Energy-optimized Path Planning for UAS in Varying Winds via Reinforcement Learning

Portia Banerjee
KBR Inc., NASA Ames Research Center
Moffett Field, CA 94035
portia.banerjee@nasa.gov

Kevin Bradner
NASA Ames Research Center
Moffett Field, CA 94035
kevin.bradner@nasa.gov

In this paper we propose a reinforcement learning (RL) algorithm for path planning of Unmanned Aviation Vehicles (UAVs) under varying wind conditions. Solutions to UAV path planning problems are becoming increasingly necessary as autonomous UAVs continue to enter commercial and government spaces. Path-planning is inherently challenging, as UAVs need to account for dynamically changing flying conditions such as weather, obstacle or no-fly zones, degraded vehicle health, and off-nominal battery power consumption. Machine learning methods such as reinforcement learning (RL) have the potential to revolutionize how vehicles navigate in such uncertain environments. In this study, we compute UAV trajectories from a pre-determined starting position to a target cell within a 7X7 grid environment by optimizing parameters for mission assurance and safety limits in addition to the energy consumption and operation time. The UAV navigates the grid by taking actions to move in any of the eight cardinal and inter-cardinal directions, under constant thrust profile. The resultant UAV state is sampled from a probability distribution which accounts for the UAV's action, local wind velocity, and the presence of obstacles or boundaries. As the unmanned airspace gets more complex due to multiple vehicles and environmental uncertainties, trade-offs between energy consumption, operation time, risk tolerance, and mission assurance need to be made. Our Markov Decision Process (MDP) environment model can capture any combination of these in the optimization objective, making it novel compared to other work in the field.

I. Introduction

The increasing popularity of unmanned aerial systems (UAS) has become evident in numerous applications such as precision agriculture, package delivery, disaster management and urban traffic surveillance operations. While UASs offer numerous benefits including reduced surveillance time and reduced human intervention in critical operations, unique challenges arise with such small rotorcrafts, especially due to their limited power storage and reduced tolerance to dynamic weather changes in the low altitude airspace. Owing to these factors, often UASs are restricted to limited usage in many practical implementations. In order to achieve efficient UAS operations in a dynamic and uncertain environment within safety bounds issued by the Federal Aviation Administration (FAA), intelligent and optimized path-planning for UASs is crucial.

The primary challenges in UAS path planning arise from (1) dynamically changing obstacles, issued by air-traffic-controllers, either in the form of other aircraft or no-fly zones and (2) dynamic weather in the form of varying wind magnitude, direction, and gusts. For a safe operation, a UAS needs to steer clear from obstacles as well as high winds or turbulence zones that are often prevalent in the low-altitude airspace where these vehicles are planned to operate. Particularly when UASs are being prepared to be used in emergency response or in urban environments, the effect of wind on the flown trajectory becomes more critical. Wind of higher sustained magnitude or gusts can deviate a UAS from its flight plan and increase risk of obstacle collision or may lead to loss of power, either of which poses high risk to the vehicle as well as the airspace. Previous studies by the authors have demonstrated the effect of wind magnitude and direction on deviation of the UAS trajectory from its commanded path [1] and on its energy consumption[2]. Such studies indicate the importance of incorporating these factors into the path planning of a UAS operation.

There are a few planning and optimization methods in the literature which compute the shortest trajectory in the presence of obstacles [3], avoiding hazardous weather [4] and restricted air space [5]. A hybrid routing and scheduling problem of an UAV delivery system studied in [6] minimized flight time. On the other hand, minimizing power consumption has been widely recognized as a key objective for modern airline trajectories. Here, aircraft trajectories have been optimized with respect to minimum fuel usage, considering fight paths constrained to a horizontal plane. In the case of UASs [7], trajectory has been optimized in terms of minimizing the jerk or cost of distance to obstacles or the unmanned airspace boundaries and exploiting wind to increase the flight time. This study focused on implementation of the aerodynamic simulation of unpowered (i.e gliding) aircrafts in windy scenarios. In Al-Sabbah's paper [8], an Markov decision process based path-planning is implemented for a 3-degrees-of-freedom UAS simulation model to compute the shortest trajectory by exploiting the wind field. They demonstrated their planned trajectory on uniform wind field as well as spatially and temporally changing wind field. In this paper, we optimize trajectories for rotorcraft vehicles. The environment model uses a lumped-mass model for simulating an octocopter trajectory including powertrain dynamics and an electrochemistry-based battery model to compute the energy consumed during the octocopter flight under varying wind conditions. Our optimization process then generates the minimum time as well as minimum energy path for that environment.

In this paper, we explore reinforcement learning (RL) methods to generate an optimized trajectory that includes the aerodynamic and kinematic response of a UAS in the presence of uncertain and dynamically changing winds. In most practical cases, it is extremely challenging to obtain the exact wind measurements along the flight trajectory, particularly for the short duration and distance covered by a typical UAS flight. The wind magnitude and direction along the trajectory has to be modelled based on weather data that are typically recorded at a much lower spatial and temporal resolution. Hence, interpolation methods such as the Gaussian Process Regression have been explored to estimate the wind measurements and the uncertainty in them [1]. The trajectory planner proposed here incorporates such uncertainties in the wind field measurements.

II. Background and Algorithm Principles

A. Markov Decision Process

One of the typical models of a sequential decision-making process that has been implemented in the field of robotic path planning is the Markov Decision Process (MDP). At each step of an MDP, the decision-maker (agent) interacts with the environment by selecting an action. After that, the environment's dynamics model transitions the agent from its current state to the next and issues a reward. In general, this transition function may be stochastic. A sequence of such interactions is called a trajectory, and the sequence of rewards in each trajectory defines a return, typically as a sum of rewards after applying exponentially decaying weights. The optimization objective is to maximize the expected return in each new episode.

Given that MDPs and their solution methods are popular and well-established, we present only a brief description of the theory in this section. An MDP is often defined by the tuple : S, A, p in which S is the set of all states, $A(s)$ is set of actions available at state s, and $p(s', r|s, a)$ is the dynamics function which defines a distribution over the successor states and rewards that can result from taking action a from state s. In some cases, we will instead use the transition function $p(s|s, a)$ which only defines a distribution over successor states. From a trajectory which leads to state s the agent selects an action a from $A(s)$, after which the environment completes the state transition and defines the tuple (s, a, r, s') which extends the earlier trajectory and allow the agent to act again. An agent acts according to a policy $\pi(a|s)$. Here, the value function of a state under a policy π is defined as the expected return from a state s which in turn is the immediate reward plus the discounted expected return under π associated with the next state s', leading to a recursive definition. An optimal policy associated with an MDP is defined by satisfying the Bellman Optimality equation, given by Eq. (2).

$$v^*(s) \equiv \max_{\pi} v_{\pi}(s) \tag{1}$$

$$v^*(s) = \max_{a} \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')] \tag{2}$$

B. Dynamic Programming

Dynamic programming refers to a series of algorithms for computing the optimal policy given a complete environment model represented as a MDP. One of the most popular dynamic programming methods is value iteration [9] , which alternates between evaluating the value of each state under a policy π and then optimizing π with respect to those estimates. Value iteration begins by initializing the value of the goal state v_0 and then each subsequent value is computed iteratively using the modified Bellman Equation, as defined in (3).

$$v_{k+1} = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')] \tag{3}$$

Under Value Iteration, the sequence v_k will converge to v^* along with $k \rightarrow \inf$. To obtain each successor v_{k+1} from v_k, iterative policy evaluation takes the same action for each state s and replaces the value estimate at state s with a new value which looks one step forward optimizes a with respect to the expected return under the current value function's estimate of each possible successor state's value. In this way, the optimal value for all states is computed. The optimal policy $\pi^*(s)$ is easily obtained by selecting the action in a state that generates the maximum value for that state, as stated in Eq. (4).

$$\pi^*(s) = arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')] \tag{4}$$

The primary assumption required to use DP to solve the MDP is that the environmental dynamics, i.e. the reward and transition probabilities, need to be completely known. In our application, the rewards are defined in terms of either the time taken for the UAS to move from one position to the next based on a rotorcraft aerodynamic model or the energy consumption by the UAS during this transition. Both these models are described in section III. The corresponding transition probabilities are defined based on interaction of the UAS flight dynamics with wind magnitude and direction as described in section IV.A.

C. Reinforcement Learning based path planning

Despite the guarantee of convergence offered by the classical dynamic programming, these methods are often infeasible to implement in real path-planning applications. The major reason behind this is that more often than not, new states need to be added in our state-space system in order to be able to capture complex environmental dynamics or real-time changes in the agent-environment interactions. As an example, path-planning for an UAS is heavily dependent on its health state. Even if a trivial binary health index is added as an additional state in the problem formulation, it leads to an increase in the number of state transitions (s, a, s') by four times. Besides, dynamic programming requires complete knowledge of the transition probabilities for all state-action pair which may not be available for complex environments or different vehicle types. As a result, reinforcement learning approaches are explored that can avoid strict model assumptions as well as manage computation time [10]. The primary idea behind reinforcement learning is that the values for the cells in a grid are randomly initialized and then updated based on the values of its neighboring cells.

D. Deep Q-Learning

There are a variety of reinforcement learning techniques other than dynamic programming. Many of these techniques fall under the category of deep reinforcement learning, which includes algorithms that use deep neural networks to solve reinforcement learning problems [11]. One such algorithm is Deep Q-Learning, which works analogously to the original Q-learning algorithm by Watkins [12]. Deep Q-Learning employs its neural network, the Deep Q-Network (DQN), to approximate the optimal Q function, defined in Eq. (5).

$$Q^*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')] \tag{5}$$

The DQN is trained using supervised learning techniques based on sample state transitions from the MDP being solved. When trained properly and applied to an appropriate MDP, Deep Q-Learning has resulted in policies capable of unprecedented performance on tasks considered impossible for tabular solution methods [13] [14]. We use Deep Q-Learning as a point of comparison for our primary solution method, which is faster and results in a more optimal solution when applied to the MDP we describe. Despite the relative performance on this task, Deep Q-Learning and other deep reinforcement learning methods may prove better choices for the more complex and realistic versions of this problem that we intend to develop in future work.

III. Simulation set-up

A. UAS flight dynamics in wind

In this study, the UAS flight dynamics are simulated based on a simple 6 degrees-of-freedom rotorcraft model using Newton-Euler equations, where inertia is described by lumped masses and followed by a linear quadratic integrator (LQI) controller. Mass and geometrical properties of the UAV are defined through a lumped-mass system: a central body sphere with mass M and radius R, n point-masses and beams with mass m and length l, respectively. The mass of the arm is assumed to be concentrated at the rotor location, therefore m refers to the total mass of propeller and arm. The term M refers to the mass of the central body comprehensive of a payload. Mass properties (m_t) and inertia moments $(I_{x_b}, I_{y_b}, I_{z_b})$ are calculated based on the assumption of lumped-mass model. The modeling equations are described in detail in previous work by the same authors in [15].

The aerodynamic drag encountered by the UAS is embedded in the vehicle dynamic equations using the standard formula

$$D_{i_b} = -\frac{1}{2} C_{D,i_b} A_{i_b} \rho \, V_{r,i_b} |V_{r,i_b}| \ , \tag{6}$$

where C_{D,i_b} is the drag coefficient, A_{i_b} is the apparent face of the vehicle, ρ is the air density, and D_{i_b} is the drag force. Vector $\vec{V}_r$ is the vector of relative velocity between air and vehicle, and V_{r,i_b} represents a single component. The symbol $|\cdot|$ indicates the absolute value. The subscript i_b indicates the direction $i = x, y, z$ in the body reference frame, so this subscript takes on x_b, y_b and z_b. The drag forces D_{i_b} are then added to the acceleration terms of the Newton-Euler equations, as stated in Eq. (7). In our application, we implemented the dynamic model in an inertial East-North-Up (ENU) reference frame centered on the first way-point of the flight, therefore, drag forces D_{i_b} are converted into the Earth-fixed reference frame to obtain D_x, D_y, D_z to be added to $\ddot{x}$, $\ddot{y}$, $\ddot{z}$.

The complete dynamic model can be written in a compact form by combining linear position and velocity variables, $\boldsymbol{s}$ and $\dot{\boldsymbol{s}}$, as well as Euler angle position and velocity variables, $[\phi, \theta, \psi]$ and $[p, q, r]$, into the state vector $\boldsymbol{x} = [\boldsymbol{s}, \dot{\boldsymbol{s}}, \phi, \theta, \psi, p, q, r]^T$. Equation (7) shows the state-space formulation $\dot{\boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{u})$ of the model, where the terms $s.$, $c.$, and $t.$ indicate sine, cosine and tangent functions, respectively.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x} + \dot{w_x} \\ \dot{y} + \dot{w_y} \\ \dot{z} + \dot{w_y} \\ (s_\theta c_\psi c_\phi + s_\phi s_\psi)\frac{T}{m_t} - \frac{D_x}{m_t} \\ (s_\theta s_\psi c_\phi - s_\phi s_\psi)\frac{T}{m_t} - \frac{D_y}{m_t} \\ -g + c_\phi c_\theta \frac{T}{m_t} - \frac{D_z}{m_t} \\ p + q s_\phi t_\theta + r c_\phi t_\theta \\ q c_\phi - r s_\phi \\ q\frac{s_\phi}{c_\theta} + r\frac{c_\phi}{c_\theta} \\ \left(\frac{I_{y_b y_b} - I_{z_b z_b}}{I_{x_b x_b}} qr + \frac{l}{I_{x_b x_b}}\tau_\phi\right) \\ \left(\frac{I_{z_b z_b} - I_{x_b x_b}}{I_{y_b y_b}} pr + \frac{l}{I_{y_b y_b}}\tau_\theta\right) \\ \left(\frac{I_{x_b x_b} - I_{y_b y_b}}{I_{z_b z_b}} qr + \frac{l}{I_{z_b z_b}}\tau_\psi\right) \end{bmatrix} \tag{7}$$

It should be noted that wind information is incorporated in the UAV kinematics and thrust reduction in state space formulation of Eq. (7), assuming that the UAV exhibits spherical aerodynamic characteristics (i.e., constant drag coefficient in all directions) and that body lift and other nonlinear aerodynamic effects are negligible.

B. UAS energy consumption in wind

Once the linear and angular motion parameters are computed based on the above aerodynamic model, the trajectory simulator is followed up with a Linear Quadratic Regulator with Integrator (LQR-I) controller derived using the Hamiltonian matrix [15]. At each time step of the simulation, the LQR-I controller provides the estimated force $F = [T, M_x, M_y, M_z]$ required to transition the UAS from its current position to the next where $\mathbf{T}$ is the total thrust and

M_x, M_y, and M_z are the three moments along the three cardinal directions.

$$F = C\Omega^2 \tag{8}$$

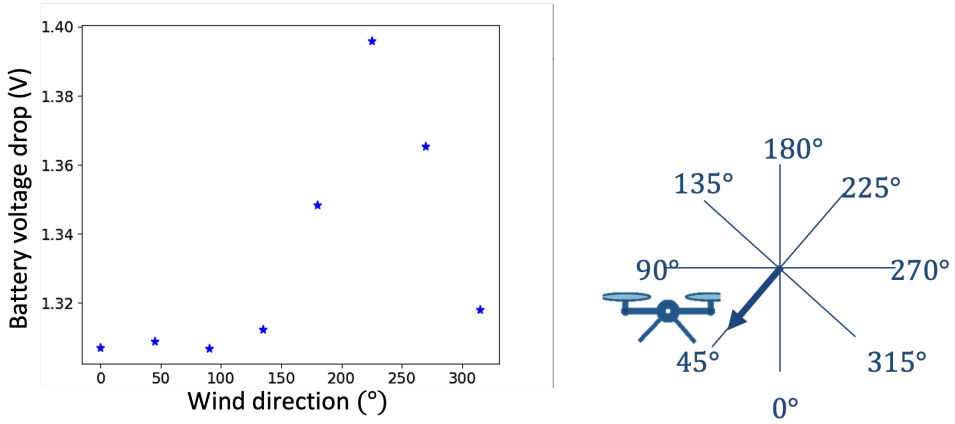
$$\Omega = [\Omega_1, \Omega_2, ..., \Omega_8] \tag{9}$$

Given the relationship of total force and the rotational speeds of each motor of the rotorcraft model as stated in Eq. 9, the approximation of the required rotational speed for each motor can be computed by inverting the control allocation matrix C mapping rotor speeds to vector forces and moments. Matrix C contains thrust and torque constants of the propellers and is dependent on the rotor geometry configuration of the vehicle. The relationship between rotor speed necessary to complete the flight and the corresponding current drawn from the motors can then be approximated using a polynomial fit of data from a motor manufacturer's data-sheet, which includes the effect of propellers on motor performance [2]. After computing the speed profile for all rotors by inverting the C matrix, we then compute the required current profiles for the individual motors to complete the mission using the polynomial fit.

Next, we used a surrogate battery model to compute the battery power consumption during the UAS flight with the current profile as an input. The surrogate battery model as developed in [2] is based on an simplified Li-ion battery model originally proposed in [16]. In that model, the surface overpotential ($V_{\eta,i}$) is derived from the Butler-Volmer equation, expressed as:

$$V_{\eta,i} = \frac{RT}{F\alpha} arcsinh(\frac{J_i}{2J_{i,0}}) \tag{10}$$

Using the above electrochemistry model, we determined that stronger winds and headwinds (wind in opposite direction to the heading angle) led to highest battery voltage drop. This is intuitive and have been further illustrated in figure 1, where the battery voltage drop is computed for a UAS flight with heading angle at 45^o and varying wind directions. The battery voltage drop was shown to be maximum for headwinds (wind direction at 225^o). Similar observations have been reported in a previous study on UAS flight characteristics under varying wind and battery conditions [2]. In order to integrate the energy consumption characteristics of the UAS, we used a surrogate model based on multi-linear regression that captures the dependency of Li-ion battery voltage drop on the wind magnitude and direction with respect to the heading angle of the UAS and integrated that in the reward definition of the MDP.



(a) (b)

Fig. 1 (a) Effect of wind direction on battery voltage drop. (b) UAV heading angle.

IV. Path-planning Results

A. MDP Parameter Settings

The problem is described as given two waypoints i,e. START and GOAL states, the objective is to determine the 'optimum' trajectory that a UAS should be following such that the time-to-goal is minimized, by exploiting the varying winds in the UAS airspace or environment. The environment is set up as a MDP. Dynamic Programming algorithms,

namely value iteration as defined in Eq. 3 is used to compute the optimal policy. In this paper, the UAS airspace is defined as a 7×7 grid with each cell length being $d = 400\ m$, as shown in Figure 2. Note that this and other similar figures in this paper are based on code from Robert Moss' notebook course on Decision Making under Uncertainty *. The red cells denote obstacles and the green cell denote the goal or target cell. The red arrows denote the wind magnitude and direction in each cell.

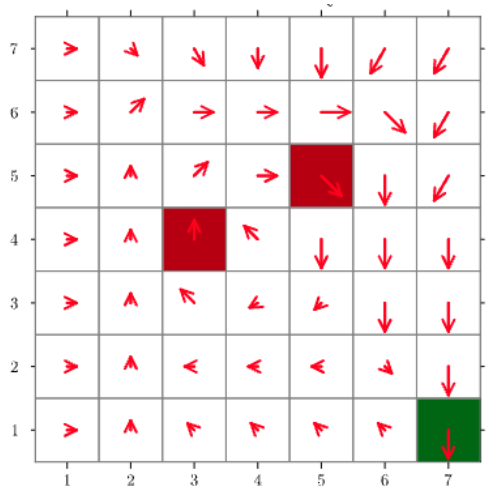


Fig. 2 Example grid-world representing UAS airspace in 2D.

- *States*: The input state space is defined as $S = \{x, y, \Psi\}$ where x and y are the horizontal positions in 2D map and Ψ is the heading angle of the UAS.
- *Actions*: In this study, we assume that the UAS can move in any of the eight directions to reach its neighboring cells i.e. $A = N, NE, E, SE, S, SW, W, NW$. An attempt to move in any of these directions will determine the heading angle associated with the successor state, where the corresponding heading angles are $\Psi = 0^o, 45^o, 90^o, 135^o, 180^0, 225^o, 270^o, 315^o$.
- *Transition Probability*: A critical parameter that defines the Markov Decision Process is the state to state transition probability given a certain action: $p(s'|s, a)$. Here, the transition probability is dictated by the uncertainty in wind field data. From the first three parameters in the UAS aerodynamic model given in Eq. (7), it can be seen that the resultant velocity of the UAS when it moves from one point to next is given by the sum of the two vectors: the commanded velocity $\dot{x}$ and the wind velocity $\dot{w}$. Therefore, if the UAS commanded velocity vector for a $cell_{i,j}$ is $V_{i,j}^{UAS}$ with heading angle $\Psi_{i,j}$ and the wind velocity vector is $W_{i,j}$ with wind direction as $\theta_{i,j}^{w}$, the resultant vector is described as:

$$V_x = V_{i,j}^{UAS} cos(\Psi_{i,j}) + W_{i,j} cos(\theta_{i,j}^{w}), \tag{11}$$

$$V_y = V_{i,j}^{UAS} sin(\Psi_{i,j}) + W_{i,j} sin(\theta_{i,j}^{w}), \tag{12}$$

$$V^{res} = \sqrt{{V_x}^2 + {V_y}^2}, \tag{13}$$

$$\omega^{res} = tan^{-1}(\frac{V_x}{V_y}). \tag{14}$$

The uncertainty in wind field is introduced as a Gaussian distribution around the resultant velocity vector, by setting ω^{res} as the mean value with standard deviation σ, in each cell. Higher the uncertainty in the wind data, higher is the value of σ that determines the probability with which the UAS will end up in one of the neighboring cells. If each of the eight possible directions is determined by a $\pi/4$ section, the transition probability is then represented by the area governed by the intersection between the Gaussian distribution curve and the range angle, as shown in Fig. 3 and given by Eq. (15).

*https://github.com/JuliaAcademy/Decision-Making-Under-Uncertainty

$$p(s'|s,a) = \frac{1}{\sigma\sqrt{2\pi}} \int_{\theta_a-\pi/8}^{\theta_a+\pi/8} e^{-0.5(\frac{v-\omega}{\sigma})^2} dv \tag{15}$$

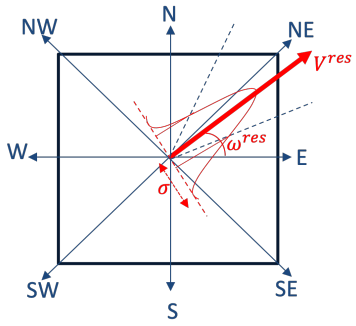


Fig. 3 Defining transition probabilities by setting resultant UAS velocity vector as mean of a Gaussian distribution with associated wind measurement uncertainty (σ).

- *Rewards*: For time-optimized trajectory, the rewards for each transition and action is set as the negative of the time taken for the UAS to move from one cell to next, with the velocity component along the direction of the specific action. Given the distance between two adjacent cells is denoted by C, the reward is defined as:

$$r(s_{i,j}) = -\frac{C}{V^{res} cos(\theta^w_{i,j} - \Psi_{i,j})}, \tag{16}$$

$$C = \begin{cases} 1\ d, & \text{if } A = \{N, E, S, W\} \\ \sqrt{2}\ d, & \text{if } A = \{NE, NW, SE, SW\} \end{cases} \tag{17}$$

  For energy-optimized trajectory, the reward of reaching a current cell is defined by the negative of the battery voltage drop as the UAS moves from its previous cell. The battery voltage drop is computed through the UAS simulation followed by the battery surrogate model as defined in section III.B.

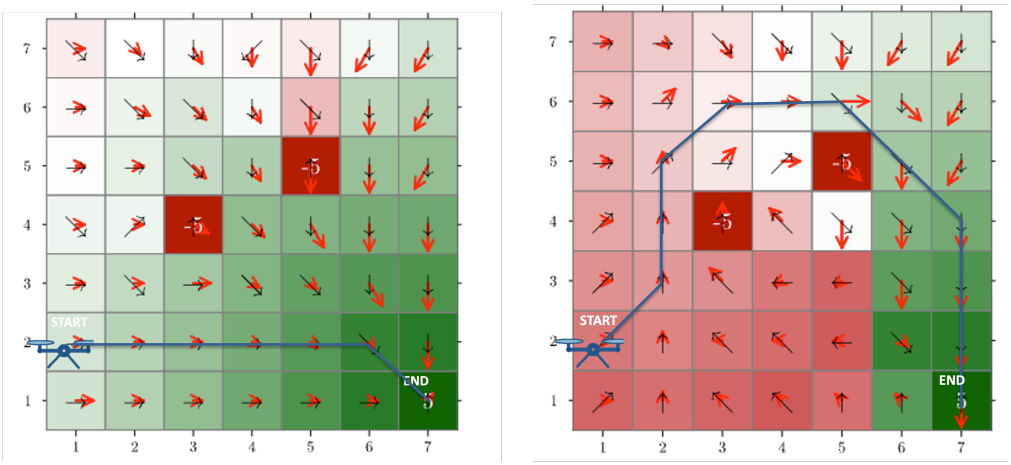
$$r(s_{i,j}) = -\delta V_{i,j} \tag{18}$$

B. Dynamic Programming Results

The dynamic programming is implemented using POMDPs.jl, an open-source framework for solving Markov decision processes (MDPs) and partially observable MDPs (POMDPs) [17]. Results from implementing the value iteration algorithm with the above parameter settings have been reported. As mentioned before, the airspace is depicted as a 7×7 grid. Figure 4 (a) and (b) shows the results based on least energy optimization for different wind conditions as represented by the red quiver plots. The red arrows depict the wind field magnitude and direction in each of the cells. The final values (expected rewards) computed for each cell are represented by the colors, red being the lowest value corresponding to -5 for the obstacle cells and green being the highest value corresponding to 5 for the target cell. The optimum policies generated based on the value iteration solver for each cell is stated by the black arrows corresponding to one out of the eight possible actions that the UAS may take when it reaches that cell. The optimum trajectory from a start cell at $[1, 2]$ to the goal cell $[7, 1]$ for the two wind scenarios are shown by the blue solid line. As observed, the optimum trajectory output changes according to the wind field. In both cases, the optimum trajectory maintains safe separation from the obstacles.

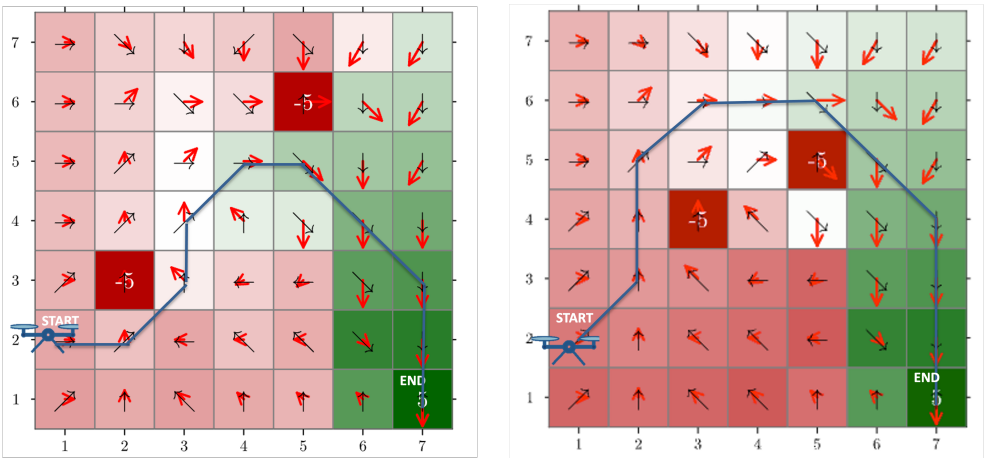
Obstacles play a critical role is defining UAS flight paths especially with wind conditions. Optimum trajectories under same wind conditions but different obstacle positions are stated in figure 5.

Next, the two plots in Figure 6 (a) and(b) compares the time-optimized and energy-optimized flight plans for the UAS, under the same wind and obstacle settings. As can be observed, the 'red' trajectory saves battery energy compared to the 'blue' even though it takes around $4min$ longer to reach the END cell. This is because the 'red' trajectory is obtained by exploiting the wind conditions such that the output path aligns with the wind direction for most of the times



(a) (b)

Fig. 4 Energy-optimized trajectory for UAS under different wind conditions (a) uniform wind field (b) circular wind field.



(a) (b)

Fig. 5 Energy-optimized trajectory for UAS with different obstacle locations.

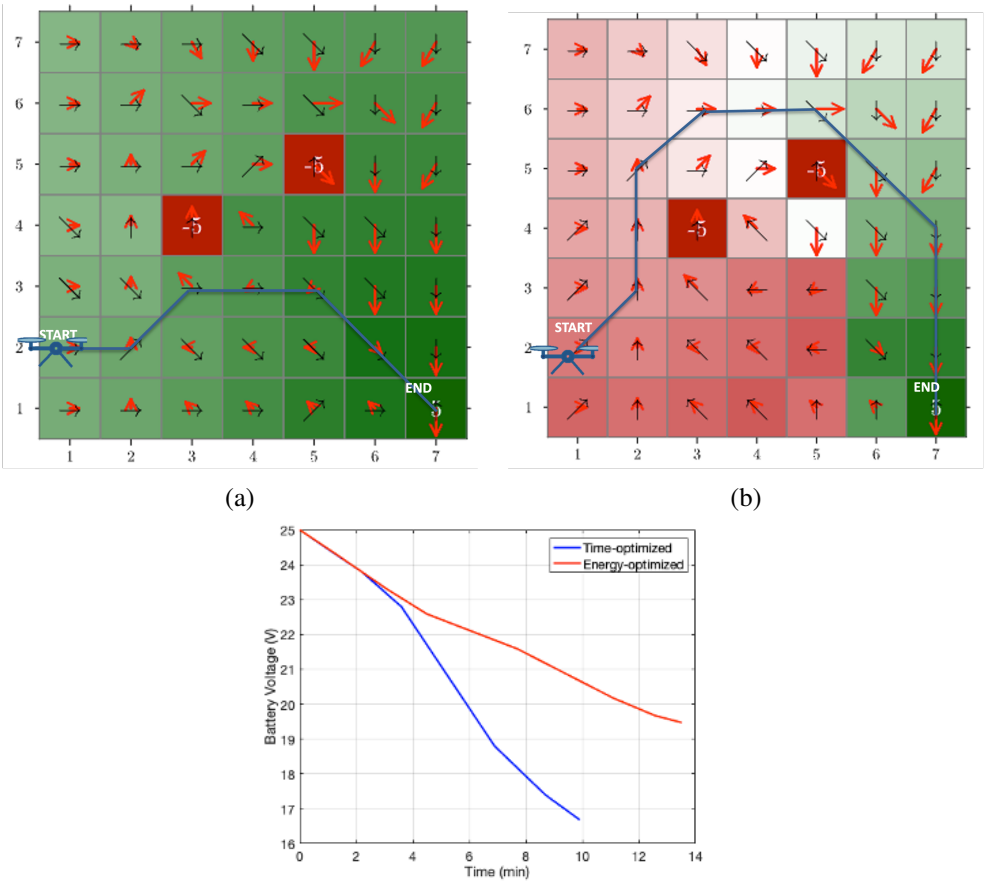
and hence saves significant battery charge. This is an important factor to be considered in missions where opting for energy efficiency may be more critical than reaching the END at the earliest. It should be noted that the increase in energy consumption specifically for turns in the trajectory or for reduced controllability in case of vehicle system faults have not been considered in this study and will be investigated more in upcoming studies.

C. Deep Q-Learning Results

The reinforcement learning tests were performed using an implementation of Deep Q-Learning from Crux, a Julia library for deep reinforcement learning [†]. Crux, in turn, is based on a more general-purpose machine learning library called Flux [18] [19], which was used directly to define the structure of the the neural network representing the policy. The network used here has a fairly simple structure. The network layers are:

1) A fully-connected layer with 2 inputs and 28 outputs, with a hardtanh (see definition below) activation function
2) A fully-connected layer with 28 inputs and 49 outputs, with a hardtanh activation function
3) A linear layer with 49 inputs and 8 outputs

[†]Crux is written by Anthony Corso and is available at https://github.com/sisl/Crux.jl



(a) (b)

(c)

Fig. 6 (a) Time-optimized trajectory and (b) Energy-optimized trajectory under same wind and obstacle conditions (c) Energy-time performance comparison for the two trajectories.

The hardtanh function, defined as $hardtanh(x) = max(-1, min(1, x))$ is a piecewise linear approximation of the hyperbolic tangent function.

Each layer described above was implemented using the fully connected 'Flux.Dense' layer type, and these layers were combined into a single neural network using 'Flux.Chain'. Each layer contained trainable weights and biases for each output, with weights initialized using Glorot Uniform initialization [20].

As a single unit, the network thus takes two inputs and maps them to eight outputs. The goal of Deep Q-Learning is to train this network to map (x, y) state coordinates to the optimal q values corresponding to taking each of the eight available actions from that state.

Training this network to the level shown in Figure 7 took 21 seconds, which is longer than it takes to solve this MDP with dynamic programming and tabular value functions. It also does not have the same convergence properties as our dynamic programming approach, as demonstrated by the brief downward spikes present in the training curve. Even so, we hope Deep Q-Learning and other deep reinforcement learning algorithms will be useful for future updates to this work, as such algorithms often generalize well enough to handle MDPs with larger state-spaces and more complex dynamics than we consider in our example. For our purpose, this may allow for the solution of variations of this problem with much larger maps, time-varying winds dependent on underlying terrain characteristics, and other modifications to bring the problem closer to a real application.

The resulting policy is visualized in Figure 8. The black arrows give the policy's choice of movement direction at each location, and the color of each cell gives the estimated value of each state.

As the DQN was trained from observations of an ϵ-greedy policy, differences in policies and value estimates between the DQN results and the dynamic programming results can be partly explained by the fact that an ϵ-greedy policy

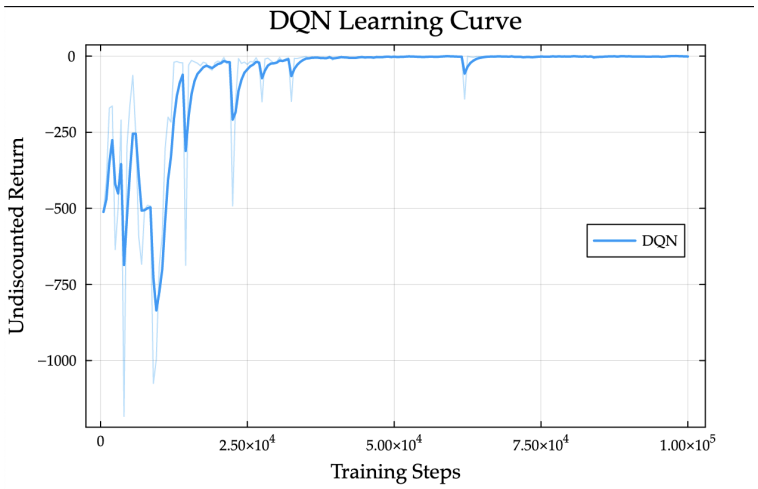


Fig. 7 Learning curve for the Deep-Q Network

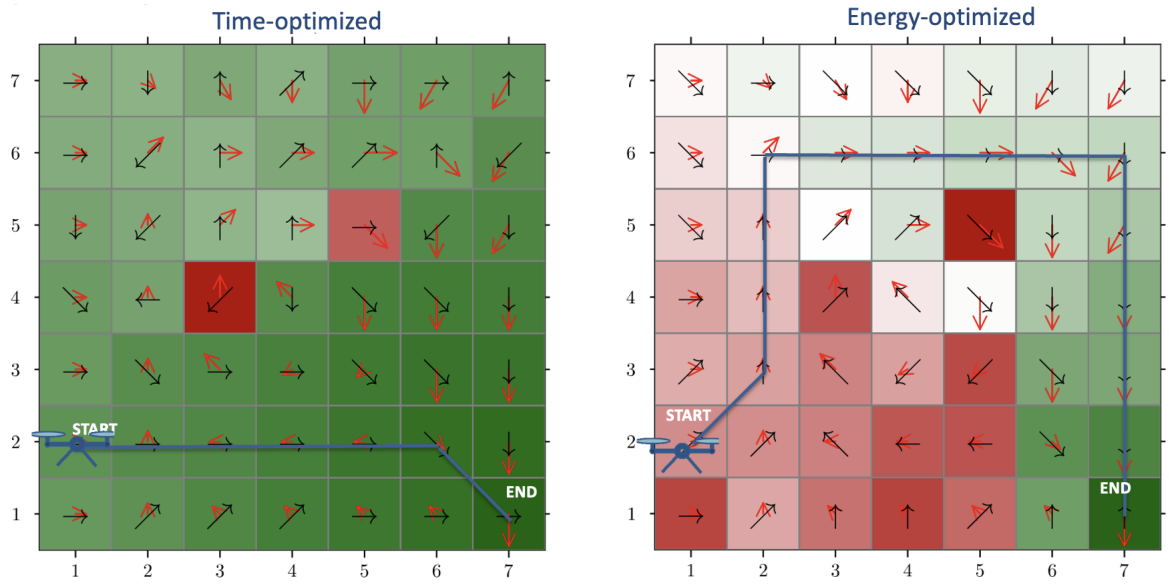


Fig. 8 Time-optimized and Energy-optimized DQN policies and resulting paths

does not spend much of its time exploring parts of the state space that are not likely to appear on optimal paths to the goal. This may not be acceptable for real applications of this work to flight hardware, and hence the robustness of deep-reinforcement learning solutions will be verified across the entire reachable state space in future work.

V. Conclusion

This paper presents a path planning approach for UASs under varying wind conditions. Previous work on this topic has been reviewed and gaps have been identified. The problem has been defined as a Markov decision process and reinforcement learning algorithms have been implemented to solve the optimization problem. The aerodynamics of a rotor craft have been introduced within the trajectory simulation to compute the time and energy consumption. Initial results from the algorithm are presented and compared under varying environmental conditions and mission objectives. For example, in a time-sensitive mission such as a UAS employed for medical supply delivery may resort to

time-optimized decision-making, whereas for a mission that demands longer flight time covering larger monitoring area such as in a surveillance operation, may benefit from using the energy-optimized decision-making, both of which ensures safety assurance at all times. Results from dynamic programming and deep reinforcement learning have been explored and compared on an example use-case. Potential advantages of using deep Q-learning techniques for larger and more complex system dynamics have been discussed. In upcoming studies, the effect of system faults and degradation on the vehicle dynamics and power consumption will be explored and integrated into the MDP models to enhance its capabilities for in-flight decision-making under varying flight conditions.

References

[1] Banerjee, P., Corbetta, M., Jarvis, K., Smalling, K., and Turner, A., "Probability of Trajectory Deviation of Unmanned Aerial Vehicle in Presence of Wind," *Journal of Air Transportation*, 2023, pp. 1–12.

[2] Pohya, A. A., Wende, G., Corbetta, M., and Kulkarni, C. S., "Comparison of Surrogate Modeling Techniques for Life Cycle Models of Advanced Air Mobility," *AIAA AVIATION 2023 Forum*, 2023, p. 3857.

[3] Tang, L., Wang, H., Li, P., and Wang, Y., "Real-time trajectory generation for quadrotors using b-spline based non-uniform kinodynamic search," *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2019, pp. 1133–1138.

[4] Garcia, M., Viguria, A., and Ollero, A., "Dynamic graph-search algorithm for global path planning in presence of hazardous weather," *Journal of Intelligent & Robotic Systems*, Vol. 69, No. 1-4, 2013, pp. 285–295.

[5] Babel, L., "Flight path planning for unmanned aerial vehicles with landmark-based visual navigation," *Robotics and Autonomous Systems*, Vol. 62, No. 2, 2014, pp. 142–150.

[6] Sajid, M., Mittal, H., Pare, S., and Prasad, M., "Routing and scheduling optimization for UAV assisted delivery system: A hybrid approach," *Applied Soft Computing*, Vol. 126, 2022, p. 109225.

[7] Chakrabarty, A., Stepanyan, V., Krishnakumar, K. S., and Ippolito, C. A., "Real-time path planning for multi-copters flying in UTM-TCL4," *AIAA Scitech 2019 forum*, 2019, p. 0958.

[8] Al-Sabban, W. H., Gonzalez, L. F., and Smith, R. N., "Wind-energy based path planning for unmanned aerial vehicles using markov decision processes," *2013 IEEE International conference on robotics and automation*, IEEE, 2013, pp. 784–789.

[9] Puterman, M. L., "Markov decision processes," *Handbooks in operations research and management science*, Vol. 2, 1990, pp. 331–434.

[10] Sutton, R. S., Barto, A. G., et al., "Reinforcement learning," *Journal of Cognitive Neuroscience*, Vol. 11, No. 1, 1999, pp. 126–134.

[11] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A., "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[12] Watkins, C. J., and Dayan, P., "Q-learning," *Machine learning*, Vol. 8, 1992, pp. 279–292.

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A., "Playing Atari with Deep Reinforcement Learning," *CoRR*, Vol. abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

[14] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D., "Rainbow: Combining improvements in deep reinforcement learning," *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32, 2018.

[15] Corbetta, M., Banerjee, P., Okolo, W., Gorospe, G., and Luchinsky, D. G., "Real-time uav trajectory prediction for safety monitoring in low-altitude airspace," *Aiaa aviation 2019 forum*, 2019, p. 3514.

[16] Daigle, M., and Kulkarni, C. S., "Electrochemistry-based battery modeling for prognostics," *Annual conference of the PHM society*, Vol. 5, 2013.

[17] Egorov, M., Sunberg, Z. N., Balaban, E., Wheeler, T. A., Gupta, J. K., and Kochenderfer, M. J., "POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty," *Journal of Machine Learning Research*, Vol. 18, No. 26, 2017, pp. 1–5. URL `http://jmlr.org/papers/v18/16-300.html`.

[18] Innes, M., Saba, E., Fischer, K., Gandhi, D., Rudilosso, M. C., Joy, N. M., Karmali, T., Pal, A., and Shah, V., "Fashionable Modelling with Flux," *CoRR*, Vol. abs/1811.01457, 2018. URL `https://arxiv.org/abs/1811.01457`.

[19] Innes, M., "Flux: Elegant Machine Learning with Julia," *Journal of Open Source Software*, 2018. https://doi.org/10.21105/joss.00602.

[20] Glorot, X., and Bengio, Y., "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.