxation of the search.

We build a graph during the node expansion phase, providing the nodes introduced follow some rules. Firstly, to reduce the number of sharp angled turns, we limit the nodes to only be generated in a finite arc of radius r and angle α such that the centre of the arc extends the line $\Gamma_n \to \Gamma_{n+1}$ as shown in Figure 1. To reduce the search space, inserted nodes within some bounded distance of nodes already explored in the environment are counted as the same node. If this is the case, then the $g(n)$ values are compared, and if the inserted node has a lower $g(n)$ value, then the explored value is updated and re-expanded to ensure optimality in the route.
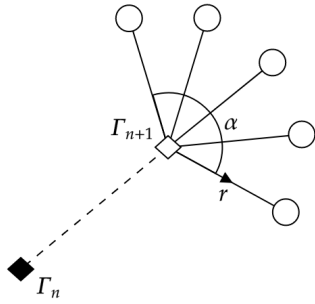

Fig. 1. New nodes are generated at regular intervals along the arc of angle α where the centre of the arc extends the path $\Gamma_n \to \Gamma_{n+1}$.

For a given previously explored point, the heuristic also needs to be calculated once. As outlined in Equation 5, the heuristic function does not include any information about previously visited nodes. With the heuristic function being this approach's most considerable computational burden, the added benefit of only calculating a heuristic once significantly improves computation speed.

During the expansion phase, nodes are categorised as either *valid* or *invalid*. A *valid* node is one which exists within the constraints of the airspace and a valid connection to a parent node exists. A valid connection is similarly defined as a connection that does not violate the constraints of the airspace or intersect with any NFZ. A node with an invalid connection is, therefore, a node with a path to its parent node that violates the constraints of either the airspace or a NFZ. To expand further, an *invalid node* is defined as any node within or too close to the region of a NFZ or outside the operational area bounds. Nodes with invalid connections are handled differently; rather than being excluded from the expansion, they are immediately placed into the *closed* list. This allows us to pre-calculate and store the heuristic, reducing the number of calculations, if the node is ever reached again via a safe route.

B. Heuristic Function

The heuristic function used in the A* algorithm adapts the approach proposed in [27]. This method first calculates the angle from the node (Γ_n) to all vertices on the polygon $\mathcal{Z}_j$ as illustrated in Equation 4. The NFZ polygons observed are primarily concave, providing a challenge for the route-finding algorithm.

$$\Theta(\Gamma_n, \mathcal{Z}_j) = \{\theta(\Gamma_n, \mathcal{Z}_j^i) - \theta(\Gamma_n, \Gamma_{goal}), \forall i \in \mathcal{Z}_j\} \quad (4)$$

This will yield a positive value if the point lies to the right of the direct line $\Gamma_n \to \Gamma_{goal}$ and a negative value otherwise. The extremes are then found from this set, that is, the point furthest left and right of the line $\Gamma_n \to \Gamma_{goal}$, this can be obtained by finding the minimum and maximum values in $\Theta(\Gamma_n, \mathcal{Z}_j)$. The heuristic, presented in Equation 5 and demonstrated in Figure 2, is then the distance between Γ_n and the extreme vertex $\mathcal{Z}_j^i$, whose absolute angle is the smallest from the line $\Gamma_n \to \Gamma_{goal}$ and the distance from $\mathcal{Z}_j^i$ to Γ_{goal}. We additionally multiply this by ω where $1.1 \leq \omega \leq 1.5$; which improves the performance of the A* algorithm in favouring exploring points closer to the goal.

$$h() = \omega(d(\Gamma_n, \mathcal{Z}_j^i) + d(\mathcal{Z}_j^i, \Gamma_{goal})) \quad (5)$$
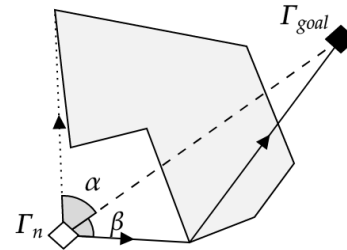

Fig. 2. The heuristic for a point whose line of sight is obscured by a NFZ is the distance to the extreme point on that polygon with the smallest angle and then the distance to the goal. This is represented by the path β, and the line α is not used as the distance $\theta_\alpha > \theta_\beta$.

C. Route-OV Deconfliction

We can predict the time an aircraft reaches a node (Γ_n) through the distance travelled from Γ_{start} and providing the cruise speed V_{cruise}, that is, the expected speed of the aircraft. In this approach, we assume that the route is only generated for the cruise phase of the flight, so any departure or arrival altitude and speed profiles are not considered. To further perform route-OV deconfliction, the system must store a representation of the airspace at discreet time intervals.

The airspace is divided into time bins of duration t_d for a given operational area. OVs are then placed into these brackets based on the start time and perceived end time of the OV. This means that an OV can exist in multiple time bins if either the t_d of the bin is shorter than that of the OV or the OV lies near the boundary of a time bin. This airspace

binning reduces the search scope to only OVs that lie near the node temporally.

Suppose a node lies within the bounds, both spatially and temporally, as an OV or the path between nodes intersects an OV, again both spatially and temporally, the node is treated as having an invalid connection. The deconfliction then occurs by observing a pair of two points, (Γ_n, Γ_{n-1}) that is the node to add and its parent, and the connecting line between them. The respective Estimated Time of Arrivals (ETAs) of both nodes are then compared with the time bins in the airspace, and NFZs in bins that overlap the ETAs are obtained. The process remains the same as deconflicting with NFZs. As is recalled from handling nodes with invalid connections for NFZs, the node is still processed but added directly to the *closed* list of the A* algorithm.

This approach prevents nodes, or routes between nodes, from directly conflicting with OVs already proposed in the airspace. We can then further decrease the possibility of an OV generated along the route conflicting with the existing OVs by imposing a distance constraint on the path between nodes (Γ_n, Γ_{n-1}) and any temporally aligned OVs.

We can further add to the deconfliction approach by considering the distribution of aircraft with an OV. As seen in the next section, we can represent the aircraft's position within an OV as a density map at any given time. We expect, within an OV, that aircraft will travel closer to the centre with fewer outliers towards the edges of the OV. Considering this distribution, the probability of observing an aircraft along a proposed connection can be calculated when an overlap is observed. This can be achieved by evaluating the probability density function for the points along the connection (Γ_{n-1}, Γ_n) of the multivariate normal distribution that can be obtained from the OV. We can then add this to the heuristic function in Equation 6 where $\beta \geq 0$ is some constant and $\delta(\Gamma_{n-1}, \Gamma_n)$ is the maximum probability of conflict expanded in Equation 7.

$$h() = \omega(d(\Gamma_n, \mathcal{Z}_j^i) + d(\mathcal{Z}_j^i, \Gamma_{goal})) + \beta\delta(\Gamma_{n-1}, \Gamma_n) \quad (6)$$

$$\delta(\Gamma_{n-1}, \Gamma_n) = \max_{\xi \in \Xi} \max_{\mathcal{C} \in \xi} P((\Gamma_{n-1}, \Gamma_n), \mathcal{C}) \quad (7)$$

V. GENERATING OPERATIONAL VOLUMES

In our previous approach, OV generation was performed using a combination of a simulator, namely BlueSky [28], and the reachability analysis tool DryVR [4]. The DryVR framework performs reachability analysis by learning a global discrepancy function based on the data provided by the simulation. The resultant reachtubes are 3-dimensional bounds between regular, discreet time intervals which we adopted to build the OVs influenced by the approach outlined in [26]. However, this approach is sensitive to the data distribution and suffers from the *wrapping effect* [5]. We propose a new approach, describing the data distribution as ellipsoids for discrete time intervals. The ellipsoids are based on the distribution of the recorded aircraft.

A. Utilising the Simulation

As with the approach referenced in [4], [26], our approach uses simulation data to build the bounding spheroids. To generate this data BlueSky [28] is used again for its variety of aircraft models along with its Python implementation, making fine control of the simulation and data extraction trivial and well-integrated with our architecture.

Once a route has been generated using the method from Section IV, the set of waypoints ($\mathcal{W}$) are generated for BlueSky. Some initial uncertainty is also injected into the system, and N aircraft are initialised in a radius around the initial waypoint w_0. The simulation then proceeds for the duration t_d, recording the state of the aircraft at regular intervals. The state of the airspace at any given time is denoted as S_t^N; therefore, the state of an individual aircraft at t is defined as $S_t^n, n \in N$. The state information stored, presented in Equation 8, includes the positional data, altitude, active waypoint ($w \in \mathcal{W}$) and the airspeed of the vehicle (V).

$$S_t^N = (lat_t^N, lon_t^N, alt_t^N, w_t^N, V_t^N) \quad (8)$$

Once the simulation has run for the duration t_d, new uncertainty is injected into the system. This is achieved by generating new aircraft proportionally relative to active waypoints and the number of aircraft approaching them. This approach naturally accounts for the spread of the aircraft in the simulation occurring from uncertainty in the initial position and external effects such as wind. To demonstrate this approach, if there are recorded three active waypoints out of the $N = 100$ aircraft in the environment $(w_\alpha, w_\beta, w_\gamma) \in \mathcal{W}$, with the distributions as $(25, 71, 4)$, then simulation run to t_d will start with new aircraft approaching the aforementioned waypoints following the distribution ($w_\alpha = 25, w_\beta = 71, w_\gamma = 4$) respectively.

B. Building OVs from Simulation Data

After each simulation has been completed to t_d, in parallel to the new aircraft generation, the OV for the previously completed run is generated. The data passed to the OV generator is in the form of S_t^N, firstly a sample of this data is taken $S_t^M \subset S_t^N$ such that $|S_t^N - S_t^M| \leq |S_t^M|$ and $|S_t^M| \geq 15$. This separates the state space into two subsets S_t^M is used to build the OV, and the remainder is later used to validate the OV includes sufficient points. The samples are drawn at random.

In this approach, we want to introduce a method that can both describe the area that the OV will occupy while simultaneously describing the distribution of these aircraft. We construct a covariance matrix (Σ_t) from the state S_t^M, specifically over the latitude, longitude and altitude components and obtain the mean position μ_t. Next, the eigenvectors v_i and eigenvalues λ_i are obtained.

The eigenvalues (λ_i) are used to derive the width and height of the resulting ellipse using Equation 9 where l_w and l_h are the lengths of the major and semi-major axes, respectively. The value z is the confidence interval for a normal distribution, described in Equation 10 where x is the desired minimum inclusion, PPF is the per cent point

function for the normal distribution and α is a constant to bloat the ellipse artificially.

$$l_{w,h} = 2z\sqrt{\lambda_{w,h}} \quad (9)$$

$$z = \left| PPF\left(\frac{1-x}{2}\right) \right| + \alpha \quad (10)$$

Finally, the ellipse's rotation angle is calculated from the eigenvectors v_i. It should be noted that for a 2×2 covariance matrix, two eigenvectors are obtained, each containing two values where v_1 and v_2 match with the eigenvalues λ_1 and λ_2 respectively. To obtain the rotation, we use the 2-argument arctangent: $\theta = \text{atan2}(v_{1,0}, v_{0,0})$.

This generates a single ellipse R_i for T_i as described in our OV definition in Section III. We use the resultant set $S = S_t^N - S_t^M$ and our desired minimum inclusion variable x to verify that the ellipse is sufficiently encompassing. To verify, we calculate the Mahalanobis Distance [29], Equation 11, first across S_t^M and then set the xth percentile distance as the threshold distance.

$$M_{Dist}(y) = \sqrt{(y - \mu_t)\Sigma_t^{-1}(y - \mu_t)'} \quad (11)$$

Then the Mahalanobis Distance is calculated for all values in S and it is asserted that:

$$\left(\frac{1}{|S|} \sum_{i=1}^{|S|} [M_{Dist}(S_i) <= Threshold] \right) >= x$$

If it is found that the number of points included falls below the threshold x the bloating factor α can be increased until the threshold is met. This is done in line with the generation of the ellipse.

As these ellipses are generated, they can be added to the OV, such that D_n consists of the covariance matrix Σ and the value z, T_n is set as the start time of the interval, and R_n while not stored can be derived as shown in this section. Repeating this process over each time interval forms a single OV of length $\mathcal{C}^{dur}$.

VI. EXPERIMENTAL RESULTS

A mock airspace environment was created to test the system, as shown in Figure 3. The airspace consists of ten arrival and departure nodes resulting in 45 unique routes. Additionally, to stress the route-finding algorithm and heuristic, the NFZs presented are concave, proposing a challenging environment for path finding algorithms.

The experiments are compared against our previous approach described in [3]. We first explore the route generation capabilities, examining route construction time and the resulting lengths of the routes generated. The OV generation is then examined. We compare the OV generation times collectively of both approaches as well as the overall volume of the generated OVs and their relative accuracy. These experiments are done in an environment with no other traffic, so deconfliction is not evaluated in these initial experiments. The final experiment stresses the proposed approach in an environment with other existing contracts, evaluating the
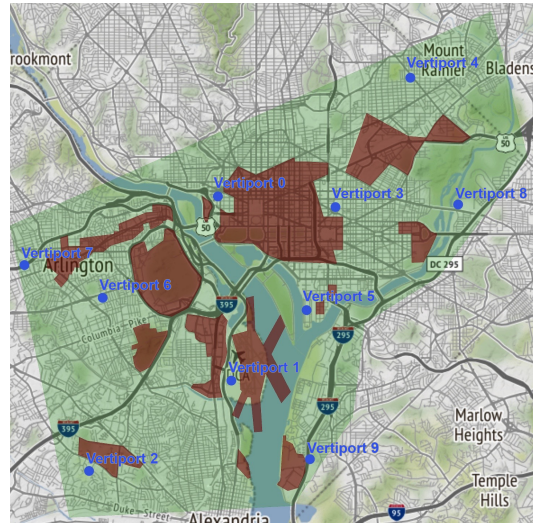

Fig. 3. The mock airspace used for testing the systems. It consists of 10 arrival/departure nodes (yellow pins) and concave NFZs (red polygons), designed to challenge the route-finding algorithm. The green lines denote the operational bounds of the airspace.

ability to generate and deconflict routes and the accompanying contract of OVs. All experiments are run using the provided Amazon drone in the BlueSky environment as part of the OpenAP [30] dataset. All experiments were run on a 2020 MacBook pro using an Intel 2GHz i5 processor and 32GB of memory.

A. Route Generation

For this experiment, both the RRT*-based route generation method from [3] and the A* method proposed in this paper are tasked with finding routes within the environment shown in Figure 3. We initially hypothesise that the A* method, being a cost-optimal algorithm, would be able to find shorter paths than the RRT*-based solution.
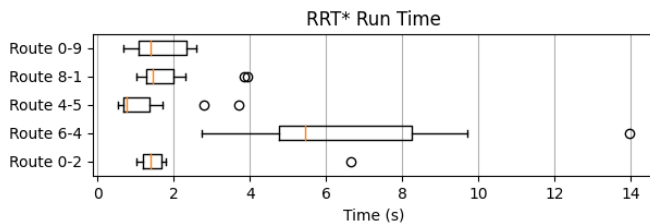
The results in Table I demonstrate that, although slower, A* produces significantly shorter routes. Due to the stochastic nature of the RRT*-based method ten experiments were run for each route and the averages used for comparison. Figures 4a and 4b, show the distribution of both the run times and distances of the RRT* route planning approach. Not that the run times remain relatively consistent likewise with the route distances over shorter routes. Route 6-4 has the largest variance in runtime. This is likely due to the origin vertiport being between two concave NFZs. The stochasticity of the RRT*s method can be a limiting factor for that method of route generation as the approach adopted returns the first found route without further exploration.

It is worth nothing that, as can be observed in Figure 7, on occasion routes may not always be optimal. Due to the nature of the A* algorithm relying on predetermined nodes, if the initial route is blocked, the resulting route generated may significantly divert from this optimal path. This performance can be improved by decreasing the distance between nodes, as well as the number of angles as visualised in Figure 1. This comes at the impedance of computation time with the
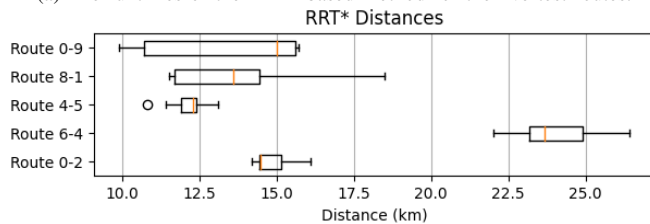
resulting routes taking longer to generate.

Route	Direct Distance (km)	Route Dist (km)		Generation Time (s)	
		A*	RRT	A*	RRT
0-2	10.01	**12.39**	14.83	10.04	**1.86**
6-4	12.48	**13.04**	23.98	7.08	**6.74**
4-5	8.39	**9.06**	12.11	3.89	**1.28**
8-1	9.48	**10.79**	13.73	13.08	**1.90**
0-9	9.19	**9.88**	13.43	3.07	**1.61**

Route	Time (s)		No. OVs		Avg Area (km^2)	
	Ours	DryVR	Ours	DryVR	Ours	DryVR
0-2	**88.01**	424.51	**15**	27	0.045	**0.032**
6-4	**86.04**	498.43	**16**	28	0.045	**0.028**
4-5	**49.19**	286.61	**11**	20	**0.043**	0.055
8-1	**67.53**	320.56	**13**	23	**0.044**	0.057
0-9	**50.62**	302.76	**12**	21	0.044	**0.037**

(a) The runtimes of the RRT*-based method for the five test routes.

(b) The distances of theRRT*-based method for the five test routes.

Fig. 4.

B. OV Generation

In this experiment, we compare the ellipse based OV generation presented in this paper against the reachability analysis method DryVR [4], used in our previous work [3]. The routes used were pregenerated using the A* route generation method. These were then supplied to both OV generation methods.

Both approaches utilise the BlueSky [28] air traffic simulator to generate the data used in OV generation. To ensure consistency, the initial starting points of each scenario were identical for each pair of experiments. Additionally, the speeds of the aircraft were uniformly distributed between $13m/s$ and $17m/s$. For the DryVR method, the overlap offset was set to 20 seconds. The results are presented in Table II.

From Table II, we see that the approach presented in this paper is significantly quicker at generating OVs than the previously adopted method. This is due to the relatively straightforward and more optimised approach adopted in this paper over that from DryVR. In this new approach, the calculations can be highly parallelised as the result of one OV does not rely on the parameters of the preceding OV, only aircraft positional data.

The method for generating reachtubes in DryVR does not account for uncertainty in the initial position. Furthermore, our approach produces significantly fewer OVs. This is caused by no offset variable being introduced in this approach. This, in turn, yields strict boundaries between OVs, as seen in [26], which the offset value introduced in [3], attempted to address. In this new approach, however, uncertainty in the initial position is included through the method outlined in Section V-A. This naturally generates overlapping OVs.

The final observation is that the average volume of the OVs generated using the DryVR approach appears to be smaller. What is not conveyed, however, is how this approach handled initial uncertainty or turns. Figure 5, demonstrates the difference in OV generation methods, where the ellipsoid OVs are outlined in white. The green and purple OVs towards the top of the figure is over the initial start point of the route. It can be seen from these two OVs that the DryVR method struggles to handle this initial uncertainty, resulting in an OV with more conservative bounds when compared to the ellipse-based approach.

We tested the accuracy by simulating 100 aircraft with added uncertainties to the latitude, longitude and speeds. The aircraft's position was recorded at 1-second intervals and then compared against the generated OVs, counting the total points within a valid OV for the given time. Both approaches generated contracts with 100% accuracy across all five routes.

C. Congested Environment

The system was tested in a high-traffic density environment for the final set of experiments. Flights were progressively created at random over a 5-minute interval. The resulting contracts and departure times were stored and used in the route generation and deconfliction of subsequent flights. This resulted in the environment progressively increasing in density. Additionally, the cruise speeds of the contracts were assigned at random between $13m/s$ and $20m/s$, and to ensure the highest level of complexity, no conflicts between OVs of other contracts were allowed. Finally, this experiment was only run with the method proposed in this paper.

This experiment aimed to reach at least 30 contracts in the environment. The system was allowed to generate random start and end node pairs with randomly assigned departure times within the interval. If no route was found, the system would look for the closest departure time, in 30-second
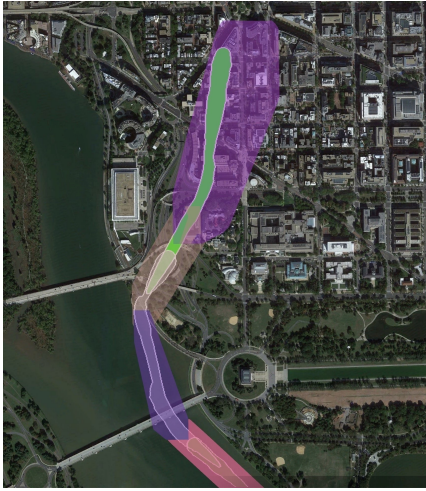
Fig. 5. The difference between the OVs generated using DryVR [4] and our method can be seen by comparing the shaded regions. The outlined regions are generated using our ellipsoid method and are demonstrated to be tighter laterally than the DryVR approach. The colours are only to aid visibility.

intervals, within the 5-minute departure slot. If no route was found, the system would look at a new pair and departure time. This correlates to one departure every 10 seconds.

The departure times were chosen randomly from a Poisson distribution, with the departure and arrival vertiports chosen from a uniform distribution. A second test was run in the BlueSky simulator, which then evaluated conflicts between aircraft following the assigned routes at the previously mentioned departure intervals.

The experiment was run until 31 contracts existed in the environment. Figure 6 shows the departure times over the 5-minute departure window. We observed that more departures were present during the first 2 minutes. This created a denser airspace in those first 2 minutes, resulting in higher computational times, with the average run time being around 3 minutes for both route and OV generation.
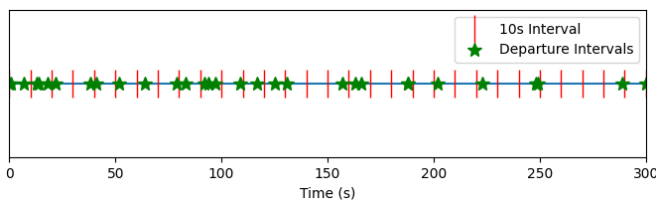

Fig. 6. The departure intervals of routes of the 31 contracts. While distributed along the 5-minute window, we observe more departures in the first 2 minutes.

When running simulations in BlueSky to verify the number of aircraft existing in confliction contracts OVs, we found that all aircraft remained deconflicted with other OVs. The further through the experiment, the more challenging the environment became for the system. However, the presented environment provides large open areas with ample space for routing. Overall, the system successfully routed and deconflicted 31 routes within a finite 5-minute departure

window. Figure 7 shows opposing routes with identical departure times. As the black route was generated later, it takes an alternative shortest route to avoid conflicts with the red OVs.
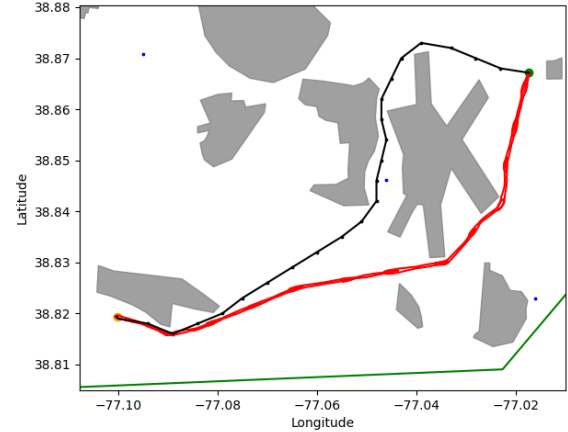

Fig. 7. The OVs (red) are the first to be generated. This takes the shortest route between the departure and arrival points. The route in black is a deconflicted second route with the opposite departure and arrival points but the same departure time. It has to take an alternative route to remain deconflicted with this existing route and others in the environment (which are not drawn here).

VII. Conclusion

The research presented in this paper builds on our previous work in [3]. To build on the previous research, we identified four critical areas of improvement we needed to address to ensure a robust system.

The first point addressed the need to build contracts where the OVs were deconflicted against static NFZs along with OVs from other contracts. This was addressed primarily in the route generation stage, where connections are deconflicted from the existing airspace obstacles when connecting nodes in the graph. We found the system could generate routes and fully deconflicting contracts of OVs in finite airspace. While the system slowed down when more contracts were included, the average full computation time for each route and contract was around 3 minutes. An improvement could arise from a better definition of an airspace manager to dynamically return time-aligned OVs to the route builder.

The second point addressed the need for improved synergy between OV and route generation. As already identified, the primary deconfliction effort occurs at the route generation stage. From Equation 6, new connections are weighted based on the maximum probability of conflict. This is obtained for any given OV from the covariance matrix and the centre point. This allows fine control over how much to consider overlaps, potentially expanding δ further to implement thresholds or minimum separations.

Our previous implementation switched the route generation method to an A* based approach. This satisfies the third requirement and is reinforced by the results in Table I, where our A* approach generates shorter routes than the previous

RRT*-based approach. We have also created a heuristic function that efficiently navigates the concave obstacles in the airspace. However, this approach is slower than the RRT*-based methods and careful parameter optimisation and adjusting the heuristic function could result in faster computational times.

The final point addressed the possibility of designing a one-shot system. We proposed that this approach should only require one iteration for route generation. Without including a re-scheduler, a system that finds a starting time for the flight, the system can generate a route and OVs that are deconflicted in a one-shot manner. With a rudimentary scheduler, the system can still run self-contained and successfully generate a route and deconflicted OVs, providing one exists. However, if a route is not found initially, then a route will have to be regenerated with a new provided departure time.

An ideal system would fall between the abilities of A* to produce cost-optimal routes along with runtime efficiency of RRT*. One middle ground could be derived from an Fast Marching Tree (FMT*) based approach [31]. Producing probabilistically near-optimal solutions it has been shown to outperform its contemporaries; RRT* and PRM*. Furthermore, it has been shown a viable option for emergency route planning of commercial aircraft in [32], lending promise to the viability of a FMT*-based approach being useful for the planning of UAS operations. Other classic approaches, such as storing the best common routes, could also significantly improve the computation time of the system at the cost of increased memory usage.

In conclusion, we have proposed a one-shot system for generating routes and deconflicted OVs for small UAS vehicles to operate in. We demonstrate that this system, in a random and congested airspace, successfully generates and deconflicts multiple contracts and routes with departure times within a 5-minute interval.

REFERENCES

[1] EASA, "Study on the societal acceptance of urban air mobility in europe," https://www.easa.europa.eu/en/downloads/127760/en, 05 2021, (accessed Oct. 07, 2022).

[2] FAA, "FAA aerospace forecast fiscal years 2020-2040," https://shorturl.at/uFIU8, 2020, (accessed Oct. 24, 2022).

[3] E. L. Thompson, Y. Xu, and P. Wei, "A framework for operational volume generation for uam strategic deconfliction," https://arxiv.org/abs/2301.12961, 2023.

[4] C. Fan, B. Qi, S. Mitra, and M. Viswanathan, "DryVR: Data-driven verification and compositional reasoning for automotive systems," in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds. Cham: Springer International Publishing, 2017, pp. 441–461.

[5] A. Neumaier, *The Wrapping Effect, Ellipsoid Arithmetic, Stability and Confidence Regions*. Vienna: Springer Vienna, 1993, pp. 175–190.

[6] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Department, Iowa State University (TR 98–11), Tech. Rep., 1998.

[7] B. Tong, Q. Liu, and C. Dai, "A RRT FN based path replanning algorithm," in *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 1, 2019, pp. 1435–1445.

[8] L. Petit and A. L. Desbiens, "RRT-Rope: A deterministic shortening approach for fast near-optimal path planning in large-scale uncluttered 3D environments," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021, pp. 1111–1118.

[9] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[11] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[12] L. R. Ford, *Network Flow Theory*. Santa Monica, CA: RAND Corporation, 1956.

[13] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 2023/05/16/ 1958, full publication date: APRIL, 1958.

[14] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, pp. 3310–3317 vol.4.

[15] S. Koenig and M. Likhachev, "D* Lite," *AAAI Conference of Artificial Intelligence (AAAI)*, pp. 476–483, 2002.

[16] J. Cyranka, M. A. Islam, S. A. Smolka, S. Gao, and R. Grosu, "Tight continuous-time reachtubes for lagrangian reachability," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6854–6861.

[17] S. Gruenbacher, J. Cyranka, M. Lechner, M. A. Islam, S. A. Smolka, and R. Grosu, "Lagrangian reachtubes: The next generation," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, dec 2020.

[18] S. Gruenbacher, M. Lechner, R. Hasani, D. Rus, T. A. Henzinger, S. Smolka, and R. Grosu, "GoTube: Scalable stochastic verification of continuous-depth models," https://arxiv.org/abs/2107.08467, 2021.

[19] J. Maidens and M. Arcak, "Reachability analysis of nonlinear systems using matrix measures," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 265–270, 2015.

[20] C. Fan, J. Kapinski, X. Jin, and S. Mitra, "Simulation-driven reachability using matrix measures," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 1, 2017.

[21] D. Sacharny and T. Henderson, *Strategic Deconfliction*. Springer International Publishing, 2022, pp. 35–56.

[22] J. R. Bertram, P. Wei, and J. Zambreno, "Scalable fastmdp for pre-departure airspace reservation and strategic de-conflict," 2020.

[23] S. Chaimatanan, D. Delahaye, and M. Mongeau, "Strategic deconfliction of aircraft trajectories," in *ISIATM 2013, 2nd International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Toulouse, France, Jul. 2013.

[24] X. Tang, Y. Zhang, P. Chen, B. Li, and S. Han, "Strategic deconfliction of 4D trajectory and perturbation analysis for air traffic control and automation system," *Discrete Dynamics in Nature and Society*, vol. 2016, Sep 2016.

[25] J. Berling, A. Lau, and V. Gollnick, "European air traffic flow management with strategic deconfliction," in *Operations Research Proceedings 2015*, K. F. Dörner, I. Ljubic, G. Pflug, and G. Tragler, Eds. Cham: Springer International Publishing, 2017, pp. 279–286.

[26] C. Hsieh, H. Sibai, H. Taylor, Y. Ni, and S. Mitra, "Skytrakx: A toolkit for simulation and verification of unmanned air-traffic management systems," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 372–379.

[27] S. Razzaq, C. Xydeas, M. E. Everett, A. Mahmood, and T. Alquthami, "Three-dimensional UAV routing with deconfliction," *IEEE Access*, vol. 6, pp. 21 536–21 551, 2018.

[28] J. Hoekstra and J. Ellerbroek, "BlueSky ATC Simulator Project: an open data and open source approach," in *Proceedings of the seventh International Conference for Research on Air Transport (ICRAT)*, 2016.

[29] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[30] J. Sun, J. M. Hoekstra, and J. Ellerbroek, "Openap: An open-source aircraft performance model for air transportation studies and simulations," *Aerospace*, vol. 7, no. 8, p. 104, 2020.

[31] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions," 2015.

[32] A. Guitart, D. Delahaye, and E. Feron, "An accelerated dual fast marching tree applied to emergency geometric trajectory generation," *Aerospace*, vol. 9, no. 4, 2022.