REINFORCEMENT LEARNING APPLICATIONS IN UNMANNED VEHICLE CONTROL: A COMPREHENSIVE OVERVIEW

Hao Liu, Bahare Kiumarsi, Yusuf Kartal, Ahmet Taha Koru, Hamidreza Modares, Frank L. Lewis

ABSTRACT

This paper briefly reviews the dynamics and the control architectures of unmanned vehicles; reinforcement learning in optimal control theory; and RL-based applications in unmanned vehicles. Nonlinearities and uncertainties in the dynamics of unmanned vehicles (e.g., aerial, underwater, and tailsitter vehicles) pose critical challenges to their control systems. Solving Hamilton-Jacobi-Bellman (HJB) equations to find optimal controllers becomes difficult in the presence of nonlinearities, uncertainties, and actuator faults. Therefore, reinforcement learning (RL)-based approaches are widely used in unmanned vehicle systems to solve the HJB equations. To this end, they learn the optimal solutions by using online data measured along the system trajectories. This approach is very practical in partially or completely model-free optimal control design and optimal fault-tolerant control design for unmanned vehicle systems.

Keywords Unmanned vehicles · Reinforcement learning · Optimal control · Uncertainty.

Note: Electronic version of an article published in *Unmanned Systems* with a DOI: 10.1142/S2301385023310027

1 Introduction

Most dynamic systems in the real world, including unmanned vehicles, are inherently nonlinear. Finding the optimal solution for nonlinear systems requires solving a nonlinear partial differential equation, namely, Hamilton–Jacobi–Bellman (HJB) equation. Explicitly solving the HJB equation is generally very difficult or even impossible. Reinforcement learning (RL) is one of the mostly employed techniques to approximate the HJB solution. A variety of RL algorithms can learn the HJB solution and thus the optimal controller by utilizing the data collected from the system's interaction with its environment, even if complete knowledge about the system dynamics is not available.

RL is a large field, and researchers from different backgrounds, including psychology, computer science, economic, control, robotics, and so on, contribute to it [1–7]. RL algorithms have received attraction in the control community by the work of [8–10]. RL algorithms aim at learning a policy, which is a map from the states to actions, based on the feedback received from the interaction with the environment. Two common and different categories of RL algorithms for implementation of learning from interaction are policy search (e.g., policy gradient) and actor-critic approaches [2]. Policy search methods directly parametrize the controllers and leverage gradient-like update laws to directly learn the optimal set of control parameters. While this approach bypasses the assignment of value to policies, there is no general way to solve the resulting optimization problem for nonlinear systems and to avoid getting traps in a local minimum. Actor-critic-based methods, which are implemented using either policy iteration or value iteration approaches, on the other hand, iteratively update control policies by learning their value functions in a policy evaluation step and improving them accordingly in a policy improvement step. The algorithms converge to the optimal solutions by iterating on these two steps. RL algorithms in the control context have mainly been used to solve optimal regulation and tracking of single-agent and multi-agent systems.

In terms of data usage, RL algorithms are categorized into two classes: on-policy and off-policy methods [2]. On-policy learning algorithms evaluate and improve the same policy as the one that is applied to the system. The data used for learning about the value of a policy must be generated from applying the same policy to the system, and past experiences collected from applying other policies to the system are of no use. On the other hand, to improve data efficiency, off-policy learning algorithms evaluate new policies using past collected data generated from different policies than the

one applied to the system. Additionally, this paper reviews robust off-policy RL methods that solves corresponding robust off-policy integral RL Bellman equations to deal with unmatched uncertainties of the unmanned vehicle system dynamics. However, for a better understanding of RL-based control of unmanned vehicles, the dynamic models and control architectures of unmanned quadrotor vehicles, unmanned underwater vehicles, and unmanned tailsitter vehicles should be overviewed. Further, their operating conditions and how to apply RL-based control design on their complex structure need to be reviewed. These will be done later in the manuscript.

The RL applications on unmanned vehicles include but are not limited to performing autonomous navigation in large-scale complex environments [11–15]. Particularly, [15] uses a decentralized collision avoidance policy by employing a reinforcement learning technique that leverages local neighbors' data to accomplish robust multiple unmanned vehicle motion planning by considering limited sensing capabilities of unmanned vehicles. Another field of RL applications for unmanned vehicles is multi-agent learning [16–22]. Learning to achieve coordination among multiple unmanned vehicles is hard to accomplish because of vehicular mobility. This problem is addressed in [18] and solved by constituting a dynamic coordination graph to model the switching topology during vehicles' interactions that work with RL approaches. Further, [21] uses multi-agent reinforcement learning to coordinate multiple UAVs performing real-time target tracking that achieves fast localization and tracking in the highly dynamic channel environment by formulating a flight decision problem as a constrained Markov decision process.

Recently, unmanned systems [23–25] received a great interest due to their capability to carry out various missions including package delivery, agricultural operations, surveillance, and so on. Therefore, one of the applications of RL in control is to learn the optimal solution for such [26–31]. Three recognized categories of RL applications in unmanned vehicle control are unknown parameters learning, unmatched uncertainties learning, and robust learning. In this manuscript, these learning categories will be addressed by reviewing integral reinforcement learning technique that is developed in continuous time, and hence more applicable to the RL-based control of unmanned vehicles.

This paper first overviews unmanned vehicle dynamics that include unmanned quadrotor vehicles, unmanned underwater vehicles, and unmanned tailsitter vehicles. Then, it recalls some on-policy and off-policy actor-critic based RL algorithms in solving the optimal control problem of unmanned systems. Eventually, it overviews how to apply these RL algorithms on unmanned vehicles that are operating in different conditions by reviewing linearization procedures and fault-tolerant control design analyses.

Organization of the paper: The remainder of this paper is organized as follows. Section 2 presents the unmanned vehicles model and the control architecture. Section 3 summarizes the application of RL in optimal control design. An overview of how to solve the optimal control of unmanned vehicles using RL algorithms is described in Section 4. Additionally, the linearization procedures of unmanned quadrotor systems and the nonstandard backstepping control technique that is compatible with the integral RL technique are revealed. Section 5 discusses the application of RL algorithms in fault-tolerant control for unmanned systems. Concluding remarks are in Section 6.

2 Vehicle Modeling and Control Architecture

This section reviews the dynamic models and control architectures of unmanned quadrotor, underwater, and tailsitter vehicles for a better understanding of RL-based control design in the following sections.

2.1 Vehicle Dynamics

To describe the position and heading of the vehicles, let $E = \{\vec{e}_x, \vec{e}_y, \vec{e}_z\}$ be the earth-fixed frame (i.e., inertial frame) and $B = \{\vec{b}_x, \vec{b}_y, \vec{b}_z\}$ be the body-fixed frame. The vehicle position with respect to E is $p = [p_x, p_y, p_z]^T \in \mathbb{R}^{3\times 1}$. The attitude of the vehicle is given by Euler angles $\Theta = [\phi, \theta, \psi]^T \in \mathbb{R}^{3\times 1}$ where ϕ indicates the roll angle, θ is the pitch angle, and ψ is the yaw angle. We use Z-Y-X Euler angles for which the rotation matrix $R_{BE} \in \mathrm{SO}(3)$ from B to E is given by

$$R_{BE} = \begin{bmatrix} \mathrm{c}\theta\,\mathrm{c}\psi & \mathrm{c}\psi\,\mathrm{s}\phi\,\mathrm{s}\theta - \mathrm{c}\theta\mathrm{s}\psi & \mathrm{s}\phi\,\mathrm{s}\psi + \mathrm{c}\phi\,\mathrm{c}\psi\,\mathrm{s}\theta \\ \mathrm{c}\theta\,\mathrm{s}\psi & \mathrm{c}\phi\,\mathrm{c}\psi + \mathrm{s}\phi\,\mathrm{s}\theta\,\mathrm{s}\psi & \mathrm{c}\phi\,\mathrm{s}\theta\,\mathrm{s}\psi - \mathrm{c}\psi\,\mathrm{s}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\,\mathrm{s}\phi & \mathrm{c}\phi\,\mathrm{c}\theta \end{bmatrix},$$

where $\mathrm{c}(\cdot)$ and $\mathrm{s}(\cdot)$ stand for cosine and sine functions, respectively. The angular velocity in B is $\omega = [\omega_x, \omega_y, \omega_z]^T \in \mathbb{R}^{3\times 1}$. The kinematic expression

$$\omega = \begin{bmatrix} 1 & 0 & -\mathrm{s}\theta \\ 0 & \mathrm{c}\theta & \mathrm{c}\theta\mathrm{s}\psi \\ 0 & -\mathrm{s}\psi & \mathrm{c}\theta\mathrm{c}\psi \end{bmatrix} \dot{\Theta}$$

states the relationship between ω and the Euler rates $\dot{\Theta}$ with respect to the rotation matrix R_{BE}.

Quadrotor Dynamics. Similar to [32–36], an unmanned quadrotor vehicle can be modeled as

$$m\ddot{p} = R_{BE}F_B - mg\vec{e}_3, \tag{1a}$$

$$I_B\ddot{\Theta} = -C(\Theta, \dot{\Theta})\dot{\Theta} + \tau_B, \tag{1b}$$

where m is the quadrotor mass, g denotes the gravity constant, $\vec{e}_3 = [0, 0, 1]^T$, and $C(\Theta, \dot{\Theta}) \in \mathbb{R}^{3\times3}$ is the Coriolis matrix shown in [35]. The force $F_B \in \mathbb{R}^{3\times1}$ relative to B is given by

$$F_B = [0, 0, f_T]^T,$$

where $f_T \in \mathbb{R}$ is the total lift. The total lift f_T and torque relative to B $\tau_B = [\tau_x, \tau_y, \tau_z]^T \in \mathbb{R}^{3\times1}$ are the control inputs. They are generated by the four rotor - propeller combinations and given by

$$f_T = k_f \sum_{j=1}^{4} \omega_{mj}^2, \tag{2a}$$

$$\tau_x = dk_f(\omega_{m2}^2 - \omega_{m4}^2), \tag{2b}$$

$$\tau_y = dk_f(\omega_{m1}^2 - \omega_{m3}^2), \tag{2c}$$

$$\tau_z = k_\tau \sum_{j=1}^{4} (-1)^{j+1}\omega_{mj}^2, \tag{2d}$$

in the plus configuration, where $\omega_{m1}, \ldots, \omega_{m4}$ are the rotational velocity of rotors, k_f is the lift coefficient, k_τ is the drag coefficient, and d is the arm length.

Underwater Vehicle Dynamics. Let $\eta_r = \left[v_r^T, \omega^T, r_{px}\right]^T \in \mathbb{R}^7$ where V_f is the current velocity with respect to E, $v_r = R_{BE}^T(\dot{p} - V_f)$ is the vehicle velocity relative to the current, and r_{px} represent the position of the moving mass in $\vec{e}_x$ of B. Then, the unmanned underwater vehicle can be modeled as

$$M\dot{\eta}_r = -\begin{bmatrix} S(\omega) & 0 & 0 \\ S(v_r) & S(\omega) & 0 \\ 0 & 0 & 0 \end{bmatrix} M\eta_r - M\eta_r + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{2}v_r^T \frac{\partial M_p(r_p)}{\partial r_{px}} v_r \end{bmatrix} + \begin{bmatrix} F_u \\ \tau_u \\ F_p \end{bmatrix}, \tag{3}$$

where m_p is the moving mass; $M_p(r_p)$ is the generalized inertia matrix for the moving mass; M is the generalized inertia matrix consisting m, I_B, and $M_p(r_p)$; F_u contains the inputs and the gravitational forces acting on the vehicle in B; τ_u is the input torques in B; and F_p contains the inputs and the gravitational forces acting on the particle in B. The function $S(x)$ constructs a skew-symmetric matrix from a vector $x = [x_1, x_2, x_3]^T$ as

$$S(x) = \begin{bmatrix} 0 & x_3 & -x_2 \\ -x_3 & 0 & x_1 \\ x_2 & -x_1 & 0 \end{bmatrix}.$$

More details about the underwater vehicle model is available in [24].

Tailsitter Dynamics. The dynamic model of the unmanned tailsitter vehicle is given by [31]

$$m\ddot{p} = R_{BE}F_B - mg\vec{e}_3, \tag{4a}$$

$$I_B\dot{\omega} = -\omega \times I_B\omega + \tau_B, \tag{4b}$$

where $F_B = [F_x, 0, F_z]^T \in \mathbb{R}^3$ is the force in B and $\tau_B = [\tau_x, \tau_y, \tau_z]^T$ is the torque in B. The force and torque generation adjusted by motors and elevons as

$$\begin{aligned} F_x &= F_{x,1}(|\dot{p}|, \omega_{m1}) + F_{x,2}(|\dot{p}|, \omega_{m2}), \\ F_z &= F_{z,1}(|\dot{p}|, \omega_{m1}, \delta_1) + F_{z,2}(|\dot{p}|, \omega_{m2}, \delta_2), \\ M_x &= dF_{z,1}(|\dot{p}|, \omega_{m1}, \delta_1) - dF_{z,2}(|\dot{p}|, \omega_{m2}, \delta_2), \\ M_y &= M_{y1}(|\dot{p}|, \omega_{m1}, \delta_1) + M_{y2}(|\dot{p}|, \omega_{m2}, \delta_2), \\ M_z &= dF_{x,1}(|\dot{p}|, \omega_{m1}) - dF_{x,2}(|\dot{p}|, \omega_{m2}), \end{aligned}$$

where ω_{m1} and ω_{m2} are the the rotational velocities of the rotors, δ_1 and δ_2 are the elevon deflections, and d is the perpendicular distance to each motor from the center of gravity of the vehicle. Explicit expressions for the functions $F_{x,1}$, $F_{x,2}$, $F_{z,1}$, $F_{z,2}$, $M_{y,1}$, and $M_{y,2}$ are available in [31].

These vehicles are underactuated with six degrees of freedom but limited numbers of actuators. Therefore, a well-established controller is needed to deal with these problems.

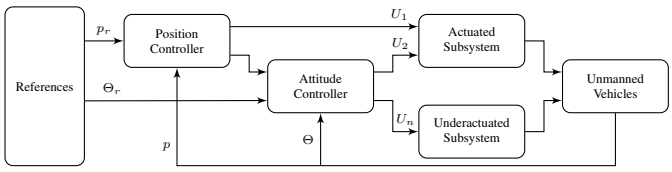

Figure 1: General control architecture of unmanned vehicle control system.

2.2 Challenges of Controlling Unmanned Vehicles

The controller for the unmanned vehicle is normally designed to achieve the position or attitude trajectory tracking. Practically, having a more accurate and complete vehicle model results in a more desirable controller for the traditional model-based control methods (see, [37]). However, a complete and accurate unmanned vehicle model is difficult to obtain due to the complex external environment and the presence of uncertainties. Furthermore, the nonlinearities and underactuation in the vehicle dynamics make it difficult, if not impossible, to obtain optimal control policy for unmanned vehicles via traditional optimal methods that includes linear quadratic regulators, linear quadratic trackers and H_∞ control methods [3]. To achieve the trajectory tracking control task, the unmanned vehicles are required to interact with the external environment to automatically find an optimal control policy without the requirement of their dynamics. The next section gives a general control architecture for unmanned vehicles.

2.3 Control Architecture

Let $p^r = \left[p_x^r, p_y^r, p_z^r\right]^T \in \mathbb{R}^{3\times 1}$ be the position reference, $\Theta^r = [\phi^r, \theta^r, \psi^r]^T \in \mathbb{R}^{3\times 1}$ be the attitude reference, and $U_1, \ldots, U_n$ be the control inputs of the unmanned vehicle. The general control architecture of the unmanned vehicles, which covers most of the studies, is shown in Fig. 1. Note that the attitude reference is not needed to get the unmanned vehicle to follow the reference position, which will be detailed later in Section 4. On the other hand, it is important to realize that the attitude controller needs to be designed compatible with the position controller as shown in Fig. 1.

As shown in Fig. 1, the control architecture of unmanned vehicles includes typically a position controller to achieve translational trajectory tracking and an attitude controller to achieve rotational trajectory tracking. For the underactuated unmanned vehicles, the vehicles are required to regulate their attitudes using an attitude controller to generate the desired control input for the position controller to track the desired position trajectory. In practical applications, the update time of the attitude controller is set to be much smaller than the position controller.

3 Reinforcement Learning Applications in Optimal Control Design

This section presents a comprehensive overview of different RL methods. We particularly overview on-policy, off-policy and robust off-policy RL algorithms to deal with unknown system parameters and unmatched dynamic uncertainties of general nonlinear time-invariant system.

Consider the nonlinear time-invariant system governed by the following equations

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \tag{5a}$$
$$y(t) = l(x(t)), \tag{5b}$$

where $x(t) \in \mathbb{R}^n$ represents the measurable state of the system, $u(t) \in \mathbb{R}^m$ is the control input, and $y(t) \in \mathbb{R}^p$ represents the output of the system, respectively. It is assumed that $f(0) = 0$ and $f(x(t)) + g(x(t))u(t)$ is locally Lipschitz, and the system is stabilizable.

The goal in the optimal control problem is to design an optimal control input to ensure the states of the system (5) converge to zero by minimizing the following cost function

$$J(x(0), u) = \int_0^\infty r(x, u)dt = \int_0^\infty \left(Q(x) + u^T R u\right) dt,$$

where $Q(x) \succeq 0$ and $R \succ 0$. The value function for the admissible control policy can be defined as

$$V(x, u) = \int_t^\infty r(x, u)d\tau = \int_t^\infty \left(Q(x) + u^T R u\right) d\tau.$$

A differential equivalent to this is

$$r(x,u) + \frac{\partial V^T}{\partial x}\left(f(x) + g(x)u\right) = 0, \quad V(0) = 0,$$

and the Hamiltonian is given by

$$H\left(x, u, \frac{\partial V^T}{\partial x}\right) = r(x,u) + \frac{\partial V^T}{\partial x}\left(f(x) + g(x)u\right).$$

The Bellman optimality equation gives the optimal value

$$r(x,u^*) + \frac{\partial V^{*T}}{\partial x}\left(f(x) + g(x)u^*\right) = 0, \tag{6}$$

which is just the Hamiton-Jacobi-Bellman (HJB) equation. The optimal control is then given as

$$u^*(t) = \arg\min_u r(x,u) + \frac{\partial V^{*T}}{\partial x}\left(f(x) + g(x)u\right) = -\frac{1}{2}R^{-1}g^T(x)\frac{\partial V^*}{\partial x}. \tag{7}$$

To find the optimal control (7), the HJB equation (6) needs to be solved first. However, solving the HJB equation analytically is difficult or even impossible. Therefore, reinforcement learning algorithms are presented to approximate HJB solution by iterating on the Bellman equations. This part discusses three algorithms: on-policy integral RL (IRL), off-policy IRL, and robust off-policy IRL.

3.1 On-policy IRL

The Bellman equation that does not involve the dynamics can be written as

$$V(x(t)) = \int_t^{t+T} \left(Q(x(\tau)) + u^T(\tau)Ru(\tau)\right) d\tau + V(x(t+T)) \tag{8}$$

This equation is called IRL Bellman equation and proposed in [38]. The on-policy RL algorithm can be implemented by iterating on the above IRL Bellman equation and updating the control policy.

Algorithm 1: On-policy IRL algorithm to find the solution of HJB

1. Set an admissible policy u_0

2. Solve for the value $V_i(x)$ using the Bellman equation

$$V_i(x(t)) = \int_t^{t+T} \left(Q(x(\tau)) + u_i^T(\tau)Ru_i(\tau)\right) d\tau + V_i(x(t+T))$$

for $i = 0, 1, \ldots$. On convergence, set $V_{i+1}(x) = V_i(x)$.

3. Update the control policy $u_{i+1}(t)$ using

$$u_{i+1}(t) = -\frac{1}{2}R^{-1}g^T(x)\frac{\partial V_i(x)}{\partial x}. \tag{9}$$

4. Stop if a stopping criterion is met. Otherwise, set $i = i + 1$ and go to step 2.

The IRL Algorithm 1 is on-line and does not require the knowledge of the drift dynamics. The value function and control inputs are updated sequentially in Algorithm 1. For a linear time-invariant (LTI) system, the algorithm converges to the optimal control law, which is determined by the algebraic Riccati equation [38], $V_i(x) \to V^*(x)$ as $i \to \infty$. Section 4 discusses this case with an example.

In general, it is hard to determine the optimal control law analytically for the nonlinear systems unlike the LTI case. A neural network structure approximates the value function $V_i(x)$ as

$$\hat{V}_i(x) = \sum_{j=1}^{L} w_{ij}\bar{\phi}_j(x) = W_{1i}^T\bar{\phi}(x), \tag{10}$$

for partially unknown nonlinear systems in [39], where L is the number of neurons on the hidden layer, $w_{ij} \in \mathbb{R}$ is the output layer weight, and $\bar{\phi}_j : \mathbb{R}^n \to \mathbb{R}$ is the basis function to compute the output of j^{th} hidden neuron, $W_{1i} \in \mathbb{R}^L$ the vector form the output layer weights, and $\bar{\phi} : \mathbb{R}^n \to \mathbb{R}^L$ is vector form of the basis functions. For this case, the update rule in (9) becomes

$$u_{i+1}(t) = -\frac{1}{2} R^{-1} g^T(x) W_{1i}^T \frac{\partial \bar{\phi}(x)}{\partial x}, \tag{11}$$

from (10). Similarly, the sequence in Algorithm 1 converge to the best estimate for the value function as $\bar{V}_i(x) \to \bar{V}^*(x) = W_1^T \bar{\phi}(x)$.

3.1.1 On-policy IRL Algorithm 1 with Experience Replay Learning Technique

In the experience replay approach [40–42], past experiences can be stored in the memory and reused for learning. This can improve the convergence of the IRL algorithm. A similar condition was proposed in [43] for adaptive control systems. Therefore, the gradient-decent update laws are obtained by minimizing not only the instantaneous error but also the errors for the stored data. Assume that the value function V(x) can be uniformly approximated as (10) in the IRL Bellman equation (8). Therefore, the approximate IRL Bellman equation becomes

$$e(t) = \hat{W}_1^T(t) \Delta\bar{\phi}(t) + \int_{t-T}^{t} \left(Q(x(\tau)) + u^T(\tau) R u(\tau) \right) d\tau,$$

where $\hat{W}_1(t) \in \mathbb{R}^L$ is the current estimation of W_1, $\Delta\bar{\phi}(t) = \bar{\phi}(x(t)) - \bar{\phi}(x(t-T))$ and $e(t) \in \mathbb{R}$ is the temporal difference error after using current value function estimator weights. Consider that the Bellman equation errors recorded in a history stack with a time sequence t_i. Now, the experience replay based gradient-descent algorithm for the value function estimator neural network is now given by

$$\dot{\hat{W}}_1(t) = -\alpha_1 \frac{\Delta\bar{\phi}(t)}{\left(1 + \Delta\bar{\phi}^T(t)\Delta\bar{\phi}(t)\right)^2} e(t) - \alpha_1 \sum_{i=K}^{l} \frac{\Delta\bar{\phi}(t_i)}{\left(1 + \Delta\bar{\phi}^T(t_i)\Delta\bar{\phi}(t_i)\right)^2} e(t_i)$$

where index i denotes the i^{th} sample data ($i = 1, \ldots, K$) stored in the history stack. As can be seen in the above equation, the first term is a gradient update law corresponding to Bellman error at time t, and the last term corresponds to errors for the stored samples in the history stack.

3.2 Off-policy IRL

Algorithm 1 requires the knowledge of input dynamics in policy improvement step. The off-policy IRL algorithm is presented in [44, 45] that solves the HIB equation (6) iteratively without knowing the system dynamics. To do so, the system dynamics (5) is rewritten as

$$\dot{x}(t) = f(x(t)) + g(x(t))u_i(t) + g(x(t))(u(t) - u_i(t)), \tag{12}$$

where $u_i(t)$ is the policy to be updated, and $u(t)$ is the behavior policy applied to the system dynamics to collect the data for learning the target policy. By differentiating $V(x)$ along with the system dynamics (12) and using the update rule in (9), one has

$$\dot{V}_i(x) = \frac{\partial V_i^T(x)}{\partial x}(f + g u_i) + \frac{\partial V_i^T(x)}{\partial x} g(u - u_i) = -Q(x) - u_i^T R u_i - 2u_{i+1}^T R(u - u_i). \tag{13}$$

The off-policy IRL Bellman equation is obtained by integrating from both sides of (13) as

$$V_i(x(t+T)) - V_i(x(t)) = \int_{t}^{t+T} \left(-Q(x) - u_i^T R u_i - 2u_{i+1}^T R(u - u_i) \right) d\tau \tag{14}$$

The following algorithm uses the off-policy IRL Bellman equation (14) to iteratively solve the HJB equation without requiring any knowledge of the system dynamics.

Algorithm 2: Off-policy IRL algorithm to find the solution of HJB

1. Apply a fixed control policy u to the system and collect data.
2. Use the data collected in step 1 to solve the IRL Bellman Equation in (14) for $V_i(x)$ and $u_{i+1}(x)$, $i = 0, 1, \ldots$.
3. Stop if a stopping criterion is met. Otherwise, set $i = i + 1$ a go to step 2.

To implement the off-policy IRL Algorithm 2, the actor-critic structure is used to approximate the value function and control policy [44]. Note that [44] uses the Weierstrass approximation theorem to approximate the value function. The reference [42] uses the same estimator to deal with unknown parameters when input constraints exist.

3.3 Robust Off-policy IRL

Consider the following nonlinear system with unmatched dynamic uncertainties

$$\dot{x}(t) = f(x(t)) + g(x(t))\left[u(t) + \Delta_x(x, w)\right], \tag{15a}$$
$$\dot{w}(t) = \Delta_w(x, w), \tag{15b}$$

where $w(t) \in \mathbb{R}^p$ is the unmeasurable part of the state with unknown order p, $\Delta_w(x, w) \in \mathbb{R}^p$ and $\Delta_x(x, w) \in \mathbb{R}^m$ are unknown locally Lipschitz functions.

To extend the IRL off-policy Algorithm 2 to the system (15) with unmatched uncertainties and find the robust off-policy IRL Bellman equation, the system dynamics (15) is re-written as

$$\dot{x}(t) = f(x(t)) + g(x(t))u_i(t) + g(x(t))v_i(t) \tag{16}$$

where $v_i = u + \Delta_x - u_i$ [44]. By differentiating $V_i(x)$ along (16) and using the update rule in (9), one has

$$\begin{aligned}\dot{V}_i &= \frac{\partial V_i^T(x)}{\partial x}(f + gu_i + gv_i) \\ &= -Q(x) - u_i^T R u_i - 2u_{i+1}^T R v_i.\end{aligned} \tag{17}$$

The robust off-policy IRL Bellman equation is obtained by integrating from both sides of (17) as

$$V_i(x(t+T)) - V_i(x(t)) = \int_t^{t+T} (-Q(x) - u_i^T R u_i - 2u_{i+1}^T R v_i) d\tau. \tag{18}$$

The value function and the control input are approximated using neural networks (NNs) as

$$\hat{V}_i(x(t)) = \sum_{j=1}^{L_1} w_{1ij}\bar{\phi}_j(x), \tag{19a}$$

$$\hat{u}_{i+1}(t) = \sum_{j=1}^{L_2} w_{2ij}\bar{\sigma}_j(x), \tag{19b}$$

where L_1 and L_2 are the numbers of neurons on the hidden layers, $w_{1ij} \in \mathbb{R}$ and $w_{2ij} \in \mathbb{R}$ are the output layer weights, and $\bar{\phi}_j : \mathbb{R}^n \to \mathbb{R}$ and $\bar{\sigma}_j : \mathbb{R}^n \to \mathbb{R}^m$ are the basis functions to compute the output of j^{th} hidden neurons. Substituting (19) in (18) yields

$$\sum_{j=1}^{L_1} w_{1ij}\left[\bar{\phi}_j(x(t+T)) - \bar{\phi}_j(x(t))\right] = \int_t^{t+T} \left(-Q(x) - \hat{u}_i^T R \hat{u}_i - 2\sum_{j=1}^{L_2} w_{2ij}\bar{\sigma}_j^T(x) R \hat{v}_i\right) d\tau. \tag{20}$$

Iterating on the IRL Bellman equation (20) provides the following robust off-policy IRL algorithm.

Algorithm 3: Robust Off-policy IRL algorithm to find the solution of HJB

1. Apply a fixed control policy u to the system and collect data.
2. Use the data collected in step 1 to solve the IRL Bellman Equation in (20) for $\hat{V}_i(x)$ and $\hat{u}_{i+1}(x)$, $i = 0, 1, \ldots$.
3. Stop if a stopping criterion is met. Otherwise, set $i = i + 1$ a go to step 2.

4 RL Applications in Optimal Control for Unmanned Vehicle

This section provides two different RL applications in optimal control for unmanned vehicle. The first example is a partially model free approach where the designer needs to know the rotational dynamics of the vehicle. The second example is a model free approach based on the linearization of the vehicle models in Section 2.

4.1 Partially Model Free RL Application in Optimal Control for Unmanned Vehicle

The main challenge of designing control methods for unmanned vehicles is the existence of coupling between the control inputs, i.e., torques τ_B and forces F_B. One way of solving this problem is to derive one of these control inputs according to the control objective and design the other one considering the constraints (1a)-(1b) (or, equivalently, (4a)-(4b)). This method is called as the nonstandard backstepping method in [29, 46].

Assume that the unmanned vehicle has thrusters that are aligned with its body up direction. Examples of such systems include helicopters and $n-$rotors systems. Then, the total force takes a specific form in the body coordinate axis system [29] where f_T is a scalar total force expression. Similarly, if the thrusters are aligned with the body north direction, the total force would have the form $F_B = [f_T, 0, 0]^T$. Examples of such systems include missiles and conventional fixed wing aircrafts. Using these special forms of F_B, the partially model-free RL algorithm can be developed. For instance, the translational dynamics of the $n-$rotors system (e.g., Equation (1a) or Equation (4a)) can be linearized around the Hover condition as

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \end{bmatrix} + \begin{bmatrix} 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\ 0 & -g & 0 & 0 \\ g & 0 & 0 & 0 \\ 0 & 0 & 0 & -1/m \end{bmatrix} U, \tag{21}$$

where $U = [\phi, \theta, \psi, f_T]^T$. Since (21) is the linear version of (5), the analysis of Section 3.1 is valid for the $n-$rotor system. Note that the nonstandard backstepping control method in [29] aims to find the desired Euler angles that need to be tracked by the inner attitude control loop. The corresponding equation for this loop is indeed rotational dynamics expression (e.g., Equation (1b) or Equation (4b)) and requires the information of inertia matrix. That's why the method given in this subsection is partially model-free. Additionally, this method assumes that the pitch angle θ and the roll angle ψ remain in the interval $[-\pi/6, \pi/6]$ to ensure that the n-rotors system shows a linear behavior as small angle approximation is valid in this interval [28]. Furthermore, since the linear model of n-rotors system is obtained around the hover flight condition, the input parameters of (21) should be summed with the nominal flight condition parameters, i.e., $U_{\text{nominal}} = [0, 0, 0, -mg]^T$, before using them as setpoints for the inner attitude control loop. The example inner loop controllers are developed in [47, 48] for the quad-rotors system.

4.2 Model Free RL Application in Optimal Control for Unmanned Vehicle

This section covers how to use the RL algorithms developed in Section 3 on unmanned vehicles. It is well-known that most of the unmanned vehicle systems have highly nonlinear dynamics. Note that the dynamics in (1), (3), and (4) are in the form of (5). Therefore, Algorithm 1 along with the NN model in (10) and the update rule in (11) in Section 3.1 optimize the control gains for such nonlinear models including (1), (3), and (4). On the other hand, their nonlinear dynamics can be linearized at a particular condition of states. Examples include the linear F-16 model developed in [49] and the quadrotor model in [50]. Thus, a linearized model of any unmanned vehicle can be given as

$$\dot{x} = Ax + Bu, \tag{22}$$

including (1), (3), and (4). The corresponding performance functional is

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt \tag{23}$$

where the pair $(A, \sqrt{Q})$ is assumed to be observable. Then, following Algebraic Riccati Equation (ARE) can be obtained by defining corresponding Hamiltonian [3]

$$0 = A^T P + PA - PBR^{-1}B^T P + Q. \tag{24}$$

To develop a model-free algorithm, rewrite the system dynamics (22) as

$$\dot{x} = A_k x + B(u_k + u) \tag{25}$$

where $u_k = K_k x$ is the control policy to be updated and $A_k = A - BK_k$. Then, the IRL form can be obtained using the Kleinman's Method [47] as

$$x^T(t+T)P_k x(t+T) - x^T(t)P_k x(t) - 2\int_t^{t+T} \left([u + K_k x]^T R K_{k+1} x\right) d\tau = \int_t^{t+T} x^T Q_k x d\tau, \tag{26}$$

where $Q_k = -Q - K_k^T R K_k$. Then, the off-policy IRL Algorihm 2 can be developed using (26) instead of (14). Since (26) is obtained using linearized system dynamics (22), the matrix P_k can be found by well established least-squares method [51] instead of the actor-critic NNs developed in Section 3.1.

5 RL Applications in Fault-tolerant Control for Unmanned Systems

This section first recalls fault tolerant control design methods [52, 53] and then gives a RL algorithm that can be employed to obtain optimal controls of unmanned vehicle system that experiences actuator faults. These methods can be applied to unmanned quadrotor vehicles, unmanned underwater vehicles, and unmanned tailsitter vehicles.

In practical applications, the potential actuator faults of unmanned systems will lead to undesirable control performances and even a sequence of catastrophes. A common actuator fault in practice is a partial loss of effectiveness (LOE), which is resulted from the propeller structure damage or the change of actuator parameters. As shown in [52, 53], the LOE actuator fault can be considered as

$$\omega_{mj}^{*2} = h_j \omega_{mj}^2 \tag{27}$$

where ω_{mj}^* is the actual rotational velocity of the j^{th} rotor, ω_{mj} is the commanded rotational velocity of the rotor, and h_j is a positive constant satisfying $0 < h_j \leq 1$. The case $h_j = 1$ indicates that the j^{th} rotor is healthy. Consider the unmanned system in (1) along with the generated forces and torques in (2). The dynamics in (1) becomes

$$\ddot{p} = m^{-1} k_f (u_p - u_{\Delta p}) - g\vec{e}_3, \tag{28a}$$

$$\ddot{\Theta} = -I_B^{-1} C(\Theta, \dot{\Theta})\dot{\Theta} + I_B^{-1}(u_\tau - u_{\Delta\tau}), \tag{28b}$$

under actuator faults in (27), where $u_p = R_{BE} F_B \in \mathbb{R}^3$ and $u_\tau = \tau_B \in \mathbb{R}^3$ are the inputs, $u_{\Delta p}$ and $u_{\Delta\Theta}$ are time-varying input uncertainties due to the actuator faults in (27). The fault-tolerant controller can be designed as

$$u_p(e_p, t) = \frac{1}{2} u_p^*(e_p) + \frac{\hat{\bar{u}}_{\Delta p}^2 R_p u_p^*(e_p)}{\|R_p u_p^*(e_p)\| \hat{\bar{u}}_{\Delta p} + \sigma_1(t)} \tag{29a}$$

$$u_\tau(e_\Theta, t) = \frac{1}{2} u_\tau^*(e_\Theta) + \frac{\hat{\bar{u}}_{\Delta\Theta}^2 R_\Theta u_\tau^*(e_\Theta)}{\|R_\Theta u_\tau^*(e_\Theta)\| \hat{\bar{u}}_{\Delta\Theta} + \sigma_2(t)} \tag{29b}$$

where $e_p = p^r - p$ is the position tracking error, $u_p^*(x)$ and $u_\tau^*(x)$ are the optimal control policies in theory, σ_1 and σ_2 are some positive functions, and $\hat{\bar{u}}_{\Delta p}$ and $\hat{\bar{u}}_{\Delta\Theta}$ are the estimated uncertainty bounds by an observer (see [54, 55] for the details). It can be observed from (29) that the optimal control policies u_p^* and u_τ^* are required in the fault-tolerant controller design.

Note that (28) is in the form of (15). Hence, robust off-policy based RL algorithms can learn the optimal control policies. In particular, let V_p be the position control performance function given by

$$V_p(e_p) = \int_t^\infty e_p^T Q_p e_p + u_p^T R_p u_p d\tau, \tag{30}$$

where $e_p = p^r - p$ is the position tracking error. Similar to Section 3.3, IRL Bellman equation is

$$V_{p,i}(e_p(t+T)) - V_{p,i}(e_p(t)) = \int_t^\infty -e_p^T Q_p e_p - u_{p,i}^T R_p u_{p,i} d\tau - \int_t^\infty 2u_{p,i+1}^T R_p (u_p - u_{\Delta p} - u_{p,i}) d\tau, \tag{31}$$

and the value function $\hat{V}_p$ and the input $\hat{u}_p$ can be approximated with NN structures in (19). Therefore, the optimal control policy u_p^* can be approximated via RL method by solving the approximated Bellman equation in (31). A similar approach is applicable to compute u_τ^*. Then, the learned optimal control policies u_p^* and u_τ^* can be used to construct the fault-tolerant controller in (29) for unmanned systems.

6 Remarks and Discussion

This paper reviewed some on-policy and off-policy RL algorithms, which have been applied to learn the optimal control policies for unmanned vehicles with uncertain dynamics. The capability of RL methods is shown to address the unknown system parameters, nonlinearities, couplings, and actuator faults in unmanned vehicle dynamics.

The feedback linearization technique shows a convenient way to design the optimal controller for unmanned vehicles in Section 4, and the partially model-free RL approach and completely model-free RL approach are presented. However, the performance of the designed controller may be affected by the nonlinear and coupled dynamics of unmanned vehicles in practical applications. Therefore, it is shown that the RL approaches combined with adaptive control methods can address the nonlinearities, couplings, and actuator faults issues in the vehicle dynamics.

In the applications of RL approaches in the mentioned literature, an admissible control policy is still required in the control design. This could be a restrictive assumption when the vehicle system is unstable, and the vehicle model is not available. In practical applications, a traditional model-free controller such as the proportional-derivative (PD) control method can be used as an initial admissible control policy to generate the vehicle system data for learning. Furthermore, the value iteration RL algorithms (see, [56]) can learn optimal control policies without requiring an initial admissible control policy at the expense of more iterations for convergence to optimal solutions.

7 Conclusions and Future Trends

This paper reviews RL methods applied to unmanned vehicles with uncertain dynamics to learn optimal control policies under the effects of nonlinearities, couplings, and actuator faults. On-policy and off-policy reinforcement learning approaches are firstly discussed in the optimal controller design. Then, the optimal control design of unmanned vehicles using a partially model-free RL approach and a completely model-free RL approach are presented. The faults in the actuators are considered, and off-policy RL approaches are introduced to construct a fault-tolerant controller without the requirement of vehicle model information. In the existing works, the RL approaches are only applied to achieve the trajectory tracking of a single unmanned vehicle or multiple unmanned vehicles with a consensus objective. However, the RL-based Nash game problem of multiple unmanned vehicles with conflicting control objectives has not been investigated yet, which can be a promising topic in the future.

Acknowledgments

This work is supported by ARO Grant W911NF-20-1-0132, and ONR Grant N00014-18-1-2221.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Belmont, MA, USA: Athena Scientific, 1996.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT press, 1998.

[3] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. New York, NY, USA: John Wiley & Sons, 2012.

[4] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and trends in Robotics*, vol. 2, no. 1-2, pp. 388–403, 2013.

[5] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[6] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2042–2062, 2017.

[7] J. Li, M. Ran, H. Wang, and L. Xie, "A behavior-based mobile robot navigation method with deep reinforcement learning," *Unmanned Systems*, vol. 9, no. 03, pp. 201–209, 2021.

[8] P. J. Werbos, "Neural networks for control and system identification," in *Proceedings of the 28th IEEE Conference on Decision and Control,*. IEEE, 1989, pp. 260–265.

[9] P. J. Werbos, W. Miller, and R. Sutton, *A menu of designs for reinforcement learning over time*. Cambridge, MA, USA: MIT press, 1990.

[10] P. Werbos, "Approximate dynamic programming for realtime control and neural modelling," *Handbook of intelligent control: neural, fuzzy and adaptive approaches*, pp. 493–525, 1992.

[11] S. Josef and A. Degani, "Deep reinforcement learning for safe local planning of a ground vehicle in unknown rough terrain," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6748–6755, 2020.

[12] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5723–5728, 2020.

[13] Y. Zhao, Y. Ma, and S. Hu, "USV formation and path-following control via deep reinforcement learning with random braking," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5468–5478, 2021.

[14] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[15] D. Wang, T. Fan, T. Han, and J. Pan, "A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3098–3105, 2020.

[16] Z. Jiandong, Y. Qiming, S. Guoqing, L. Yi, and W. Yong, "UAV cooperative air combat maneuver decision based on multi-agent reinforcement learning," *Journal of Systems Engineering and Electronics*, vol. 32, no. 6, pp. 1421–1438, 2021.

[17] A. Sacco, F. Esposito, G. Marchetto, and P. Montuschi, "Sustainable task offloading in uav networks via multi-agent reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 5003–5015, 2021.

[18] C. Yu, X. Wang, X. Xu, M. Zhang, H. Ge, J. Ren, L. Sun, B. Chen, and G. Tan, "Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 735–748, 2019.

[19] X. Liu, Y. Liu, and Y. Chen, "Reinforcement learning in multiple-UAV networks: Deployment and movement design," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8036–8049, 2019.

[20] Y.-J. Chen, D.-K. Chang, and C. Zhang, "Autonomous tracking using a swarm of UAVs: A constrained multi-agent reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 702–13 717, 2020.

[21] Z. Xia, J. Du, J. Wang, C. Jiang, Y. Ren, G. Li, and Z. Han, "Multi-Agent Reinforcement Learning Aided Intelligent UAV Swarm for Target Tracking," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 931–945, 2021.

[22] K. Menda, Y.-C. Chen, J. Grana, J. W. Bono, B. D. Tracey, M. J. Kochenderfer, and D. Wolpert, "Deep reinforcement learning for event-driven multi-agent decision processes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 4, pp. 1259–1268, 2018.

[23] S. M. Wong, H. W. Ho, and M. Z. Abdullah, "Design and fabrication of a dual rotor-embedded wing vertical take-off and landing unmanned aerial vehicle," *Unmanned Systems*, vol. 9, no. 01, pp. 45–63, 2021.

[24] K. Z. Ang, J. Q. Cui, T. Pang, K. Li, K. Wang, Y. Ke, and B. M. Chen, "Design and implementation of a thrust-vectored unmanned tail-sitter with reconfigurable wings," *Unmanned Systems*, vol. 3, no. 02, pp. 143–162, 2015.

[25] A. Hamissi, K. Busawon, L. Dala, Y. Bouzid, M. Zaouche, and M. Hamerlain, "A new nonlinear control design strategy for fixed wing aircrafts piloting," *Unmanned Systems*, vol. 9, no. 04, pp. 293–308, 2021.

[26] B. Kiumarsi, W. Kang, and F. L. Lewis, "H_∞ control of nonaffine aerial systems using off-policy reinforcement learning," *Unmanned Systems*, vol. 4, no. 01, pp. 51–60, 2016.

[27] W. Zhao, H. Liu, and F. L. Lewis, "Data-driven fault-tolerant control for attitude synchronization of nonlinear quadrotors," *IEEE Transactions on Automatic Control*, vol. 66, no. 11, pp. 5584–5591, 2021.

[28] H. Liu, F. Peng, H. Modares, B. Kiumarsi, and F. L. Lewis, "Heterogeneous formation control of multiple rotorcrafts with unknown dynamics using reinforcement learning," in *IEEE Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 3617–3622.

[29] Y. Kartal, P. Kolaric, V. Lopez, A. Dogan, and F. Lewis, "Backstepping approach for design of PID controller with guaranteed performance for micro-air UAV," *Control Theory and Technology*, vol. 18, no. 1, pp. 19–33, 2020.

[30] Z.-q. Su, M. Zhou, F.-f. Han, Y.-w. Zhu, D.-l. Song, and T.-t. Guo, "Attitude control of underwater glider combined reinforcement learning with active disturbance rejection control," *Journal of Marine Science and Technology*, vol. 24, no. 3, pp. 686–704, 2019.

[31] S. Verling, B. Weibel, M. Boosfeld, K. Alexis, M. Burri, and R. Siegwart, "Full attitude control of a VTOL tailsitter UAV," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3006–3012.

[32] W. Zhao, H. Liu, and F. L. Lewis, "Robust formation control for cooperative underactuated quadrotors via reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4577–4587, 2020.

[33] B. Xiao and S. Yin, "A new disturbance attenuation control scheme for quadrotor unmanned aerial vehicles," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2922–2932, 2017.

[34] H. Liu, T. Ma, F. L. Lewis, and Y. Wan, "Robust formation control for multiple quadrotors with nonlinearities and disturbances," *IEEE transactions on cybernetics*, vol. 50, no. 4, pp. 1362–1371, 2018.

[35] W. Zhao, H. Liu, F. L. Lewis, K. P. Valavanis, and X. Wang, "Robust visual servoing control for ground target tracking of quadrotors," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 5, pp. 1980–1987, 2019.

[36] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29–39, 2010.

[37] H. Mo and G. Farid, "Nonlinear and adaptive intelligent control techniques for quadrotor UAV–a survey," *Asian Journal of Control*, vol. 21, no. 2, pp. 989–1008, 2019.

[38] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[39] D. Vrabie and F. Lewis, "Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems," *Neural Networks*, vol. 22, no. 3, pp. 237–246, 2009.

[40] R. Kamalapurkar, J. R. Klotz, and W. E. Dixon, "Concurrent learning-based approximate feedback-Nash equilibrium solution of $N-$player nonzero-sum differential games," *IEEE/CAA journal of Automatica Sinica*, vol. 1, no. 3, pp. 239–247, 2014.

[41] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for approximate optimal regulation," *Automatica*, vol. 64, pp. 94–104, 2016.

[42] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, vol. 50, no. 1, pp. 193–202, 2014.

[43] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *International Journal of Adaptive Control and Signal Processing*, vol. 27, no. 4, pp. 280–301, 2013.

[44] Y. Jiang and Z.-P. Jiang, "Robust adaptive dynamic programming and feedback stabilization of nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 882–893, 2014.

[45] B. Luo, H.-N. Wu, and T. Huang, "Off-policy reinforcement learning for H_∞ control design," *IEEE transactions on cybernetics*, vol. 45, no. 1, pp. 65–76, 2014.

[46] Y. Kartal, K. Subbarao, A. Dogan, and F. Lewis, "Optimal game theoretic solution of the pursuit-evasion intercept problem using on-policy reinforcement learning," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 16, pp. 7886–7903, 2021.

[47] Y. Kartal, K. Subbarao, N. R. Gans, A. Dogan, and F. Lewis, "Distributed backstepping based control of multiple UAV formation flight subject to time delays," *IET Control Theory & Applications*, vol. 14, no. 12, pp. 1628–1638, 2020.

[48] S. Ristevski, A. T. Koru, T. Yucelen, K. M. Dogan, and J. A. Muse, "Experimental results of a quadrotor uav with a model reference adaptive controller in the presence of unmodeled dynamic," in *AIAA Scitech 2022 Forum*, 2022, p. 1381.

[49] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[50] M. Bergamasco and M. Lovera, "Identification of linear models for the dynamics of a hovering quadrotor," *IEEE transactions on control systems technology*, vol. 22, no. 5, pp. 1696–1707, 2014.

[51] Y. Jiang and Z.-P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, no. 10, pp. 2699–2704, 2012.

[52] R. C. Avram, X. Zhang, and J. Muse, "Nonlinear adaptive fault-tolerant quadrotor altitude and attitude tracking with multiple actuator faults," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 701–707, 2017.

[53] W. Zhao, H. Liu, K. Valavanis, and F. Lewis, "Fault-tolerant formation control for heterogeneous vehicles via reinforcement learning," *IEEE Transactions on Aerospace and Electronic Systems*, 2021.

[54] W. Zhao, H. Liu, and Y. Wan, "Data-driven fault-tolerant formation control for nonlinear quadrotors under multiple simultaneous actuator faults," *Systems & Control Letters*, vol. 158, p. 105063, 2021.

[55] X.-J. Li and G.-H. Yang, "Robust adaptive fault-tolerant control for uncertain linear systems with actuator failures," *IET control theory & applications*, vol. 6, no. 10, pp. 1544–1551, 2012.

[56] S. A. A. Rizvi and Z. Lin, "Reinforcement learning-based linear quadratic regulation of continuous-time systems using dynamic output feedback," *IEEE Transactions on Cybernetics*, vol. 50, no. 11, pp. 4670–4679, 2019.