Scilight

Advanced Mechatronics
https://www.sciltp.com/journals/am

Review

Reinforcement Learning Based Optimal Control: A Survey of Adaptive Dynamic Programming for Manipulators and Wheeled Mobile Robots

Chi Pui Chan [1], Ningwei Bai [1], Qichen Yin [1], Junan Wang [2], Boming Hu [3], Yunda Yan [1] and Zezhi Tang [1,*]

[1] Department of Computer Science, University College London (UCL), London WC1E 6BT, UK
[2] Department of Electrical and Electronic Engineering, University of Manchester, Manchester M13 9PL, UK
[3] Midea Global Innovation Center, Shanghai 201702, China
* Correspondence: zezhi.tang@ucl.ac.uk

How To Cite: Chan, C.P.; Bai, N.; Yin, Q.; et al. Reinforcement Learning Based Optimal Control: A Survey of Adaptive Dynamic Programming for Manipulators and Wheeled Mobile Robots. *Adv. Mechatron.* **2026**, *1*(1), 1.

Publication History

Received: 26 February 2026
Revised: 29 April 2026
Accepted: 26 May 2026
Published: 15 July 2026

Abstract: Adaptive Dynamic Programming (ADP) has emerged as a promising Reinforcement Learning (RL)-based optimal control approach for robotics, offering a middle ground between model-based optimal control and purely data-driven deep RL. While previous surveys have organized the ADP literature by algorithm type or control problem, the significance of platform-specific physical constraints for ADP controller design is not widely addressed. This survey presents a robotics-oriented review of ADP for two representative platforms, namely robotic manipulators and Mobile Wheeled Robots (MWRs). The literature is reviewed on the basis of how platform characteristics shape controller architectures, modelling assumptions, and stability analysis. We compare studies employing typical ADP structures (actor–critic, single-critic, decentralized, and event-triggered formulations), approaches to robustness guarantees, hardware validation, and practical deployment limitations. Several recurring patterns are identified, including the predominance of Uniform Ultimate Boundedness (UUB) as the primary stability guarantee, the limited scale of hardware validation, and the tension between scalability, safety, and real-time implementation. Finally, open challenges and future directions are discussed, including certified safety, high-dimensional scalability, sample efficiency, sim-to-real transfer, and perception-driven ADP.

Keywords: Adaptive Dynamic Programming; Reinforcement Learning; robotics; manipulators; Mobile Wheeled Robots

1. Introduction

Modern robotic systems are increasingly used in complex and unstructured environments. These include busy indoor areas, urban road networks, offshore industrial platforms, and low-Earth-orbit space stations [1–4]. Operating in these environments demands control strategies capable of real-time adjustment while adhering to safety and energy constraints. RL provides a robust framework by formulating control issues as Markov Decision Process (MDP) [5, 6]. However, traditional tabular methods do not scale well to high-dimensional state spaces [7, 8], and deep RL methods usually need large amounts of training data and provide only probabilistic safety guarantees, not deterministic ones [9–12].

ADP serves as a middle ground between model-based control and data-heavy deep learning. By incorporating Policy Iteration (PI) and Value Iteration (VI) within the Hamilton–Jacobi–Bellman (HJB) framework, ADP estimates the optimal cost-to-go using neural critics while ensuring Lyapunov-style stability guarantees [13, 14]. The foundations of ADP go back several decades. Bellman formalized Dynamic Programming (DP) in the 1950s [15, 16]. Later, Werbos proposed training neural network critics via backpropagated Bellman errors. This work led to the creation of Heuristic Dynamic Programming (HDP), Dual Heuristic Programming (DHP), and Globalised Dual Heuristic Programming (GDHP) architectures [17, 18]. Sutton and Barto developed Temporal Difference (TD) learning and the actor–critic framework [5, 19]. Bertsekas and Tsitsiklis contributed with their work on Neural Dynamic Programming (NDP) [20]. Lewis combined adaptive control and neural architectures by creating the first passivity-based neural controller for robot manipulators [21]. Doya expanded RL to continuous-time systems using the HJB framework [22].

Key extensions have increased the use of ADP. Integral Reinforcement Learning (IRL) reformulates the Bellman equation in integral form to allow model-free learning without needing explicit knowledge of dynamics [23, 24]. H_∞-ADP treats control as a two-player zero-sum game to effectively

reject disturbances [25–27]. Qin et al. [28] further extended this zero-sum game formulation to input-saturated nonlinear systems with strict state constraints, using a smooth mapping function and a single-critic neural network to reduce controller update rates while maintaining safety. Event-triggered ADP cuts down on computational and communication demands by updating control signals only when needed [29–31]. Qin et al. [32] proposed a novel event-triggered fault-tolerant control method for saturated nonlinear systems with full-state constraints and actuator faults, using an observer-based compensation scheme combined with experience replay to achieve near-optimal safety control without constructing an event-triggered HJB equation. Recent advances include hardware-efficient single-network critics [33], adaptive event-triggered H_∞ designs [34], and meta-learning warm-start methods [35].

The use of ADP in robotics started with Lyapunov-guaranteed manipulator tracking [21]. It has now grown to include mobile robots [36–39], wheel-legged platforms [40,41], flexible manipulators [42] and multi-robot systems [43–45]. These applications demonstrate that ADP's combination of online learning, Lyapunov-based stability, and constraint handling solves problems that traditional controllers struggle with. Figure 1 illustrates how ADP has developed from its theoretical roots to current robotic applications.

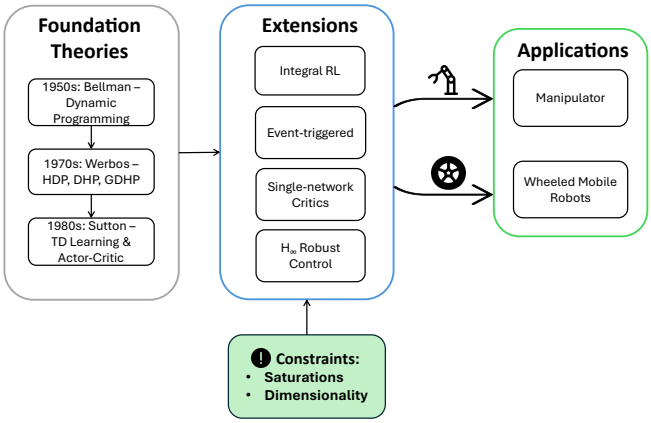

Figure 1. Development of ADP.

1.1. Related Surveys and Motivation

While there have been several surveys previously on ADP and RL-based optimal control, they were all organized by algorithms or categories of problems instead of by robotic platform [13, 14, 46]. The work of Liu et al. [13] was organized based on problem category such as optimal regulation, game theory, large-scale systems, and comparing ADP network structures like HDP, DHP, GDHP, and single-critic vs. actor-critic approaches at the algorithmic level. While the manipulation system was part of the large-scale systems category in terms of fault-tolerant decentralized control and event-triggered decentralized tracking of modular reconfigurable robots, this survey did not explore the constraints posed by the manipulator and did not mention MWRs. Similarly, the survey conducted by Kiumarsi et al. [14] was organized by the type of problem such as Discrete Time (DT) regulation, Continuous Time (CT) regulation, H_∞, Nash games, and graphical games, where the examples of robot manipulator tracking and nonholonomic wheeled mobile robot control were briefly mentioned without any discussion on constraints related to the robot platform itself on ADP approach selection. Liu et al. [46] reviewed ADP for unmanned vehicles such as Unmanned Ground Vehicle (UGV) and Unmanned Aerial Vehicle (UAV) based on ADP methods (robust ADP, event-triggered ADP, H_∞-ADP) and mentioned manipulators only for event-triggered decentralized tracking.

This survey adopts a fundamentally different perspective. Rather than surveying ADP algorithms and appending robotic examples, we take manipulators and MWRs as the organizing principle and systematically examine how their physical constraints shape ADP design. The novelty of this perspective lies not only in the choice of organization, but in the insights it enables.

1. Constraint-driven design principles, demonstrating how platform-specific constraints (actuator saturation, joint limits, input delays, nonholonomic kinematics, and safety requirements) dictate the choice of ADP architecture, cost function structure, and stability analysis.
2. Deployment-oriented evaluation dimensions, introducing comparison axes (centralized vs. decentralized, simulation vs. hardware, single-critic vs. actor–critic) that reveal practical trade-offs between scalability, safety, and real-time feasibility.

Table 1 summarizes the scope differences with prior surveys.

Table 1. Comparison of this survey with prior ADP/RL surveys

Survey	Year	Organized by Robot	Arch. Comparison	Platform Constraints	Stability Taxonomy	Sim vs. Hardware
Liu [13]	2020	–	Partial	Partial	–	–
Kiumarsi [14]	2017	–	–	Partial	–	–
Liu [46]	2025	–	Partial	Partial	–	Partial
This work	2026	✓	✓	✓	✓	✓

The remainder of this paper is organized as follows. Section 2 presents the theoretical foundations of ADP. Section 3 reviews representative ADP applications across manipulators and MWRs. Section 4 discusses future research directions, and Section 5 concludes the survey.

2. ADP Foundations

ADP provides a foundation for optimal control problem solving in the context of robotics by incorporating both policy evaluation and improvement into the HJB problem formulation [17,18]. As opposed to purely heuristic learning methods, ADP remains directly linked to the theoretical principles of optimal control, employing function approximation to address the challenges imposed by high-dimensional nonlinear robotic dynamics [13,47].

In discussing ADP as a framework for optimal control throughout the paper, it is worth noting that ADP falls short of providing a full optimal control solution for several reasons. Given the utilization of function approximators with limited capacity, namely critics and actors, in nonlinear robotic systems, the exact solution of the HJB equation is not possible. Hence, a near-optimal controller is obtained whose performance is inferior to the optimal one by some residual determined by the approximation error, the degree of excitation of the training data and exploration noise injected during online learning [48,49]. The stability guarantees reviewed in Section 2.3 reflect this fact, which is why UUB rather than asymptotic optimality is the dominant claim across the literature.

2.1. Problem Formulation

Consider a robotic system described by the discrete-time affine nonlinear dynamics:

$$x_{k+1} = f(x_k) + g(x_k)u_k, \tag{1}$$

where $x \in \mathbb{R}^n$ denotes the system state, $u \in \mathbb{R}^m$ represents the control input, $f : \mathbb{R}^n \to \mathbb{R}^n$ characterizes the drift dynamics, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ represents the control effectiveness matrix. Both $f(\cdot)$ and $g(\cdot)$ are assumed to be continuous functions with $f(0) = 0$ to ensure that the origin is an equilibrium. The stage cost (instantaneous cost) is defined as a quadratic function:

$$U\left(x_k, u_k\right) = x_k^\top Q x_k + u_k^\top R u_k, \tag{2}$$

where $Q \succeq 0$ penalizes state deviations and $R \succ 0$ penalizes control effort. The objective is to find a control policy $u_k = \pi(x_k)$ that minimizes the infinite-horizon discounted cost function:

$$J\left(x_k\right) = \sum_{i=k}^{\infty} \gamma^{i-k}\left(x_i^\top Q x_i + u_i^\top R u_i\right), \tag{3}$$

where $0 < \gamma \leq 1$ is the discount factor that determines the relative importance of future costs.

Continuous-Time Counterpart

Many robotic platforms are naturally modelled in continuous time, as reflected in the manipulator dynamics of Section 3.1 and the MWR kinematics of Section 3.2. The CT analogue of (1) is the affine nonlinear system

$$\dot{x}(t) = f(x(t)) + g(x(t))\,u(t), \tag{4}$$

with infinite-horizon cost

$$V(x(t)) = \int_t^\infty \left(x^\top(\tau)Qx(\tau) + u^\top(\tau)Ru(\tau)\right) d\tau, \tag{5}$$

and the associated HJB equation

$$0 = \min_u \left\{x^\top Qx + u^\top Ru + \nabla V^\top(x)\big(f(x) + g(x)u\big)\right\}. \tag{6}$$

The DT and CT forms are related through zero-order-hold sampling, which means DT-based ADP derivations transfer to sampled CT implementations on digital hardware. IRL [23,24] reformulates the CT Bellman equation over a finite integration window, removing the explicit dependence on $f(x)$ and $g(x)$ and enabling direct CT learning. To keep notation uniform, the HDP, DHP, and GDHP updates in the remainder of this section are presented in DT form. Their CT counterparts, in which the Bellman residual is replaced by the Hamiltonian, are developed in [22,48–50].

Convergence of the critic weights in all ADP variants requires that the collected data is sufficiently rich, a condition formalized as Persistent Excitation (PE) [49]. PE is enforced by adding probing noise $e_p(t)$ to the control signal so that $u(t) = u_{\text{adp}}(t) + e_p(t)$, which guarantees weight convergence but introduces transient performance loss and potential constraint violations on physical hardware. Experience replay [24] and concurrent learning [51] weaken or remove the PE requirement by reusing stored transitions, reducing the need for persistent probing.

2.2. ADP Architectures

Most ADP controllers adopt an actor–critic framework, where neural networks or basis functions are employed for function approximation, as illustrated in Figure 2. In this framework, the actor approximates the control input directly based on the control policy $u = \pi(x)$, thereby avoiding explicit computation of the HJB equation. The critic approximates either the value function $V(x)$ or its gradient $\nabla_x V(x)$, which is used to update the actor's weights.

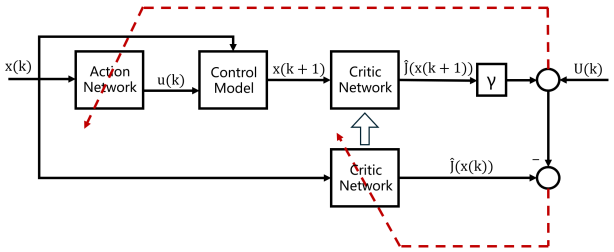

Figure 2. Actor–critic architecture for ADP.

The actor–critic approximations are defined as

$$\begin{aligned} V(x_k) &= W_c^\top \phi(x_k), \\ u_k &= W_a^\top \psi(x_k), \end{aligned} \tag{7}$$

where W_c and W_a denote the weight vectors of the critic and actor, respectively, and $\phi(x)$ and $\psi(x)$ are the corresponding basis functions or neural networks. According to the classification established by Prokhorov and Wunsch [18], ADP architectures are categorized into three fundamental types: HDP, DHP, and GDHP.

2.2.1. Heuristic Dynamic Programming (HDP)

In HDP, the critic network approximates the value function $V(x)$ directly and is trained to minimize the temporal difference (Bellman) error:

$$\begin{aligned} \mathcal{L}_{\text{HDP}} &= \sum_k \delta_{(1)k}^2, \\ \delta_{(1)k} &= V(x_k) - [U(x_k, u_k) + \gamma V(x_{k+1})], \end{aligned} \tag{8}$$

where $\delta_{(1)k}$ represents the TD error at time step k. The actor weights are updated using gradient descent:

$$\Delta W_a = -\eta_a \frac{\partial V(x_k)}{\partial u_k} \frac{\partial u_k}{\partial W_a}, \tag{9}$$

where $\eta_a > 0$ is the actor learning rate.

2.2.2. Dual Heuristic Programming (DHP)

In DHP, the critic network approximates the gradient of the value function (costate) $\nabla_x V(x)$ rather than the value function itself. The training objective minimizes the gradient error:

$$\begin{aligned} \mathcal{L}_{\text{DHP}} &= \delta_{(2)k}^\top \delta_{(2)k}, \\ \delta_{(2)k} &= \frac{\partial V(x_k)}{\partial x_k} - \left[\frac{\partial U(x_k, u_k)}{\partial x_k} + \gamma \frac{\partial V(x_{k+1})}{\partial x_k} \right]. \end{aligned} \tag{10}$$

Each component of the gradient error is computed as:

$$\begin{aligned} \delta_{(2)k,j} = \frac{\partial V(x_k)}{\partial x_{k,j}} - \gamma \frac{\partial V(x_{k+1})}{\partial x_{k,j}} - \frac{\partial U(x_k, u_k)}{\partial x_{k,j}} \\ - \sum_{i=1}^{m} \frac{\partial U(x_k, u_k)}{\partial u_{k,i}} \frac{\partial u_{k,i}}{\partial x_{k,j}}. \end{aligned} \tag{11}$$

The actor is updated according to:

$$\Delta W_a = -\eta_a \left[\frac{\partial U(x_k, u_k)}{\partial u_k} + \gamma \frac{\partial V(x_{k+1})}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial u_k} \right]^\top \frac{\partial u_k}{\partial W_a}, \tag{12}$$

and the critic weights are updated as:

$$\Delta W_c = -\eta_c \sum_{j=1}^{n} \delta_{(2)k,j} \frac{\partial^2 V(x_k)}{\partial x_{k,j} \partial W_c}, \tag{13}$$

where $\eta_c > 0$ is the critic learning rate.

2.2.3. Globalised Dual Heuristic Programming (GDHP)

GDHP combines the objectives of both HDP and DHP by training the critic to minimize errors with respect to both the value function and its gradient:

$$\Delta W_c = -\eta_{c1} \delta_{(1)k} \frac{\partial V(x_k)}{\partial W_c} - \eta_{c2} \sum_{j=1}^{n} \delta_{(2)k,j} \frac{\partial^2 V(x_k)}{\partial x_{k,j} \partial W_c}, \tag{14}$$

where η_{c1} and η_{c2} are the learning rates for the value and gradient components, respectively. The actor update follows the same formulation as in DHP (12).

2.2.4. Action-Dependent Variants

The action-dependent variants (ADHDP, ADDHP, ADGDHP) extend these architectures by having the critic approximate functions of both state and action (x, u). Specifically, ADHDP learns the action-value function $Q(x, u)$, ADDHP learns its gradient $\nabla Q(x, u)$, and ADGDHP learns both. In these variants, the actor is updated directly via $\partial Q / \partial u$, eliminating the need for a separate dynamics model.

2.2.5. "Model-Free" in ADP

"Model-free" in ADP means that there is no need for the drift term $f(x)$ in (1) [23, 24]. However, the algorithm will still require the specification of the stage cost $U(x, u)$, and the optimal controller $u^* = -\frac{1}{2} R^{-1} g^\top(x) \nabla V^*(x)$ is a linear function of $g(x)$. Thus, the vast majority of algorithms will assume on having at least the form of the control effectiveness matrix. Furthermore, IRL and action-dependent methods will further relax this using integrated Bellman forms [23], but they are still model-dependent on costs and measurements rather than being fully data-driven.

2.3. Stability Notions in ADP Analysis

The ADP literature mentions several different forms of stability, each having a unique practical implication for the application of robots. The most powerful notion of asymptotic stability requires

$$\lim_{t \to \infty} \|x(t)\| = 0, \tag{15}$$

to hold along any trajectory initiated from a sufficiently small neighbourhood of the equilibrium [52], a form of stability guaranteed in ADP methods only if the residual errors associated with the critic and actor approximators have zero residual error.

UUB is more commonly discussed and guarantees that

$$\exists\, T(x_0, \rho) \geq 0 \text{ such that } \|x(t)\| \leq \rho,\ \forall t \geq T, \tag{16}$$

where $T(x_0, \rho)$ denotes the finite transient time after which the trajectory starting from x_0 remains within the residual set of radius ρ, with ρ determined by the approximation errors and learning gains [47,52]. Boundedness, another notion of stability, guarantees only the existence of a forward-invariant set without providing an explicit bound on its size.

Finite-time stability strengthens (15) to

$$x(t) = 0,\ \forall t \geq T(x_0), \quad T(x_0) < \infty, \tag{17}$$

with the settling time depending on the initial condition [53]. Fixed-time stability further requires $T(x_0) \leq T_{\max}$ independently of x_0 [54]. Both rely on non-Lipschitz feedback terms that can induce chattering on physical hardware. Practical stability relaxes the target to a prescribed tolerance set,

$$\|x(t)\| \leq \delta,\ \forall t \geq T, \quad \delta > 0 \text{ prescribed}, \tag{18}$$

which is useful when an engineering-acceptable neighbourhood suffices [55].

In the manipulator control design studies surveyed in Section 3.1 and MWR applications discussed in Section 3.2, the guarantees are mostly UUB (16), along with some cases where finite-time and asymptotic convergence results via Lyapunov function approach are stated. This tendency is explained by the problem that approximation errors in neural critics render the derivative of the Lyapunov function non-negative definite, thus providing bounds on estimation error only.

2.4. Safety and Constraint Handling in ADP

Robotic deployment of ADP involves limitations on the joints, actuators, and obstacles, which must be obeyed both when converging and learning transiently. Two approaches are used in the literature. Soft constraints are enforced by augmenting the stage cost with a penalty term:

$$U_{\text{soft}}(x, u) = x^\top Q x + u^\top R(x) u + \lambda\, \max\{0,\, h_v(x)\}^2, \tag{19}$$

where $h_v(x)$ denotes a constraint violation, while $R(x)$ grows closer to joint limits [33]. A saturation function of the type (26) limits the torque input without guaranteeing the state safety. The resulting guarantees are only soft preferences because the effectiveness of constraints will depend on the multiplier λ, while exploration noise introduced for PE is capable of violating the constraint during learning [11,56].

Hard constraints provide deterministic forward invariance of a safe set. A Control Barrier Function (CBF) $h(x)$ with $\{x \mid h(x) \geq 0\}$ taken as the safe set enforces invariance through

$$L_f h(x) + L_g h(x)\, u \ \geq\ -\alpha(h(x)), \tag{20}$$

for some class-$\mathcal{K}$ function α [57, 58]. In ADP, the learned action $u_{\text{adp}}(x)$ is composed with the CBF quadratic program

$$u^*(x) = \arg\min_u \tfrac{1}{2}\, \|u - u_{\text{adp}}(x)\|^2 \ \text{ s.t. } \ L_f h + L_g h\, u \geq -\alpha(h), \tag{21}$$

which projects the nominal policy onto the safe action set with minimum deviation from optimality [59,60]. The projection requires knowledge of $g(x)$, which conflicts with the model-free premise of many ADP schemes, and designing $h(x)$ for high-Degrees of Freedom (DOF) manipulators with simultaneous joint, velocity, and obstacle constraints remains task-specific.

Another approach involves the incorporation of the ADP policy within a predictive safety filter, which computes a solution to a short-horizon optimization problem that overrides the nominal control input if the resulting action will take the system outside a pre-computed robust invariant set [61,62]. Hamilton-Jacobi reachability yields the strongest theoretical foundations for the safety filter by computing the backward reachable set of the unsafe region offline [63,64]. However, its scalability is quite poor for problems involving more than six state dimensions. The overall picture is that the ADP papers surveyed in Sections 3.1 and 3.2 handle safety almost exclusively through (19) and saturation, and that a principled coupling between the UUB Lyapunov analysis of ADP and the set-invariance analysis behind (20) and (21) remains largely open.

3. ADP in Robotic Manipulation and Mobile Wheeled Systems

The following section discusses ADP in manipulators (Section 3.1) and MWR (Section 3.2). The discussion for each type of robot starts with stating its standard system model, followed by a synthesis of the literature based on four overarching issues, which are architecture evolution and design trade-offs, assumptions used in literature and their shortcomings, stability analysis, and validation status. Tables 2 and 3 present a systematic comparison of several methods. Orthogonal to the above categorization based on robot classes, the discussed ADP methods are further classified according to three dimensions, namely single-critic versus actor–critic, centralized versus decentralized, and simulation-only versus hardware-validated. These dimensions are reflected in the architecture paragraphs of Sections 3.1.2 and 3.2.2, the validation-status subsections (Sections 3.1.5 and 3.2.5), and the Architecture and Hardware Validation columns of Tables 2 and 3.

Table 2. Comparison of ADP architectures for manipulator control.

Reference	System	Model Knowledge	Architecture	Stability	Constraints	Hardware Validation	Key Limitation
Gierlak et al. [65]	3-DOF rigid	Partial (dynamics model)	Dual DHP	UUB	None	Yes (Scorbot 4PC)	Torque transients during exploration
Szuster et al. [66]	3-DOF rigid	Partial ($M(q)$)	Dual DHP	UUB	None	Yes (Scorbot-ER 4pc)	Limited to low-DOF systems
Li et al. [33]	3-DOF planar	Partial (M_0, C_0, G_0)	Single-critic RBF	UUB	Joint limit + saturation (cost-shaped)	No	Requires smooth ∇V
Zhang et al. [67]	Multi-manipulator	None	Critic + sliding mode	UUB	Fault tolerance	No	Requires known fault bounds
Yang et al. [68]	Large-scale	None	Decentralized	Asymptotic	Mismatched coupling	No	Requires pseudo-inverse of $g(x)$
Zhao et al. [69]	Modular reconfigurable	None	Decentralized + event-trigger	Asymptotic	Event threshold	No	Threshold tuning per configuration
Zhou et al. [70]	2-DOF modular	None	Decentralized + fault-tolerant	UUB	Actuator faults	Yes (2-DOF modular)	Strong coupling degrades
Sveen et al. [71]	4-DOF serial	None (off-policy)	Decentralized	UUB	Rank-based excitation	Yes (Quanser Qarm)	Sensitive to initial condition + noise

Table 3. Comparison of ADP-based control approaches for MWR.

Reference	System	Model Knowledge	Architecture	Stability	Constraints	Hardware Validation	Key Limitation
Luy et al. [72]	Diff. drive	Partial	Single-network	UUB	Vision feedback	Yes	Low-light performance
Dian et al. [73]	Magnetic wheel	Partial	Dual DHP	UUB	None	No	Platform-specific
Wang et al. [74]	Diff. drive	Partial (B)	Critic-only	Fixed-time	None	Yes (QBot 3)	Non-Lipschitz chattering
Li et al. [75]	Diff. drive	None	A-C + resistance NN	UUB (L-K)	Delay compensation	No	Assumes bounded delay
Li et al. [76]	Diff. drive	Partial	Model-A-C + APF	UUB	Obstacle avoidance	No	Untested in dynamic crowds
Ghasemzadeh et al. [77]	Tractor-trailer	Partial ($g(X)$)	Critic-only	Lyapunov	Kinematic-dynamic separation	No	Retuning per payload
Luo et al. [78]	Multi-robot (3)	Partial	Critic-only	UUB	Target circumnavigation	No	Known target kinematics
Zhang et al. [79]	Multi-robot	Partial	Critic-only	UUB	Containment control	No	Graph connectivity required

3.1. Manipulators

Robotic manipulators were among the earliest platforms for ADP research, with work spanning trajectory tracking and force regulation under model uncertainty and actuator constraints.

3.1.1. Dynamics of Manipulators

Consider an n-DOF manipulator governed by the standard rigid-body dynamics

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau, \tag{22}$$

where $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$ are the joint position, velocity, and acceleration; $M(q) \in \mathbb{R}^{n\times n}$ is the symmetric positive definite inertia matrix, $C(q,\dot{q})\dot{q}$ represents Coriolis and centrifugal terms, $G(q)$ is the gravitational torque, and $\tau \in \mathbb{R}^n$ is the control input.

The goal is to design a control law $\tau = u(q,\dot{q})$ such that the manipulator either tracks a desired trajectory $q_d(t)$:

$$\lim_{t\to\infty} \big(q(t) - q_d(t)\big) = 0, \tag{23}$$

or stabilizes to a desired equilibrium configuration q^*:

$$\lim_{t\to\infty} q(t) = q^*. \tag{24}$$

3.1.2. Architecture Evolution and Design Trade-offs

The basic design trade-off in manipulator ADP concerns the balance between generality and tractability. Three main categories of architecture have evolved in the research community, namely dual network actor–critic designs [65,66], single-critic design [33,67,80], and decentralized design [68–71].

Dual-Network Actor–Critic

The manipulation ADP methods were first developed based on a traditional DHP-based actor-critic approach with a critic learning the value gradient $\nabla_x V(x)$, and an actor producing the control action [65,66]. Both networks are learned in an online fashion, although partial model information (the dynamic model for predicting the next state in [65] and the moment of inertia $M(q)$ for torque transformation in [66]) is used. An output from the actor network is augmented by a PD controller and Lyapunov-based supervisory terms to guarantee stability at the early stages of learning. The schematic representation of this learning framework is given in Figure 3. The direct gradient learning aligns closely to the HJB equation in continuous time. Hence, more accurate policies can be obtained compared with value-only critics. However, higher levels of gradient noise in the critic and sensitivity to approximation errors may lead to oscillatory torque generation during exploration. Such effects become critical in assembly operations when there is a risk of gear destruction.

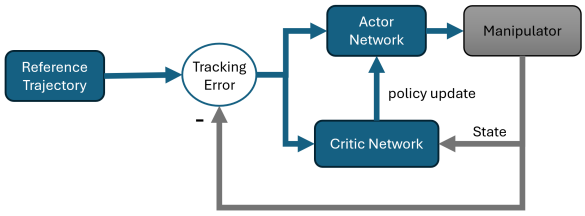

Figure 3. Simplified model-free ADP architecture used in [66].

Single-Critic (Actor-Free)

A common issue with dual-network architectures is the computational complexity and potential instability that arises with the need to simultaneously update the actor and critic networks. Some researchers have overcome this by eliminating the actor and solving for the controller by directly satisfying the HJB equation [33,67,80]. In such a system, only a critic network is required to approximate the value function from which the optimal torque control input can be calculated. Li et al. [33] give a representative application of this method to a 3-DOF planar manipulator accounting for constraints, based on the use of nominal dynamics (M_0, C_0, G_0). According to Li et al. [33], an augmented state $x = [z,\ e]^\top$ can be derived from the tracking error $z = q - q_r$, where q_r results from the Closed-Loop Inverse Kinematics block through the damped pseudo-inverse $J_c^+ = J_c^\top (J_c J_c^\top + \delta I)^{-1}$, via a sliding surface $e = \Lambda z + \dot{z}$. A single Radial Basis Function (RBF) critic then approximates

$$\hat{J}(x) = \hat{W}^\top S(x), \qquad \nabla \hat{J}(x) = \nabla S(x)^\top \hat{W}, \tag{25}$$

with $S(x)$ containing radial basis functions and $\hat{W}$ being the estimate of the adaptive weights. The resultant $\nabla \hat{J}(x)$ will be directly used in the optimal control law. Prior to applying the torque, joint constraints are considered using a state-dependent input weighting $R(q)$ that penalizes cost near joint bounds, and the torque is passed through an asymmetric saturation model:

$$u(\tau) = \begin{cases} \gamma\left(e^{\tau - \theta u_{\max}} - 1\right) + \theta u_{\max}, & \tau > \theta u_{\max}, \\ \gamma\left(1 - e^{\theta u_{\min} - \tau}\right) + \theta u_{\max}, & \tau < \theta u_{\min}, \\ \tau, & \text{otherwise.} \end{cases} \tag{26}$$

This entire control scheme is illustrated in Figure 4. Such a single critic approach can significantly reduce the learning variance, and is hence more suitable for embedded controllers with higher bandwidth. Nonetheless, such an approach needs the gradients of the value function to be smooth, making its application to contact-intensive environments, where dynamics become discontinuous such as grinding and insertion tasks, less suitable. A different issue tackled by Zhang et al. [67] is combining critic-only configurations with sliding mode control for fault-tolerant multi-manipulator tracking.

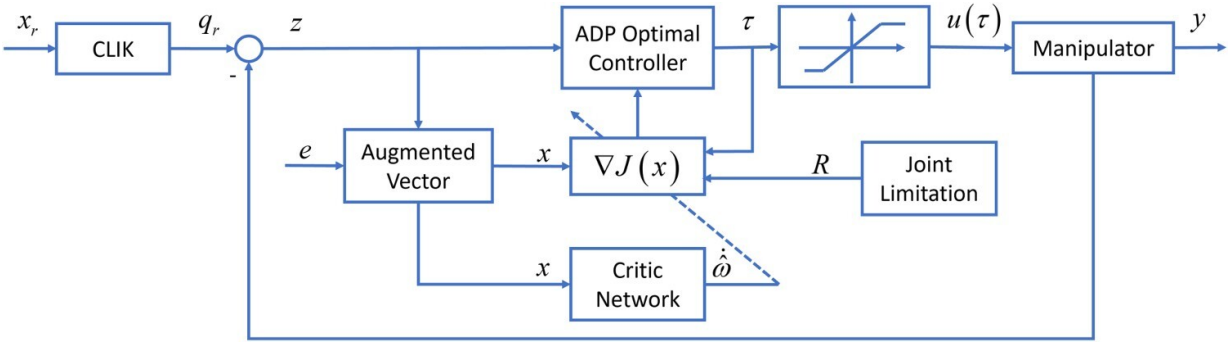

Figure 4. Single Network ADP in manipulator control diagram [33].

Decentralized Architectures

With more DOF manipulators, centralized controllers have to deal with exponentially increasing number of inputs. In the context of decentralized ADP methods, the control task is broken down into separate sub-problems for each joint i,

$$M_i(q)\ddot{q}_i + C_i(q,\dot{q})\dot{q}_i + G_i(q) + \sum_{j \neq i} h_{ij}(q,\dot{q},\ddot{q}) = \tau_i, \tag{27}$$

where h_{ij} denotes inter-joint coupling. Sveen et al. [71] define each of the local torques as $\tau_i = Y_i\hat{\theta}_i - K_i s_i$, with Y_i being the regression matrix, $\hat{\theta}_i$ being the adaptive estimate of the parameter vector, and $s_i = \dot{e}_i + \lambda_i e_i$ as the filtered sliding variable. The ADP part relies on offline policy value function estimation using recorded data and tested in a 4-DOF Quanser Qarm model. The method still needs an exploratory phase with noise to ensure full rank in critic convergence, and sensitivity to initial conditions and exploration noise remains a practical challenge. The simplified schematic of this architecture is given in Figure 5. This framework has been extended in many ways. First, Yang and He [68] provided a proof for the asymptotic stability of decentralized ADP when there are mismatches in the interconnection topology, although the method requires computing the Moore-Penrose pseudo-inverse of the control matrix beforehand. Secondly, Zhao and Liu [69] proposed an event-triggered technique that allows for reduced computation and communication cost in modular reconfigurable robots with asymptotic stability. Lastly, Zhou et al. [70] showed actuator fault tolerance in a critic-based decentralized control system applied to a 2-DOF physical modular robot platform. The main constraint that exists with all forms of decentralization is that of bounded coupling assumption. Where there is heavy coupling between joints, for instance in the case of humanoid wrists or dexterous hands, performance becomes compromised.

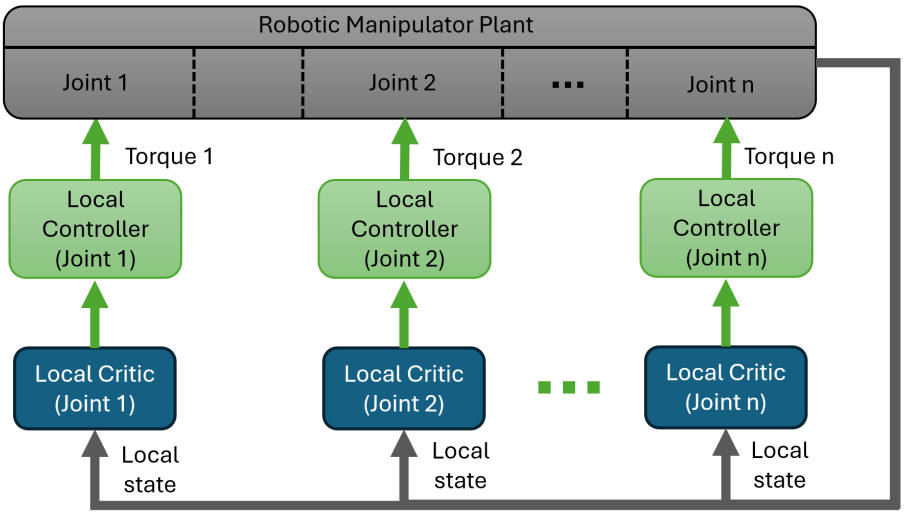

Figure 5. Simplified decentralized ADP architecture used in [71].

3.1.3. Common Assumptions and Their Limitations

Some assumptions appear repeatedly throughout the literature on manipulator ADP. First, the most common one is the assumption that the system dynamics satisfy the condition of Lipschitz continuity. This supports the argument about neural networks' universal approximation capability used in almost all ADP stability proofs [33, 66, 71]. However, while it can be applied to the case of a rigid body manipulator, it is not applicable to cases with a lot of contact points where dynamics have intrinsic discontinuities. Another assumption involves knowledge about the model. Even though the approach is referred to as "model-free," some manipulator ADP algorithms utilize partial information about the model, either in terms of the dynamics model used for state prediction [65], inertia matrix for torque transformation [66] or nominal dynamic model (M_0, C_0, G_0) [33]. The degree of model dependence varies across methods and should be considered when evaluating their practical applicability.

The second popular assumption is that of having complete state information (q, $\dot{q}$), which is the case in all studied manipulator works. However, practical velocity data could be corrupted, or even missing, necessitating the use of state observers, which inevitably incur an extra approximation error that was not considered during the stability proof. Regarding the learning part, PE or similar rank conditions are often needed for the critic weights to converge [65, 66, 71]. This condition can be met by adding perturbation to the control input. While this approach is reasonable in simulations, it poses safety risks in real-world applications.

The decentralized approach also requires that there exists bounded inter-joint coupling ($\|h_{ij}\| \leq \bar{h}$) such that the joints can learn independently of one another [68, 71]. While this is valid for loosely coupled industrial manipulators, it does not hold for highly coupled parallel manipulators and anthropomorphic hands, where coupling forces dominate the dynamics.

3.1.4. Stability Guarantees

The stability guarantees in the manipulator ADP literature range widely. In the majority of works, such as [33,65–67,70,71], the authors promise only UUB. UUB implies the convergence of tracking error to a bounded residual set with an error radius determined by the neural network approximation error, which is theoretically proven but hardly ever quantitatively assessed in the literature. More promising results are obtained by Yang and He [68] and Zhao and Liu [69]. In both works, asymptotic stability is proven for decentralized and event-triggered architectures respectively. Another remarkable work is Wang et al. [81], where the authors obtain finite-time stability us-

ing backstepping in conjunction with Pareto-optimized ADP. Nevertheless, designs for finite-time stability demand non-Lipschitz terms, which might lead to chattering when implemented in practice. Despite the differences in the approaches utilized in each case, the stability analysis in all of the reviewed methods is conducted under the assumption of bounded approximation error without discussing scaling properties of the error with respect to the dimensionality of the problem. It creates a gap between the theoretical and practical aspects of high-DOF system.

3.1.5. Validation Status

There is a gap between the theoretical developments and experimental verification of manipulator ADP algorithms. While several of them have only been implemented using simulations for 3-DOF rigid or planar manipulators [33,67–69], some have been validated by experiments carried out in physical systems. For example, dual-DHP control strategies have been experimentally tested on a Scorbot-ER 4pc (Figure 6) by Gierlak [65] and Szuster [66]. In addition, Zhou et al. [70] performed experiments on a two-link manipulator system with decentralized fault-tolerant control, whereas Sveen et al. [71] used off-policy decentralized learning in a 4-DOF Quanser Qarm. These systems, although validated in their corresponding experiments, still consist of low-DOF (2–4 joints). Furthermore, there is no test for different disturbances, varying payloads, and environmental changes encountered during the application of such an algorithm in industrial settings. The major impediments to the implementation of the approach physically include possible generation of mechanical wear due to the occurrence of learning transients, computational overhead due to online updating of the neural network at servo frequencies ($\geq$1 kHz), and the necessity for exploration noise violating safety conditions.

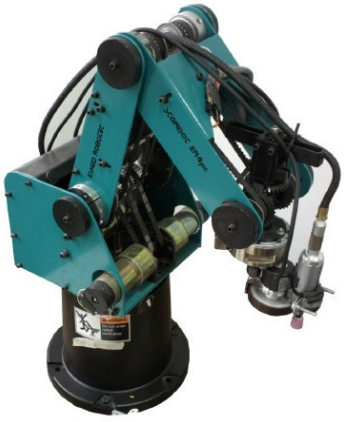

Figure 6. Scorbot-ER 4pc manipulator used for hardware validation of dual-DHP ADP tracking control [66].

3.2. MWR

For MWRs, ADP has been applied in order to address some of the problems associated with input delay, nonholonomic constraints, and multi-body interaction. The research can be classified into three categories depending on the scale of robot operation, namely single-robot tracking [72–76], articulated robots [77], and multiple robots [78,79].

3.2.1. Dynamics of MWR

The kinematic model of the differential drive MWR relates the pose of the robot $\xi = [x,\ y,\ \theta]^\top$ to wheel velocities through

$$\dot{\xi} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{28}$$

where v and ω denote the linear and angular velocities. When dynamic effects are significant, the model extends to include actuator dynamics:

$$M_w \dot{v}_w + C_w v_w + D_w v_w = \tau_w, \tag{29}$$

where M_w, C_w, and D_w represent inertia, Coriolis, and friction effects, respectively, and τ_w is the control input.

Unlike manipulators, MWRs function under nonholonomic constraints such that they cannot move laterally directly. Other complications like wheel slippage, time-varying loads, and communication delay are experienced when these systems are put into practice. The need for model-independent controller design arises out of these issues.

3.2.2. Architecture Evolution: From Single-Robot to Multi-Robot

Single-Robot Tracking

In the case of single MWRs, ADP designs have evolved to accommodate the limitations imposed by platforms. Robust single-network schemes with visual feedback [72] have been demonstrated on real-world wheeled robot hardware, and DHP-based schemes were designed for specific platforms like magnetic wheeled robots [73]. Contemporary developments include efforts to achieve fixed-time convergence using a critic-only scheme with a known dynamics matrix B, this approach is validated on the QBot 3 platform [74]. Li et al. [76] integrated a model-action-critic structure with artificial potential fields for obstacle-rich environments.

An especially significant practical issue is that of input delay due to sampling, communication, and computation lags, which may cause instability in fast-tracking systems. Li et al. [75] addressed this problem through an actor-critic framework equipped with a resistance neural network, incorporating the input delay system within the ADP objective function. The objective function consists of four categories of terms. The first category represents penalties associated with the current pose, tracking error, and control action at time step n,

$$J_{\text{now}} = \xi^T(n)\, Q_1\, \xi(n) + \Delta^T(n)\, Q_0\, \Delta(n) + \tau^T(n)\, R_1\, \tau(n). \tag{30}$$

The second group penalizes the delayed state and the delayed control input, where $T_{1,0} + T_{1,n}$ and $T_{2,0} + T_{2,n}$ denote the total state and input delays,

$$J_{\text{del}} = \xi\big(n-(T_{1,0}+T_{1,n})\big)\, Q_3\, \xi^T\big(n-(T_{1,0}+T_{1,n})\big) + \tau^T\big(n-(T_{2,0}+T_{2,n})\big)\, R_3\, \tau\big(n-(T_{2,0}+T_{2,n})\big). \tag{31}$$

The third group captures cross-coupling between the current and delayed signals,

$$J_{\text{cross}} = 2\,\xi^T(n)\, Q_2\, \xi^T\big(n-(T_{1,0}+T_{1,n})\big) + 2\,\tau^T(n)\, R_2\, \tau\big(n-(T_{2,0}+T_{2,n})\big). \tag{32}$$

The full recursive cost is then $J(\xi(n)) = J_{\text{now}} + J_{\text{del}} + J_{\text{cross}} + J(\xi(n+1))$. The neural network critic estimates the resultant value function, whereas the Lyapunov-Krasovskii functional provides UUB stability to the closed-loop system [82]. This approach shows one such technique which is quite typical in ADP for a single robot scenario, where the practical constraint is resolved through enhancement of the cost function instead of the learning structure. This enables real-time delay compensation without explicit delay models.

Articulated Systems

In both logistics and agriculture, wheeled vehicles are usually articulated into units such as tractor trailers, which makes the task of modeling the multi-body dynamics extremely hard due to the coupled nature. Ghasemzadeh and Amjadifard [77] overcome this challenge by separating kinematic and dynamic control layers. The critic part learns the optimal cost function, whose weights are tuned using the equation below:

$$\dot{\hat{W}} = -\alpha_c \frac{\phi}{(1+\phi^\top\phi)^2}\left(\phi^\top\hat{W} + r(e,\tau)\right). \tag{33}$$

The stability of the system is guaranteed using the Lyapunov analysis method. Even though the drift dynamics $f(X)$ are treated as not known, it is assumed that the input dynamics $g(X)$ are known to the controller. The partial model knowledge reduces the necessity for accurate kinematic modeling of the tractor-trailer systems. However, the training regime demands much room for maneuvers, and the resulting control policy needs to be reconfigured for different payloads.

Multi-Robot Coordination

In cooperative scenarios, ADP needs to be generalized to a multi-agent setting. There have been two proposed methods. One of them is called distributed containment control [79] in which follower agents converge to and stay inside a convex hull defined by the leaders, through a critic-only architecture that learns an approximate solution of the HJB equations. Another one is called moving-target circumnavigation [78], in which a group of three agents keeps a constant angular speed while circumnavigating a target:

$$e_v = v - v_d, \quad v_d = \omega_d \begin{bmatrix} -(y-y_t) \\ (x-x_t) \end{bmatrix} + \dot{p}_t. \tag{34}$$

In both scenarios, backstepping is used to take care of the kinematics of the robots, while critic-only ADP network solves the HJB equation for dynamics of the robots. The transition of ADP from a single robot to multiple robots brings about a qualitatively different set of challenges. The distributed training must be performed with restricted communication between agents while the value function becomes increasingly complex for larger fleets, which motivate graph-based or consensus driven architectures [79].

3.2.3. Common Assumptions and Practical Constraints

In MWR research, some key underlying assumptions are worthy of consideration. Bounded disturbances are assumed in single-robot schemes, with most schemes depending on partial models of the robot dynamics such as input dynamics matrix [74] or system structure [72,73] while treating wheel-ground contact dynamics as unknown and needing compensation. In delay-aware models, bounded and slowly changing delay is considered [75], this assumption could be challenged by network switching during operations in factory-like settings.

When multiple robots work together, the assumption made is that the interconnectivity graph is connected, and there is sufficient information regarding the relative state of other agents [78,79]. The assumptions do not consider agent disconnection or network issues that might result in loss of connectivity. These infrastructure-related assumptions highlight significant discrepancies between theoretical achievements and field application.

Platform-specific modeling aspects may also limit the applicability of conclusions from the reviewed literature. The magnetic wheels used in [73] operate very differently compared to standard rubber wheels, meaning that the learned policies cannot be easily applied across platforms.

3.2.4. Stability and Robustness Guarantees

The stability results for MWR ADP with regard to the manipulator are similar in nature although extended to include multi-robots systems. In particular, the techniques for ensuring single robot UUB stability rely on traditional Lyapunov functions [72,73] or Lyapunov-Krasovskii functional if delays exist [75]. The work in [74] by Wang et al. is important in that it demonstrates fixed-time stability where tracking errors become convergent under bounded time independent of initial states.

Regarding articulated robots, Ghasemzadeh et al. [77] ensure Lyapunov stability and convergence of weights. However, Lyapunov stability is only proven under the assumption that the separation of kinematics and dynamics is maintained which is not always possible in aggressive movements like jackknife recovery.

The stability problem in multi-robot systems is much more complicated as it is now dependent on factors such as the dynamics of learning and communication graph structure. In both references [78,79], UUB can be guaranteed but under assumptions regarding graph connectivity and the bounding of target movement which may pose challenges in scaling beyond 5–10 robots.

3.2.5. Validation Status

From the aforementioned MWR studies, some of them have been physically implemented on actual hardware. Specifically, Luy et al. [72] implemented their proposed single-network method on a physical wheeled robot system, and Wang et al. [74] showed the feasibility of fixed-time tracking on the QBot 3 platform. In contrast, the rest of the single-robot methods [73,75,76], articulated robots [77], and multi-robot methods [78,79] are only studied using simulation. In the case of multi-robot cooperation, this gap is quite large because the simulations are based on ideal assumptions that include accurate localization, no communication delays, and similar robot dynamics. Closing this gap would require incorporating methods for robust perception and communication-dependent learning.

4. Challenges and Future Directions

This section identifies key challenges limiting the practical deployment of ADP in robotic systems and discusses potential mitigations and future research directions.

4.1. Critical Challenges and Mitigations

4.1.1. Scalability to High-Dimensional Systems

Centralized critic networks suffer from exponentially increasing computational complexity with the dimensionality of the state space, making them impractical for high-DOF systems such as multi-robot wheel systems with more than 10 robots and high-DOF manipulator systems. To overcome this, decentralized ADP systems have been proposed [68,71], where the original problem is divided into subproblems, each solved locally. However, existing decentralized ADP algorithms assume bounded inter-subsystem couplings, making them impractical for systems with high couplings. Future research directions could include the study of hierarchical subproblem decomposition [83], the application of graph neural networks to exploit sparsity in the state space of multi-robot systems [84], and the incorporation of attention mechanisms to focus the learning on the relevant subspaces of the state space [85].

4.1.2. Certified Safety Guarantees

Existing ADP algorithms are based on soft constraints, where the constraints are enforced using cost functions [33] or include penalty functions that are barrier-like. However, this does not guarantee constraint satisfaction in safety-critical applications such as multi-robot wheel systems, where robots must operate safely near humans, and collaborative manipulator systems, where deterministic constraint satisfaction is required. To overcome this, CBFs [58] can be incorporated into the ADP framework, providing a certificate of safety while maintaining the optimality of the solution [59], as well as predictive safety filters, where the control actions are overridden when the constraints are violated. Recent work by Qin et al. moves in this direction. Qin et al. [86] addressed interconnected nonlinear systems under discontinuous state constraints by designing a boundary reconstruction function and embedding a Barrier Lyapunov Function within the ADP cost, combined with a dynamic event-triggered mechanism. Qin et al. [87] tackled secure tracking control for interconnected systems with mismatched uncertainties by embedding a CBF directly into the cost function and formulating the problem as an event-triggered zero-sum game solved via parallel learning. Together, these works offer a template for coupling learned value functions with hard-constraint certificates rather than relying on post-hoc filters.

4.1.3. Sample Efficiency and Convergence Speed

Model-free ADP generally requires thousands of iterations to converge, which may not be suitable for online learning on physical robots, where each sample must be executed in real time and may cause physical damage. Mechanisms for experience replay [24] can be used to improve the efficiency of the data used, and event-triggered ADP [29,31] can be used to update the policy less frequently without compromising performance. Meta-learning strategies [35] for the critic network using data from similar tasks and offline RL strategies for the agent using logged data before online fine-tuning are promising future research directions.

4.1.4. Robustness to Model Uncertainty

Although the IRL-based ADP [23, 24] eliminates the need for knowledge of the dynamics of the system, the learned policy may still perform poorly if the operational conditions change significantly from those in the training data, for example, in the presence of payloads, friction coefficients, and temperature effects. H_∞-ADP [25,26] can be used to improve the robustness of the policy to disturbances in the system, as it models the disturbances as a game. However, H_∞-ADP is inadequate because the H_∞ performance guarantee is dependent on disturbance assumptions, the min-max principle is conservative for nominal performance, and safety constraints such as saturation, collision avoidance, or state constraints are not inherently satisfied. In the future, the policy's performance in the presence of disturbances should be quantified and bounded.

4.1.5. Real-Time Implementation

For using ADP on physical robots, some compromises have to be done in terms of computation rates, power consumption, and deterministic latency [88]. For low DOF systems, with relatively simple critics, embedded CPU processors can be employed. For more complex critics, one may utilize edge GPUs [89], albeit at the expense of greater power consumption. Field-programmable gate arrays (FPGA) allow for high-frequency deterministic execution but cannot adapt with architectural changes [90]. The next step should be to perform benchmarks for ADP inference on these platforms, including a hybrid approach with an FPGA for real-time inference and a GPU for periodic weight updates.

4.1.6. Sim-to-Real Gap

As mentioned in Sections 3.1.5 and 3.2.5, although some studies of ADP discussed have performed experiments on hardware using low-DOF systems, most algorithms have been tested exclusively using simulation-based methods. Simulated policies can perform poorly when transferred to the hardware because of frictional forces, backlash, sensor latency, communication jitter, and payload variations that were not considered during training in the simulated environment. Bridging this research gap is one of the key issues in data-driven ADP, and some promising approaches in this area include domain randomization for critic pre-training, residual online adaptation layered on top of a simulation-warm-started policy, and benchmarks using hardware-in-the-loop testing.

4.2. Emerging Research Directions

4.2.1. Contact-Rich Manipulation

Extension of ADP to problems of intermittent contacts, such as assembly and grinding, is still difficult to handle due to the discontinuous nature of the motion. Hybrid ADP formulations, in which the control between contact and free motion is switched, along with the integration of force feedback, could be a solution to this problem.

4.2.2. Perception-Driven Control

Currently, ADP algorithms rely on ground truth information about the states. Integration with vision pipelines, handling occlusions during state estimation, and learning from high-dimensional sensory modalities such as images and point clouds are unexplored areas that could expand the scope of ADP.

4.2.3. Multi-Agent Coordination

To expand the applicability of ADP to cooperative and competitive multi-robot scenarios, distributed learning of the value functions with limited communication must be achieved. Multi-agent Hamilton–Jacobi–Isaacs (HJI) formulations have been proposed in [78,79], which provide a theoretical foundation. However, designing algorithms for more than 10 robots remains an open challenge.

4.2.4. Standardized Benchmarks for ADP

While deep RL methods enjoy the advantage of having a set of established benchmarks, which include OpenAI Gym and MuJoCo, for evaluating the effectiveness of their algorithms, there is no standard benchmark suite that the ADP field can utilize. The evaluated papers have employed a diverse range of measures for their algorithm's convergence, such as simulation time, iteration count, or time steps, in various dynamic systems and dimensionalities. These differences pose challenges to comparing one method against another, making replication challenging. Development of open-source robotics ADP benchmarks with standardized task descriptions and convergence criteria will help advance research and facilitate the evaluation of competing paradigms.

5. Conclusions

In this survey, recent advances in ADP applied to robotic manipulators and MWRs were presented through a robotics-focused lens. In contrast to previous surveys which tend to structure the literature into algorithmic approaches or problems addressed within the control system, this survey emphasizes the way characteristics and physical constraints of the robot platform itself impact the development of ADP. For both robotic manipulator systems and MWRs, the literature indicates that ADP is a suitable approach to reconcile optimal control theory and machine learning thanks to its ability to adapt online and to rely on Lyapunov-stability-based proofs of convergence. However, there are some consistent limitations to consider. The majority of existing approaches rely on near-optimal guarantees such as UUB rather than asymptotic stability, their hardware implementation is only demonstrated on smaller-scale platforms, and practical issues of scalability, safe learning, sample-efficiency, and real-time application are left unanswered. In general, the ADP technique remains a good approach towards designing an effective robot controller, and future development is contingent on overcoming the barrier that separates theory and implementation by developing safe, efficient, and hardware-friendly learning algorithms.

Author Contributions

Conceptualization, C.P.C. and Z.T.; Methodology, C.P.C., N.B. and Q.Y.; Investigation and literature review, C.P.C., N.B. and Q.Y.; Writing—original draft preparation, C.P.C.; Writing—review, editing, and proofreading, C.P.C, N.B., Q.Y., J.W., B.H., Y.Y. and Z.T.; Discussion and feedback, Z.T., J.W. and Y.Y.; Supervision, Z.T. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

Not applicable.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

No AI tools were utilized for this paper.

Nomenclature

ADP	Adaptive Dynamic Programming
DP	Dynamic Programming
RL	Reinforcement Learning
HJB	Hamilton-Jacobi-Bellman
HJI	Hamilton-Jacobi-Isaacs
H_∞	disturbance attenuation/robust control
HDP	Heuristic Dynamic Programming
DHP	Dual Heuristic Programming
GDHP	Globalised Dual Heuristic Programming
ADHDP	Action-Dependent Heuristic Dynamic Programming
VI	Value Iteration
PI	Policy Iteration
IRL	Integral Reinforcement Learning
TD	Temporal Difference
CT	Continuous Time
DT	Discrete Time
RBF	Radial Basis Function
FPGA	Field-Programmable Gate Array
DOF	Degrees of Freedom
UAV	Unmanned Aerial Vehicle
MDP	Markov Decision Process
UUB	Uniform Ultimate Boundedness
CBF	Control Barrier Function
MWR	Mobile Wheeled Robot
PE	Persistent Excitation
NDP	Neural Dynamic Programming
ADDHP	Action-Dependent Dual Heuristic Programming
ADGDHP	Action-Dependent Generalized Dual Heuristic Dynamic Programming

References

1. Truong, X.; Ngo, T. Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model. *IEEE Trans. Autom. Sci. Eng.* **2017**, *14*, 1743–1760.
2. Coelho, D.; Oliveira, M. A review of end-to-end autonomous driving in urban environments. *IEEE Access* **2022**, *10*, 75296–75311.
3. Gehring, C.; Fankhauser, P.; Isler, L.; et al. Anymal in the field: Solving industrial inspection of an offshore HVDC platform with a quadrupedal robot. In Proceedings of the Field and Service Robotics: Results of The 12th International Conference, Tokyo, Japan, 29–31 August 2019; pp. 247–260.
4. Bualat, M.; Barlow, J.; Fong, T.; et al. Astrobee: Developing a free-flying robot for the international space station. In Proceedings of the AIAA SPACE 2015 Conference and Exposition, Pasadena, CA, USA, 31 August–2 September 2015; p. 4643.
5. Sutton, R.; Barto, A. *Reinforcement Learning: An Introduction*; MIT press: Cambridge, UK; 1998; Vol. 1.
6. Kober, J.; Bagnell, A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274.
7. Kaelbling, L.; Littman, M.; Moore, A. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285.
8. Puterman, M. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
9. Mnih, V.; Kavukcuoglu, K.; Silver, D.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.
10. Schulman, J.; Wolski, F.; Dhariwal, P.; et al. Proximal policy optimization algorithms. *arXiv* **2017**. *preprint*, arXiv:1707.06347.
11. Garcıa, J.; Fernández, F. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **2015**, *16*, 1437–1480.
12. Recht, B. A tour of reinforcement learning: The view from continuous control. *Annu. Rev. Control Robot. Auton. Syst.* **2019**, *2*, 253–279.
13. Liu, D.; Xue, S.; Zhao, B.; et al. Adaptive dynamic programming for control: A survey and recent advances. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *51*, 142–160.
14. Kiumarsi, B.; Vamvoudakis, K.; Modares, H.; et al. Optimal and autonomous control using reinforcement learning: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 2042–2062.
15. Bellman, R.; Corporation, R.; Collection, K.M.R. *Dynamic Programming*; Rand Corporation research study, Princeton University Press: Princeton, NJ, USA, 1957.
16. Powell, W. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*; John Wiley & Sons: Hoboken, NJ, USA, 2007; Vol. 703.
17. Werbos, P. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD Thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA, USA, 1974.
18. Prokhorov, D.; Wunsch, D. Adaptive critic designs. *IEEE Trans. Neural Netw.* **1997**, *8*, 997–1007.
19. Sutton, R. Learning to predict by the methods of temporal differences. *Mach. Learn.* **1988**, *3*, 9–44.
20. Bertsekas, D.; Tsitsiklis, J. Neuro-dynamic programming: An overview. In Proceedings of 1995 34th IEEE Conference on Decision and Control, New Orleans, LA, USA, 13–15 December 1995; Vol. 1, pp. 560–564.
21. Lewis, F.; Liu, K.; Yesildirek, A. Neural net robot controller with guaranteed tracking performance. *IEEE Trans. Neural Netw.* **1995**, *6*, 703–715.
22. Doya, K. Reinforcement learning in continuous time and space. *Neural Comput.* **2000**, *12*, 219–245.
23. Vrabie, D.; Pastravanu, O.; AbuKhalaf, M.; et al. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica* **2009**, *45*, 477–484.
24. Modares, H.; Lewis, F.; NaghibiSistani, M. Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica* **2014**, *50*, 193–202.
25. Lv, Y.; Na, J.; Ren, X. Online H_∞ control for completely unknown nonlinear systems via an identifier-critic-based ADP structure. *Int. J. Control* **2019**, *92*, 100–111.
26. Tang, Z.; Rossiter, J.A.; Panoutsos, G. A reinforcement learning-based approach for optimal output tracking in uncertain nonlinear systems with mismatched disturbances. In Proceedings of the 2024 UKACC 14th International Conference on Control (CONTROL), Winchester, UK, 10–12 April 2024; pp. 169–174.
27. Tang, Z.; Rossiter, J.A.; Dong, Y.; et al. Reinforcement learning-based output stabilization control for nonlinear systems with generalized disturbances. In Proceedings of the 2024 IEEE International Conference on Industrial Technology (ICIT), Bristol, UK, 25–27 March 2024; pp. 1–6.
28. Qin, C.; Pang, M.; Wang, Z.; et al. Event-Triggered Zero-Sum Game for Input Saturated Nonlinear Systems with State Con-

straints. *IEEE Trans. Consum. Electron.* **2026**, *72*, 455–465.
29. Wang, N.; Jia, W.; Wu, H.; et al. Event-triggered self-organizing swarm control of distributed unmanned surface vehicles. *IEEE Trans. Intell. Transp. Syst.* **2024**, *26*, 3431–3445.
30. Dong, L.; Zhong, X.; Sun, C.; et al. Event-triggered adaptive dynamic programming for continuous-time systems with control constraints. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 1941–1952.
31. Bai, N.; Chan, C.P.; Yin, Q.; et al. Deep Reinforcement Learning Optimization for Uncertain Nonlinear Systems via Event-Triggered Robust Adaptive Dynamic Programming. *arXiv* **2025**, *preprint*, arXiv:math/2512.15735.
32. Qin, C.; Pang, M.; Wang, Z.; et al. Observer based fault tolerant control design for saturated nonlinear systems with full state constraints via a novel event-triggered mechanism. *Eng. Appl. Artif. Intell.* **2025**, *161*, 112221.
33. Li, B.; Zhan, H.; Zhang, X.; et al. Single-network based adaptive dynamic programming optimal control for robotic manipulator with joint limitation under input saturation. *Int. J. Syst. Sci.* **2024**, *55*, 3355–3370.
34. Xue, S.; Liu, Z.; Wang, L.; et al. Adaptive dynamic programming based event-triggered multi-H_∞ control. *Neurocomputing* **2025**, *638*, 130157.
35. Wang, Q.; Zhang, C.; Hu, B. Dynamic programming with meta-reinforcement learning: A novel approach for multi-objective optimization. *Complex Intell. Syst.* **2024**, *10*, 5743–5758.
36. Lin, W.; Yang, P. Adaptive critic motion control design of autonomous wheeled mobile robot by dual heuristic programming. *Automatica* **2008**, *44*, 2716–2723.
37. Szuster, M.; Hendzel, Z. Discrete Globalised Dual Heuristic Dynamic Programming in Control of the Two-Wheeled Mobile Robot. *Math. Probl. Eng.* **2014**, *2014*, 628798.
38. Stojanović, V.; Djordjević, V.; Dubonjić, L.; et al. ADP-based control of a two-wheeled self-balancing mobile robot. *Eng. Today* **2024**, *3*. https://doi.org/10.5937/engtoday2400018S.
39. Azimi, A.; Shamshiri, R.; Ghasemzadeh, A. Adaptive dynamic programming for robust path tracking in an agricultural robot using critic neural networks. *Agric. Eng.* **2025**, *80*. https://doi.org/10.15150/ae.2025.3327.
40. Liu, X.; Sun, Y.; Han, Q.; et al. A novel adaptive dynamic optimal balance control method for wheel-legged robot. *Appl. Math. Model.* **2025**, *137*, 115737.
41. Liang, J.; Tang, S.; Jia, B. Control of Parallel Quadruped Robots Based on Adaptive Dynamic Programming Control. *Machines* **2024**, *12*, 875.
42. Wei, X.; Ye, J.; Xu, J.; et al. Adaptive dynamic programming-based cross-scale control of a hydraulic-driven flexible robotic manipulator. *Appl. Sci.* **2023**, *13*, 2890.
43. Xia, H.; Xue, D.; Wang, Y.; et al. Critic-Identifier Structure ADP Based Near-optimal Decentralized Tracking Control of Modular and Reconfigurable Robots. In Proceedings of the 2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC), Jinzhou, China, 6–8 June 2019; pp. 440–445.
44. Tang, Y.; Dou, L.; Zhang, R.; et al. Adaptive dynamic programming–based fault-tolerant control of multi-UAV formation system. *Trans. Inst. Meas. Control* **2025**, *47*, 1893–1905.
45. Luo, Z.; Zhang, P.; Ding, X.; et al. Adaptive affine formation maneuver control of second-order multi-agent systems with disturbances. In Proceedings of the 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV), Shenzhen, China, 13–15 December 2020; pp. 1071–1076.
46. Liu, H.; Yi, X.; Liu, D.; et al. A review of unmanned vehicle control with adaptive dynamic programming implementations. *J. Intell. Robot. Syst.* **2025**, *111*, 10.
47. Lewis, F.; Vrabie, D. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **2009**, *9*, 32–50.
48. AbuKhalaf, M.; Lewis, F. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* **2005**, *41*, 779–791.
49. Vamvoudakis, K.; Lewis, F. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **2010**, *46*, 878–888.
50. Jiang, Y.; Jiang, Z. Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica* **2012**, *48*, 2699–2704.
51. Kamalapurkar, R.; Andrews, L.; Walters, P.; et al. Model-based reinforcement learning for infinite-horizon approximate optimal tracking. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 753–758.
52. Khalil, H.K.; Grizzle, J.W. *Nonlinear Systems*; Prentice Hall: Upper Saddle River, NJ, USA, 2002; Vol. 3.
53. Bhat, S.P.; Bernstein, D.S. Finite-time stability of continuous autonomous systems. *SIAM J. Control Optim.* **2000**, *38*, 751–766.
54. Polyakov, A. Nonlinear feedback design for fixed-time stabilization of linear control systems. *IEEE Trans. Autom. Control* **2011**, *57*, 2106–2106.
55. Lakshmikantham, V.; Leela, S.; Martynyuk, A.A. *Practical Stability of Nonlinear Systems*; World Scientific: Singapore, 1990.
56. Brunke, L.; Greeff, M.; Hall, A.W.; et al. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annu. Rev. Control Robot. Auton. Syst.* **2022**, *5*, 411–444.
57. Ames, A.D.; Xu, X.; Grizzle, J.W.; et al. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Autom. Control* **2016**, *62*, 3861–3876.
58. Ames, A.D.; Coogan, S.; Egerstedt, M.; et al. Control barrier functions: Theory and applications. In Proceeding of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 3420–3431.
59. Cheng, R.; Orosz, G.; Murray, R.M.; et al. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Vol. 33, pp. 3387–3395.
60. Taylor, A.; Singletary, A.; Yue, Y.; et al. Learning for safety-critical control with control barrier functions. In Proceedings of the 2nd Conference on Learning for Dynamics and Control, Virtual Event, 10–11 June 2020; pp. 708–717.
61. Wabersich, K.P.; Zeilinger, M.N. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica* **2021**, *129*, 109597.
62. Hewing, L.; Wabersich, K.P.; Menner, M.; et al. Learning-based model predictive control: Toward safe learning in control. *Annu. Rev. Control Robot. Auton. Syst.* **2020**, *3*, 269–296.
63. Bansal, S.; Chen, M.; Herbert, S.; et al. Hamilton-jacobi reachability: A brief overview and recent advances. In Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, VIC, Australia, 12–15 December; pp. 2242–2253.

64. Fisac, J.F.; Akametalu, A.K.; Zeilinger, M.N.; et al. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Trans. Autom. Control* **2018**, *64*, 2737–2752.
65. Gierlak, P.; Szuster, M.; Żylski, W. Discrete dual–heuristic programming in 3DOF manipulator control. In Proceedings of the International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 13–17 June 2010; pp. 256–263.
66. Szuster, M.; Gierlak, P. Approximate dynamic programming in tracking control of a robotic manipulator. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 16.
67. Zhang, X.; Yang, Z.; Liu, H.; et al. Optimal Sliding Mode Fault-Tolerant Control for Multiple Robotic Manipulators via Critic-Only Dynamic Programming. *Sensors* **2025**, *25*, 5410.
68. Yang, X.; He, H. Adaptive dynamic programming for decentralized stabilization of uncertain nonlinear large-scale systems with mismatched interconnections. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *50*, 2870–2882.
69. Zhao, B.; Liu, D. Event-triggered decentralized tracking control of modular reconfigurable robots through adaptive dynamic programming. *IEEE Trans. Ind. Electron.* **2019**, *67*, 3054–3064.
70. Zhou, F.; Nie, F.; An, T.; et al. Decentralized fault tolerant control of modular manipulators system based on adaptive dynamic programming. *Int. J. Control Autom. Syst.* **2022**, *20*, 3252–3253.
71. Sveen, E.M.; Zhou, J.; Ebbesen, M.K.; et al. Decentralised adaptive learning-based control of robot manipulators with unknown parameters. *J. Autom. Intell.* **2025**, *4*, 136–144.
72. Luy, N.T. Robust adaptive dynamic programming based online tracking control algorithm for real wheeled mobile robot with omni-directional vision system. *Trans. Inst. Meas. Control* **2017**, *39*, 832–847.
73. Dian, S.; Fang, H.; Zhao, T.; et al. Modeling and trajectory tracking control for magnetic wheeled mobile robots based on improved dual-heuristic dynamic programming. *IEEE Trans. Ind. Inform.* **2020**, *17*, 1470–1482.
74. Wang, C.; Zhan, H.; Guo, Q.; et al. Adaptive Dynamic Programming-Based Fixed-Time Optimal Control for Wheeled Mobile Robot. *IEEE Robot. Autom. Lett.* **2024**, *10*, 176-183
75. Li, S.; Ding, L.; Gao, H.; et al. ADP-based online tracking control of partially uncertain time-delayed nonlinear system and application to wheeled mobile robots. *IEEE Trans. Cybern.* **2019**, *50*, 3182–3194.
76. Li, X.; Wang, L.; An, Y.; et al. Dynamic path planning of mobile robots using adaptive dynamic programming. *Expert Syst. Appl.* **2024**, *235*, 121112.
77. Ghasemzadeh, A.; Amjadifard, R.; KeymasiKhalaji, A. Adaptive dynamic programming for trajectory tracking control of a tractor-trailer wheeled mobile robot. *IET Control Theory Appl.* **2025**, *19*, e12784.
78. Luo, Y.; Li, Y.; Ding, J.; et al. Optimal moving-target circumnavigation control of multiple wheeled mobile robots based on adaptive dynamic programming. *IEEE Trans. Netw. Sci. Eng.* **2024**, *11*, 4679–4688.
79. Zhang, Y.; Zhao, B.; Liu, D. Distributed optimal containment control of wheeled mobile robots via adaptive dynamic programming. *IEEE Trans. Syst. Man Cybern. Syst.* **2025**, *55*, 5876–5886.
80. Lewis, F.; Vrabie, D.; Syrmos, V. *Optimal Control*; John Wiley & Sons: Hoboken, NJ, USA, 2012.
81. Wang, Y.; An, T.; Dong, B.; et al. Adaptive dynamic programming-based finite-time optimal backstepping force/position control of reconfigurable robot manipulators via Pareto optimal. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 10660-10671.
82. Tang, Z.; Rossiter, J.A.; Jin, X.; et al. Output Tracking for Uncertain Time-Delay Systems via Robust Reinforcement Learning Control. In Proceedings of the 2024 43rd Chinese Control Conference (CCC), Kunming, China, 28–31 July 2024; pp. 2219–2226.
83. Nachum, O.; Gu, S.S.; Lee, H.; et al. Data-efficient hierarchical reinforcement learning. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS 2018), Montréal, Canada, 3–8 December 2018; Vol. 31.
84. Lee, E.S.; Zhou, L.; Ribeiro, A.; et al. Graph neural networks for decentralized multi-agent perimeter defense. *Front. Control Eng.* **2023**, *4*, 1104745.
85. Vaswani, A.; Shazeer, N.; Parmar, N.; et al. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Vol. 30.
86. Qin, C.; Sun, K.; Sun, A.; et al. Dynamic event-triggered optimal safety control of interconnected nonlinear systems with discontinuous state constraints by adaptive dynamic programming. *Neurocomputing* **2025**, *669*, 132467.
87. Qin, C.; Hou, S.; Pang, M.; et al. Reinforcement learning-based secure tracking control for nonlinear interconnected systems: An event-triggered solution approach. *Eng. Appl. Artif. Intell.* **2025**, *161*, 112243.
88. Chen, J.; Ran, X. Deep learning with edge computing: A review. *Proc. IEEE* **2019**, *107*, 1655–1674.
89. Mittal, S. A survey on optimized implementation of deep learning models on the nvidia jetson platform. *J. Syst. Archit.* **2019**, *97*, 428–442.
90. Guo, K.; Zeng, S.; Yu, J.; et al. [DL] A survey of FPGA-based neural network inference accelerators. *ACM Trans. Reconfig. Technol. Syst. (TRETS)* **2019**, *12*, 1–26.