Data-Driven Multi-UAV Navigation in Large-Scale Dynamic Environments Under Wind Disturbances

Baoqian Wang*, Junfei Xie†, Jun Chen‡

In the near future, large amount of unmanned aerial vehicles (UAVs) are expected to appear in the airspace. To ensure the safety of the airspace, there are many daunting technical problems to tackle, one of which is how to navigate multiple UAVs safely and efficiently in the large-scale airspace with both static and dynamic obstacles under wind disturbances. This paper solves this problem by developing a novel data-driven multi-UAV navigation framework that combines A^* algorithm with a state-of-the-art deep reinforcement learning (DRL) method. The A^* algorithm generates a sequence of waypoints for each UAV and the DRL ensures that the UAV can reach each waypoint in order while satisfying all dynamic constraints and safety requirements. Furthermore, our framework significantly expedites the online planning procedure by offloading most computations to offline and limiting online computing to only path fine-tuning and dynamic obstacle avoidance. The simulation studies show the good performance of the proposed framework.

I. Introduction

Unmanned aerial vehicles (UAVs) have been widely used in many civilian and commercial applications such as infrastructure inspection,[1] agriculture spraying,[2] fire detection[3] and package delivery.[4,5] It is predicted by the Federal Aviation Administration that millions of UAVs will appear in the U.S. airspace by 2040.[6] To ensure the safety of the airspace, how to manage large amount of UAV traffic efficiently is a critical problem to address. Although past experiences on managing the air traffic are valuable resources to learn from, they cannot be directly applied for UAV traffic management (UTM), considering the numerous differences between manned civilian aircraft and small UAVs, such as the high variety and uncertainty in UAV trajectories, heterogeneity of UAV types, and sensitivity of UAV dynamics to environmental disturbances. To address these challenges, the National Aeronautics and Space Administration (NASA) has devoted great efforts on developing a research platform for the UTM system to determine how UAVs can access the low-altitude airspace safely, efficiently and fairly.[7,8] Despite the significant progress that has been made,[9,10] there are still many unsolved research challenges. In this paper, we aim to address one of the research challenges, that is how to navigate multiple UAVs in a complicated large-scale environment with both static and dynamic obstacles under wind disturbances. The difficulty of this problem lies in the complexity and large scale of the environment, which makes real-time decision-making hard to achieve.

UAV navigation plays an important role in many UAV applications, which aims at driving the UAV to reach a target while avoiding obstacles and minimizing certain costs such as flight time, travel distance and control efforts. To achieve UAV navigation, many existing approaches adopt a two-step procedure.[11–14] In particular, the first step applies a path planning algorithm to find a feasible path. Typical path planning algorithms include: 1) *search-based algorithms* such as the Dijkstra algorithm,[15] A* algorithm[16] and D* algorithm,[17] 2) *sampling-based algorithms* like the Rapidly-exploring Random Trees (RRT),[18,19] Probabilistic Roadmaps (PRM)[20] and Expansive Space Trees (EST),[21] which are usually more time-efficient but generate less optimal paths than the search-based algorithms, and 3) *numerical optimization methods* such as integer linear programming,[22] particle swarm optimization[23] and ant colony optimization.[24] The second step applies

*Ph.D. student, Department of Electrical and Computer Engineering, University of California-San Diego, San Diego State University. Email: bwang4848@sdsu.edu.

†Assistant professor, Department of Electrical and Computer Engineering, San Diego State University, AIAA member. Email: jxie4@sdsu.edu. Corresponding Author.

‡Assistant professor, Department of Aerospace Engineering, San Diego State University, AIAA member. Email:jun.chen@sdsu.edu

a trajectory generation algorithm such as Bezier curves[12] and spline interpolation[25] to make the path flyable by smoothing the path and imposing UAV dynamic and kinematic constraints.

To simplify the process, some studies explored optimal control techniques,[26,27] which generate a sequence of control signals that drive the UAV to navigate in the environment. However, these approaches require prior knowledge of system dynamics and the environment. In scenarios where the system dynamic or the environment is unknown, reinforcement learning[28] can be used to learn the optimal actions to take during navigation. However, solving the optimal control problem or training a reinforcement learning control strategy is time-consuming, which is not suitable for complicated environments that are large in scale and/or change over time.[29] Moreover, in a large-scale environment, the reward is sparse making the training of reinforcement learning suffer from local optimality easily.[29] To address this challenge, authors in[29] develop a hierarchical method that combines PRM for path planning with reinforcement learning for motion planning. In particular, the PRM is first used to generate a sequence of waypoints, and the reinforcement learning is then applied to ensure these waypoints can be followed while satisfying the dynamic constraints and collision-free requirements. This method, nevertheless, does not consider dynamic obstacles or the impact of wind disturbances, and is designed for a single robot.

In this paper, we introduce a data-driven multi-UAV navigation framework to enable safe and efficient UAV operations in a large-scale dynamic environment under wind disturbances. To enable long-range UAV navigation, we combine the A^* algorithm[30] with deep reinforcement learning (DRL),[28] where the A^* algorithm provides roadmaps that will enable the UAVs to reach their target positions and the DRL learns point-to-point navigation policies to ensure the UAVs can navigate safely from one point to another without colliding with any dynamic obstacles. Furthermore, to enable real-time decision-making in a large-scale environment, we move most of the computations to offline through leveraging historical solutions for similar scenarios and limit online computing to only path fine-tuning and dynamic obstacle avoidance. This framework is promising in that it significantly relieves the burden for online computing by offloading the path planning overhead to offline while taking the wind disturbances into consideration. It also leverages DRL to achieve collision avoidance with dynamic obstacles. The performance of this framework is demonstrated through comprehensive simulation studies. Of note, a similar idea was presented in paper,[31] but it does not consider dynamic obstacles and adopts an optimal control based technique, which can be computationally expensive for large-scale scenarios.

In the rest of the paper, we first describe system models and formulate the problem to be solved in Section II. Section III then describes the proposed data-driven multi-UAV navigation framework. Simulation results are presented in Section IV. Finally, Section V concludes the paper.

II. Preliminaries

In this section, we first describe system models including the wind influence model and the UAV dynamic model. We then provide the mathematical formulation for the multi-UAV navigation problem to be solved in this study.

II.A. Wind Influence Model

As wind disturbances have a great impact on UAV dynamics, ignoring their impacts can cause safety issues. To model wind dynamics, we adopt the wind influence model presented in,[32] which captures the spatiotemporal wind spread dynamics. In particular, consider a 2-dimensional (2-D) space, denoted by $\mathcal{W} \in \mathbb{R}^2$. The wind field in the environment space $\mathcal{W}$ is discretized into L regions represented by $\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_L\}$. The wind in each region l at time step t is described by the wind speed $w_l[t]$, which takes a discrete value from the range $[w_{min}, w_{max}]$ with resolution Δ_w. Therefore, the possible wind statuses are $\{w_{min}, w_{min} + \Delta_w, \ldots, w_{min} + (K-1)\Delta_w\}$, where $K = \lceil \frac{w_{max} - w_{min}}{\Delta_w} \rceil$ is the total number of wind statuses. To describe the wind state of each region l at time t, a one-hot state vector $\mathbf{s}_l[t] \in \mathbb{R}^{K\times 1}$ is used in the influence model, whose i-th element is 1 if at the i-th wind status, where $i \in [K] := \{1, 2, \ldots, K\}$. The whole wind field's state at time step t can then be described by $\mathbf{S}[t] = [\mathbf{s}_1^T[t], \mathbf{s}_2^T[t], \ldots, \mathbf{s}_L^T[t]]^T$.

The evolution of each region's wind state is affected by the region's previous state and its neighbors through two parameters. First, a local transition matrix $\mathbf{A}_{l,\tilde{l}} \in \mathbb{R}^{K\times K}$ is used to describe the impact of region l's previous state on the current state of region $\tilde{l}$. In particular, each (i,j)-th entry of $\mathbf{A}_{l,\tilde{l}}$ is the probability of transiting to state j in region $\tilde{l}$ given the previous state i in region l. Second, a scalar

$c_{l,\tilde{l}} \in [0,1]$ is used to capture how frequently region l affects region $\tilde{l}$. Given the state of each region at the previous time step $t-1$, the probability mass function of each region's state at the current time step t can then be determined by

$$\sum_{\tilde{l}=1}^{L} c_{l,\tilde{l}} \mathbf{s}_{\tilde{l}}[t-1] \mathbf{A}_{l,\tilde{l}}, \tag{1}$$

which in turn is used to generate the wind state $\mathbf{s}_l[t]$ of each region l. In this study, we assume that wind dynamics are time-invariant over the planning horizon. Therefore, each snapshot of the wind field $\mathbf{S}[t]$ is considered as one wind scenario. Without loss of information, we use $\mathbf{S}$ to denote a wind scenario in the following sections,

II.B. UAV Dynamic Model

We apply a point-mass model to describe the dynamics of a UAV.[26] Here we assume all UAVs fly at the same altitude. The state of a UAV $\mathbf{p}[t] = [x[t], y[t]]^T \in \mathcal{W}$ at time t captures the position of the UAV. The control input $\mathbf{u}[t]$ is specified by $\mathbf{u}[t] = [v[t], \theta[t]] \in \mathcal{U}$, where $v[t] \in [0, v_{max}]$ and $\theta[t]$ are the speed and heading angle at time t, respectively, v_{max} is the maximum speed, and $\mathcal{U}$ is the control input space. The dynamics of the UAV in a windless environment are then described by:

$$\dot{\mathbf{p}}[t] = f(\mathbf{p}[t], \mathbf{u}[t]) = \begin{bmatrix} v[t]\cos(\theta[t]) \\ v[t]\sin(\theta[t]) \end{bmatrix} \tag{2}$$

When winds are present but mild, existing UAVs are usually capable of stabilizing themselves, by applying disturbance rejection control strategies, like the ones described in.[33,34] However, when winds are strong, UAVs of small size can be displaced or flipped,[35] despite how robust their control systems to disturbances are. To capture UAV dynamics in a windy environment, we hence adopt the simple impact model described in.[36] Specifically, we assume the UAV is able to stabilize itself when the wind speed is below a certain threshold $\zeta > 0$, in which case its dynamics are captured by (2). On the other hand, when the wind speed exceeds ζ, we assume the UAV is unable to maintain normal operations and will crash.

II.C. Problem Formulation

In this study, we consider the scenario where N number of UAVs aim to navigate safely in a large-scale environment with both static and dynamic obstacles (e.g., buildings, trees, birds, other aircraft) and under wind disturbances. Each UAV i, $i \in [N]$, aims to reach a given target position $\mathbf{p}_{i,g}$ from a starting position $\mathbf{p}_{i,a}$, while avoiding static and dynamic obstacles as well as areas with strong winds.

To describe this problem mathematically, we introduce C_{free} to capture the free space in which the UAVs can operate safely at the presence of static obstacles and wind disturbances. Particularly, C_{free} is given by the following equation

$$C_{free} = \{\mathbf{p} | \mathbf{p} \in \mathcal{W}_l, \mathbf{p} \notin C_{obs}, w_l \leq \zeta, l \in [L]\},$$

where C_{obs} represents the static obstacle space. Furthermore, suppose there are M dynamic obstacles and the position of each dynamic obstacle $j \in [M]$ at time t is denoted by $\mathbf{o}_j[t]$. Assuming that $\mathbf{p}_{i,g}, \mathbf{p}_{i,a} \in C_{free}$, $\forall i \in [N]$, the multi-UAV navigation problem to be solved can then be described by following equations

$$\begin{aligned}
\underset{\mathbf{u}_i[0:T_i], \forall i \in [N]}{\text{minimize}} \quad & J = \sum_{i=1}^{N} T_i \\
\text{subject to} \quad & \dot{\mathbf{p}}_i[t] = f(\mathbf{p}_i[t], \mathbf{u}_i[t]), && \forall i \in [N] \\
& \mathbf{p}_i[T_i] = \mathbf{p}_{i,g}, && \forall i \in [N] \\
& \mathbf{p}_i[0] = \mathbf{p}_{i,a}, && \forall i \in [N] \\
& ||\mathbf{p}_i[t] - \mathbf{o}_j[t]|| \geq r_i + r_{o,j}, && \forall i \in [N], j \in [M] \\
& ||\mathbf{p}_i[t] - \mathbf{p}_j[t]|| \geq r_i + r_j, && \forall i \in [N], j \in [N], j \neq i \\
& \mathbf{p}_i[t] \in C_{free}, && \forall i \in [N] \\
& \mathbf{u}_i[t] \in \mathcal{U}, && \forall i \in [N]
\end{aligned}$$

where T_i is the flight time of UAV i and $\sum_{i=1}^{N} T_i$ is the total flight time we aim to minimize. To ensure safe operations, the UAVs should keep a safe distance from each other, as well as from any obstacles. By regarding each UAV as a sphere with radius of r_i, the safe distance between two UAVs, i and j, is thus $r_i + r_j$. Similarly, the safe distance between each UAV i and a dynamic obstacle j is $r_i + r_{o,j}$, where $r_{o,j}$ is the radius of the dynamic obstacle.

III. Data-Driven Multi-UAV Navigation Framework

In this section, we introduce a data-driven multi-UAV navigation framework to solve the problem formulated in the previous section efficiently.

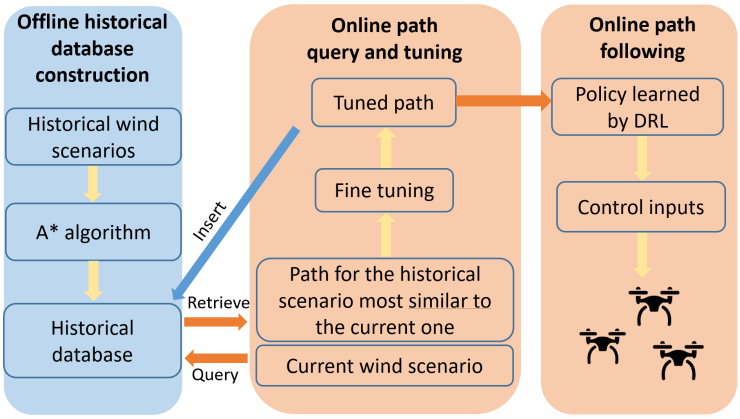


Figure 1: Overview of the proposed data-driven multi-UAV navigation framework.

Our framework (see Figure 1) consists of three phases: 1) the *offline historical database construction* phase, 2) the *online path query and tuning* phase and 3) the *online path following* phase. In the offline historical database construction phase, a historical database is constructed offline, which stores the historical wind scenarios tagged with corresponding paths generated by the A^* algorithm. The online path query and tuning phase is triggered when a new mission is received and performed before UAV departure. In this phase, the UAVs query the historical database using the current wind scenario to retrieve the most similar historical wind scenario and associated paths. The retrieved paths are then fine tuned to address the uniqueness of the current wind scenario. With the corrected paths, the UAVs then depart and enter the online path following phase. In this phase, the UAVs follow these paths and make proper adjustments to avoid dynamic obstacles, by applying the control policy learned by a deep reinforcement learning (DRL) based algorithm.[28] In the rest of this section, let's describe each phase in more detail.

Table 1: Structure of the Historical Database

Index	Wind scenario	Starting positions	Target positions	Paths
1	$\mathbf{S}_1$	$\mathbf{P}_a^{(1)}$	$\mathbf{P}_g^{(1)}$	$\mathcal{P}^{(1)}$
2	$\mathbf{S}_2$	$\mathbf{P}_a^{(2)}$	$\mathbf{P}_g^{(2)}$	$\mathcal{P}^{(2)}$
...	...	...	...	...
I	$\mathbf{S}_I$	$\mathbf{P}_a^{(I)}$	$\mathbf{P}_g^{(I)}$	$\mathcal{P}^{(I)}$

III.A. Offline Historical Database Construction

The historical database stores offline wind scenarios and associated paths for the UAVs generated by the A^* algorithm. Table 1 illustrates the data stored in the database, where $\mathbf{P}_a^{(j)} = \{\mathbf{p}_{1,a}^{(j)}, \mathbf{p}_{2,a}^{(j)}, \ldots, \mathbf{p}_{N,a}^{(j)}\}$ and $\mathbf{P}_g^{(j)} = \{\mathbf{p}_{1,g}^{(j)}, \mathbf{p}_{2,g}^{(j)}, \ldots, \mathbf{p}_{N,g}^{(j)}\}$ are the starting and target positions of the N UAVs in the j-th scenario,

$\mathcal{P}^{(j)} = \{\mathcal{P}_1^{(j)}, \mathcal{P}_2^{(j)}, \ldots, \mathcal{P}_N^{(j)}\}$ are the associated paths generated by the A^* algorithm, and I is the total number of wind scenarios in the database.

Algorithm 1: A* algorithm

Input: $\mathbf{P}_a$,$\mathbf{P}_g$, $\mathbf{S}$, C_{obs}, ζ, β
Output: $\mathcal{P}$

1 $\mathcal{P} \leftarrow \emptyset$
2 Construct $C_{free} \leftarrow \{\mathbf{p} | \mathbf{p} \in \mathcal{W}_l, \mathbf{p} \notin C_{obs}, w_l \leq \zeta, l \in [L]\}$
3 **foreach** $i \in [N]$ **do**
4 Construct $\mathcal{G}(V, E)$, with $V = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m\}$, $\mathbf{p}_1 = \mathbf{p}_{i,a}$, and $\mathbf{p}_m = \mathbf{p}_{i,g}$
5 $\mathcal{O} \leftarrow \{\mathbf{p}_1\}$; $\mathcal{C} \leftarrow \{\}$
6 $g_1 \leftarrow 0$; $g_k \leftarrow \infty, \forall k \in V \backslash \{\mathbf{p}_1\}$
7 **while** $\mathbf{p}_m \notin \mathcal{C}$ **do**
8 $\mathcal{O} \leftarrow \mathcal{O} \setminus \{\mathbf{p}_k\}$, where $\mathbf{p}_k = \arg\min_{\mathbf{p}_k \in \mathcal{O}} f_k$ and $f_k = g_k + \epsilon ||\mathbf{p}_k - \mathbf{p}_m||$
9 $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{p}_k\}$
10 **for** $j \in \{j | \mathbf{p}_j$ *is connected with* $\mathbf{p}_k\}$, $j \notin \mathcal{C}, j \in \mathcal{C}_{free}$ **do**
11 **if** $g_j > g_k + ||\mathbf{p}_k - \mathbf{p}_j||$ **then**
12 $g_j \leftarrow g_k + ||\mathbf{p}_k - \mathbf{p}_j||$
13 $B(\mathbf{p}_j) \leftarrow \mathbf{p}_k$
14 **if** $\mathbf{p}_j \in \mathcal{O}$ **then**
15 $f_j \leftarrow g_j + \epsilon ||\mathbf{p}_j - \mathbf{p}_m||$
16 **else**
17 $\mathcal{O} \leftarrow \mathcal{O} \cup \{\mathbf{p}_j\}$
18 $\mathcal{P}_i \leftarrow \{\mathbf{p}_m\}$
19 $\hat{\mathbf{p}} \leftarrow \mathbf{p}_m$
20 **while** $\mathbf{p}_1 \notin \mathcal{P}_i$ **do**
21 $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup B(\hat{\mathbf{p}})$
22 $\hat{\mathbf{p}} \leftarrow B(\hat{\mathbf{p}})$
23 $\mathcal{P} \leftarrow \{\mathcal{P}, \mathcal{P}_i\}$
24 **Return** $\mathcal{P}$

As the first step to enable long-range multi-UAV navigation, the A^* algorithm is used to generate the path for each UAV, given a wind scenario and map of the operating environment. The derived path avoids static obstacles and areas with strong winds and is composed of a sequence of waypoints. Note that collision avoidance with dynamic obstacles is tackled in the online path following phase. Algorithm 1 summarizes the procedures to generate the paths $\mathcal{P}$ using the A^* algorithm, given a wind scenario $\mathbf{S}$, starting positions $\mathbf{P}_a$ and target positions $\mathbf{P}_g$ of the UAVs.

To generate the paths $\mathcal{P}$, the first step is to construct the free space C_{free} based on the wind scenario $\mathbf{S}$ and the map of the operating environment, i.e., C_{obs} (Line 2). For each UAV i, we then construct a graph $\mathcal{G}(V, E)$ (Line 4) by decomposing the operating environment $\mathcal{W}$ into a collection of grids, with the size of each grid to be $\beta \times \beta, \beta > 0$. In this graph, the set of vertices V are formed by the centroids of the grids, and the edges E are formed by connecting adjacent grids, where grids that share a common edge/vertex are considered as adjacent to each other. Moreover, in case when the starting position $\mathbf{p}_{i,a}$ (or target position $\mathbf{p}_{i,g}$) is not the centroid of any grid, we expand the graph to include this position, by appending V with $\mathbf{p}_{i,a}$ (or $\mathbf{p}_{i,g}$) and adding the edge that connects $\mathbf{p}_{i,a}$ (or $\mathbf{p}_{i,g}$) with the centroid of the grid containing $\mathbf{p}_{i,a}$ (or $\mathbf{p}_{i,g}$) into E. For convenience of reference, we let $\mathbf{p}_k$ denote the k-th vertex in V, and $\mathbf{p}_{i,a} = \mathbf{p}_1$, $\mathbf{p}_{i,g} = \mathbf{p}_m$, where $m = |V|$ is the total number of vertices. With $\mathcal{G}$, the A^* algorithm is then performed to find the paths $\mathcal{P}$ (Line 3-23). In particular, Line 5 introduces a set $\mathcal{O}$ to store the vertices to visit and a set $\mathcal{C}$ to store the vertices that have been visited. g_k is then introduced in Line 6 to denote the estimated minimum cost from the starting position $\mathbf{p}_1$ (i.e., $\mathbf{p}_{i,a}$) to each vertex $\mathbf{p}_k \in V$, which is initialized to ∞, $\forall \mathbf{p}_k \in V \backslash \{\mathbf{p}_1\}$. After that, an iterative procedure is performed to optimize g_k until the target position $\mathbf{p}_m$ (i.e., $\mathbf{p}_{i,g}$) has been visited. In each iteration, the next vertex to visit is determined by a priority score defined as $f_k = g_k + \epsilon ||\mathbf{p}_k - \mathbf{p}_m||$, where ϵ is a weight factor that balances between the efficiency and optimality.

In Line 13, $B(\mathbf{p}_k)$ is used to store the vertex right before $\mathbf{p}_k$ in the estimated shortest path from $\mathbf{p}_1$ to $\mathbf{p}_k$. Finally, the path $\mathcal{P}_i$ for UAV i is constructed based on $B(\mathbf{p}_k)$ (Line 18 to Line 23).

III.B. Online Path Query and Tuning

With the current wind scenario generated by forecasting tools, the online path query and tuning phase quickly generates feasible paths for the UAVs to reach their target positions. This is achieved in two steps. The first step queries the historical database to retrieve the most similar wind scenario and associated paths. The second step fine tunes the retrieved paths to address any safety concerns incurred due to the differences between the current and retrieved wind scenarios.

To find the historical scenario most similar to the current wind scenario, we apply the similarity search algorithm developed in our previous studies.[37,38] This algorithm utilizes a multiresolution distance measure[39] to quantify the similarity between different spatiotemporal wind scenarios. As wind dynamics are assumed to be time-invariant over the planning horizon in this study, wind scenarios are compared only along the spatial dimension. In particular, to calculate the similarity between two scenarios $\mathbf{S}_i$ and $\mathbf{S}_j$, we adopt a spatial moving window of increasing size to scan two scenarios simultaneously and calculate their distance after each scan by using the following equation:

$$\mathcal{D}_{i,j,z} = \sum_{\phi_{l,z}\in\Phi_z} \frac{1}{|\phi_{l,z}|} \left[\sum_{\tilde{l}\in\phi_{l,z}} \left(\frac{w_{\tilde{l}}^{(i)}}{\lambda_{\tilde{l},z}} - \frac{w_{\tilde{l}}^{(j)}}{\lambda_{\tilde{l},z}} \right) \right] \tag{3}$$

In the above equation, $\phi_{z,l}$ denotes the spatial window of size z centered at region l, which includes all regions that are within z hops away from l. $\Phi_z = \{\phi_{z,l} | l \in [L]\}$ denotes the set of all possible windows of size z. $\lambda_{\tilde{l},z} = \sum_{\phi_{l,z}\in\{\phi_{l,z}|\tilde{l}\in\phi_{l,z}\}} \frac{1}{|\phi_{l,z}|}$is the spatial contribution factor to address the unbalanced contributions of boundary regions. $w_{\tilde{l}}^{(j)}$ is the wind speed in region $\tilde{l}$ of wind scenario j. The overall distance $\mathcal{D}_{i,j}$ between the two scenarios is the weighted average of the distances obtained at each window size. In particular,

$$\mathcal{D}_{i,j} = \sum_{z=1}^{z_{\max}} \mathcal{D}_{i,j,z} \frac{\sigma_z}{\sum_{z=1}^{z_{\max}} \sigma_z} \tag{4}$$

where z_{max} is the maximum window size, $\sigma_z = e^{-\alpha(z-1)}$ is a weighting factor that decreases with the increase of the window size, indicating less contributions of larger window sizes with coarser resolutions. Based on this distance measure, the similarity search algorithm[37,38] adopts an iterative procedure to quickly find the scenario most similar to the query scenario. The key idea is to use the bounds of the distance measure tightened at each iteration to progressively prune the search space until the scenario with the smallest distance is found. For more detailed descriptions, please refer to papers.[37,38]

Given the current wind scenario $\mathbf{S}_c$, the path retrieved from the historical database can then be described by

$$\begin{aligned} \mathcal{P}^* &= \mathcal{P}^{(q)} \\ q &= \underset{j\in[I]}{arg\ min}\ \ \mathcal{D}_{c,j} \end{aligned} \tag{5}$$

As the current wind scenario $\mathbf{S}_c$ is not exactly same as scenario $\mathbf{S}_q$ in the database that is most similar to it, the retrieved path $\mathcal{P}^*$ may not satisfy the safety requirements. Therefore, fast online tuning is desired. We here develop a simple heuristic online tuning algorithm (see Algorithm 2) that performs a safety check for each waypoint along the retrieved path in order and makes adjustments whenever a waypoint violates the safety requirements. In particular, for each UAV, we start from the first waypoint in its retrieved path and check whether this waypoint is in the free space $\mathcal{C}_{free}$ constructed based on $\mathbf{S}_c$ or not. If yes, we move on to the next waypoint. Otherwise, we run the A^* algorithm by setting the previous waypoint as the starting position, and setting the next waypoint in $\mathcal{C}_{free}$ as the end position. The generated sequence of waypoints is then used to correct the retrieved path by replacing the corresponding path segment with these waypoints. This procedure is repeated until the end of the path. With the corrected path, each UAV then departs and performs online path following described in the next subsection to reach its target position. At the same time, we also insert the corrected path and the current wind scenario into the historical database to facilitate future planning.

Algorithm 2: Online Tuning Algorithm

Input: Retrieved paths $\mathcal{P}^* = \{\mathcal{P}_1^*, \mathcal{P}_2^*, \ldots, \mathcal{P}_N^*\}$, current wind scenario $\mathbf{S}_c$
Output: Corrected paths $\mathcal{P}$

$C_{free} \leftarrow \{\mathbf{p} | \mathbf{p} \in \mathcal{W}_l, \mathbf{p} \notin C_{obs}, w_l \leq \zeta, l \in [L]\}$
foreach $i \in [N]$ **do**
  $k \leftarrow 1$
  $\mathcal{P}_i \leftarrow \mathcal{P}_i^*(k)$, where $\mathcal{P}_i^*(k)$ is the k-th waypoint in path $\mathcal{P}_i^*$.
  while $k \leq n_i := |\mathcal{P}_i^*|$ **do**
    if $\mathcal{P}_i^*(k) \notin C_{free}$ **then**
      $j \leftarrow k + 1$
      while $j \leq n_i$ **and** $j \notin C_{free}$ **do**
        $j \leftarrow j + 1$
      Run A^* algorithm with $\mathcal{P}_i^*(k-1)$ as the starting position and $\mathcal{P}_i^*(j)$ as the end position to generate path $\mathcal{P}_s$
      $\mathcal{P}_s \leftarrow \mathcal{P}_s \setminus \{\mathcal{P}_i^*(k-1), \mathcal{P}_i^*(j)\}$
      $\mathcal{P}_i \leftarrow [\mathcal{P}_i, \mathcal{P}_s]$
      $k \leftarrow j$
    else
      $\mathcal{P}_i \leftarrow [\mathcal{P}_i, \mathcal{P}_i^*(k)]$
      $k \leftarrow k + 1$
  if $\mathcal{P}_i^*(n_i) \in \mathcal{P}_i$ **then**
    $\mathcal{P} \leftarrow \{\mathcal{P}, \mathcal{P}_i\}$
  else
    Return NaN, as no feasible is found.
Return $\mathcal{P}$

III.C. Online Path Following

In the online path following phase, the UAVs depart from their starting positions and navigate towards their target positions by following the paths $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$ generated from the previous phase. To avoid dynamic obstacles, a state-of-the-art DRL-based motion planning approach, called GA3C-CADRL (GPU/CPU Asynchronous Advantage Actor-Critic for Collision Avoidance with Deep Reinforcement Learning),[28] is adopted to learn point-to-point navigation policies. A brief description of GA3C-CADRL is provided as follows.

Define $\tilde{\mathbf{p}}_i[t] = [\tilde{\mathbf{p}}_i^o[t], \tilde{\mathbf{p}}_i^h[t]]$ as the new state vector for UAV i, where $\tilde{\mathbf{p}}_i^o[t] = [\mathbf{p}_i[t], \dot{\mathbf{p}}_i[t], r_i]$ is the observable state including the position, velocity, and radius of the UAV i, $\tilde{\mathbf{p}}_i^h[t] = [\mathbf{p}_{i,k}, v_{max}, \theta[t]]$ is the hidden state with $\mathbf{p}_{i,k}$ being the k-th waypoint in the path $\mathcal{P}_k$, $k \in [n_i]$. Moreover, let $\tilde{\mathbf{p}}_i^{o-}[t]$ denote the observable state vectors of all UAVs except i, and $\tilde{\mathbf{o}}_j[t] = [\mathbf{o}_j[t], \dot{\mathbf{o}}_j[t], r_{o,j}]$ denote the state vector of obstacle j. Of note, in this path following phase, we also consider static obstacles and strong wind areas as dynamic obstacles but with zero velocities.

Based on above definitions, the GA3C-CADRL learns a stochastic policy $\pi(\mathbf{u}_i[t] \mid \tilde{\mathbf{p}}_i[t], \tilde{\mathbf{p}}_i^{o-}[t], \tilde{\mathbf{o}}_j[t], \forall j \in [M])$ that outputs the probability density function of the control inputs $\mathbf{u}_i[t]$, given the states of all UAVs, $\tilde{\mathbf{p}}_i[t]$ and $\tilde{\mathbf{p}}_i^{o-}[t]$, as well as the states of the obstacles $\tilde{\mathbf{o}}_j[t], \forall j \in [M]$. As the number of obstacles may differ in different scenarios, to make the DRL system general to an arbitrary number of obstacles, the policy $\pi(\mathbf{u}_i[t] | \tilde{\mathbf{p}}_i[t], \tilde{\mathbf{p}}_i^{-o}[t], \tilde{\mathbf{o}}_j[t], \forall j \in [M])$ is represented by a neural network that consists of a Long Short Term Memory (LSTM) cell and two fully connected layers, as illustrated in Figure 2. The usage of LSTM allows the policy to accept an arbitrary number of obstacles' states as the inputs.

To learn the policy, multiple UAVs are operated to interact with the environment through simulations or real experiments, where the UAVs learn a common policy. The reward (or cost) signals received from the environment are used to optimize the policy by minimizing the expected flight time $T_{i,k}$ from one waypoint

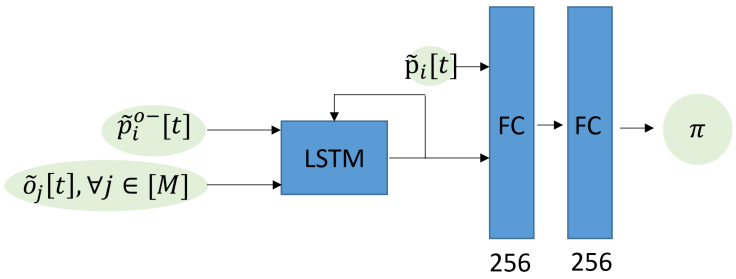


Figure 2: Neural network architecture of policy function.

$\mathbf{p}_{i,k}$ to the next $\mathbf{p}_{i,k+1}$, $k \in [n_i - 1]$. The mathematical formulation is given as follows

$$\begin{aligned} \underset{\pi(\mathbf{u}_i[t]|\tilde{\mathbf{p}}_i[t],\tilde{\mathbf{p}}_i^o[t],\tilde{\mathbf{o}}_j[t],\forall j\in[M])}{\text{argmin}} \quad & \mathbb{E}\left[T_{i,k}|\tilde{\mathbf{p}}_i[t_k],\tilde{\mathbf{p}}_i^o[t_k],\tilde{\mathbf{o}}_j[t_k],\forall j\in[M],\pi\right] \\ \text{subject to} \quad & \dot{\mathbf{p}}_i[t] = f(\mathbf{p}_i[t],\mathbf{u}_i[t]) \\ & \mathbf{p}_i[T_{i,k}] = \mathbf{p}_{i,k+1} \\ & \mathbf{p}_i[t_k] = \mathbf{p}_{i,k} \\ & \|\mathbf{p}_i[t] - \mathbf{p}_j[t]\| \geq r_i + r_j, \forall j \in [N]\backslash\{i\} \\ & \|\mathbf{p}_i[t] - \mathbf{o}_j[t]\| \geq r_i + r_{o,j}, \forall j \in [M] \\ & \mathbf{p}_i[t] \in C_{free} \\ & \mathbf{u}_i[t] \in \mathcal{U} \end{aligned} \tag{6}$$

For more details, please refer to the paper.[28]

IV. Simulation Studies

In this section, we evaluate the performance of the proposed data-driven multi-UAV navigation framework through simulation studies. In particular, we first illustrate the procedures to construct the historical database. We then show the effectiveness of the path query and tuning phase in accelerating the online path planning procedure, as well as the effectiveness of the path following phase in enabling safe multi-UAV navigation at the presence of dynamic obstacles. Finally, we demonstrate that combining A^* for path planning with DRL for motion planning enables long-range navigation in a dynamic environment, which cannot be achieved by either A^* or DRL alone. In the following simulation studies, all experiments were run on an Alienware Desktop with 32GB memory and 16-core 3.6GHz CPU.

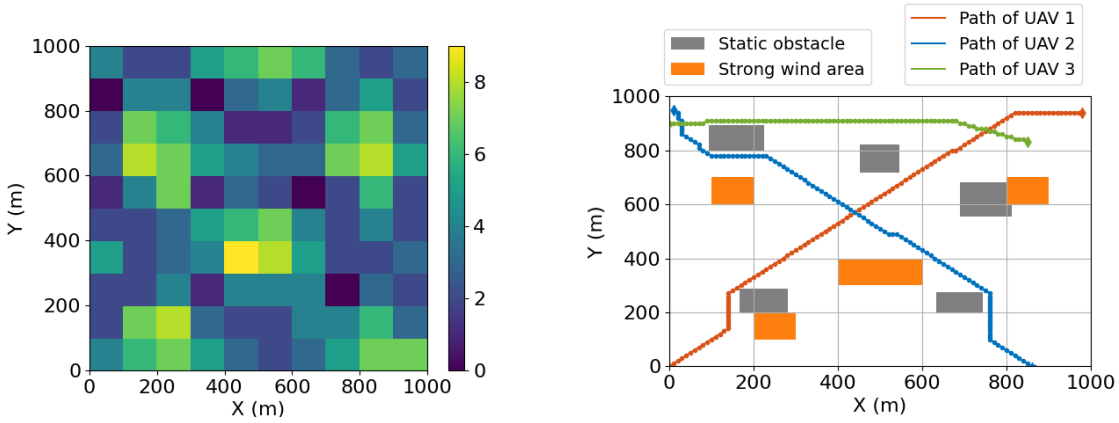
IV.A. Offline Historical Database Construction

To construct the historical database, we consider $N = 3$ UAVs navigating in a wind field of size $[0, 1000m] \times [0, 1000m]$. The starting positions of the three UAVs are set to $[0, 0]$, $[860m, 5m]$, $[0, 900m]$, respectively, and their target positions are set to $[980m, 940m]$, $[10m, 950m]$, $[850m, 830m]$, respectively. The wind field is equally divided into $L = 100$ regions, with the size of each region to be $100m \times 100m$. $I = 100$ spatial wind scenarios are then generated using the wind influence model, with each snapshot of the wind field generated at each time step considered as one scenario. Specifically, the range of the wind speed is set to $[0m/s, 10m/s)$ with resolution of $\Delta_w = 1m/s$. Therefore, the total number of wind statuses is $K = 10$. The initial wind speed in each region l is determined by first applying the following equation,[40]

$$\sqrt{\left[2\sin\left(\frac{x_l}{100}\right) + 4\cos\left(\frac{y_l}{100}\right) + \delta\right]^2 + \left[2\cos\left(\frac{x_l}{100}\right) + 4\sin\left(\frac{y_l}{100}\right) + \delta\right]^2} \tag{7}$$

and then rounding the value to the nearest wind status value. Here, δ is a standard normal random variable, i.e., $\delta \sim \mathcal{N}(0, 1)$, and $[x_l, y_l]^T$ is the center position of region l.

For each wind scenario, we then run the A^* algorithm (Algorithm 1) offline to generate near-optimal paths for the three UAVs, with the threshold of wind speed for safe operation set as $\zeta = 7m/s$. An example wind scenario and the corresponding paths of the three UAVs are shown in Figure 3. Of note, the average computation time of the A^* algorithm to generate paths for the three UAVs is $183s$, which does not meet the real-time decision requirement.



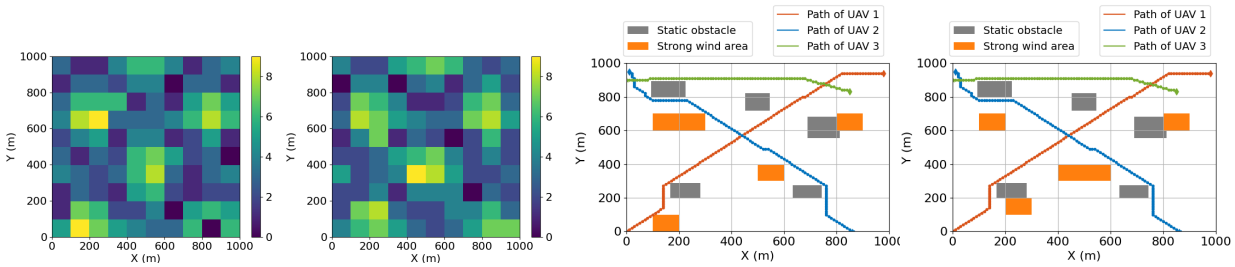
(a) (b)

Figure 3: An example of a) the wind scenario and b) corresponding paths of the three UAVs.

IV.B. Online Path Query and Tuning

In this subsection, we demonstrate the capability of the online path query and tuning phase in our framework through three experimental studies. In these and following studies, the parameters of the similarity search algorithm are set to $\alpha = 1$ and $z_{max} = 8$.

In the first experiment, we generate a new wind scenario that is very similar to the one retrieved from the historical database, as shown in Figure 4 (a)-(b). In this case, the retrieved paths can be directly applied without correction. Figure 4(d) and 4(c) show the retrieved paths before and after applying the online tuning procedure (Algorithm 2). As no corrections are actually made to the retrieved paths, the "corrected" paths are exactly same as the retrieved paths. The time taken to perform path query and tuning is $4s$, demonstrating the efficiency of our framework.



(a) (b) (c) (d)

Figure 4: Illustration of the a) current wind scenario, b) retrieved wind scenario, c) corrected paths, and d) retrieved paths in the first experiment.

In the second experiment, we generate a new wind scenario that is quite different from the one retrieved from the historical database, as shown in Figure 5(a)-(b). In this case, the retrieved paths cannot be directly used and need to be corrected. Figure 5(c) shows the corrected paths after applying the online tuning procedure on the retrieved paths shown in Figure 5(d). As we can see, the corrected paths differ from the retrieved ones and can successfully avoid strong wind areas that do not appear in the retrieved wind scenario. Compared with the first experiment, it takes 26 more seconds to perform path query and tuning in this case

due to path corrections. Nevertheless, the overall computation time, which is $30s$, is still much shorter than the time required for directly running the A^* algorithm, which is $183s$.

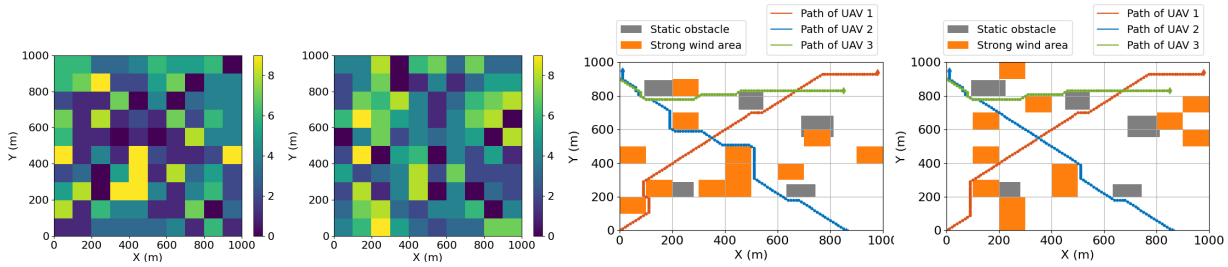


Figure 5: Illustration of the a) current wind scenario, b) retrieved wind scenario, c) corrected paths, and d) retrieved paths in the second experiment.

In the third experiment, we randomly generate 10 new wind scenarios, and run the path query and tuning procedures for each wind scenario. The average computation time is $34s$, which further demonstrates the effectiveness of the path query and tuning phase in expediting the online path planning procedure.

IV.C. Online Path Following

The online path following phase enables the UAVs to follow the paths generated from the online path query and tuning phase, while avoiding dynamic obstacles. To demonstrate its performance, we introduce a dynamic obstacle that moves along the same path as the first UAV but in the opposite direction. Therefore, collisions will happen if the UAV does not act properly. To implement online path following, we adopt the same settings as the ones used in paper[28,41] to generate DRL policies for the UAVs. The control frequency is set to 20Hz. The radius of the three UAVs and the dynamic obstacle are all set to $0.6m$. During the movement, both static and dynamic obstacles within the range of $8m$ can be sensed by the UAVs, and the sensing information include positions, sizes, and velocities of the obstacles. Figure 6 shows the trajectories of the three UAVs and the dynamic obstacle. The current wind scenario and the retrieved paths are shown in Figure 4(a) and Figure 4(d), respectively. As we can see, all three UAVs can successfully reach their target positions without crashing into any static/dynamic obstacles or entering areas with strong winds.

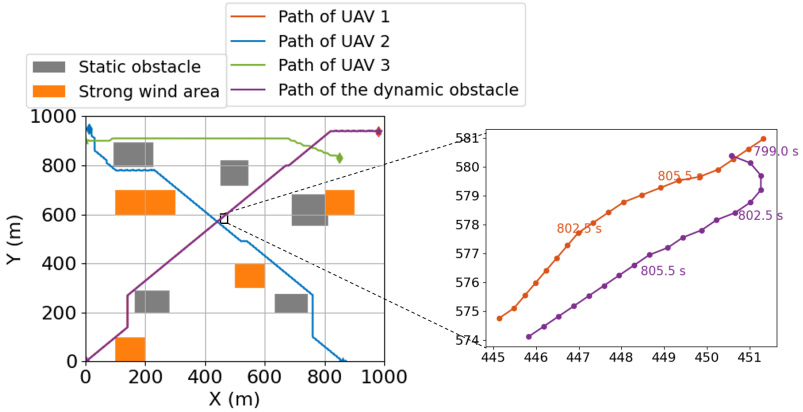


Figure 6: Trajectories of three UAVs and the dynamic obstacle.

IV.D. Long-Range Multi-UAV Navigation

To further demonstrate the effectiveness of our framework that combines A^* with DRL, we show in this subsection that directly applying the DRL is infeasible for long-range multi-UAV navigation. Note that the A^* algorithm alone is unable to handle dynamic obstacles.

Consider the wind scenario shown in Figure 4(a), the trajectories of the three UAVs generated by directly applying the DRL are shown in Figure 7. As we can see, all UAVs move in a circle and fail to reach their target positions.

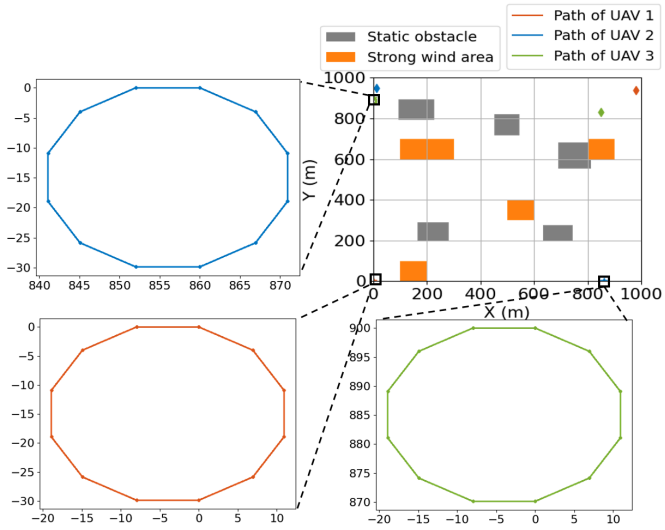


Figure 7: Trajectories of the UAVs operating in a large-scale environment when directly applying the DRL.

For demonstration purpose, we also run a small-scale experiment and show that DRL is indeed feasible for short-range multi-UAV navigation. In this experiment, we consider the scenario where three UAVs operate in a small space of size $[-5m, 15m] \times [-5m, 15m]$. The starting positions of the three UAVs are set to $[0m, 0m]$, $[-1m, -1m]$ and $[-1m, 2m]$, respectively, and their target positions are set to $[-1m, 4m]$, $[12m, 10m]$ and $[8m, 6m]$, respectively. The trajectories of the three UAVs by applying the DRL are shown in Figure 8. As we can see, all UAVs can reach their target positions and successfully avoid both static obstacles and other UAVs.

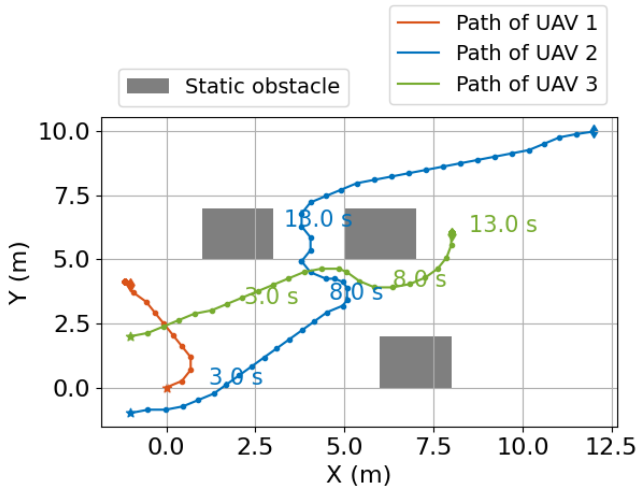


Figure 8: Trajectories of the UAVs operating in a small-scale environment when directly applying the DRL.

V. Conclusion

In this paper, we developed a data-driven multi-UAV navigation framework that combines A^* algorithm with DRL to enable multiple UAVs to navigate safely in a large-scale dynamic environment under wind

disturbances. To speed up the online planning procedure, this framework offloads most computations to offline by leveraging a historical database that stores paths planned for historical wind scenarios. Given a new wind scenario, instead of planning from scratch, our framework retrieves the most similar historical scenario and associated paths from the database and fine tunes the paths to address the uniqueness of the current scenario online. Simulation results show that, by using our framework, the UAVs can successfully and safely navigate to their target positions in a large-scale complicated environment, with the existence of both static/dynamic obstacles and wind disturbances, which cannot be achieved by either the A^* or DRL alone. The results also show that our data-driven framework significantly reduces the computation time for online path planning.

Acknowledgments

This work is partially supported by the National Science Foundation under Grant CNS-1953048 and ECCS-1953049, and by San Diego State University under the University Grants Program.

References

[1]K. Máthé and L. Buşoniu, "Vision and control for uavs: A survey of general methods and of inexpensive platforms for infrastructure inspection," *Sensors*, vol. 15, no. 7, pp. 14 887–14 916, 2015.

[2]F. G. Costa, J. Ueyama, T. Braun, G. Pessin, F. S. Osório, and P. A. Vargas, "The use of unmanned aerial vehicles and wireless sensor network in agricultural applications," in *Proceedings of the 2012 IEEE International Geoscience and Remote Sensing Symposium*. Munich, Germany: IEEE, July 2012.

[3]C. Yuan, Z. Liu, and Y. Zhang, "Uav-based forest fire detection and tracking using image processing techniques," in *Proceedings of the 2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. Denver, CO: IEEE, July 2015.

[4]E. T. Bokeno, T. M. Bort, S. S. Burns, M. Rucidlo, W. Wei, D. L. Wires *et al.*, "Package delivery by means of an automated multi-copter uas/uav dispatched from a conventional delivery vehicle," Jul. 14 2016, uS Patent App. 14/989,870.

[5]J. Chen, "Fast planning for joint routing and charging of autonomous drone delivery system," in *Proceedings of AIAA Scitech 2020 Forum*. Orlando, Florida: AIAA, January 2020.

[6]F. aerospace Forecasts, "Fiscal years 2020-2040, faa office of aviation policy and plans," Tech. Rep., 2020.

[7]P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. Robinson, "Unmanned aircraft system traffic management (utm) concept of operations," in *Proceedings of 2016 AIAA Aviation Forum*. Washington, DC USA: AIAA, June 2016.

[8]T. Prevot, J. Rios, P. Kopardekar, J. E. Robinson III, M. Johnson, and J. Jung, "Uas traffic management (utm) concept of operations to safely enable low altitude flight operations," in *Proceedings of 16th AIAA Aviation Technology, Integration, and Operations Conference*. Washington, DC: AIAA, June 2016.

[9]M. Johnson, J. Jung, J. Rios, J. Mercer, J. Homola, T. Prevot, D. Mulfinger, and P. Kopardekar, "Flight test evaluation of an unmanned aircraft system traffic management (utm) concept for multiple beyond-visual-line-of-sight operations," 2017.

[10]A. S. Aweiss, B. D. Owens, J. Rios, J. R. Homola, and C. P. Mohlenbrink, "Unmanned aircraft systems (uas) traffic management (utm) national campaign ii," in *Proceedings of 2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, Kissimmee, Florida, January 2018.

[11]W. Ren and R. W. Beard, "Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 5, pp. 706–716, 2004.

[12]J. Pérez, J. Godoy, J. Villagrá, and E. Onieva, "Trajectory generator for autonomous vehicles in urban environments," in *Proceedings of 2013 IEEE International Conference on Robotics and Automation*. Karlsruhe: IEEE, May 2013.

[13]O. K. Sahingoz, "Generation of bezier curve-based flyable trajectories for multi-uav systems with parallel genetic algorithm," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 499–511, 2014.

[14]R. Song, Y. Liu, and R. Bucknall, "Smoothed a* algorithm for practical unmanned surface vehicle path planning," *Applied Ocean Research*, vol. 83, pp. 9–20, 2019.

[15]Y. Singh, S. Sharma, R. Sutton, and D. Hatton, "Optimal path planning of an unmanned surface vehicle in a real-time marine environment using a dijkstra algorithm," in *Marine Navigation*. CRC Press, 2017, pp. 399–402.

[16]P. E. Hart, N. J. 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.

[17]D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, 2005, pp. 1–10.

[18]P. Wu, L. Li, J. Xie, and J. Chen, "Probabilistically guaranteed path planning for safe urban air mobility using chance constrained rrt," in *Proceedings of AIAA AVIATION 2020 FORUM*, virtual, June 2020.

[19]P. Wu, J. Xie, and J. Chen, "Safe path planning for unmanned aerial vehicle under location uncertainty," in *Proceedings of 2020 IEEE 16th International Conference on Control & Automation (ICCA)*, virtual, June 2020.

[20]S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[21]J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour, "Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states," in *Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, Washington: IEEE, May 2015.

[22]J. Xie, L. R. G. Carrillo, and L. Jin, "An integrated traveling salesman and coverage path planning problem for unmanned aircraft systems," *IEEE control systems letters*, vol. 3, no. 1, pp. 67–72, 2018.

[23]V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning," *IEEE Transactions on industrial informatics*, vol. 9, no. 1, pp. 132–141, 2012.

[24]C. Zhang, Z. Zhen, D. Wang, and M. Li, "Uav path planning method based on ant colony optimization," in *Proceedings*

of 2010 Chinese Control and Decision Conference. Xuzhou, China: IEEE, May 2010.

[25]L. R. Howell and B. D. Allen, "Spline trajectory algorithm development: Bézier curve control point generation for uavs," in *Proceedings of 8th AIAA Atmospheric and Space Environments Conference*, Washington, D.C., June 2016.

[26]K. Bollino and L. R. Lewis, "Collision-free multi-uav optimal path planning and cooperative control for tactical applications," in *Proceedings of 2008 AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, August 2008.

[27]Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su, "Uav path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407–1420, 2016.

[28]M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain: IEEE, October 2018.

[29]A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *Proceedings of 2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia: IEEE, May 2018.

[30]S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[31]C. He and Y. Wan, "Clustering stochastic weather scenarios using influence model-based distance measures," in *Proceedings of AIAA Aviation 2019 Forum*. Dallas, Texas: AIAA, June 2019.

[32]C. Asavathiratham, "The influence model: A tractable representation for the dynamics of networked markov chains," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[33]Y. Liu, S. Rajappa, J. M. Montenbruck, P. Stegagno, H. Bülthoff, F. Allgöwer, and A. Zell, "Robust nonlinear control approach to nontrivial maneuvers and obstacle avoidance for quadrotor uav under disturbances," *Robotics and Autonomous Systems*, vol. 98, pp. 317–332, 2017.

[34]C. Zhang, X. Zhou, H. Zhao, A. Dai, and H. Zhou, "Three-dimensional fuzzy control of mini quadrotor uav trajectory tracking under impact of wind disturbance," in *Proceedings of 2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*. Melbourne, Australia: IEEE, November 2016.

[35]K. Cole and A. M. Wickenheiser, "Trajectory generation for uavs in unknown environments with extreme wind disturbances," *arXiv preprint arXiv:1906.09508*, 2019.

[36]——, "Reactive trajectory generation for multiple vehicles in unknown environments with wind disturbances," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1333–1348, 2018.

[37]J. Xie, A. Reddy Kothapally, Y. Wan, C. He, C. Taylor, C. Wanke, and M. Steiner, "Similarity search of spatiotemporal scenario data for strategic air traffic management," *Journal of Aerospace Information Systems*, pp. 187–202, 2019.

[38]J. Xie, H. Nguyen, and Y. Wan, "Similarity search of spatiotemporal scenario data for strategic air traffic management," in *Proceedings of 2018 Aviation Technology, Integration, and Operations Conference*. Atlanta, Georgia: AIAA, June 2018.

[39]J. Xie, Y. Wan, Y. Zhou, S.-L. Tien, E. P. Vargo, C. Taylor, and C. Wanke, "Distance measure to cluster spatiotemporal scenarios for strategic air traffic management," *Journal of Aerospace Information Systems*, vol. 12, no. 8, pp. 545–563, 2015.

[40]S. Yang, N. Wei, S. Jeon, R. Bencatel, and A. Girard, "Real-time optimal path planning and wind estimation using gaussian process regression for precision airdrop," in *Proceedings of 2017 American Control Conference (ACC)*. Seattle, Washington: IEEE, May 2017.

[41]M. Everett, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning," 2020. [Online]. Available: https://github.com/mit-acl/rl_collision_avoidance