IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

A Differential Dynamic Programming Framework for Inverse Reinforcement Learning

Kun Cao, Xinhang Xu, Wanxin Jin, Karl H. Johansson, Lihua Xie

***Abstract*—A differential dynamic programming (DDP)-based framework for inverse reinforcement learning (IRL) is introduced to recover the parameters in the cost function, system dynamics, and constraints from demonstrations. Different from existing work, where DDP was usually used for the inner forward problem, our proposed framework uses it to efficiently compute the gradient required in the outer inverse problem with equality and inequality constraints. The equivalence between the proposed and existing methods based on Pontryagin's Maximum Principle (PMP) is established. More importantly, using this DDP-based IRL with an open-loop loss function, a closed-loop IRL framework is presented. In this framework, a loss function is proposed to capture the closed-loop nature of demonstrations. It is shown to be better than the commonly used open-loop loss function. We show that the closed-loop IRL framework reduces to a constrained inverse optimal control problem under certain assumptions. Under these assumptions and a rank condition, it is proven that the learning parameters can be recovered from the demonstration data. The proposed framework is extensively evaluated through four numerical robot examples and one real-world quadrotor system. The experiments validate the theoretical results and illustrate the practical relevance of the approach.**

***Index Terms*—Inverse Reinforcement Learning, Inverse Problems, Differential Dynamical Programming, Constrained Optimal Control, Inverse Optimal Control**

I. Introduction

Recent years have witnessed a significant surge in advancements within the field of Reinforcement Learning (RL), which iteratively learns an optimal policy that maximizes a human-designed accumulative reward by repeatedly interacting with the environment, has demonstrated remarkable capabilities in dealing with challenging tasks such as game playing [1], motion planning [2], portfolio optimization [3], and energy system operation [4]. Despite these achievements, one of the principal challenges in RL remains the design of an appropriate cost function that reliably induces desired behaviors, especially for complex and high-dimensional tasks [5]. Typically, the design process of a cost function involves an iterative process of trial and error, requiring substantial manual effort and even strong prior knowledge and expertise.

To address this, the inverse RL (IRL) problem has been proposed to automate the critical task of designing cost functions by learning from the observed behaviors of (possibly non-) experts. Over the past decades, many IRL formulations have been proposed, with different approaches emphasizing different learning criteria. Representative works include apprenticeship learning [6], which matches the feature vectors of demonstration and predicted trajectory, MaxEnt [7], which maximizes the entropy of the trajectory distribution subject to a reward expectation constraint, and Max-margin [8], which maximizes the margin between the objectives of demonstration and predicted trajectory.

Despite different cost update criteria, existing approaches share a common bi-level algorithmic design: the cost function is updated in the outer loop and the corresponding RL is optimized in the inner loop. For the inner level, optimizing an RL agent is primarily driven by agent sampling via interacting with the environment. This sampling-based optimization process may take a large number of training epochs to converge, which ultimately leads to the inefficiency of the entire IRL framework. To alleviate this, the authors in [9] proposed the Pontryagin Differential Programming (PDP) framework (specifically, the IRL mode is referred to throughout the paper), where the inner level uses a parameterized optimal control (OC) problem and can be efficiently solved by a model-based solver. Furthermore, the proposed analytical gradient by differentiating the equilibrium condition (i.e., PMP) of the inner problem makes the end-to-end update of the cost function possible. A similar framework has also been proposed in [10], [11] to consider the IRL where there are stage-wise state and/or control constraints in the inner RL agent.

While the above IRL frameworks building upon a differentiable inner loop achieve computational efficiency, a limitation, which we have empirically observed but have been largely overlooked in [9]–[11], is their imitation-based loss function in the outer level. Specifically, [9]–[12] proposes minimizing a

K. Cao is with the Department of Control Science and Engineering, College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China, the Shanghai Institute of Intelligent Science and Technology, National Key Laboratory of Autonomous Intelligent Unmanned Systems, and Frontiers Science Center for Intelligent Autonomous Systems, Ministry of Education, Beijing 100816, China, and was with School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798 and School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, SE-10044 Stockholm, Sweden caokun@tongji.edu.cn

X. Xu, and L. Xie (corresponding author) are with School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798 xu0021ng@e.ntu.edu.sg; elhxie@ntu.edu.sg

W Jin is with the School for Engineering of Matter, Transport, and Energy, Arizona State University. wanxinjin@gmail.com

K. H. Johansson is with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, and also with Digital Futures, SE-10044 Stockholm, Sweden. kallej@kth.se.

This work was supported in part by the Fundamental Research Funds for the Central Universities, the National Natural Science Foundation of China under Grant 62088101, 62503367, Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0100), Wallenberg-NTU Presidential Postdoctoral Fellowship, Swedish Research Council Distinguished Professor Grant 2017-01078, Knut and Alice Wallenberg Foundation Wallenberg Scholar Grant, and Swedish Strategic Research Foundation FUSS SUCCESS Grant, Ministry of Education, Singapore, under AcRF TIER 1 Grant RG64/23. (corresponding author: Lihua Xie)

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

mean square outer-level loss, which is a discrepancy between the reproduced trajectory and the demonstrations; thus, the IRL formulation can be viewed as a nonlinear least square problem. The use of imitation loss implies that the expert demonstrated trajectory is a result of open-loop control, and that the trajectory data has been polluted by temporally independent noise. However, this assumption may be invalid (we have later shown this analytically and numerically) for observation data generated by the expert with closed-loop policies. In fact, data collection from a closed-loop policy agent is often the case, for better stability and robustness. Due to the nature of the closed-loop policy, the noise along an observed trajectory is not temporally independent. Thus, the choice of imitation loss would lead to bias in the cost function.

In this paper, we rethink the problem of IRL via the differentiation of the inner layer. However, different from existing work, we consider both the inner-level optimal control agent and the outer-level learning loss from a closed-loop perspective. Specifically,

- We formulate a closed-loop optimal control problem at the inner level via the process of DDP, upon which we propose a new way of carrying out the differentiation via the corresponding Bellman optimality equation.
- We propose a new loss function that directly captures the feedback nature of the expert data generation, which leads to unbiased learning of the cost function, compared to the open-loop definition of the loss function.

A. Related Work

Bi-level optimization was first realized in the field of game theory in the seminal work [13] to solve a hierarchical decision-making problem, where the inner-level problem can be defined by different programs [14], such as linear programs, nonlinear programs, games, and multi-stage programs. Generally, there are two classes of approaches for solving this problem. The first one reduces it to a single-level problem by replacing the inner-level problem with its optimality conditions as constraints and then simultaneously updates the outer and inner decision variables. The second approach maintains the bi-level structure and alternates between the updates of the outer and inner decision variables, where the inner-level problem can be solved by existing solvers and the gradient required by the outer level is obtained by differentiating the inner-level equilibrium conditions [9]–[11]. In the spirit of the second approach, this work focuses on developing a new way of efficient differentiation for general constrained OC problems.

The dynamics of inner-level multi-stage programs can be either modeled by a Markov Decision Process (MDP) or a state-space equation. In this paper, we only focus on the deterministic optimal control problem, where the dynamics are modeled by the state-space equation. We categorize existing deterministic optimal control techniques into open-loop methods, which directly solve a trajectory as a function of time, and closed-loop methods, which seek a mapping from current observation to an optimal control action. The first category is based on the PMP [15], which is derived from the calculus of variations. Popular methods include shooting methods [16] and collocation methods [17]. However, these methods optimize based on the initial conditions and hence are susceptible to model errors or disturbances during deployment.

Another category of methods is based on dynamic programming and specifically the Bellman optimality equation [18], which characterizes the mathematical condition that a control input in each step should satisfy w.r.t. the current state, hence it leads to a closed-loop policy. Differential dynamical programming [19] is a numerical algorithm that aims to find the solution to this equation by iteratively quadratizing the cost function and dynamic equation. It enjoys the linear time complexity (w.r.t. horizon) and local quadratic convergence [20]. Subsequently, this algorithm has also been generalized to the case with inequality constraints via three major methods: 1) penalty-based methods (i.e., convert the constrained problems to unconstrained ones, e.g., [21]), which are easy to implement while potentially suffer from ill-conditioning, slow convergence, etc.; 2) active-set methods (i.e., identify the active inequality constraints and then solve the equality-constrained OC problem, e.g., [22]), which can be computationally efficient with accurately predicted or identified active sets, but may suffer from a combinatorial complexity in the worst case; 3) interior-point methods (i.e., introduce a constrained version of the Bellman equation and solve a relaxed equilibrium condition, e.g., [23], [24]), which avoids the modification of the objective function and the identification of active sets and enjoys provable local quadratic convergence. Equality constraints have also been explicitly considered in previous studies: the linearized point-wise dynamic constraints are added to the active inequality constraints in the forward pass [25], which naturally inherits the issue with active-set methods; the augmented Lagrangian formulation has been proposed in [26], which does not come with convergence guarantees; a globalization strategy was adopted in [27] to establish local convergence with a controlled rate for equality constraints, but also suffers from a combinatorial complexity when extended to inequality constraints [28]. In the spirit of the interior-point method for inequality constraints, this work presents a DDP-based algorithm to solve general constrained OC problems with an inherited property of local convergence, which paves the way to compute the gradient in the subsequent inverse problems. In addition, observe that in the aforementioned results, DDP-based methods are only used to compute the optimal trajectory, which is the inner loop of the IRL problem, and have not been exploited for the update in the outer loop. In this paper, we develop a new way of differentiation by virtue of DDP-based methods. A related line of research can be found in the context of unconstrained optimal estimation and control [29]–[31], where DDP is used to enable direct optimization over time-invariant parameters with theoretical convergence guarantees. Another related work is [32], which uses the techniques of vector-Jacobian products (VJPs) to compute the gradient of an outer-level objective w.r.t. the parameter and enjoys $\mathcal{O}(1)$ memory complexity. However, these works cannot handle our proposed closed-loop IRL problem since they consider the derivatives of the cost function up to the second order (i.e., DDP itself) while the design of the closed-loop loss uses the quantities in DDP and

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

its gradient-based optimization requires the derivatives of the cost function up to the third order (i.e., the derivatives of all the quantities in DDP w.r.t. the parameter). In this paper, we first introduce a DDP-based gradient solver to explicitly compute all the quantities in DDP and the gradient of the trajectory w.r.t. the parameter, then derive the derivatives of the former and obtain the gradient of the closed-loop loss along with the latter.

The inverse optimal control (IOC) problem, which is highly related to IRL while assuming that the system dynamics is known or being identified beforehand by system identification techniques, has been considered in the control community. A popular and efficient approach to solving IOC is residual minimization, which finds a set of parameters such that the violation of optimality conditions (e.g., Karush-Kuhn-Tucker conditions [33], [34] and PMP equations [35]–[37]) is minimized when evaluated along with collected demonstrations. By exploiting the special structure of the cost function, it can be shown that the optimality conditions are linear in the parameter, and the latter can be decoupled from the collected demonstrations. Therefore, some rank equality conditions based only on demonstrations can be derived as a sufficient condition for recovering the parameter. Moreover, owing to the linearity, these methods only need to solve a quadratic programming problem, which avoids solving optimal control problems in an inner loop as in the bi-level optimization, and hence are generally more efficient. However, these methods did not take into consideration stage-wise constraints, which often appear in real applications. The authors in [38] extended their work [36] to the case with only control constraints, where an additional index set was introduced to remove these constraints and convert the problem back into an unconstrained problem. However, the presented method is limited to the control constraints and is difficult to be extended to the case with more general constraints. This paper will establish the recoverability condition for the general constrained IRL problem and include the above-mentioned condition as a special case.

B. Contributions

In this work, we propose a DDP-based IRL framework, where it is shown that the terms required to update the outer loop can be computed by using DDP algorithms. In particular, by observing that the intermediate matrices that appear in DDP recursions are exactly the terms which we require for obtaining the analytical gradient, we introduce an augmented system with the learning parameter being an additional state and show that the gradient can be generated by performing a one-step DDP recursion on that augmented system. Moreover, in order to incorporate the closed-loop nature of data collection, we propose a new type of loss function based on the above-mentioned intermediate matrices, where the main idea is that one should try to match the reproduced and demonstrated feedback policies instead of matching the reproduced and demonstrated trajectories. Furthermore, thanks to the general form of this new loss function, it naturally leads to a generalized set of recoverability conditions for the constrained IOC problem under some assumptions.

The contributions of this paper lie in five-folds:

- We propose a unified DDP-based IRL framework to learn the parameters in the cost function, system dynamics, and general constraints;
- We show that the required gradient term for updating the learning parameter can be obtained efficiently via performing one-step DDP recursion on an augmented system and establish the equivalence between DDP-based methods and PDP-based methods;
- We propose a new type of loss function which by definition outperforms the traditionally adopted imitation loss on the closed-loop demonstrations and develop an efficient algorithm alongside;
- We establish the recoverability conditions for the general constrained IRL problem, whose specialization under some assumptions is also a generalization of the traditional unconstrained IOC recoverability condition;
- We apply the proposed theoretical results to simulation examples and real-world experiments.

The rest of this paper is structured as follows. Section II formally formulates the problem to be studied. Section III presents our proposed DDP-based IRL framework with a commonly used open-loop loss. Section IV details the DDP-based IRL framework with the proposed closed-loop loss. Numerical simulations and real-world experiments are provided in Secs. V and VI. Finally, Section VII concludes this paper.

Notations: In this paper, $\|\mathbf{x}\|$ denotes the 2-norm of $\mathbf{x} \in \mathbb{R}^n$ and $\|\mathbf{x}\|_{\mathbf{A}}^2 = \mathbf{x}^\top \mathbf{A}\mathbf{x}$. Denote by $\mathbf{A}^\top$ and $\mathbf{A}^{-1}$ the transpose and inverse of $\mathbf{A} \in \mathbb{R}^{n\times n}$, respectively. Let $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ be the n-dimensional identity matrix and $\mathbf{1}_n$ be the n-dimensional column vector with all entries of 1. Denote the vectorization operation by $\mathrm{vec}(\cdot)$, i.e., $\mathrm{vec}([\mathbf{a},\mathbf{b}]) = [\mathbf{a}^\top, \mathbf{b}^\top]^\top$. Let $\mathrm{col}(\{\mathbf{A},\mathbf{B}\}) = [\mathbf{A}^\top, \mathbf{B}^\top]^\top$. Let $\mathsf{D}(\cdot)$ denote the transformation from a vector to a diagonal matrix or the extraction of the diagonal elements from a square matrix to a vector. Let $\otimes$, $\odot$, and $\oplus$ denote the Kronecker product, the tensor contraction, and the quaternion product operation, respectively. Let $\mathcal{I}_n = \{0,\dots,n-1\}$. Let $(\cdot)_{\mathbf{a}} := \frac{\partial(\cdot)}{\partial \mathbf{a}}$ and $(\cdot)_{\mathbf{ab}} := \frac{\partial^2(\cdot)}{\partial \mathbf{b}\partial \mathbf{a}}$, and define $\frac{\mathrm{dvec}(\mathbf{A})}{\mathrm{d}x}$ and $\frac{\partial \mathrm{vec}(\mathbf{A})}{\partial \mathbf{x}}$ by $\mathring{\nabla}_x \mathbf{A}$ and $\mathring{\partial}_x \mathbf{A}$, respectively. Let $[\mathbf{A}]_i$ denote the i-th slice of tensor $\mathbf{A}$ and $[\cdot]_\times$ denote the cross product operation. Let $\mathbf{C}^{n,m}$ denotes the commutation matrix which satisfies $\mathbf{C}^{n,m}\,\mathrm{vec}(\mathbf{A}) = \mathrm{vec}(\mathbf{A}^\top)$, where $\mathbf{A} \in \mathbb{R}^{n\times m}$.

II. Problem Formulation

Consider the following general nonlinear constrained optimal control problem

$$\begin{aligned} \min_{\mathcal{U}} \;\; & W(\mathcal{Z};\boldsymbol{\theta}) := \sum_{k\in\mathcal{I}_N} \ell(\mathbf{x}_k,\mathbf{u}_k;\boldsymbol{\theta}) + \wp(\mathbf{x}_N;\boldsymbol{\theta}) \\ \text{s.t.} \;\; & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k;\boldsymbol{\theta}), \mathbf{x}_0 \text{ is given}, \\ & \mathbf{g}(\mathbf{x}_k,\mathbf{u}_k;\boldsymbol{\theta}) \le \mathbf{0}, \mathbf{h}(\mathbf{x}_k,\mathbf{u}_k;\boldsymbol{\theta}) = \mathbf{0}, \end{aligned} \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^{m_\mathbf{x}}$ and $\mathbf{u}_k \in \mathbb{R}^{m_\mathbf{u}}$ denote the state and control input at time instant k, respectively; $\mathcal{U} := \{\mathbf{u}_k\}_{k\in\mathcal{I}_N}$ is the collection of control inputs and N is the control horizon; $\mathcal{Z} := \{\mathbf{x}_k\}_{k\in\mathcal{I}_{N+1}} \cup \mathcal{U}$ denotes the entire system trajectory; $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ denotes the variable parameterizing the following functions:

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

- stage cost $\ell : \mathbb{R}^{m_{\mathbf{x}}} \times \mathbb{R}^{m_{\mathbf{u}}} \times \mathbb{R}^{m_{\boldsymbol{\theta}}} \to \mathbb{R}$;
- terminal cost $\wp : \mathbb{R}^{m_{\mathbf{x}}} \times \mathbb{R}^{m_{\boldsymbol{\theta}}} \to \mathbb{R}$;
- system dynamics $\mathbf{f} : \mathbb{R}^{m_{\mathbf{x}}} \times \mathbb{R}^{m_{\mathbf{u}}} \times \mathbb{R}^{m_{\boldsymbol{\theta}}} \to \mathbb{R}^{m_{\mathbf{x}}}$;
- inequality constraint $\mathbf{g} : \mathbb{R}^{m_{\mathbf{x}}} \times \mathbb{R}^{m_{\mathbf{u}}} \times \mathbb{R}^{m_{\boldsymbol{\theta}}} \to \mathbb{R}^{m_{\mathrm{in}}}$;
- equality constraint $\mathbf{h} : \mathbb{R}^{m_{\mathbf{x}}} \times \mathbb{R}^{m_{\mathbf{u}}} \times \mathbb{R}^{m_{\boldsymbol{\theta}}} \to \mathbb{R}^{m_{\mathrm{eq}}}$.

We assume that the above functions are twice-differentiable. Note that for the sake of clarity, $\ell, \mathbf{f}, \mathbf{g}$ and $\mathbf{h}$ presented here do not explicitly depend on the time instant k, however, our analysis in the sequel can be easily extended to the case where $\ell, \mathbf{f}, \mathbf{g}$ and $\mathbf{h}$ are time-dependent.

Denote a sampled trajectory of the entire system trajectory $\mathcal{Z}$ as $\mathcal{Z}_{\mathcal{S}} := \{\mathbf{x}_k\}_{k\in\mathcal{S}} \cup \{\mathbf{u}_k\}_{k\in\mathcal{S}}$, where $\mathcal{S} \subseteq \mathcal{I}_{N+1}$ denotes the set of sampling time instants. Given a specific value of $\boldsymbol{\theta}$, one can use a nonlinear programming solver to obtain a trajectory $\mathcal{Z}(\boldsymbol{\theta})$. We assume that the mapping from $\boldsymbol{\theta}$ to $\mathcal{Z}(\boldsymbol{\theta})$ always exists and is unique for the local set Θ, where the required regularity conditions can be found in [10, Lemma 1].

The problem of interest is that given a set of $|\mathcal{D}|$ expert demonstrations $\mathcal{D} = \{\mathcal{Z}_{\mathcal{S}_i}(\boldsymbol{\theta}^*)\}_{i=1,\ldots,|\mathcal{D}|}$ generated from some unknown parameter $\boldsymbol{\theta}^*$, find a $\boldsymbol{\theta}$ which matches these expert demonstrations most, i.e.,

$$\begin{aligned} \min_{\boldsymbol{\theta}\in\Theta} \;& L(\mathcal{D}, \mathcal{Z}, \boldsymbol{\theta}) \\ \text{s.t.} \;& \mathcal{Z} \text{ with } \mathcal{U} \text{ being solved from (1).} \end{aligned} \tag{2}$$

In the above, L denotes the loss function which characterizes the closeness between the demonstration $\mathcal{Z}_{\mathcal{S}_i}(\boldsymbol{\theta}^*)$ and the solved trajectory $\mathcal{Z}$. A commonly used loss function in the literature [9], [10] is the mean-square-error loss[1]

$$L^{\mathrm{ol}} := \tfrac{1}{2}\|\mathcal{Z}_{\mathcal{S}}(\boldsymbol{\theta}^*) - \mathcal{Z}\|_2^2, \tag{3}$$

where an additional regularization term $\|\boldsymbol{\theta}\|_2^2$ for $\boldsymbol{\theta}$ can be added when required. In the sequel, we denote this loss as the open-loop loss, as it views the state and control input in the demonstration independently, which usually is not the case in the demonstration generation process. Nevertheless, in the subsequent section, we develop efficient algorithms for optimizing this loss. In Section IV, to explicitly take into consideration the feedback nature of demonstrations, we propose a new so-called closed-loop loss, which will be demonstrated to be superior to the open-loop loss.

III. Open-loop IRL

In this section, we shall develop a new IRL algorithm to solve the optimization problem (2) with the open-loop loss L^{ol} by exploiting the vanilla DDP algorithm and its variants. It can be found that problem (2) is of the form of the bi-level optimization, where the low-level inner optimization solves the constrained multi-stage optimal control problem (1) and the higher-level outer optimization optimizes the loss function L^{ol}. Hence, the commonly used gradient descent method can be adopted to solve this bi-level optimization problem, i.e.,

$$\boldsymbol{\theta}_{t+1} = \mathrm{proj}_{\Theta}\left[\boldsymbol{\theta}_t - \eta_t\left(\frac{\partial L^{\mathrm{ol}}}{\partial \mathcal{Z}}\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial L^{\mathrm{ol}}}{\partial \boldsymbol{\theta}}\right)\right], \tag{4}$$

[1]We omit the demonstration index i in the sequel and assume a single demonstration for the sake of simplicity, while the subsequent analysis can be easily extended to the case with multiple demonstrations.

where $\boldsymbol{\theta}_t$ is the current estimate of the learning parameter $\boldsymbol{\theta}$ in iteration t with its initial value being $\boldsymbol{\theta}_0$; η_t is the learning rate; $\frac{\partial L^{\mathrm{ol}}}{\partial \mathcal{Z}}$ and $\frac{\partial L^{\mathrm{ol}}}{\partial \boldsymbol{\theta}}$ denote the partial derivatives of the loss function w.r.t. the solved trajectory and the learning parameter; $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$ denotes the derivative of the solved trajectory w.r.t. the learning parameter. At iteration t, one can solve the trajectory $\mathcal{Z}(\boldsymbol{\theta}_t)$ from (1) with a specific value $\boldsymbol{\theta}_t$. This can be done by either an external solver or the method developed in Section III-A, and we call this the trajectory solver in the sequel. Then, with a known loss function, one can evaluate $\frac{\partial L^{\mathrm{ol}}}{\partial \mathcal{Z}}$ and $\frac{\partial L^{\mathrm{ol}}}{\partial \boldsymbol{\theta}}$ easily with the analytic differentiation or auto-differentiation in existing machine learning frameworks (e.g., PyTorch [39]). However, for $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$, $\mathcal{Z}$ is the optimal solution of an optimization program, which is obtained from at least tens of iterations, instead of an explicit functional mapping from inputs (initial state and parameter). Auto-differentiation on this optimization procedure unrolls computational graphs in each iteration and hence results in prohibitive memory and time complexity. On the other hand, notice that in order to be the optimal solution, $\mathcal{Z}$ should satisfy some equilibrium conditions, which implicitly characterize the relationship between inputs and the optimal solution. In Section III-B, these conditions are resorted to design an efficient gradient solver for obtaining $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$.

In what follows, we shall first introduce a DDP-based trajectory solver. Although the trajectory can also be solved by any other external solvers, we present this algorithm as a generalization of the previous IPDDP algorithm [24] to the case with equality constraints and also for paving the way to develop the gradient solver in Sec. III-B.

A. DDP-based trajectory solver

In this subsection, we shall present a DDP-based algorithm to solve the optimal control problem with both inequality and equality constraints. The proposed algorithm inherits the general structure of the traditional DDP algorithm, i.e., solving a Bellman equation via iterative backward and forward recursions. The backward recursions compute control inputs to minimize a quadratic approximation of the cost-to-go in the vicinity of the current solution, and the forward recursions update the current solution. However, in order to deal with equality constraints which have not been considered in [24], both the Bellman equation and the iterative process should be redesigned, which are detailed as follows.

Let us first denote the cost-to-go and optimal cost-to-go at time instant k as Q_k and V_k, respectively. In the following, for the sake of clarity, we shall omit the subscript $(\cdot)_k$ if no confusion is caused and let $(\cdot)^+$ denote $(\cdot)_{k+1}$. Applying the Bellman principle of optimality, one has:

$$V = \min_{\mathbf{u}} \; \ell + V^+(\mathbf{x}^+) \quad \text{s.t. } \mathbf{g} \le \mathbf{0}, \mathbf{h} = \mathbf{0},$$

which is a general nonlinear programming problem. One possible solution is using nested optimization, i.e., in the inner loop, each $\mathbf{u}_k$ is solved by calling a general nonlinear programming solver, and in the outer loop, the trajectory $\mathcal{Z}$ is updated. However, it can be found that in this solution, each outer loop calls a solver N times, which results in high computational complexity. Observe that the above process

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

works in a similar manner to the barrier method [40, Chapter 19.6], where a full optimization problem is solved in the inner loop (i.e., "centering"), which inspires us to use the primal-dual interior-point method as an alternative way to solve (1). To this end, we introduce dual variables $\boldsymbol{\lambda}, \boldsymbol{\gamma}$ for the inequality and equality constraints, respectively. As a result, one has the following interior-point min-max Bellman equation:

$$V = \min_{\mathbf{u}} \max_{\boldsymbol{\lambda}\geq\mathbf{0},\boldsymbol{\gamma}} Q := \ell + V^{+} + \boldsymbol{\lambda}^{\top}\mathbf{g} + \boldsymbol{\gamma}^{\top}\mathbf{h}. \tag{5}$$

In contrast to the cost-to-go function that appears in the traditional DDP algorithm and is only a function of $\mathbf{x}, \mathbf{u}$, Q in the above is also a function of the dual variables $\boldsymbol{\lambda}, \boldsymbol{\gamma}$.

After introducing (5), we shall aim to solve it iteratively by resorting to its local approximation. Taking the second order variation of the above Q and V, one has

$$\delta Q = \frac{1}{2} \sum_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\{\mathbf{x},\mathbf{u},\boldsymbol{\lambda},\boldsymbol{\gamma}\}} \delta\boldsymbol{\alpha}^{\top} Q_{\boldsymbol{\alpha}} + \delta\boldsymbol{\alpha}^{\top} Q_{\boldsymbol{\alpha}\boldsymbol{\beta}} \delta\boldsymbol{\beta} + Q_{\boldsymbol{\beta}}^{\top} \delta\boldsymbol{\beta}$$

and

$$\delta V = \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x} \end{bmatrix}^{\top} \begin{bmatrix} 0 & V_{\mathbf{x}}^{\top} \\ V_{\mathbf{x}} & V_{\mathbf{x}\mathbf{x}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \end{bmatrix}. \tag{6}$$

By definition of Q, one has the following equations

$$\begin{aligned} &Q_{\boldsymbol{\alpha}} = \ell_{\boldsymbol{\alpha}} + \mathbf{f}_{\boldsymbol{\alpha}}^{\top} V_{\mathbf{x}}^{+} + \mathbf{g}_{\boldsymbol{\alpha}}^{\top}\boldsymbol{\lambda} + \mathbf{h}_{\boldsymbol{\alpha}}^{\top}\boldsymbol{\gamma}, Q_{\boldsymbol{\lambda}} = \mathbf{g}, Q_{\boldsymbol{\gamma}} = \mathbf{h}, \\ &Q_{\boldsymbol{\lambda}\boldsymbol{\alpha}} = \mathbf{g}_{\boldsymbol{\alpha}}, Q_{\boldsymbol{\gamma}\boldsymbol{\alpha}} = \mathbf{h}_{\boldsymbol{\alpha}}, Q_{\boldsymbol{\lambda}\boldsymbol{\lambda}} = \mathbf{0}, Q_{\boldsymbol{\lambda}\boldsymbol{\gamma}} = \mathbf{0}, Q_{\boldsymbol{\gamma}\boldsymbol{\gamma}} = \mathbf{0}, \\ &Q_{\boldsymbol{\alpha}\boldsymbol{\beta}} = \ell_{\boldsymbol{\alpha}\boldsymbol{\beta}} + \mathbf{f}_{\boldsymbol{\alpha}}^{\top} V_{\mathbf{x}\mathbf{x}}^{+} \mathbf{f}_{\boldsymbol{\beta}} + V_{\mathbf{x}}^{+} \odot \mathbf{f}_{\boldsymbol{\alpha}\boldsymbol{\beta}} + \boldsymbol{\lambda} \odot \mathbf{g}_{\boldsymbol{\alpha}\boldsymbol{\beta}} + \boldsymbol{\gamma} \odot \mathbf{h}_{\boldsymbol{\alpha}\boldsymbol{\beta}} \end{aligned}$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \{\mathbf{x}, \mathbf{u}\}$. These equations compute the partial derivatives of the cost-to-go $Q_{(\cdot)}$ from the optimal cost-to-go at the next time instant. In order to complete the backward recursion, an update rule from the cost-to-go Q to the optimal cost-to-go V is required. To this end, we consider the solution to the following problem,

$$\min_{\delta\mathbf{u}} \max_{\delta\boldsymbol{\lambda},\delta\boldsymbol{\gamma}} \delta Q \text{ s.t. } \boldsymbol{\lambda} + \delta\boldsymbol{\lambda} \geq \mathbf{0}, \tag{7}$$

which is first-order variation of (5). If $(\mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\gamma})$ is the stationary point of (5), $(\delta\mathbf{u}, \delta\boldsymbol{\lambda}, \delta\boldsymbol{\gamma})$ should satisfy the following conditions:

- for the minimizing variable $\delta\mathbf{u}$, it must satisfy the stationarity condition:

  $$\tfrac{\delta Q}{\delta \mathbf{u}} = Q_{\mathbf{u}} + Q_{\mathbf{u}\mathbf{x}}\delta\mathbf{x} + Q_{\mathbf{u}\mathbf{u}}\delta\mathbf{u} + Q_{\mathbf{u}\boldsymbol{\lambda}}\delta\boldsymbol{\lambda} + Q_{\mathbf{u}\boldsymbol{\gamma}}\delta\boldsymbol{\gamma} = \mathbf{0}.$$

- for the maximizing variable $\delta\boldsymbol{\lambda}$ related to the inequality constraint, it must satisfy the dual feasibility condition $\boldsymbol{\lambda} + \delta\boldsymbol{\lambda} \geq \mathbf{0}$ and the complementary condition:

  $$\mathsf{D}(\boldsymbol{\lambda} + \delta\boldsymbol{\lambda})(Q_{\boldsymbol{\lambda}} + Q_{\boldsymbol{\lambda}\mathbf{x}}\delta\mathbf{x} + Q_{\boldsymbol{\lambda}\mathbf{u}}\delta\mathbf{u}) = \mathbf{0}.$$

  Omitting the second-order terms, adding a perturbation vector $\mu\mathbf{1}$ on the left-hand side where μ is a perturbation variable [2], and rearranging the above equation, one has

  $$\delta\boldsymbol{\lambda} = -[\mathsf{D}(\mathbf{g})]^{-1}(\mathbf{r}_{\text{in}} + \mathsf{D}(\boldsymbol{\lambda})Q_{\boldsymbol{\lambda}\mathbf{x}}\delta\mathbf{x} + \mathsf{D}(\boldsymbol{\lambda})Q_{\boldsymbol{\lambda}\mathbf{u}}\delta\mathbf{u}),$$

  where $\mathbf{r}_{\text{in}} := \mathsf{D}(\boldsymbol{\lambda})\mathbf{g} + \mu\mathbf{1}$.

- for the maximizing variable $\delta\boldsymbol{\gamma}$ related to the equality constraint, it must satisfy the primal feasibility condition:

  $$Q_{\boldsymbol{\gamma}} + Q_{\boldsymbol{\gamma}\mathbf{x}}\delta\mathbf{x} + Q_{\boldsymbol{\gamma}\mathbf{u}}\delta\mathbf{u} = \mathbf{0}.$$

  Inspired by the perturbed complementarity equation for equality constraints, e.g. [41, Eq. (6.22)], adding a perturbation term $\mu(\boldsymbol{\gamma} + \delta\boldsymbol{\gamma})$ on the right-hand side and one has:

  $$\delta\boldsymbol{\gamma} = \mu^{-1}(Q_{\boldsymbol{\gamma}\mathbf{x}}\delta\mathbf{x} + Q_{\boldsymbol{\gamma}\mathbf{u}}\delta\mathbf{u}) - \mathbf{r}_{\text{eq}},$$

  where $\mathbf{r}_{\text{eq}} := \boldsymbol{\gamma} - \mu^{-1}Q_{\boldsymbol{\gamma}} = \boldsymbol{\gamma} - \mu^{-1}\mathbf{h}$.

Substituting $\delta\boldsymbol{\lambda}$ and $\delta\boldsymbol{\gamma}$ defined above back into the stationarity condition, one has the following feedback control law:

$$\delta\mathbf{u} = \mathbf{k} + \mathbf{K}\delta\mathbf{x} \tag{8}$$

where $\mathbf{k} = -\hat{Q}_{\mathbf{u}\mathbf{u}}^{-1}\hat{Q}_{\mathbf{u}}, \mathbf{K} = -\hat{Q}_{\mathbf{u}\mathbf{u}}^{-1}\hat{Q}_{\mathbf{u}\mathbf{x}}$, $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \{\mathbf{x}, \mathbf{u}\}$,

$$\begin{aligned} \hat{Q}_{\boldsymbol{\alpha}} &= Q_{\boldsymbol{\alpha}} - Q_{\boldsymbol{\alpha}\boldsymbol{\lambda}}[\mathsf{D}(\mathbf{g})]^{-1}\mathbf{r}_{\text{in}} - Q_{\boldsymbol{\alpha}\boldsymbol{\gamma}}\mathbf{r}_{\text{eq}}, \\ \hat{Q}_{\boldsymbol{\alpha}\boldsymbol{\beta}} &= Q_{\boldsymbol{\alpha}\boldsymbol{\beta}} - Q_{\boldsymbol{\alpha}\boldsymbol{\lambda}}[\mathsf{D}(\mathbf{g})]^{-1}\,\mathsf{D}(\boldsymbol{\lambda})Q_{\boldsymbol{\lambda}\boldsymbol{\beta}} + \tfrac{1}{\mu}Q_{\boldsymbol{\alpha}\boldsymbol{\gamma}}Q_{\boldsymbol{\gamma}\boldsymbol{\beta}}. \end{aligned} \tag{9}$$

Compared with [24], it can be found from the above definitions of $\hat{Q}_{(\cdot)}$ that an additional term (i.e., the third term) was introduced to deal with the equality constraint $\mathbf{h}$. Next, one also substitutes the feedback control law (8) into $\delta\boldsymbol{\lambda}$ and $\delta\boldsymbol{\gamma}$ to obtain their expressions in feedback form:

$$\delta\boldsymbol{\lambda} = \mathbf{k}_{\text{in}} + \mathbf{K}_{\text{in}}\delta\mathbf{x}, \quad \delta\boldsymbol{\gamma} = \mathbf{k}_{\text{eq}} + \mathbf{K}_{\text{eq}}\delta\mathbf{x}, \tag{10}$$

where

$$\begin{aligned} \mathbf{k}_{\text{in}} &= -[\mathsf{D}(\mathbf{g})]^{-1}(\mathbf{r}_{\text{in}} + \mathsf{D}(\boldsymbol{\lambda})Q_{\boldsymbol{\lambda}\mathbf{u}}\mathbf{k}), \\ \mathbf{K}_{\text{in}} &= -[\mathsf{D}(\mathbf{g})]^{-1}(\mathsf{D}(\boldsymbol{\lambda})Q_{\boldsymbol{\lambda}\mathbf{x}} + \mathsf{D}(\boldsymbol{\lambda})Q_{\boldsymbol{\lambda}\mathbf{u}}\mathbf{K}), \\ \mathbf{k}_{\text{eq}} &= -\mathbf{r}_{\text{eq}} + \mu^{-1}Q_{\boldsymbol{\gamma}\mathbf{u}}\mathbf{k}, \quad \mathbf{K}_{\text{eq}} = \mu^{-1}(Q_{\boldsymbol{\gamma}\mathbf{x}} + Q_{\boldsymbol{\gamma}\mathbf{u}}\mathbf{K}). \end{aligned}$$

After finding the solution of $\delta\mathbf{u}$, we shall update the derivatives related to optimal cost-to-go by following the traditional DDP algorithm, i.e.,

$$\begin{aligned} V_{\mathbf{x}} = \hat{Q}_{\mathbf{x}} - \hat{Q}_{\mathbf{u}\mathbf{x}}^{\top}\hat{Q}_{\mathbf{u}\mathbf{u}}^{-1}\hat{Q}_{\mathbf{u}} &= \hat{Q}_{\mathbf{x}} - \mathbf{K}^{\top}\hat{Q}_{\mathbf{u}\mathbf{u}}\mathbf{k}, \\ V_{\mathbf{x}\mathbf{x}} = \hat{Q}_{\mathbf{x}\mathbf{x}} - \hat{Q}_{\mathbf{u}\mathbf{x}}^{\top}\hat{Q}_{\mathbf{u}\mathbf{u}}^{-1}\hat{Q}_{\mathbf{u}\mathbf{x}} &= \hat{Q}_{\mathbf{x}\mathbf{x}} - \mathbf{K}^{\top}\hat{Q}_{\mathbf{u}\mathbf{u}}\mathbf{K}, \end{aligned} \tag{11}$$

where $\hat{Q}_{\mathbf{x}}$ and $\hat{Q}_{\mathbf{x}\mathbf{x}}$ can be obtained by replacing the subscript $(\cdot)_{\mathbf{u}}$ in (9) with $(\cdot)_{\mathbf{x}}$. Repeating the above alternating update of the cost-to-go Q and the optimal cost-to-go V for $k = N-1, \ldots, 0$, one can obtain a set of control gains $\{\mathbf{k}, \mathbf{K}, \mathbf{k}_{\text{in}}, \mathbf{K}_{\text{in}}, \mathbf{k}_{\text{eq}}, \mathbf{K}_{\text{eq}}\}$ and this completes the backward recursions.

In the forward recursions, we aim to obtain an updated trajectory $\mathcal{Z}^{\dagger}$ by using the system dynamics and the control gains obtained above, i.e., repeating the following computation

$$\begin{aligned} [\mathbf{u}^{\dagger}, \boldsymbol{\lambda}^{\dagger}, \boldsymbol{\gamma}^{\dagger}] &= [\mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\gamma}] + [\mathbf{k}, \mathbf{k}_{\text{in}}, \mathbf{k}_{\text{eq}}] \\ &\quad + \mathsf{D}(\{\mathbf{K}, \mathbf{K}_{\text{in}}, \mathbf{K}_{\text{eq}}\}(\mathbf{x}^{\dagger} - \mathbf{x}), \\ \mathbf{x}^{+\dagger} &= \mathbf{f}(\mathbf{x}^{\dagger}, \mathbf{u}^{\dagger}), \end{aligned} \tag{12}$$

for $k = 0, \ldots, N-1$ with the fixed initial condition $\mathbf{x}_0^{\dagger} = \mathbf{x}_0$.

We summarize the proposed generalized interior-point DDP-based trajectory in Algorithm 1, where the above-mentioned backward and forward recursions can be found in lines 3 to 7, and lines 9 to 11, respectively. Note that in the

[2] Similar to the primal-dual interior-point method, μ governs the duality gap and is essential for maintaining the centrality of iterates during optimization. Specifically, it is used in the complementary slackness condition and allows the algorithm to balance feasibility and optimality, progressively reducing the duality gap by controlling $\mu \to 0$.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

practical implementation of the above algorithm, it is usually the case that the inequality constraint is a control bound and the equality constraint is a fixed waypoint constraint. We normally initialize with a trajectory with a constant velocity. For more complex cases, one can extend the infeasible interior-point DDP [24] to allow for an infeasible initial solution guess. In addition, regularization terms should be added in the backward recursions to guarantee the positive-definiteness of $\hat{Q}_{\mathbf{uu}}$. The perturbation variable μ should be updated following [24]. Line-search methods should be added to the forward recursions to preserve the primal and dual feasibility, i.e., $\mathbf{g} < \mathbf{0}$ and $\boldsymbol{\lambda} > \mathbf{0}$.

Algorithm 1 DDP-based trajectory solver

Input: system (1), parameter θ, initial state $\mathbf{x}_0$, initial solution $\mathcal{U}_0$, initial Lagrangian multipliers $\boldsymbol{\lambda}_0, \boldsymbol{\gamma}_0$ and tolerance tol, initial perturbation μ_0

Output: optimal solution $\mathcal{U}$

while $\mathrm{merit} > \mathrm{tol}$ **do**
  set $V_{\mathbf{x},N} = \wp_{\mathbf{x}}$, $V_{\mathbf{xx},N} = \wp_{\mathbf{xx}}$
  for $k = N-1, \ldots, 0$ **do**
    evaluate $\hat{Q}_{(\cdot)}$ using (9)
    compute control gains in (8) and (10)
    update $V_{\mathbf{x}}$, $V_{\mathbf{xx}}$ using (11)
  end for
  set $\mathbf{x}_0^\dagger = \mathbf{x}_0$
  for $k = 0, \ldots, N-1$ **do**
    Update $\mathbf{u}_k^\dagger$, $\boldsymbol{\lambda}_k^\dagger$, $\boldsymbol{\gamma}_k^\dagger$, and $\mathbf{x}_{k+1}^\dagger$ according to (12)
  end for
  Update μ
end while

Remark III.1. *Under the assumption that $\hat{Q}_{\mathbf{uu}}$ is positive-definite for all $k \in \mathcal{I}_N$, one can establish the local quadratic convergence by following the proof of [24, Theorem 2] with the vector-valued merit function being defined by* $\mathrm{merit} = [Q_{\mathbf{u}}^\top, \mathbf{r}_{\mathrm{in}}^\top, \mathbf{r}_{\mathrm{eq}}^\top]^\top$.

Remark III.2. *Another algorithm called active-set DDP algorithm, which also adopts the backward-forward structure, can also be used to solve the general nonlinear constrained optimal control problem. In the backward recursions, it first identifies the active inequality constraint while excluding the inactive parts which do not contribute to the optimal solution. Then it considers solving the following problem:*

$$\min_{\delta\mathbf{u}} \delta Q^\diamond \text{ s.t. } \mathbf{h}^\diamond(\mathbf{x}+\delta\mathbf{x}, \mathbf{u}+\delta\mathbf{u}; \boldsymbol{\theta}) = \mathbf{0}, \tag{13}$$

where $\mathbf{h}^\diamond$ concatenates $\mathbf{h}$ and the rows of $\mathbf{g}$ which equals $\mathbf{0}$. To solve this, the following KKT condition is used:

$$\begin{bmatrix} Q_{\mathbf{uu}}^\diamond & (\mathbf{h}_{\mathbf{u}}^\diamond)^\top \\ \mathbf{h}_{\mathbf{u}}^\diamond & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{u} \\ \gamma^\diamond \end{bmatrix} = -\begin{bmatrix} Q_{\mathbf{ux}}^\diamond \\ \mathbf{h}_{\mathbf{x}}^\diamond \end{bmatrix} \delta\mathbf{x} - \begin{bmatrix} Q_{\mathbf{u}}^\diamond \\ \mathbf{0} \end{bmatrix}, \tag{14}$$

where $Q^\diamond := \ell + (V^\diamond)^+$ here is redefined with the new optimal cost-to-go $V^\diamond$ and $\boldsymbol{\gamma}^\diamond$ is the Lagrangian multiplier for the equality constraint $\mathbf{h}^\diamond = \mathbf{0}$. We refer to [22] for a more detailed process.

B. DDP-based gradient solver

In the last subsection, we have presented a DDP-based trajectory solver for obtaining the optimal solution $\mathcal{U}$ of the constrained multi-stage optimal control problem (1) given the current parameter $\boldsymbol{\theta}$, from which both $\frac{\partial L^{\mathrm{ol}}}{\partial \mathcal{Z}}$ and $\frac{\partial L^{\mathrm{ol}}}{\partial \boldsymbol{\theta}}$ can be easily computed. In this subsection, we aim to present an efficient algorithm, which is referred to as DDP-based gradient solver, to obtain the remaining term $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$ in order to update $\boldsymbol{\theta}_t$ as in (4).

1) A motivating example: Let us first take a detour to consider the computation of the optimal solution w.r.t. the parameter for the following single-stage unconstrained optimization problem $\mathbf{x}^* = \arg\min_{\mathbf{x}} c(\mathbf{x}; \boldsymbol{\theta})$, where $c : \mathbb{R}^{m_{\mathbf{x}}} \times \mathbb{R}^{m_{\boldsymbol{\theta}}} \to \mathbb{R}$ is a scalar function parameterized by $\boldsymbol{\theta}$. Then $\mathbf{x}^*$ should satisfy the first-order necessary equilibrium condition $c_{\mathbf{x}}(\mathbf{x}^*; \boldsymbol{\theta}) = \mathbf{0}$. To obtain the gradient of $\mathbf{x}^*$ w.r.t $\boldsymbol{\theta}$, one approach is to differentiate the above equation w.r.t. $\boldsymbol{\theta}$, i.e., $c_{\mathbf{xx}}\mathbf{x}_{\boldsymbol{\theta}} + c_{\mathbf{x}\boldsymbol{\theta}} = \mathbf{0}$. If $c_{\mathbf{xx}}$ is invertible, one can obtain $\mathbf{x}_{\boldsymbol{\theta}} = -(c_{\mathbf{xx}})^{-1} c_{\mathbf{x}\boldsymbol{\theta}}$ by following the implicit function theorem [42]. Alternatively, one can write $\mathbf{y} := [\boldsymbol{\theta}^\top, \mathbf{x}^{*\top}]^\top$ and take the variation of $\bar{c}_{\mathbf{x}}(\mathbf{y}) := c_{\mathbf{x}}(\mathbf{x}^*; \boldsymbol{\theta})$ w.r.t. $\mathbf{y}$ to get $\bar{c}_{\mathbf{xy}}\delta\mathbf{y} = [\bar{c}_{\mathbf{x}\boldsymbol{\theta}} \ \bar{c}_{\mathbf{xx}}][\delta\boldsymbol{\theta}^\top, \delta\mathbf{x}^\top]^\top = \mathbf{0}$, from which one can obtain $\mathbf{x}_{\boldsymbol{\theta}} = \frac{\delta\mathbf{x}}{\delta\boldsymbol{\theta}} = -(\bar{c}_{\mathbf{xx}})^{-1}\bar{c}_{\mathbf{x}\boldsymbol{\theta}}$.

Indeed, observing that for the constrained multi-stage optimal control problem, we can view

$$\bar{\hat{Q}}_{\mathbf{u}}(\mathbf{y}, \mathbf{u}) := \hat{Q}_{\mathbf{u}}(\mathbf{x}, \mathbf{u}; \boldsymbol{\theta}) = \mathbf{0}$$

as the equilibrium condition of the optimization problem (7) at each time instant k, where the dependence on the dual variables has been removed by substitution and we have slightly abused the notation $\mathbf{y} := [\boldsymbol{\theta}^\top, \mathbf{x}^\top]^\top$. Taking the variation, one has $[\bar{\hat{Q}}_{\mathbf{uy}} \ \bar{\hat{Q}}_{\mathbf{uu}}][\delta\mathbf{y}^\top, \delta\mathbf{u}^\top]^\top = \mathbf{0}$, from which one can obtain

$$\frac{\delta\mathbf{u}}{\delta\boldsymbol{\theta}} = -\bar{\hat{Q}}_{\mathbf{uu}}^{-1}\bar{\hat{Q}}_{\mathbf{uy}} \begin{bmatrix} \mathbf{I} \\ \frac{\delta\mathbf{x}}{\delta\boldsymbol{\theta}} \end{bmatrix}. \tag{15}$$

In the above, both $\frac{\delta\mathbf{u}}{\delta\boldsymbol{\theta}}$ and $\frac{\delta\mathbf{x}}{\delta\boldsymbol{\theta}}$ are exactly the elements in $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$ and they are connected for each time instant k. However, currently, we are only given $\frac{\delta\mathbf{x}_0}{\delta\boldsymbol{\theta}} = \mathbf{0}$ (since $\mathbf{x}_0$ is fixed), which is insufficient to compute $\{\frac{\delta\mathbf{x}_k}{\delta\boldsymbol{\theta}}\}_{k=0}^N$ and $\{\frac{\delta\mathbf{u}_k}{\delta\boldsymbol{\theta}}\}_{k=0}^{N-1}$. To address this, one should be able to compute the matrix $\bar{\hat{Q}}_{\mathbf{uy}}$, and also $\{\frac{\delta\mathbf{x}_k}{\delta\boldsymbol{\theta}}\}_{k=1}^N$.

2) DDP-based gradient solver design: We shall consider the following augmented system:

$$\begin{aligned} \min_{\mathcal{U}} \ & \sum_{k=0}^{N-1} \bar{\ell}(\mathbf{y}_k, \mathbf{u}_k) + \bar{\wp}(\mathbf{y}_N) \\ \text{s.t.} \ & \mathbf{y}_{k+1} = \bar{\mathbf{f}}(\mathbf{y}_k, \mathbf{u}_k) = [\boldsymbol{\theta}^\top, \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k; \boldsymbol{\theta})^\top]^\top, \\ & \bar{\mathbf{g}}(\mathbf{y}_k, \mathbf{u}_k) \le \mathbf{0}, \bar{\mathbf{h}}(\mathbf{y}_k, \mathbf{u}_k) = \mathbf{0}, \mathbf{y}_0 \text{ is given} \end{aligned} \tag{16}$$

where all the functions $f(\ldots; \boldsymbol{\theta}), f \in \{\ell, \wp, \mathbf{f}, \mathbf{g}, \mathbf{h}\}$ parameterized by $\boldsymbol{\theta}$ in (1) have been replaced by their counterpart $\bar{f}$ with $\mathbf{y}$ being the new state variable for the augmented system.

Define the following quantities: $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \{\mathbf{y}, \mathbf{u}\}$,

$$\begin{aligned} \bar{\hat{Q}}_{\boldsymbol{\alpha}} &= \bar{\ell}_{\boldsymbol{\alpha}} + \bar{\mathbf{f}}_{\boldsymbol{\alpha}}^\top \bar{V}_{\mathbf{y}}^+ - \mu \bar{\mathbf{g}}_{\boldsymbol{\alpha}}^\top [\mathsf{D}(\bar{\mathbf{g}})]^{-1}\mathbf{1} + \mu^{-1}\bar{\mathbf{h}}_{\boldsymbol{\alpha}}^\top \bar{\mathbf{h}}, \\ \bar{\hat{Q}}_{\boldsymbol{\alpha}\boldsymbol{\beta}} &= \bar{\ell}_{\boldsymbol{\alpha}\boldsymbol{\beta}} + \bar{\mathbf{f}}_{\boldsymbol{\alpha}}^\top \bar{V}_{\mathbf{yy}}^+ \bar{\mathbf{f}}_{\boldsymbol{\beta}} + \bar{V}_{\mathbf{y}}^+ \odot \bar{\mathbf{f}}_{\boldsymbol{\alpha}\boldsymbol{\beta}} \\ &\quad + \mu \bar{\mathbf{g}}_{\boldsymbol{\alpha}}^\top [\mathsf{D}(\bar{\mathbf{g}})]^{-2} \bar{\mathbf{g}}_{\boldsymbol{\beta}} - \mu([\mathsf{D}(\bar{\mathbf{g}})]^{-1}\mathbf{1}) \odot \bar{\mathbf{g}}_{\boldsymbol{\alpha}\boldsymbol{\beta}} \\ &\quad + \mu^{-1}\bar{\mathbf{h}}_{\boldsymbol{\alpha}}^\top \bar{\mathbf{h}}_{\boldsymbol{\beta}} + \mu^{-1}\bar{\mathbf{h}} \odot \bar{\mathbf{h}}_{\boldsymbol{\alpha}\boldsymbol{\beta}}, \end{aligned} \tag{17}$$

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

where no dual variables were involved. The following result establishes the connection between one iteration of backward-forward recursion on this augmented system and the gradient of the optimal trajectory w.r.t. the learning parameter.

Theorem III.3. *Suppose $\mathcal{Z}$ is the optimal solution to* (1) *with μ, and $\bar{\hat{Q}}_{\mathbf{uu}}$ is invertible for $k = 0, \dots, N-1$. The derivative of the solved trajectory w.r.t. the learning parameter $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$ can be obtained by iteratively updating* (15) *and*

$$\begin{bmatrix} \frac{\delta \boldsymbol{\theta}^+}{\delta \boldsymbol{\theta}} \\ \frac{\delta \mathbf{x}^+}{\delta \boldsymbol{\theta}} \end{bmatrix} = \bar{\mathbf{f}}_{\mathbf{y}} \begin{bmatrix} \mathbf{I} \\ \frac{\delta \mathbf{x}}{\delta \boldsymbol{\theta}} \end{bmatrix} + \bar{\mathbf{f}}_{\mathbf{u}} \frac{\delta \mathbf{u}}{\delta \boldsymbol{\theta}}. \tag{18}$$

for $k \in \mathcal{I}_N$, with $\frac{\delta \mathbf{x}_0}{\delta \boldsymbol{\theta}} = \mathbf{0}$ and $\bar{\hat{Q}}_{(\cdot)}$ being defined in (17).

Theorem III.3 implies that the gradient of the trajectory w.r.t. the parameter can be computed by performing a single backward-forward recursion on the augmented system with the augmented optimal trajectory being the initial solution. Intuitively, during each iteration in the backward recursion, the solver finds the affine relationship between the variation of input $\delta\mathbf{u}$ and that of augmented state $\delta\mathbf{y}$, which also leads to the affine relationship between the gradients [see (15)]. Next, during each iteration in the forward recursion, an affine relationship between the gradients [see (18)] can also be established by utilizing an affine relationship among the variation of augmented state at next time $\delta\mathbf{y}^+$, that of input $\delta\mathbf{u}$, and that of augmented state $\delta\mathbf{y}$. Consequently, this DDP-based gradient solver enjoys the linear time complexity $\mathcal{O}(N)$.

On the other hand, if the active-set method was used as the trajectory solver (either the off-the-shelf commercial solver or the active-set DDP-based approach mentioned in Remark III.2), it is equivalent to solving the equality-constrained ($\mathbf{h}^\diamond = \mathbf{0}$) optimal control problem. In order to compute the gradient, we consider the following augmented system:

$$\begin{aligned} \min_{\mathcal{U}} \quad & \sum_{k=0}^{N-1} \bar{\ell}(\mathbf{y}_k, \mathbf{u}_k) + \bar{\wp}(\mathbf{y}_N) \\ \text{s.t.} \quad & \mathbf{y}_{k+1} = \bar{\mathbf{f}}(\mathbf{y}_k, \mathbf{u}_k), \bar{\mathbf{h}}^\diamond = \mathbf{0}, \mathbf{y}_0 \text{ is given}, \end{aligned} \tag{19}$$

where $\bar{f}$, $f \in \{\ell, \wp, \mathbf{f}\}$ shares the same definitions of those in (16) while the active equality constraint is defined as $\bar{\mathbf{h}}^\diamond(\mathbf{y}_k, \mathbf{u}_k) := \mathbf{h}^\diamond(\mathbf{x}, \mathbf{u}; \boldsymbol{\theta})$.

Define $\bar{Q}^\diamond := \bar{\ell} + (\bar{V}^\diamond)^+$, by which one can obtain its partial derivatives $\bar{Q}^\diamond_{(\cdot)}$ by definition: $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \{\mathbf{y}, \mathbf{u}\}$,

$$\begin{aligned} \bar{Q}^\diamond_{\boldsymbol{\alpha}} &= \bar{\ell}_{\boldsymbol{\alpha}} + \bar{\mathbf{f}}_{\boldsymbol{\alpha}}^\top (\bar{V}^\diamond_{\mathbf{y}})^+, \\ \bar{Q}^\diamond_{\boldsymbol{\alpha}\boldsymbol{\beta}} &= \bar{\ell}_{\boldsymbol{\alpha}\boldsymbol{\beta}} + \bar{\mathbf{f}}_{\boldsymbol{\alpha}}^\top (\bar{V}^\diamond_{\mathbf{yy}})^+ \mathbf{f}_{\boldsymbol{\beta}} + (\bar{V}^\diamond_{\mathbf{y}})^+ \odot \bar{\mathbf{f}}_{\boldsymbol{\alpha}\boldsymbol{\beta}}. \end{aligned} \tag{20}$$

Additionally, define $\mathbf{A} := [\bar{\mathbf{h}}^\diamond_{\mathbf{u}} (\bar{Q}^\diamond_{\mathbf{uu}})^{-1} (\bar{\mathbf{h}}^\diamond_{\mathbf{u}})^\top]^{-1} \bar{\mathbf{h}}^\diamond_{\mathbf{u}} (\bar{Q}^\diamond_{\mathbf{uu}})^{-1}$. The following result establishes the relationship between the one-time backward-forward recursion on the augmented system and the gradient of the optimal trajectory w.r.t. the learning parameter. Its proof can be found in Appendix B.

Theorem III.4. *Suppose $\mathcal{Z}$ is the optimal solution to the optimal control problem* (1)*, $\bar{Q}^\diamond_{\mathbf{uu}}$ is invertible and $\bar{\mathbf{h}}^\diamond_{\mathbf{u}}$ is full row-rank for $k \in \mathcal{I}_N$. The derivative of solved trajectory w.r.t. the learning parameter $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$ can be obtained by iteratively updating* (18) *and*

$$\frac{\delta \mathbf{u}}{\delta \boldsymbol{\theta}} = \left[(\bar{Q}^\diamond_{\mathbf{uu}})^{-1} [\mathbf{I} - (\bar{\mathbf{h}}^\diamond_{\mathbf{u}})^\top \mathbf{A}] \quad \mathbf{A}^\top \right] \begin{bmatrix} \bar{Q}^\diamond_{\mathbf{uy}} \\ \bar{\mathbf{h}}^\diamond_{\mathbf{y}} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \frac{\delta \mathbf{x}}{\delta \boldsymbol{\theta}} \end{bmatrix} \tag{21}$$

for $k \in \mathcal{I}_N$, with $\frac{\delta \mathbf{x}_0}{\delta \boldsymbol{\theta}} = \mathbf{0}$ and $Q^\diamond_{(\cdot)}$ being defined in (20).

Note that Theorems III.3 and III.4 consider the most general multi-stage constrained optimal control problem and they can be reduced to the unconstrained case by ignoring all the terms related to the constraints (i.e., $\mathbf{g}, \mathbf{h}, \boldsymbol{\lambda}, \boldsymbol{\gamma}$ for Theorem III.3, and $\mathbf{h}^\diamond, \boldsymbol{\gamma}^\diamond$ for Theorem III.4), which is detailed as follows.

Corollary III.5. *Suppose $\mathcal{Z}$ is the optimal solution to the unconstrained optimal control problem* (1) *with $\mathbf{g}, \mathbf{h} := \mathbf{0}$, and $\bar{Q}_{\mathbf{uu}}$ is invertible for $k \in \mathcal{I}_N$. The derivative of solved trajectory w.r.t. the learning parameter $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$ can be obtained by iteratively updating* (15) *and* (18) *for $k \in \mathcal{I}_N$, with $\frac{\delta \mathbf{x}_0}{\delta \boldsymbol{\theta}} = \mathbf{0}$, where $\hat{Q}_{(\cdot)}$ is defined in* (17) *with $\bar{\mathbf{g}}, \bar{\mathbf{h}} := \mathbf{0}$.*

Algorithm 2 DDP-based gradient solver

Input: system (1), optimal trajectory $\mathcal{Z}$, parameter θ, perturbation μ
Output: $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$

1: construct the augmented system (16)
2: set $\bar{V}_{\mathbf{y},N} = \bar{\wp}_{\mathbf{y}}$, $\bar{V}_{\mathbf{yy},N} = \bar{\wp}_{\mathbf{yy}}$
3: **for** $k = N-1, \dots, 0$ **do**
4: evaluate $\bar{\hat{Q}}_{(\cdot)}$ using (17)
5: compute control gains in (15)
6: update $\bar{V}_{\mathbf{y}}$, $\bar{V}_{\mathbf{yy}}$ similarly as in (11)
7: **end for**
8: set $\frac{\delta \mathbf{x}_0}{\delta \boldsymbol{\theta}} = \mathbf{0}$
9: **for** $k = 0, \dots, N-1$ **do**
10: update $\frac{\delta \mathbf{u}}{\delta \boldsymbol{\theta}}$ according to (15)
11: update $\frac{\delta \mathbf{x}^+}{\delta \boldsymbol{\theta}}$ according to (18)
12: **end for**
13: collect $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$

We summarize our proposed DDP-based gradient solver in Algorithm 2, where the backward and forward recursions are detailed in lines 3 to 7, lines 9 to 12, respectively.

3) Relationship to previous studies: As mentioned in Section I, another framework called PDP (and its variant SafePDP) has been proposed in [9], [10] as a gradient solver, where the PMP conditions are differentiated to obtain the implicit relationships between the learning parameter and the optimal trajectory. Due to the close relationship between dynamic programming and PMP on optimal control problems, it is natural to ask if DDP-based and PDP-based gradient solvers, which are their respective differentiated versions, will inherit this relationship. We provide an affirmative answer to this. The computation of the gradient term from the DDP-based method as in Theorem III.3 is equivalent to [10, Theorem 2(c)]. This can be shown by viewing the Hamiltonian function L and the dual variable $\boldsymbol{\lambda}$ for the dynamics constraint defined in [10] as the cost-to-go Q defined in (5) and $V_{\mathbf{x}}$ defined in (11), respectively. Similarly, one can also show that the computation of the gradient term for the optimal control

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

problem from the active-set DDP-based method is equivalent to [10, Theorem 1]; Corollary III.5 implies that the gradient is computed by performing a single backward-forward traditional DDP recursion on an augmented unconstrained system, i.e., (16) with constraints being removed. One can show that this recursion is equivalent to [9, Lemma 5.2] by viewing the Hamiltonian function H and the dual variable $\boldsymbol{\lambda}$ for the dynamics constraint as the cost-to-go $Q := \ell + V^+$ and $V_{\mathbf{x}}$, respectively. In addition, this equivalence will also be validated by numerical simulations in Section V-B. Compared to the PDP-based method, our derivation of the one-step DDP on the augmented system is more compact and easier to be interpreted, i.e., the affine relationship among the gradients follows from that among the variations.

In terms of computation, it has been shown in [10] that the PDP-based gradient solver is of time complexity $\mathcal{O}(N)$. It has been widely perceived that the DDP method is computationally expensive due to the introduction of a 3-dimensional tensor $\mathbf{f_{ab}}$, $\mathbf{a}, \mathbf{b} \in \{\mathbf{x}, \mathbf{u}\}$. However, the tensor evaluation can be avoided if we view $\mathbf{c} \odot \mathbf{f_{ab}}$ as a matrix-valued function with $(\mathbf{c}, \mathbf{a}, \mathbf{b})$ as its arguments, which only has the same cost of evaluating other matrix-valued functions (e.g., $\ell_{\mathbf{ab}}$) and does not introduce much overhead. Consequently, as will be shown by numerical simulations in Section V-B, the DDP-based gradient solver consumes less computational time for systems with high dimensions, which benefits from the compact form of our derivation. Most importantly, as in the optimal control problem where DDP can provide a closed-loop feedback policy for subsequent control and hence provide a more robust performance than PMP, the proposed DDP-based gradient solver also provides some intermediate matrices as byproducts, which can be further used to construct a new closed-loop loss function. As will be seen from Section IV, this new loss function leads to a better performance compared to the case using the open-loop loss function.

Note that the DDP-based gradient solver for constrained problems can be reduced to the one for unconstrained problems; conversely, the solver for constrained problems involves more terms (i.e., $\bar{\mathbf{g}}, \bar{\mathbf{h}}$-related terms in (17)) to deal with these constraints. To compute these terms, more symbolic evaluations are performed, which result in longer computational time than that for unconstrained problems. However, it can be easily shown that (17) is indeed the intermediate matrices for the unconstrained system with modified stage cost (i.e., $\bar{\ell}(\mathbf{y}_k, \mathbf{u}_k) - \mu \mathbf{1}^\top \log(-\bar{\mathbf{g}}) + 1/(2\mu)\|\bar{\mathbf{h}}\|^2$). Therefore, one can also solve the gradient for constrained problems by resorting to the solver for unconstrained problems with a modified objective function. This is consistent with the idea of barrier method in optimization literature and we call this BarrierDDP-based gradient solver. In practice, as will be shown in numerical simulations in Section V-B, the modification in the stage cost does not introduce much overhead for the symbolic evaluation of stage cost while saving significant overhead for that of constraints-related terms.

Additionally, Differentiable Optimal Control (DOC) was proposed in [32] as an alternative to the PDP-based method for the unconstrained case. In light of the above statement of equivalence between DDP-based and PDP-based methods, our method obtains the same final outer-level gradient as DOC. However, they use different intermediate steps: DOC uses the techniques of VJP, while we explicitly compute the intermediate Jacobian first and then compute the outer-level gradient. In terms of computational complexity, our method can also achieve $\mathcal{O}(1)$ memory complexity by combining Line 4 of Algorithm 3 and Line 8-12 of Algorithm 2 together to avoid storage of intermediate quantities. More importantly, our approach enables the gradient computation for the closed-loop loss in the sequel.

C. Open-loop IRL algorithm

Equipped with the introduced trajectory solvers and gradient solvers, it is now ready to summarize the entire IRL algorithm with the open-loop loss, as seen in Algorithm 3. Note that Algorithm 3 only shows the case where the interior-point DDP-based gradient solver is adopted, for the case of the active-set DDP-based gradient solver, one can replace the involved (optimal) cost-to-go accordingly. Furthermore, if the trajectory was solved by any (interior-point, active-set, or traditional) DDP-based trajectory solver, the (optimal) cost-to-go computed in the last iteration of the trajectory solver can be saved and then can be reused in the backward recursion of the gradient solver.

Algorithm 3 Open-loop IRL Algorithm

Input: demonstrative trajectories $\mathcal{D}$, system (1), loss function L^{ol}, initial parameter θ_0, maximum iteration $t_{\max}$

Output: θ

1: **for** $t = 0, \ldots, t_{\max}$ **do**
2: call external solver, Algorithm 1, or active-set DDP-based trajectory solver to solve (1) with $\boldsymbol{\theta} = \boldsymbol{\theta}_t$ (perhaps save some intermediate matrices related to the cost-to-go)
3: collect $\mathcal{Z}$
4: evaluate $\frac{\partial L^{\mathrm{ol}}}{\partial \mathcal{Z}}$ and $\frac{\partial L^{\mathrm{ol}}}{\partial \boldsymbol{\theta}}$
5: call Algorithm 2 to obtain $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$
6: update $\boldsymbol{\theta}_t$ according to (4)
7: $t \leftarrow t + 1$
8: **end for**

Remark III.6. *As shown in Algorithm 3, the open-loop IRL algorithm is essentially a first-order gradient-descent algorithm to solve a generic bi-level optimization problem. By following [43, Theorem 2.1], one can establish the global convergence result and iteration complexity of the full problem with assumptions on strong convexity and smoothness conditions for the functions in the lower-level optimization problem. However, these conditions are too restrictive for commonly considered unconstrained infinite-horizon linear-quadratic regulator problems, let alone the multi-stage optimal control problem. Nevertheless, in practice, if one assumes that for each $\boldsymbol{\theta} \in \Theta$, the solution to the lower-level optimization problem always exists and is unique, along with smoothness conditions, the result of local convergence to a stationary point can be easily obtained.*

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE I: Entire trajectories for the simple example (22).

	state at $k=0$	control at $k=0$	state at $k=1$	control at $k=1$	state at $k=2$
reproduced traj. $\mathcal{Z}$	$\mathbf{x}_0$	$-\frac{2\theta+1}{2\theta+3}\mathbf{x}_0$	$\frac{2}{2\theta+3}\mathbf{x}_0$	$-\frac{1}{2\theta+3}\mathbf{x}_0$	$\frac{1}{2\theta+3}\mathbf{x}_0$
noise-free demo. $\mathcal{Z}^*$	$\mathbf{x}_0$	$-\frac{2\theta^*+1}{2\theta^*+3}\mathbf{x}_0$	$\frac{2}{2\theta^*+3}\mathbf{x}_0$	$-\frac{1}{2\theta^*+3}\mathbf{x}_0$	$\frac{1}{2\theta^*+3}\mathbf{x}_0$
noisy demo. $\mathcal{Z}^{**}$	$\mathbf{x}_0$	$-\frac{2\theta^*+1}{2\theta^*+3}\mathbf{x}_0$	$\frac{2}{2\theta^*+3}\mathbf{x}_0+\mathbf{n}_1$	$-\frac{1}{2\theta^*+3}\mathbf{x}_0-\frac{1}{2}\mathbf{n}_1$	$\frac{1}{2\theta^*+3}\mathbf{x}_0+\frac{1}{2}\mathbf{n}_1+\mathbf{n}_2$

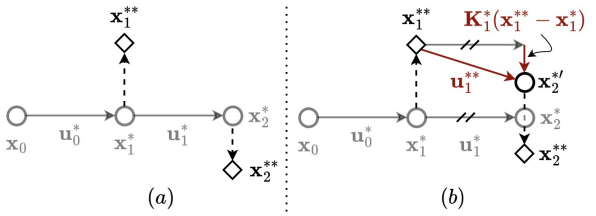


Fig. 1: Illustration of a collection of open-loop and closed-loop trajectories. The gray part denotes the nominal optimal trajectory under ideal environments. For the collection of the open-loop trajectory (left), it is implicitly assumed that the noise process (denoted by the dashed arrow) only affects the measurement afterward. On the contrary, for the collection of the closed-loop trajectory, the next control input will take into consideration this noise and make a correction (denoted by the red arrow).

IV. Closed-loop IRL

In the previous section, we tackled the IRL problem with the open-loop loss L^{ol}, and we can expect that $\boldsymbol{\theta}_t \to \boldsymbol{\theta}^*$ as $t \to \infty$ if $\boldsymbol{\theta}_0$ is in the vicinity of $\boldsymbol{\theta}^*$ for noise-free demonstrations. It is also expected that this type of least-square formulation can tolerate some noise in the collected demonstration signal. However, this formulation implicitly assumes that the noise only appears after the optimal trajectory is solved and executed precisely, or mathematically speaking, it adds some perturbations on the optimal demonstrations $\mathcal{Z}_{\mathcal{S}_i}(\boldsymbol{\theta}^*)$ afterward, see Fig. 1(a). However, this is often not the case in the real data collection process, where the action is performed in a feedback manner to counter the uncertainty.

Let us first take a detour to consider the following simple example of an optimal control problem

$$\begin{aligned} \min_{\mathcal{U}} \quad & \sum_{k=0,1} \tfrac{1}{2}(\theta \mathbf{x}_k^\top \mathbf{x}_k + \mathbf{u}_k^\top \mathbf{u}_k) + \tfrac{1}{2}\mathbf{x}_2^\top \mathbf{x}_2 \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k, \mathbf{x}_0 \text{ is given}, \end{aligned} \tag{22}$$

where θ is the parameter to be learned. Solving the above problem, one can find that the optimal feedback policy is given by $\mathbf{u}_0 = -\frac{2\theta+1}{2\theta+3}\mathbf{x}_0, \mathbf{u}_1 = -\frac{1}{2}\mathbf{x}_1$. Therefore, given an estimated parameter θ (resp. the true parameter θ^*), the reproduced trajectory $\mathcal{Z}$ (resp. noise-free demonstration $\mathcal{Z}^*$) can be explicitly expressed as the second (resp. third) row in Table I. However, if there is some process and/or measurement noise $\mathbf{n}_k$ which comes from a noise distribution $\mathcal{N}_k$ (see the fourth and sixth column of the last row in Table I), the control input at $k=1$ will change correspondingly (see the fifth column of the last row in Table I) and the noisy demonstration $\mathcal{Z}^{**}$ can be obtained. In this case, the open-loop loss L^{ol} defined in (3) can be written as

$$\begin{aligned} L^{\mathrm{ol}} = \tfrac{1}{2}\mathbb{E}_{\mathbf{n}_k\sim\mathcal{N}_k} & \underbrace{\mathbf{0} + \|\tilde{\theta}\mathbf{x}_0 + \mathbf{n}_1\|^2 + \|\tfrac{1}{2}(\tilde{\theta}\mathbf{x}_0 + \mathbf{n}_1)\|^2}_{\sum_{k=0,1,2}\|\mathbf{x}_k - \mathbf{x}_k^{**}\|_2^2} \\ & + \underbrace{\|\tilde{\theta}\mathbf{x}_0\|^2 + \|\tfrac{1}{2}(\tilde{\theta}\mathbf{x}_0 + \mathbf{n}_1) + \mathbf{n}_2\|^2}_{\sum_{k=0,1}\|\mathbf{u}_k - \mathbf{u}_k^{**}\|_2^2}, \end{aligned}$$

where $\tilde{\theta} := \frac{2\theta+1}{2\theta+3} - \frac{2\theta^*+1}{2\theta^*+3}$. By some mathematical operations, one can find that the optimal solution for L^{ol} is given by $-\mathbb{E}_{\mathbf{n}_k\sim\mathcal{N}_k}\frac{3\mathbf{n}_1+\mathbf{n}_2}{5\mathbf{x}_0}$, which means that $\tilde{\theta} \to 0$ only if the noise is of zero-mean and state-independent and one has collected a sufficiently large amount of data. In other words, nonzero-mean or state-dependent noise, or limited size of data will lead to a biased estimation of θ^*. Furthermore, for either a longer horizon $N > 2$ or more general linear system dynamics, L^{ol} involves higher-order terms of θ and one cannot solve the stationary point analytically from $\frac{\mathrm{d}L^{\mathrm{ol}}}{\mathrm{d}\theta} = 0$. However, one has that $\frac{\mathrm{d}L^{\mathrm{ol}}}{\mathrm{d}\theta}|_{\theta=\theta^*}$ is again a linear combination of noise $\{\mathbf{n}_k\}_{k=1}^N$, which implies that zero-mean and state-independent noise and a sufficiently large amount of data are necessary conditions for unbiased estimation of θ^*.

For a general nonlinear optimal control problem, in addition to the numerically computed nominal optimal open-loop input, an additional feedback term should be implemented to correct the deviation $(\mathbf{x}_1^{**} - \mathbf{x}_1^*)$, where $\mathbf{x}_k^{**}$ is the observed current state and is not necessarily equal to the ideal current state $\mathbf{x}_k^*$ due to process noise in $\mathbf{f}$, see Fig. 1(b) for illustration. In the following, we assume that the demonstrations are collected from a closed-loop controller solved by a DDP-based trajectory solver, i.e., instead of having $\mathcal{U}$ as the output, it additionally records the feedback gain $\mathbf{K}_k^*$. During the physical roll-out, the control input is recomputed as[3]

$$\mathbf{u}_k^{**} = \mathbf{u}_k^* + \mathbf{K}_k^*(\mathbf{x}_k^{**} - \mathbf{x}_k^*). \tag{23}$$

Denote the collected demonstration as $\mathcal{Z}_{\mathcal{S}}^{**}$. In this case, if we use the open-loop loss L^{ol} for this type of noisy demonstration, $\boldsymbol{\theta}_t$ does not converge to $\boldsymbol{\theta}^*$ as $t \to \infty$ since $\frac{\mathrm{d}L^{\mathrm{ol}}}{\mathrm{d}\boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}$ is a nonlinear function of noise $\{\mathbf{n}_k\}_{k=1}^N$ and is not equal to $\mathbf{0}$ almost surely (This will also be validated by numerical simulations in Section V-C). To tackle this, we propose a new IRL problem:

$$\min_{\boldsymbol{\theta}\in\Theta} \quad L^{\mathrm{cl}} := \tfrac{1}{2}\sum_{k\in\mathcal{S}} \| \underbrace{\hat{Q}_{\mathbf{u}} + \hat{Q}_{\mathbf{ux}}(\mathbf{x}^{**} - \mathbf{x}) + \hat{Q}_{\mathbf{uu}}(\mathbf{u}^{**} - \mathbf{u})}_{=:\boldsymbol{\epsilon}} \|^2, \tag{24}$$

where $\hat{Q}_{\mathbf{u}}$, $\hat{Q}_{\mathbf{ux}}$, and $\hat{Q}_{\mathbf{uu}}$ are computed in (9), and $\mathcal{Z}$ is solved from (1). We refer to L^{cl} as the closed-loop loss since it is motivated by the closed-loop controller (23) and captures the

[3]Specifically, for the infinite-horizon LQR problem, this means the optimal gain is used for generating the demonstrations in a feedback manner.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

feedback nature. In particular, firstly, notice that $\hat{Q}_{\mathbf{u}} \equiv \mathbf{0}$, since $\mathcal{Z}$ is the optimal trajectory of system (1)[4]. Secondly, $\boldsymbol{\epsilon}$ recovers (23) if $\hat{Q}_{\mathbf{uu}}^{-1}$ is multiplied in each term and all the quantities related to $\mathcal{Z}$ are replaced by the optimal trajectory $\mathcal{Z}(\boldsymbol{\theta}^*)$[5]. By the second point, it can be seen that $\boldsymbol{\theta}^*$ is a global minimum for (24). Note that currently $\boldsymbol{\epsilon}$ only relates the residuals of the current input to the current state, while it is still possible to consider the opposite direction, i.e., including the dynamics residual $\|\mathbf{x}^{**+} - \mathbf{f}(\mathbf{x}^{**}, \mathbf{u}^{**}; \boldsymbol{\theta})\|_2^2$, which relates the next state to current input. However, due to its least-square form, this residual only works well for additive process noise but not for other types of noise. Nevertheless, the addition only brings a marginal overhead in terms of computation (as its required gradient term has already been computed by Algorithm 1). Alternatively, one can use this residual to initialize the parameter to be estimated. On the other hand, if the collected demonstrations are generated in a closed-loop manner other than (23), e.g., model predictive control, the proposed loss can be interpreted as finding an affine time-varying feedback controller which matches the closed-loop demonstrations.

To solve the new IRL problem (24), one can resort to the gradient descent method similar to (4):

$$\begin{aligned}\frac{\mathrm{d}L^{\mathrm{cl}}}{\mathrm{d}\boldsymbol{\theta}} &= \sum_{k\in\mathcal{S}} (\frac{\mathrm{d}\boldsymbol{\epsilon}}{\mathrm{d}\boldsymbol{\theta}})^\top \boldsymbol{\epsilon} \\ &= \sum_{k\in\mathcal{S}} \sum_{\mathbf{z}\in\{\mathbf{x},\mathbf{u}\}} (\bar{\hat{Q}}_{\mathbf{uz}} \frac{\mathbf{z}_k}{\delta\boldsymbol{\theta}} + [(\mathbf{z}^{**} - \mathbf{z}_k)^\top \otimes \mathbf{I}_m] \mathring{\nabla}_{\boldsymbol{\theta}} \bar{\hat{Q}}_{\mathbf{uz}})^\top \boldsymbol{\epsilon} \\ &= \sum_{k\in\mathcal{S}} (\bar{\hat{Q}}_{\mathbf{u}\boldsymbol{\theta}} + \sum_{\mathbf{z}\in\{\mathbf{x},\mathbf{u}\}} [(\mathbf{z}^{**} - \mathbf{z}_k)^\top \otimes \mathbf{I}_m] \mathring{\nabla}_{\boldsymbol{\theta}} \bar{\hat{Q}}_{\mathbf{uz}})^\top \boldsymbol{\epsilon}.\end{aligned}$$

In the above, the first equality is from the fact that both the trajectory $\mathcal{Z}$ itself and the intermediate matrices $\bar{\hat{Q}}_{(\cdot)}$ (which are evaluated at the current trajectory $\mathcal{Z}$) are functions of the learning parameter $\boldsymbol{\theta}$. The second equality results from (15) and implies that it is not necessary to compute $\frac{\delta\mathbf{u}_k}{\delta\boldsymbol{\theta}}$ and $\frac{\delta\mathbf{x}_k}{\delta\boldsymbol{\theta}}$ explicitly since the required term $\bar{\hat{Q}}_{\mathbf{u}\boldsymbol{\theta}}$ has been computed as part of $\bar{\hat{Q}}_{\mathbf{uy}}$ in (17). Nonetheless, one can find that this term is tightly related to the first-order derivative of the trajectory w.r.t. the parameter. However, notice that the above gradient also involves $\mathring{\nabla}_{\boldsymbol{\theta}} \bar{\hat{Q}}_{(\cdot)}$, which is the gradient of the intermediate matrices w.r.t. the learning parameter and has not been obtained in Section III-B. Intuitively speaking, this relates to the second-order derivatives of the trajectory w.r.t. the parameter. This is as expected since in the open-loop loss formulation, one tries to find a parameter to match the solved trajectory with the demonstrations, while in the closed-loop one, one aims to find a parameter to match their variations in the differential sense.

In order to compute $\mathring{\nabla}_{\boldsymbol{\theta}} \bar{\hat{Q}}_{(\cdot)}$, we differentiate $\hat{Q}_{\mathbf{uu}}$ in (17)[6] w.r.t. $\boldsymbol{\theta}$, i.e.,

$$\begin{aligned}\mathring{\nabla}_{\boldsymbol{\theta}} \bar{\hat{Q}}_{\mathbf{uu}} = &\mathring{\nabla}_{\boldsymbol{\theta}} \{\bar{\ell}_{\mathbf{uu}} + \bar{\mathbf{f}}_{\mathbf{u}}^\top \bar{V}_{\mathbf{yy}}^+ \bar{\mathbf{f}}_{\mathbf{u}} + \bar{V}_{\mathbf{y}}^+ \odot \bar{\mathbf{f}}_{\mathbf{uu}}\} \\ &+ \mathring{\nabla}_{\boldsymbol{\theta}} \{\mu \bar{\mathbf{g}}_{\mathbf{u}}^\top [\mathsf{D}(\bar{\mathbf{g}})]^{-2} \bar{\mathbf{g}}_{\mathbf{u}} - \mu([\mathsf{D}(\bar{\mathbf{g}})]^{-1}\mathbf{1}) \odot \bar{\mathbf{g}}_{\mathbf{uu}}\} \\ &+ \mathring{\nabla}_{\boldsymbol{\theta}} \{\mu^{-1} \bar{\mathbf{h}}_{\mathbf{u}}^\top \bar{\mathbf{h}}_{\mathbf{u}} + \mu^{-1} \bar{\mathbf{h}} \odot \bar{\mathbf{h}}_{\mathbf{uu}}\},\end{aligned} \tag{25}$$

where the second and third rows denote the terms related to inequality and equality constraints. For the sake of clarity, we only show the derivation of the first row. The first term reads

$$\mathring{\nabla}_{\boldsymbol{\theta}} \bar{\ell}_{\mathbf{uu}} = \mathring{\partial}_{\boldsymbol{\theta}} \bar{\ell}_{\mathbf{uu}} + \mathring{\partial}_{\mathbf{x}} \bar{\ell}_{\mathbf{uu}} \frac{\delta\mathbf{x}}{\delta\boldsymbol{\theta}} + \mathring{\partial}_{\mathbf{u}} \bar{\ell}_{\mathbf{uu}} \frac{\delta\mathbf{u}}{\delta\boldsymbol{\theta}},$$

which comes from the fact that $\bar{\ell}_{\mathbf{uu}}$ is the function of $(\boldsymbol{\theta}, \mathbf{x}, \mathbf{u})$. The second term is obtained by using the matrix calculus [44]:

$$\begin{aligned}\mathring{\nabla}_{\boldsymbol{\theta}} \{\bar{\mathbf{f}}_{\mathbf{u}}^\top \bar{V}_{\mathbf{yy}}^+ \bar{\mathbf{f}}_{\mathbf{u}}\} = &(\mathbf{C}^{m,m} + \mathbf{I}_{m^2})(\mathbf{I}_m \otimes \bar{\mathbf{f}}_{\mathbf{u}}^\top \bar{V}_{\mathbf{yy}}^+) \mathring{\nabla}_{\boldsymbol{\theta}} \bar{\mathbf{f}}_{\mathbf{u}} \\ &+ (\bar{\mathbf{f}}_{\mathbf{u}}^\top \otimes \bar{\mathbf{f}}_{\mathbf{u}}^\top) \mathring{\nabla}_{\boldsymbol{\theta}} \bar{V}_{\mathbf{yy}}^+,\end{aligned}$$

where the term $\mathring{\nabla}_{\boldsymbol{\theta}}(\cdot)$ involved can be obtained similarly as in the above equation. For the third term, by the definition of tensor contraction, one has

$$\mathring{\nabla}_{\boldsymbol{\theta}} \{\bar{V}_{\mathbf{y}}^+ \odot \bar{\mathbf{f}}_{\mathbf{uu}}\} = \sum_{i=1,\dots,n} \mathring{\nabla}_{\boldsymbol{\theta}} [\bar{V}_{\mathbf{y}}^+]_i [\bar{\mathbf{f}}_{\mathbf{uu}}]_i + [\bar{V}_{\mathbf{y}}^+]_i \mathring{\nabla}_{\boldsymbol{\theta}} [\bar{\mathbf{f}}_{\mathbf{uu}}]_i.$$

The second and third rows of (25) and $\mathring{\nabla}_{\boldsymbol{\theta}} \bar{\hat{Q}}_{\mathbf{ux}}$ can be obtained similarly by following the above derivations.

Note that in order to accelerate the learning process, we use the Levenberg–Marquardt algorithm, i.e., updating the parameter using the following rule:

$$[\mathbf{J}^\top \mathbf{J} + \eta' \mathbf{I}] \delta\boldsymbol{\theta} = \mathbf{J}^\top \boldsymbol{\epsilon}^{\mathcal{S}} \tag{26}$$

where η' is a damping factor adjusted at each iteration, $\mathbf{J} := \mathrm{vec}(\{\frac{\mathrm{d}\boldsymbol{\epsilon}}{\mathrm{d}\boldsymbol{\theta}}\}_{k\in\mathcal{S}})$ and $\boldsymbol{\epsilon}^{\mathcal{S}} := \mathrm{vec}(\{\boldsymbol{\epsilon}\}_{k\in\mathcal{S}})$ are the concatenated gradient and residual terms for the closed-loop loss.

We summarize the closed-loop IRL algorithm in Algorithm 4. In line 4, Algorithm 2 is called to obtain the intermediate matrices as well as the first-order gradient for both computing the loss and preparing for calculating $\mathring{\nabla}_{\boldsymbol{\theta}} \bar{\hat{Q}}_{(\cdot)}$. Lines 7 to 10 detail the backward iteration for computing $\mathring{\nabla}_{\boldsymbol{\theta}} \bar{\hat{Q}}_{(\cdot)}$.

The above content only details the computational aspects of our proposed algorithm. Next, we aim to provide some theoretical characterization of the condition for recoverability, i.e., under which conditions the algorithm can find $\boldsymbol{\theta}^*$. Its proof can be found in Appendix C.

Theorem IV.1. *Suppose that the level set $\{\boldsymbol{\theta} \mid L^{\mathrm{cl}}(\boldsymbol{\theta}) \leq L^{\mathrm{cl}}(\boldsymbol{\theta}^*)\}$ is bounded and that the residual function $\boldsymbol{\epsilon}$ is Lipschitz continuously differentiable in a neighborhood of L^{cl}. Assume that for each t, the approximate solution $\delta\boldsymbol{\theta}$ of (26) satisfies the inequality*

$$L^{\mathrm{cl}}(\boldsymbol{\theta}_t) - L^{\mathrm{cl}}(\boldsymbol{\theta}_t + \delta\boldsymbol{\theta}) \geq c_1 \|\mathbf{J}^\top \boldsymbol{\epsilon}^{\mathcal{S}}\| \min(\Delta_t, \|\mathbf{J}^\top \boldsymbol{\epsilon}^{\mathcal{S}}\| / \|\mathbf{J}^\top \mathbf{J}\|),$$

for some positive constant c_1, and in addition $\|\delta\boldsymbol{\theta}\| \leq c_2 \Delta_t$ for some constant $c_2 \geq 1$, where Δ_t is the trust-region radius in its counterpart trust-region method such that $\eta'(\delta\boldsymbol{\theta} - \Delta_t) = 0$, then Algorithm 4 converges to the stationary point, i.e., $\lim_{t\to\infty} \frac{\mathrm{d}L^{\mathrm{cl}}}{\mathrm{d}\boldsymbol{\theta}} = \lim_{t\to\infty} \mathbf{J}^\top \boldsymbol{\epsilon}^{\mathcal{S}} = \mathbf{0}$. Furthermore, $\boldsymbol{\theta}^*$ can be fully recovered only if $\mathrm{Rank}(\mathbf{J}) = m_{\boldsymbol{\theta}}$.

[4]It is still kept in (24) for the subsequent content of specialization.

[5]An alternate form of the residual $\boldsymbol{\epsilon}' := (\mathbf{u}_k^{**} - \mathbf{u}_k) + \mathbf{K}_k(\mathbf{x}_k^{**} - \mathbf{x}_k)$ may be more obvious to understand the design, while it breaks the tie with the subsequent content of specialization and we do not present here. Nevertheless, it is still applicable for closed-loop IRL of general nonlinear problems and shares nearly the same subsequent algorithmic computation of gradients.

[6]Note that the subsequent derivation is based on the interior-point DDP-based gradient solver, while a similar derivation can be easily performed on the active-set DDP-based gradient solver.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Algorithm 4 Closed-loop IRL Algorithm

Input: demonstrative trajectories $\mathcal{D}$, system (1), loss function L^{cl}, initial parameter θ_0, maximum iteration $t_{\max}$
Output: θ

for $t = 0, \dots, t_{\max}$ **do**
  call external solver, Algorithm 1, or active-set DDP-based trajectory solver to solve (1) with $\boldsymbol{\theta} = \boldsymbol{\theta}_t$ (perhaps save some intermediate matrices related to the cost-to-go)
  collect $\mathcal{Z}$
  call Algorithm 2 to obtain $\frac{\mathrm{d}\mathcal{Z}}{\mathrm{d}\boldsymbol{\theta}}$ and save $\bar{\hat{Q}}_{(\cdot)}$
  evaluate $\boldsymbol{\epsilon}$
  set $\mathring{\nabla}_{\boldsymbol{\theta}}\bar{V}_{\mathbf{y},N} = \mathring{\nabla}_{\boldsymbol{\theta}}\bar{\wp}_{\mathbf{y}}$, $\mathring{\nabla}_{\boldsymbol{\theta}}\bar{V}_{\mathbf{yy},N} = \mathring{\nabla}_{\boldsymbol{\theta}}\bar{\wp}_{\mathbf{yy}}$
  for $k = N-1, \dots, 0$ **do**
    evaluate $\mathring{\nabla}_{\boldsymbol{\theta}}\bar{\hat{Q}}_{(\cdot)}$ using (25)
    update $\mathring{\nabla}_{\boldsymbol{\theta}}\bar{V}_{\mathbf{y}}$, $\mathring{\nabla}_{\boldsymbol{\theta}}\bar{V}_{\mathbf{yy}}$ similarly as in (11)
  end for
  collect $\mathring{\nabla}_{\boldsymbol{\theta}}\bar{\hat{Q}}_{(\cdot)}$ to compute $\mathbf{J}$
  update $\boldsymbol{\theta}_t$ according to (26)
  $t \leftarrow t+1$
end for

If the LQR problem is considered, following the definition of the residual term in (24),

$$\begin{aligned}\boldsymbol{\epsilon}_{\mathrm{lqr}} &= \hat{Q}_{\mathbf{u}} + \hat{Q}_{\mathbf{ux}}(\mathbf{x}^{**} - \mathbf{x}) + \hat{Q}_{\mathbf{uu}}(\mathbf{u}^{**} - \mathbf{u}) \\ &= \hat{Q}_{\mathbf{ux}}\mathbf{x}^{**} + \hat{Q}_{\mathbf{uu}}\mathbf{u}^{**},\end{aligned} \tag{27}$$

where the second equality follows from the optimality condition of LQR. Defining $\mathbf{J}_{\mathrm{lqr}} := \sum_{\mathbf{z}\in\{\mathbf{x},\mathbf{u}\}}[(\mathbf{z}^{**})^\top \otimes \mathbf{I}_m]\mathring{\nabla}_{\boldsymbol{\theta}}\hat{Q}_{\mathbf{uz}}$, one can have the following result.

Corollary IV.2. *The learning parameter $\boldsymbol{\theta}^*$ for LQR can be fully recovered only if* $\mathrm{Rank}(\mathbf{J}_{\mathrm{lqr}}) = m_{\boldsymbol{\theta}}$.

Remark IV.3. *Note that a two-step strategy has been proposed in [45], where a gain matrix $\mathbf{K}^{**}$ is firstly solved from a least square problem [45, Eq. (15)] and a bi-level problem with a cost function* $\mathrm{Tr}\{(\mathbf{K}-\mathbf{K}^{**})^\top(\mathbf{K}-\mathbf{K}^{**})\}$ *is then iteratively solved. It should be noted that Algorithm 4 can also be applied to this scheme if we use* $\mathring{\nabla}_{\boldsymbol{\theta}}\mathbf{K} = -\mathring{\nabla}_{\boldsymbol{\theta}}\hat{Q}_{\mathbf{uu}}^{-1}\hat{Q}_{\mathbf{ux}} = -(\mathbf{I}_n\otimes\hat{Q}_{\mathbf{uu}}^{-1})[(\mathbf{K}^\top\otimes\mathbf{I}_m)\mathring{\nabla}_{\boldsymbol{\theta}}\hat{Q}_{\mathbf{uu}} + \mathring{\nabla}_{\boldsymbol{\theta}}\hat{Q}_{\mathbf{ux}}]$*, which requires the same gradient terms derived in* (25)*. Moreover, our presented algorithm is applicable to IRL of general nonlinear systems subject to constraints.*

Note that due to the nature of nonlinearity of the intermediate matrices w.r.t. the learning parameter, the above rank condition depends on the collected demonstrations, the solved trajectory, and the current parameter. In the following, we shall show that under specific assumptions, $\mathbf{J}$ is linear in $\boldsymbol{\theta}$ and each element only depends on collected demonstrations. Before that, we present an assumption and some definitions which will be used.

Assumption IV.4. *1) The termination condition for solving* (1) *is set as $\mathcal{Z} = \mathcal{Z}_{\mathcal{S}}^{**}$; 2) The demonstrations satisfy the interior-point min-max Bellman equation* (5) *with perturbation μ; 3) The stage cost is linearly parameterized by $\boldsymbol{\theta}$, i.e., $\ell = \boldsymbol{\theta}^\top\boldsymbol{\phi}(\mathbf{x},\mathbf{u})$; and 4) The terminal cost $\wp$, dynamics $\mathbf{f}$, and constraints $\mathbf{g},\mathbf{h}$ are independent of $\boldsymbol{\theta}$ and known.*

Let $(\cdot) \in \{\mathbf{x},\mathbf{u}\}$, $\mathbf{c}_{(\cdot)} := \mu\mathbf{g}_{(\cdot)}^\top[\mathsf{D}(\mathbf{g})]^{-1}\mathbf{1} + \mu^{-1}\mathbf{h}_{(\cdot)}^\top\mathbf{h}$, $V_{\mathbf{x},1:m} := \mathrm{col}(\{V_{\mathbf{x}}\}_{k=1}^m)$, $\boldsymbol{\phi}_{(\cdot),1:m}^\top := \mathrm{col}(\{\boldsymbol{\phi}_{(\cdot)}^\top\}_{k=1}^m)$, $\mathbf{c}_{(\cdot),1:m} := \mathrm{col}(\{\mathbf{c}_{(\cdot)}\}_{k=1}^m)$, $\mathbf{B}_{0:m} := \mathsf{D}(\{\mathbf{f}_{\mathbf{u}}^\top\}_{k=0}^m)$, $\mathbf{E}_{m+1} := [\mathbf{0},\dots,\mathbf{I}]^\top \in \mathbb{R}^{(m+1)n_{\mathbf{x}}\times n_{\mathbf{x}}}$, $\mathbf{A}_{1:m} := \mathbf{I} + \begin{bmatrix}\mathbf{0} & -\mathsf{D}(\{\mathbf{f}_{\mathbf{x}}^\top\}_{k=1}^m)\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$. Furthermore, define $\mathbf{J}_{\mathrm{lin},1:2} := [\mathbf{J}_{\mathrm{lin},1},\mathbf{J}_{\mathrm{lin},2}] := [\boldsymbol{\phi}_{\mathbf{u},1:m}^\top - \mathbf{B}_{0:m}\mathbf{A}_{1:m}^{-1}\boldsymbol{\phi}_{\mathbf{x},1:m}^\top, \mathbf{B}_{0:m}\mathbf{A}_{1:m}^{-1}\mathbf{E}_{m+1}]$, $\mathbf{J}_{\mathrm{lin},3} := -\mathbf{B}_{0:m}\mathbf{A}_{1:m}^{-1}\mathbf{c}_{\mathbf{x},1:m} + \mathbf{c}_{\mathbf{u},1:m}$.

Corollary IV.5. *Under Assumption IV.4 and $\mathbf{J}_{\mathrm{lin},3} \neq \mathbf{0}$, if $\lim_{\mu\to 0}\mathrm{Rank}(\mathbf{J}_{\mathrm{lin},1:2}) = m_{\boldsymbol{\theta}} + m_{\mathbf{x}}$, then the learning parameter $\boldsymbol{\theta}^*$ can be recovered from the demonstration as*

$$\begin{aligned}\boldsymbol{\theta}^* &= [\lim_{\mu\to 0}\arg\min_{[\boldsymbol{\theta}^\top, V_{\mathbf{x},m+2}^\top]^\top}\|\boldsymbol{\epsilon}^{\mathcal{S}}\|^2]_{1:m_{\boldsymbol{\theta}}} \\ &= [\lim_{\mu\to 0} -(\mathbf{J}_{\mathrm{lin},1:2}^\top\mathbf{J}_{\mathrm{lin},1:2})^{-1}\mathbf{J}_{\mathrm{lin},1:2}^\top\mathbf{J}_{\mathrm{lin},3}]_{1:m_{\boldsymbol{\theta}}}.\end{aligned}$$

The proof can be found in Appendix D. It has been shown that the above rank condition only depends on the collected demonstrations and this property resembles that in [36], [37]. However, due to the introduction of constraints, the rank condition is quite different. Moreover, unlike [38] where only control constraints can be considered, our method can deal with general nonlinear constraints.

V. Numerical Experiments

In this section, we first present several examples to validate the equivalence between our proposed DDP-based methods and PDP-based methods. Then, we apply our proposed closed-loop IRL algorithm on these examples to show its advantage over open-loop IRL. Also, we provide an example to demonstrate the proposed recoverability conditions on both the general IRL problem and the specialized constrained inverse optimal control problem.

A. System settings

For simulations, we consider four systems of different dimensions (complexities), which have been commonly used in the literature [9], [11], [21], [22], [46], [47] and are described in Table II. The control task is to drive the system to a prescribed desired state $\mathbf{x}_{\mathrm{d}}$. We consider the following stage and terminal costs $\ell := (\mathbf{x}-\mathbf{x}_{\mathrm{d}})^\top\mathsf{D}(\boldsymbol{\theta}_{\mathbf{x}})(\mathbf{x}-\mathbf{x}_{\mathrm{d}}) + \boldsymbol{\theta}_{\mathbf{u}}\|\mathbf{u}\|^2$, $\wp := (\mathbf{x}-\mathbf{x}_{\mathrm{d}})^\top\mathsf{D}(\boldsymbol{\theta}_{\mathbf{x}})(\mathbf{x}-\mathbf{x}_{\mathrm{d}})$, where $\boldsymbol{\theta}_{\mathbf{x}}, \boldsymbol{\theta}_{\mathbf{u}}$ denote the weights for state and control, and we set $\boldsymbol{\theta}_{\mathbf{u}} = 0.1$ to avoid ambiguity. In addition, we set norm-bounded constraints for both state and control vectors and add a waypoint as an equality constraint. We set both the system parameters and the cost parameters as the parameters to be learned, as shown in the sixth column of Table II.

B. PDP-based vs. DDP-based methods for gradient computation

We first consider the unconstrained IRL problem with the open-loop loss. For the above-mentioned four examples, we temporarily exclude the equality and norm-bounded constraints and their involved waypoints and upper bounds

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE II: System settings for numerical experiments.

Examples	State vector $\mathbf{x}$	Control input $\mathbf{u}$	Desired state $\mathbf{x}_\mathrm{d}$	Constraints	Parameters to be learned	References
CartPole	cart position x, velocity $\dot{x}$ pole angle q angular velocity $\dot{q}$	force f	$[0, 0, \pi, 0]^\top$	$\lvert x\rvert \leq x_\mathrm{ub}$ $\lvert f\rvert \leq f_\mathrm{ub}$ $\dot{x}[k] = x_\mathrm{wp}$	cart mass, pole mass pole length, $\boldsymbol{\theta}_\mathbf{x}$ bounds x_ub, f_ub, waypoint x_wp	[11]
Quadrotor	position $\mathbf{p}_\mathrm{w}$, velocity $\mathbf{v}_\mathrm{w}$ orientation $\mathbf{q}_\mathrm{b}$ angular velocity $\boldsymbol{\omega}_\mathrm{b}$	thrust $\boldsymbol{\tau}$	$[\mathbf{0}_3, \mathbf{0}_3, \mathbf{q}_\mathrm{d}, \mathbf{0}_3]^\top$	$\lVert\mathbf{p}_\mathrm{w}\rVert \leq r$ $\lVert\mathbf{u}\rVert_\infty \leq u_\mathrm{ub}$ $\mathbf{p}_\mathrm{w}[k] = \mathbf{p}_\mathrm{wp}$	mass, moment of inertia wing length, $\boldsymbol{\theta}_\mathbf{x}$ bounds r, u_ub, waypoint $\mathbf{p}_\mathrm{wp}$	[9]
RobotArm	angles and angular velocities of both links $q_1, q_2, \dot{q}_1, \dot{q}_2$	torques τ_1, τ_2	$[\pi/2, 0, 0, 0]$	$\lvert q_i\rvert \leq q_\mathrm{ub}$ $\lVert\mathbf{u}\rVert_\infty \leq u_\mathrm{ub}$ $q_2[k] = q_\mathrm{wp}$	link lengths, $\boldsymbol{\theta}_\mathbf{x}$ bounds $q_\mathrm{ub}, u_\mathrm{ub}$, waypoint q_wp	[9], [47]
Rocket	position $\mathbf{p}_\mathrm{w}$, velocity $\mathbf{v}_\mathrm{w}$ orientation $\mathbf{q}_\mathrm{b}$ angular velocity $\boldsymbol{\omega}_\mathrm{b}$	vectored thrust $\mathbf{u}$	$[\mathbf{0}_3, \mathbf{0}_3, \mathbf{q}_\mathrm{d}, \mathbf{0}_3]^\top$	$\mathrm{tr}(\mathbf{I}_3 - \mathbf{R}_\mathrm{d}^\top \mathbf{R}) \leq \alpha_\mathrm{ub}$ $\lVert\mathbf{u}\rVert_2 \leq u_\mathrm{ub}$ $\mathbf{p}_\mathrm{w}[k] = \mathbf{p}_\mathrm{wp}$	mass, moment of inertia, $\boldsymbol{\theta}_\mathbf{x}$ bounds $\alpha_\mathrm{ub}, u_\mathrm{ub}$, waypoint $\mathbf{p}_\mathrm{wp}$	[9], [10]

from the optimal control problem and the learning parameter, respectively. We use both the PDP-based [9] and our proposed DDP-based algorithms to compute the required gradient. Next, we present the comparison of SafePDP [10] and our proposed IPDDP-based algorithms, which are used for the IRL problem with constraints. Additionally, we implement the BarrierDDP-based method mentioned in Sec. III-B3, which incorporates the constraints into the stage cost via barrier functions. We run the gradient descent algorithm for 1000 steps and repeat for 5 times. For the sake of clarity, we show the difference between gradients computed by PDP-based and DDP-based algorithms for the initial 20 steps in Fig. 2. Similarly, the gradient difference between gradients computed by PDP-based and DDP-based algorithms on constrained problems is recorded in Fig. 3. Table III shows the computational time for the gradient computation in each gradient descent step adopting the above-mentioned algorithms. Table IV shows the wall-clock time for running the open-loop IRL with different gradient solvers.

All the parameters are initialized with an additive Gaussian noise of $\mathcal{N}(0, 0.2)$ as a perturbation to the ground-truth parameters. Based on the above results, we have the following comments.

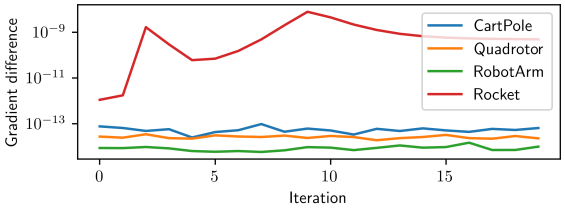


Fig. 2: The difference between the gradients computed by PDP-based and proposed DDP-based algorithms on unconstrained problems.

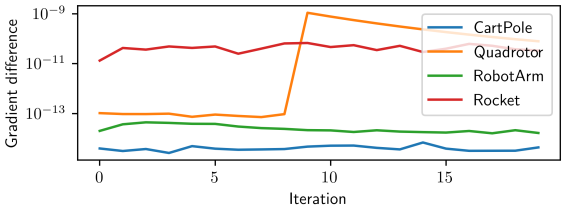


Fig. 3: The difference between the gradients computed by PDP-based and proposed DDP-based algorithms on constrained problems.

1) In terms of the gradient difference, it can be seen from Figs. 2 and 3 that the residual is negligible for the tested examples, which verifies our theoretical result of the equivalence of the gradient computation from the algorithms.

2) It can be found from Tables III and IV that for the systems with a lower dimension (cartpole and robot arm), the computational time is marginally the same, while for those with a higher dimension (quadrotor and rocket), DDP-based algorithm is faster since our derivation is more compact in the sense that it uses a vectorized form of many small terms which are also used in PDP-based algorithms.

3) As seen from Tables III and IV, compared to SafePDP, the IPDDP-based algorithm is worse for the cartpole and robot arm examples while marginally better in the quadrotor and rocket examples. The reason is that although the compact derivation saves computational time (as explained in the unconstrained case), the IPDDP-based algorithm introduces the dual variables as the control variable, which increases the problem size and leads to a bit longer computational overhead for the symbolic evaluation of (17). However, this is not the case for BarrierDDP since its implementation does not increase the problem size as in the IPDDP-based algorithm while inheriting the advantage of DDP over PDP on problems with higher dimensions, which can be seen from Table III.

C. Advantages of closed-loop IRL over open-loop IRL

We define the following metrics to evaluate the performance of our proposed algorithms.

• **Parameter residual:** this measures the error between the learned parameter $\boldsymbol{\theta}$ and the ground truth $\boldsymbol{\theta}^*$, i.e., $r_\mathrm{para}(\boldsymbol{\theta}) := \lVert\boldsymbol{\theta} - \boldsymbol{\theta}^*\rVert^2$, $r_\mathrm{para} = 0$ means an exact recovery of the true parameter.

• **Trajectory residual:** this measures the distance between the demonstration trajectories $\mathcal{Z}(\boldsymbol{\theta}^*)$ and the rollout trajectories $\mathcal{Z}_\mathrm{rollout}(\boldsymbol{\theta})$, i.e., $r_\mathrm{traj}(\boldsymbol{\theta}) := \lVert\mathcal{Z}(\boldsymbol{\theta}^*) - \mathcal{Z}_\mathrm{rollout}(\boldsymbol{\theta})\rVert_2^2$, This metric resembles the open-loop loss L^ol while differs in that the rollout trajectories $\mathcal{Z}_\mathrm{rollout}(\boldsymbol{\theta})$ are not obtained by directly solving (1) but by performing the feedback policy $\{\mathbf{k}, \mathbf{K}\}$ on the system with the true dynamics, i.e., $\mathbf{f}(\cdot; \boldsymbol{\theta}^*)$, which is possibly contaminated by a process noise.

• **Suboptimality gap:** this measures the performance gap between the testing demonstrations $\mathcal{Z}(\boldsymbol{\theta}^*)$ and the rollout trajectories $\mathcal{Z}_\mathrm{rollout}(\boldsymbol{\theta})$ evaluated at the performance index under the parameter $\boldsymbol{\theta}^\mathrm{e}$, i.e., $r_\mathrm{sub}(\boldsymbol{\theta}; \boldsymbol{\theta}^\mathrm{e}) := W(\mathcal{Z}_\mathrm{rollout}(\boldsymbol{\theta}); \boldsymbol{\theta}^\mathrm{e}) - W(\mathcal{Z}(\boldsymbol{\theta}^*); \boldsymbol{\theta}^\mathrm{e})$, Specifically, $\boldsymbol{\theta}^\mathrm{e}$ can be chosen among the true $\boldsymbol{\theta}^*$ and the final value of the learned parameter. Note that this

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE III: The computational time for each call of different gradient solvers. (unit: second)

Examples	Unconstrained		Constrained		
	PDP	DDP	SafePDP	IPDDP	BarrierDDP
CartPole	9.11e-03±7.20e-03	7.37e-03±6.94e-03	8.68e-03±3.80e-03	1.27e-02±4.54e-03	1.14e-02±4.73e-03
Quadrotor	1.62e-02±9.33e-03	7.99e-03±6.07e-03	3.57e-02±2.96e-02	2.25e-02±8.92e-03	2.14e-02±7.44e-03
RobotArm	1.37e-02±6.86e-03	1.09e-02±7.19e-03	1.43e-02±5.75e-03	1.77e-02±5.64e-03	1.58e-02±6.17e-03
Rocket	5.54e-02±1.98e-02	3.65e-02±1.61e-02	1.73e-01±3.79e-01	5.02e-02±2.84e-02	4.97e-02±1.89e-02

TABLE IV: The wallclock time for open-loop IRL with different gradient solvers. (unit: second)

Examples	Unconstrained		Constrained		
	PDP	DDP	SafePDP	IPDDP	BarrierDDP
CartPole	2.79e+01±1.30e+00	2.62e+01±1.10e+00	1.00e+02±3.81e+01	1.02e+02±3.18e+01	1.02e+02± 3.12e+01
Quadrotor	3.66e+01±2.20e+00	2.84e+01±1.64e+00	2.35e+02±6.78e+01	2.08e+02±7.58e+01	2.05e+02± 8.52e+01
RobotArm	2.82e+01±2.35e+00	2.54e+01±2.51e+00	1.40e+02±3.95e+01	1.45e+02±4.02e+01	1.45e+02± 3.98e+01
Rocket	1.39e+02±3.90e+00	1.20e+02±3.16e+00	6.31e+03±2.49e+03	4.67e+03±6.57e+02	4.57e+03± 5.64e+02

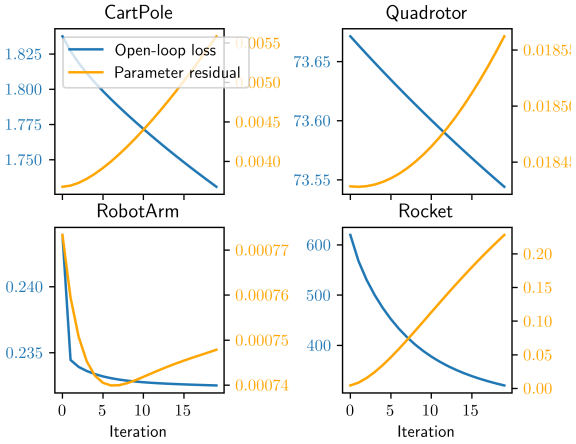


Fig. 4: Traces of loss and parameter estimation error by adopting the open-loop IRL on unconstrained problems. The stepsizes are set as $10^{-3}, 10^{-4}, 10^{-2}, 10^{-4}$, and the horizons are set as $N = 12, 10, 10, 40$.

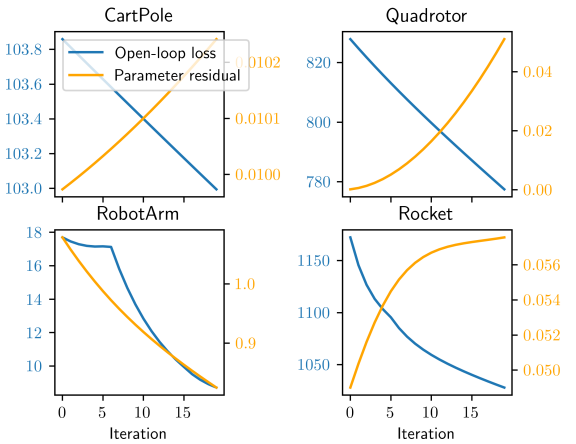


Fig. 5: Traces of loss and parameter estimation error by adopting the open-loop IRL on constrained problems.

suboptimality gap can be negative even if $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ due to different noise realizations.

We first present a qualitative comparison between the open-loop and the closed-loop IRL algorithms. For the open-loop IRL, we use the gradients calculated when generating Figs. 2 and 3 to update the parameter according to Algorithm 3 and record the trace of the open-loop loss L^{ol} and the parameter residual r_{para} in Figs. 4 and 5. For the closed-loop IRL, we implement Algorithm 4 for 5 trials in each simulation example and record the trace of the closed-loop loss L^{cl} and the parameter residual r_{para} in Fig. 6, where each type of line denotes a different trial. For the first trial denoted by solid lines, it uses the same initial condition as that in Fig. 4. While for the other trials, we have applied a multiplicative uniform noise $[0.5, 1.5]$ to the initial parameter to demonstrate the robustness of our proposed algorithm. In the meantime, we record its open-loop loss L^{ol} during the learning process, denoted by the green lines. The computational time for the gradient computation and the wall-clock time for running closed-loop IRL are recorded in Table V. In order to quantitatively demonstrate the advantages of our proposed closed-loop IRL, we test the learned parameter $\boldsymbol{\theta}^{\mathrm{ol}}$ and $\boldsymbol{\theta}^{\mathrm{cl}}$

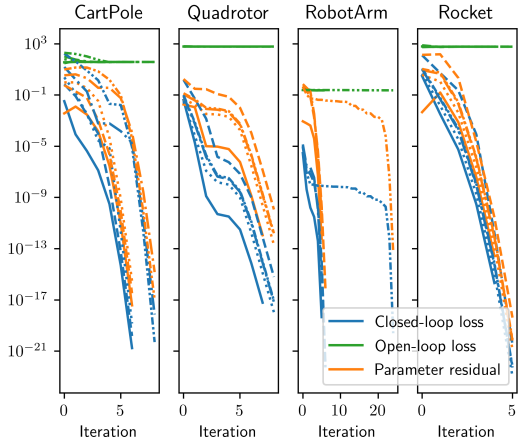


Fig. 6: Traces of loss and parameter estimation error by adopting the closed-loop IRL, i.e., Algorithm 4. Each type of line denotes a different trial.

TABLE V: The computational time for the gradient solver and wallclock time in closed-loop IRL. (unit: second)

Examples	Time for gradient computation	Wallclock time
CartPole	6.03e-02±3.11e-03	6.00e-01±1.33e-02
Quadrotor	2.09e-01±2.11e-02	2.31e+00±1.62e-01
RobotArm	6.38e-02±1.04e-02	7.60e-01±2.68e-01
Rocket	7.76e-01±2.95e-02	4.78e+00±1.70e-01

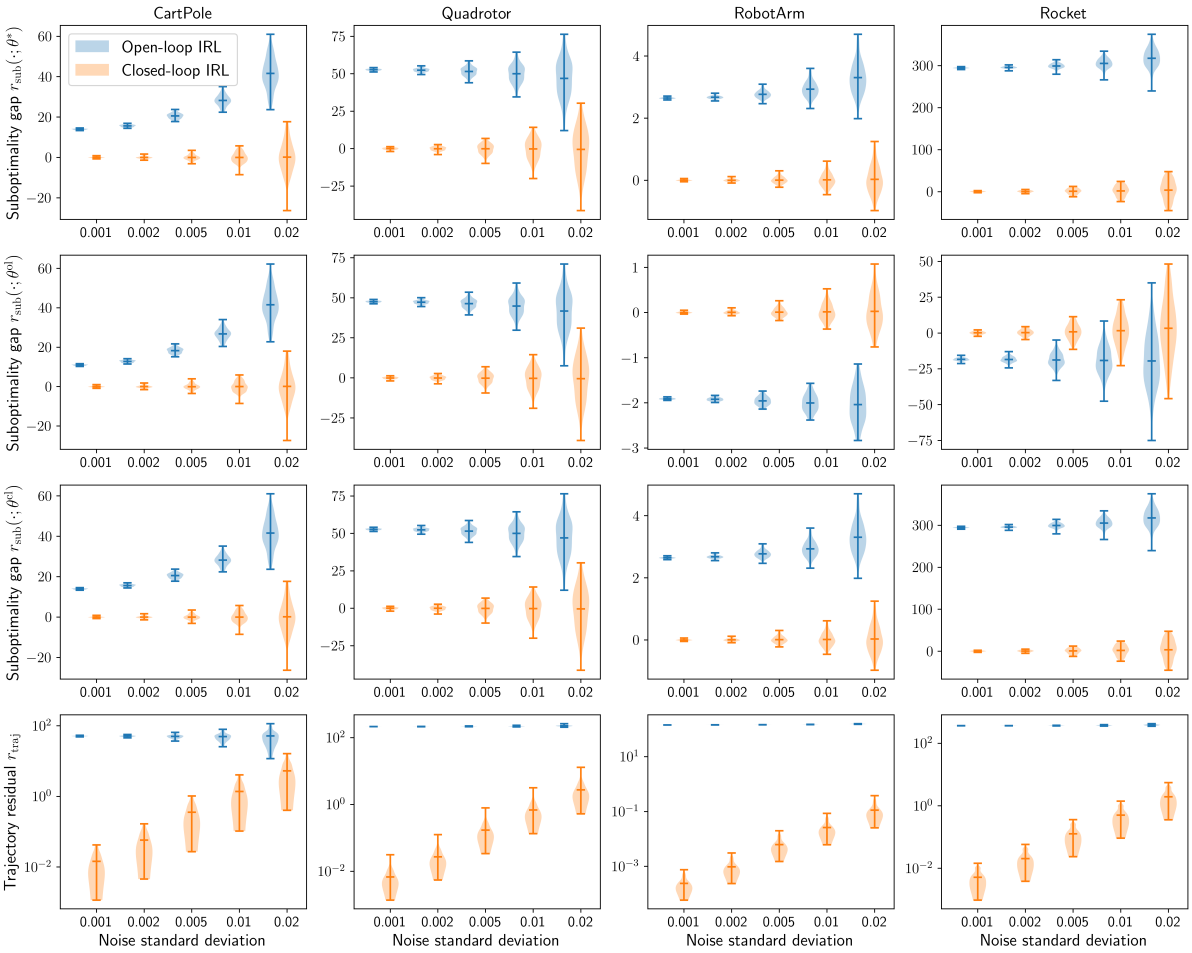


Fig. 7: Performance evaluation on different metrics with parameters learned from open-loop and closed-loop IRL algorithms. The lower and upper bars denote the range and the middle bar denotes the mean. The shaded area shows the probability density of the data at different values.

from the algorithms in the following new setting which is different from training. Specifically, we set the horizon as 20, randomly choose a new initial condition, and use $\boldsymbol{\theta}^{\text{ol}}$ and $\boldsymbol{\theta}^{\text{cl}}$ to compute its corresponding feedback policy. During rollout, we randomly add multiplicative process noise to the system dynamics and record the entire trajectory. We repeat the simulation 100 times for each algorithm under the noise of different standard deviations. Additionally, we go through the same process with the true parameter $\boldsymbol{\theta}^*$ to generate the test dataset. Then, we evaluate these trajectories with the above-defined sub-optimality gaps $r_{\text{sub}}(\boldsymbol{\theta};\boldsymbol{\theta}^{\text{e}})$ with $\boldsymbol{\theta}^{\text{e}} \in \{\boldsymbol{\theta}^*, \boldsymbol{\theta}^{\text{ol}}, \boldsymbol{\theta}^{\text{cl}}\}$ and the trajectory residual r_{traj}, as shown in each row in Fig. 7. Based on the above results, we have the following comments.

1) As seen from Fig. 4, the loss decreases slowly as expected for a gradient descent algorithm. Further, as explained in Sec. IV, due to the different nature of the demonstration (closed-loop) and the loss function (open-loop), the optimizing direction for L^{ol} does not necessarily coincide with the optimizing direction for the parameter residual r_{para}. This can be seen from the rocket example (the last column of Fig. 4), where the parameter residual is indeed increasing. A similar phenomenon can be observed from Fig. 5, i.e., the parameter residuals for cartpole, quadrotor, and rocket systems increase even when the open-loop losses decrease.

2) It can be found from Fig. 6 that, different from the open-loop IRL, the parameter residual of the closed-loop IRL decreases as the closed-loop loss decreases. In the meantime, the open-loop loss is recorded (not used for iteration), from which one can find that it remains a large value even if the parameter residual is negligible. This is expected since our closed-loop design has incorporated the closed-loop nature of demonstrations while not seeking to minimize the discrepancy between demonstrated and reproduced trajectories. As seen from Table V, the computational time for the gradient in the closed-loop IRL is around 5-25 times more than that in the open-loop IRL, with a smaller overhead for low-dimensional systems (cartpole and robot arm) and a larger overhead for high-dimensional systems (quadrotor and rocket), since we need to evaluate the third-order derivatives of the cost function. Thanks to the usage of the LM algorithm, it only takes tens of iterations and a wallclock time of seconds to converge to a very small residual, which is significantly faster than the gradient-descent-based closed-loop IRL.

3) For the first three rows of Fig. 7, the range and variance

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

of suboptimality gaps for both algorithms increase as the standard deviation of noise increases, while the mean of those for closed-loop IRL is approximately zero, indicating that it achieves a similar level of performance (in the sense of cost function) as the policy induced from the true parameter. As seen from the first row of Fig. 7, closed-loop IRL significantly outperforms open-loop IRL in terms of the suboptimality gap evaluated at the true parameter, i.e. $r_{\mathrm{sub}}(\boldsymbol{\theta};\boldsymbol{\theta}^*)$. The third row which corresponds to the suboptimality gap evaluated at closed-loop IRL learned parameter $\boldsymbol{\theta}^{\mathrm{cl}}$ resembles the first row since the parameter residual $r_{\mathrm{para}}(\boldsymbol{\theta}^{\mathrm{cl}})$ is negligible. We cannot guarantee the advantage of closed-loop IRL over open-loop IRL in terms of suboptimality gap evaluated at open-loop IRL learned parameter $\boldsymbol{\theta}^{\mathrm{ol}}$ (the second row of Fig. 7), since in this case the latter is exactly optimized under $\boldsymbol{\theta}^{\mathrm{ol}}$ and is expected to outperform the former. Nevertheless, we observe that the former still outperforms the latter in the cartpole and quadrotor example and they are close in the rocket example, since in these cases open-loop IRL wrongly estimates the parameter in system dynamics, while the rollout is performed on the dynamic system with the true parameter $\boldsymbol{\theta}^*$.

4) As seen from the last row of Fig. 7, closed-loop IRL outperforms open-loop IRL by at least one order of magnitude in terms of trajectory residual r_{traj}, which means that the rollout trajectory generated from the learned parameter is much closer to the one generated from the true parameter. Different from the previous three rows where the mean for closed-loop IRL is always approximately zero, the mean in this row increases as the standard deviation of noise increases, this is because different noise realizations lead to distinct rollout trajectories and hence a strictly positive trajectory residual r_{traj}, and the difference between two trajectories increases. This can also be understood with a simplified case where the rollout trajectory is assumed to be a linear function of parameter with additive noise, then under a negligible parameter residual, i.e., $\boldsymbol{\theta} = \boldsymbol{\theta}^*$, the mean of trajectory residual r_{traj} is exactly two times of the variance of the noise.

5) As can be observed from all of the subplots in Fig. 7, the ranges of data for two algorithms overlap (or are going to be overlapping) with each other as the noise gets larger, this is because the rollout trajectory deviates too much from the nominal trajectory which is used for computing the feedback policy, and hence the policy cannot be guaranteed to perform well in this case.

D. Properties of closed-loop IRL

In the previous section, we have demonstrated the advantages of our proposed closed-loop IRL over the open-loop one by implicitly assuming that both algorithms are trained with sufficient data. In this section, we aim to provide an in-depth analysis of how much data is required. As before, we first present a set of qualitative examples, where we set $|\mathcal{S}| = 2$, i.e., 2 sampling instants within horizon N, and apply Algorithm 4 subsequently. We use the same initial conditions (5 trials) as in Fig. 6 and record the trace of loss L^{cl} and parameter residual r_{para} in Fig. 8. Next, we vary the length of the demonstration $|\mathcal{S}|$ from 1 to 10, and only record the final closed-loop loss L^{cl} and parameter residual r_{para}, as shown in Fig. 9.

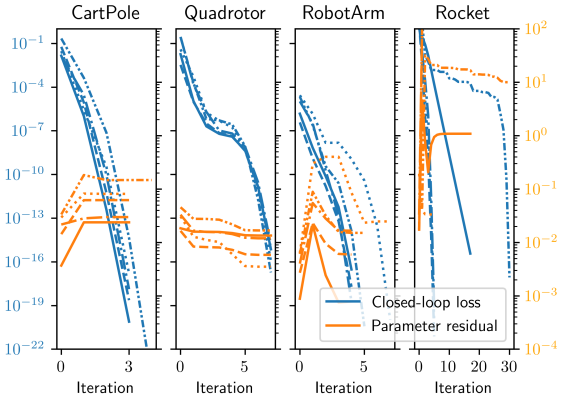


Fig. 8: Traces of loss and parameter estimation error by adopting Algorithm 4 with a short length of demonstrations ($|\mathcal{S}| = 2$).

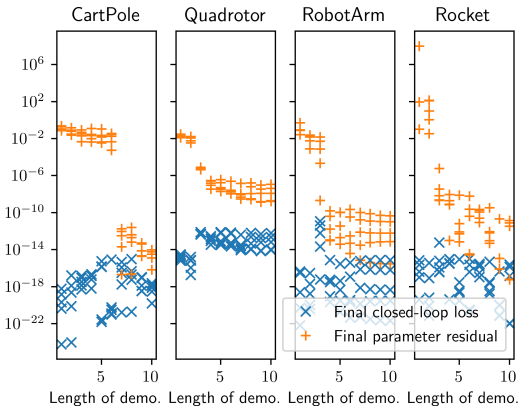


Fig. 9: Final loss and parameter estimation error by adopting Algorithm 4 with different lengths of demonstrations.

It can be found from Fig. 8 that although the closed-loop loss L^{cl} decreases rapidly, the parameter residual stops updating and remains a non-negligible value. The reason is that for $\mathrm{Rank}(\mathbf{J}) < m_{\boldsymbol{\theta}}$, there exists another set of parameters except for $\boldsymbol{\theta}^*$ such that the closed-loop loss L^{cl} is zero. This result can be more easily seen in Fig. 9. One can find an obvious parameter residual r_{para} drop when $|\mathcal{S}|$ is near to $\lceil m_{\boldsymbol{\theta}}/m_{\mathbf{u}} \rceil$ since in this case $\mathrm{Rank}(\mathbf{J}) = m_{\boldsymbol{\theta}}$ in general, e.g. for the quadrotor example, $\lceil m_{\boldsymbol{\theta}}/m_{\mathbf{u}} \rceil = \lceil 9/4 \rceil = 3$. For a longer length of demonstration, i.e., $|\mathcal{S}| > \lceil m_{\boldsymbol{\theta}}/m_{\mathbf{u}} \rceil$, both the closed-loop loss L^{cl} and the parameter residual r_{para} remain negligible values since $\mathrm{Rank}(\mathbf{J})$ is non-decreasing w.r.t. the increase of $|\mathcal{S}|$.

E. Constrained inverse optimal control

In this section, we present an example of an LQR problem to validate Corollary IV.5. We consider the linear system $\mathbf{x}^+ = \left[\begin{smallmatrix} -1 & 1 \\ 0 & 1 \end{smallmatrix}\right]\mathbf{x} + \left[\begin{smallmatrix} 1 \\ 3 \end{smallmatrix}\right]\mathbf{u}$ with the stage cost $\ell := \mathbf{x}^\top \mathrm{D}(\boldsymbol{\theta}_{\mathbf{x}})\mathbf{x} + \boldsymbol{\theta}_{\mathbf{u}}\mathbf{u}^\top\mathbf{u}$ and the terminal cost $\wp := 0$, where the true parameter $\boldsymbol{\theta}^* = [\boldsymbol{\theta}_{\mathbf{x}}^{*\top}, \boldsymbol{\theta}_{\mathbf{u}}^{*\top}]^\top = [0.1, 0.3, 0.6]^\top$. Alternatively, one can rewrite $\ell = \boldsymbol{\phi}^\top\boldsymbol{\theta}$ with $\boldsymbol{\phi} = [\mathbf{x}^\top \otimes \mathbf{x}^\top, \mathbf{u}^\top \otimes \mathbf{u}^\top]^\top$, which satisfies Assumption IV.5-3). We use the inequality constraint

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

$\|[\mathbf{x}]_1\mathbf{u}\| \leq 0.1$, which is a more general nonlinear constraint than the control-only constraint considered in [38]. We generate the initial state $\mathbf{x}_0$ randomly and produce a trajectory with horizon $N = 50$. For this trajectory, we use different lengths (from 1 to 100) of observation and perturbation to construct the matrix $\mathbf{J}_{\text{lin},i}, i = 1, 2, 3$ as defined in Corollary IV.5. The rank of $\mathbf{J}_{\text{lin},1:2}$ and the parameter residual are recorded in Fig. 10. It can be found that when the observation length is not long

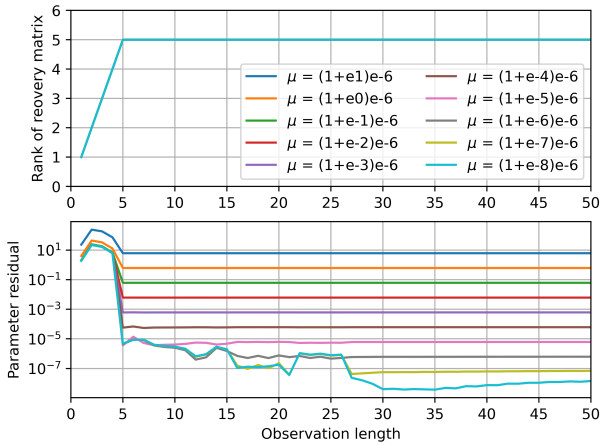


Fig. 10: Rank of $\mathbf{J}_{\text{lin},1:2}$ and parameter residual w.r.t. different observation length and perturbation.

enough, i.e., $|\mathcal{S}| < \lceil (m_{\boldsymbol{\theta}} + m_{\mathbf{x}})/m_{\mathbf{u}} \rceil = 5$, $\mathbf{J}_{\text{lin},1:2}$ is rank-deficient, and it will be rank 5 when it is sufficiently long, i.e., $|\mathcal{S}| \geq 5$. On the other hand, if $\mathbf{J}_{\text{lin},1:2}$ is rank-deficient, the solution is not unique and may be of no physical meaning. If $\mathbf{J}_{\text{lin},1:2}$ is full column rank, the parameter residual denotes the error between the true and the estimated parameters under an assumed perturbation μ, and it decreases as μ decreases.

VI. Real-world Experiments

In this section, we aim to demonstrate the advantages of our proposed closed-loop IRL over open-loop IRL via a real-world task, quadrotor navigation in partially unknown environments.

1) Experiment setup: We verify the advantage of our proposed approach using a self-made tethered quadrotor in a $5\text{m} \times 5\text{m} \times 2\text{m}$ indoor area, which is equipped with a motion capture system. Specifically, as seen in Fig. 11, the quadrotor is connected to a ground power supply using a cable to support long-duration operation. It uses a motion capture system for localization in the environment and is equipped with an i7 computer for onboard computation. The quadrotor performs trajectory planning onboard by solving an optimal control, with a linear dynamics model as commonly used in drone control [48]. Denote the position, velocity, and acceleration by $\mathbf{p}$, $\mathbf{v}$, and $\mathbf{u}$, respectively. Due to the physical limitations and safety considerations, we set $\|\mathbf{v}\|_\infty \leq 1$ and $\|\mathbf{u}\|_\infty \leq 0.5$ to limit both the velocity and acceleration. With the partially unknown information, the task of trajectory planning is to minimize the following costs $\wp := \|\mathbf{p} - \mathbf{p}_\text{d}\|^2$, $\ell := \sum_{i=1,2} \boldsymbol{\theta}_i \exp\{-10^{i-3}(k - k_{\text{g,i}})^2\}\|\mathbf{p} - \mathbf{c}_{\text{g,i}}\|^2 + \boldsymbol{\theta}_{\mathbf{u}}\|\mathbf{u}\|^2$. This type of formulation has been used in [12], [49]. Here,

Fig. 11: Quadrotor robot experiment system setup (top view). A self-made quadrotor is powered via a cable connected to a ground power supply. It relies on the motion capture system (not shown) for localization and uses an onboard computer for high-level trajectory planning and a flight controller for low-level tracking. The task is to navigate the quadrotor from the starting position (upper-left) to the goal position (bottom-right red plus sign) while flying through two gates (formed by the vertical pole and two tripods) sequentially.

the cost function only encodes the approximate locations of each gate, $\mathbf{c}_{\text{g},1}$ and $\mathbf{c}_{\text{g},2}$, which can be represented by the position of any point on the gates. By a partially unknown environment, we mean the accurate size (geometry) of the gate is unknown, which is typically required for navigation. Therefore, we aim to learn the cost function weights, which encode how the quadrotor safely flies through the gate. In the testing and generalization scenarios, we will vary the location of the two gates. Note that in this case, the stage cost is time-dependent, as mentioned in Sec. II, all of our presented methods still apply. We assume that $\boldsymbol{\theta}_{\mathbf{u}} = 0.01$ to avoid ambiguity and set $\boldsymbol{\theta} := [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2]^\top$ as the learning parameter, where $\boldsymbol{\theta}_i \geq 0, i = 1, 2$. We add an additional constraint $\mathbf{1}^\top\boldsymbol{\theta} = 1$ on the learning parameter. The planned high-level trajectory is tracked by a low-level cascaded PID controller. Note that both the physical setup (disturbance brought by power cable during motion) and software stack (hierarchical control architecture) necessitate the use of closed-loop control.

a) Training, test and generalization settings: As seen in Fig. 11, we collect the demonstration trajectory by recording the real-time position obtained by the motion capture system and the high-level control command sent to the low-level controller. We set the initial position as $[1.5, 1, 1]^\top$ and the initial velocity as $\mathbf{0}$. The desired position is set as $[-0.5, -1, 1]^\top$. We set $\mathbf{c}_{\text{g},i}$ as the center of two gates with $\mathbf{c}_{\text{g},1} = [1, 0, 1]^\top$ and $\mathbf{c}_{\text{g},2} = [0.5, 1, 1]^\top$, the height of both gates as 2m, and the width of two gates as 1m and 1.5m. The planning horizon is set as $N = 30$. In the sequel, we shall refer to this setting as both the training and test setting.

Different from the environment for training and testing, we will set new ones by varying the following settings (The other setting is kept the same as the training setting.): 1) longer planning horizon with $N = 40$; 2) new initial condition $[1.5, 1.5, 1]^\top$; 3) new desired position $[0, -1.5, 0]^\top$; 4) new

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

gate position $\mathbf{c}_{\mathrm{g},2} = [0, 1, 1]^\top$, which is further away from gate 1; and 5) new hardware system dynamics including both the mass and the moment of inertia, where we have removed the redundant components (e.g., battery and sensors) of 387 grams from the quadrotor (total weight 1213 grams).

With these new settings, we use the learned parameters to compute the feedback policy for the high-level trajectory of the quadrotor. We check if the trajectories executed in the real-world experiments can successfully complete the task goal: flying through two gates sequentially and arriving at the vicinity of the desired position. We use the following metrics

- **minimum distance to each gate center:** $\min_{k\in\mathcal{N}} \|\mathbf{p}_k - \mathbf{c}_{\mathrm{g},i}\|, i = 1, 2,$
- **final distance to the goal:** $\|\mathbf{p}_N - \mathbf{p}_\mathrm{d}\|,$

to quantitatively evaluate the generalization performance.

2) Results and analysis: We run both open-loop and closed-loop algorithms with the above-collected demonstration. The final learned parameters given by these algorithms are $\boldsymbol{\theta}^{\mathrm{ol}} = [0.74, 0.26]^\top$ and $\boldsymbol{\theta}^{\mathrm{cl}} = [0.45, 0.55]^\top$, respectively. In the sequel, we shall refer to the trajectories generated by parameters learned from open-loop and closed-loop IRL as OL and CL trajectories, respectively. By checking the value of these parameters, one can expect that the OL trajectory will put more weight on gate 1 and less weight on gate 2 than the CL trajectory.

We first test the performance in the test setting. A set of trajectories (one OL trajectory and one CL trajectory) is recorded in Fig. 12(a). We further test the generalization of the learned parameter, or equivalently, the cost function in the generalization settings. The generalizations of the learned cost function to a longer planning horizon of $N = 40$ and a new initial position are shown in Figs. 12(b) and 12(c). Figure 12(d) and 12(e) present the generalization of the learned cost function to a new desired position and new gate positions, as seen from the top view. Figure 12(f) shows the generalization of the learned cost function to new hardware system dynamics. *Note that all the experiments are performed in the area shown in Fig. 11 and trajectories are recorded by the motion capture system and visualized in Fig. 12.* Furthermore, for each case, we have repeated 5 times and computed quantitative measures for the recorded trajectories, as shown in Table VI. Based on these results, we have the following comments. From the test of learned weights, as seen from Fig. 12(a) and Table VI, CL trajectory takes a larger detour on flying through gate 2 than OL trajectory. The average final distance to the goal is slightly smaller. Under a longer horizon, new initial position, new desired position, new gate position, and new hardware system dynamics, Fig. 12(b)-12(f) show that the generalized CL trajectories still fly through two gates sequentially, and arrive at the vicinity of the desired position. However, generalized OL trajectories fail to fly through gate 2. Specifically, as seen in Fig. 12(b) and Table VI, with a longer planning horizon, both OL and CL trajectories will be closer to gate 1 center. Then, both of them take a larger detour towards the center of gate 2. In this case, the apex of the CL trajectory gets closer to the center of gate 2 and results in a large drop in terms of minimum distance to gate 2 center. However, this is not the case for OL trajectory, since the increase of the horizon only reshapes its segment near gate 1. Nevertheless, this detour changes the velocity profile of the OL trajectory and results in a smaller terminal velocity and overshoot.

Note that for the generalization of gate position in Fig. 12(e), the movement of gate 2 enlarges the width of the curve (see the top view) due to the attraction force from its center. However, this (discrete) change of environment is too significant to be followed by a continuous adaptation of the trajectory, which results in an increase of minimum distance to gate 2 center, especially for OL trajectory as it does not reshape in Y-direction but the gate moves further away in X-direction. We also report a failure case where we further move gate 2 away from the initial position, i.e., $\mathbf{c}_{\mathrm{g},2} = [-0.2, 1, 1]^\top$, as visualized in Fig. 13. It can be seen that the CL trajectory fails to reshape itself to fly through gate 2. Moreover, the change of gate position completely alters the landscape of the cost function, enlarging the minimum distance to gate 1 and failing to arrive at the vicinity of the desired position. This result clearly shows the bounds of the generalizability, i.e., the learned cost function can only be applied to some unseen scenarios that are close to the training setting.

VII. Conclusion

In this work, we have proposed a DDP-based framework for IRL with general constraints, where the DDP was exploited to compute the gradient required in the outer loop. We have established the equivalence between DDP-based and PDP-based methods in terms of computation. In addition, inspired by the DDP condition, we have proposed the closed-loop IRL with the closed-loop loss function to capture the nature of collected demonstrations. Moreover, we have shown that this new formulation can be reduced to a general constrained IOC problem under certain conditions, which leads to a generalized recoverability condition. Simulations and experiments demonstrated the superiority of the closed-loop algorithm. Future work can be on the extension of this framework to multi-agent systems and stochastic systems.

Appendix

A. Proof of Theorem III.3

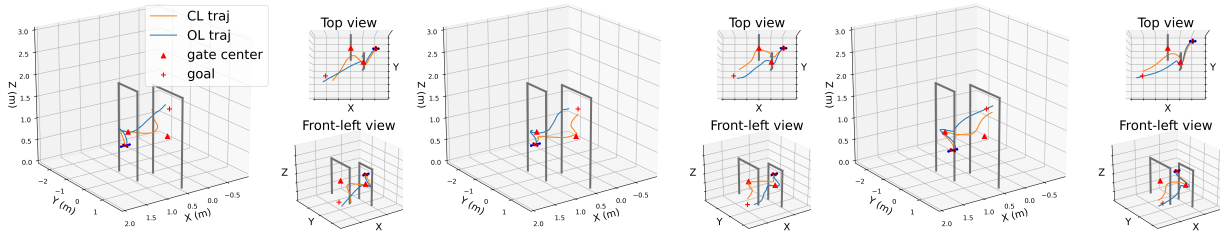
Proof. Suppose we are using the DDP-based trajectory solver to find the optimal solution of the augmented system (16). Following the process in Section III-B, redefine the cost-to-go function as $\bar{Q} := \bar{\ell} + \bar{\boldsymbol{\lambda}}^\top \bar{\mathbf{g}} + \bar{\boldsymbol{\gamma}}^\top \bar{\mathbf{h}} + \bar{V}^+$, where $\bar{V}^+(\mathbf{y})$ is a redefined optimal cost-to-go. Then the backward recursion reads $\delta\mathbf{u} = -\hat{\bar{Q}}_{\mathbf{uu}}^{-1}(\hat{\bar{Q}}_{\mathbf{u}} + \hat{\bar{Q}}_{\mathbf{uy}}\delta\mathbf{y})$, where the terms $\hat{\bar{Q}}_{(\cdot)}$ related to the redefined cost-to-go are expressed as follows:$\boldsymbol{\alpha} \in \{\mathbf{y}, \mathbf{u}\}$,

$$\begin{aligned}\hat{\bar{Q}}_{\mathbf{u}} &= \bar{Q}_{\mathbf{u}} - \bar{Q}_{\mathbf{u}\bar{\boldsymbol{\lambda}}}[\mathsf{D}(\bar{\mathbf{g}})]^{-1}\bar{\mathbf{r}}_{\mathrm{in}} - \bar{Q}_{\mathbf{u}\bar{\boldsymbol{\gamma}}}\bar{\mathbf{r}}_{\mathrm{eq}},\\ \hat{\bar{Q}}_{\mathbf{u}\boldsymbol{\alpha}} &= \bar{Q}_{\mathbf{u}\boldsymbol{\alpha}} - \bar{Q}_{\mathbf{u}\bar{\boldsymbol{\lambda}}}[\mathsf{D}(\bar{\mathbf{g}})]^{-1}\,\mathsf{D}(\bar{\boldsymbol{\lambda}})\bar{Q}_{\bar{\boldsymbol{\lambda}}\boldsymbol{\alpha}} + \tfrac{1}{\mu}\bar{Q}_{\mathbf{u}\bar{\boldsymbol{\gamma}}}\bar{Q}_{\bar{\boldsymbol{\gamma}}\boldsymbol{\alpha}}.\end{aligned} \tag{28}$$

Notice that all the terms $\bar{(\cdot)}$ involved in the right-hand side are redefined with the new state variable $\mathbf{y}$. Letting $\bar{\boldsymbol{\lambda}} \equiv \boldsymbol{\lambda}$ and $\bar{\boldsymbol{\gamma}} \equiv \boldsymbol{\gamma}$, one can find that $\hat{\bar{f}} = \hat{\bar{f}}$ for $f \in \{Q_{\mathbf{u}}, Q_{\mathbf{uy}}, Q_{\mathbf{uu}}\}$.

By assumption that $\mathcal{Z}$ is the optimal solution, or equivalently, the optimization problem (1) has been solved in the sense of $[Q_{\mathbf{u}}^\top, \mathbf{r}_{\mathrm{in}}^\top, \mathbf{r}_{\mathrm{eq}}^\top]^\top = \mathbf{0}$, which implies that $\boldsymbol{\lambda} =$

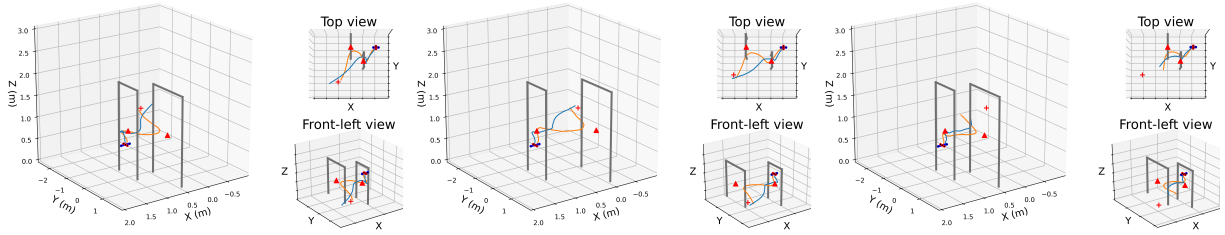
IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.



(a) Trajectory planning of the learned cost function test under the training setting.

(b) Generalization of the learned cost function on trajectory planning with a longer horizon.

(c) Generalization of the learned cost function on trajectory planning with a new initial condition.



(d) Generalization of the learned cost function on trajectory planning with a new desired position.

(e) Generalization of the learned cost function on trajectory planning with a new gate position.

(f) Generalization of the learned cost function on trajectory planning with new hardware system dynamics.

Fig. 12: Trajectory planning of the cost function learned from open-loop and closed-loop IRL (a) test under the training setting and its generalization to new settings, i.e., (b) longer horizon, (c) new initial positions, (d) new desired position, (e) new gate position, (f) new hardware system dynamics. Blue and orange solid lines denote the planned trajectories using the parameters learned from open-loop and closed-loop IRL, respectively. The initial state of the quadrotor is denoted by a red-blue icon. The red plus sign is the desired position. We use gray bars and red triangles to denote the gate and its center, respectively. *All the experiments are performed in the area shown in Fig. 11 and trajectories are recorded by the motion capture system.* All the quantitative measures are presented in Table VI.

TABLE VI: Measure of the trajectory planning test and its generalization results. (unit: meter)

Scenario	Minimum distance to gate 1 center		Minimum distance to gate 2 center		Final distance to the goal	
	CL	OL	CL	OL	CL	OL
Fig. 12(a)	0.20 ± 0.14	0.10 ± 0.12	0.58 ± 0.10	0.87 ± 0.09	0.41 ± 0.12	0.46 ± 0.14
Fig. 12(b)	0.09 ± 0.13	0.05 ± 0.04	0.28 ± 0.13	0.89 ± 0.02	0.40 ± 0.18	0.34 ± 0.11
Fig. 12(c)	0.06 ± 0.05	0.08 ± 0.05	0.65 ± 0.11	0.78 ± 0.05	0.39 ± 0.15	0.56 ± 0.41
Fig. 12(d)	0.10 ± 0.06	0.09 ± 0.05	0.60 ± 0.14	0.94 ± 0.08	0.43 ± 0.19	0.55 ± 0.50
Fig. 12(e)	0.31 ± 0.12	0.20 ± 0.05	0.47 ± 0.08	0.76 ± 0.06	0.82 ± 0.06	0.96 ± 0.18

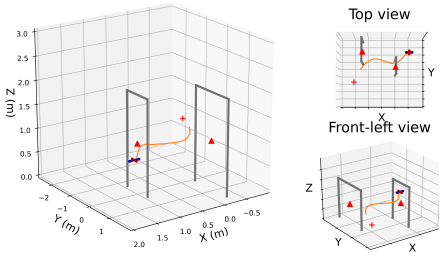


Fig. 13: Failure case of generalization of the cost function learned from closed-loop IRL on trajectory planning with a new gate position. Gate 2 is further away from gate 1 compared with Fig. 12(e). The legend is the same as that in Fig. 12.

$-\mu[\mathsf{D}(\mathbf{g})]^{-1}\mathbf{1}$ and $\boldsymbol{\gamma} = \mu^{-1}\mathbf{h}$. Substituting these to the Lagrange multipliers $\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\gamma}}$ in (28), one can obtain (17).

After obtaining the matrices $\bar{\hat{Q}}_{\mathbf{uy}}$ and $\bar{\hat{Q}}_{\mathbf{uu}}$, we are now at the stage of figuring out how to compute $\{\frac{\delta \mathbf{x}_k}{\delta \boldsymbol{\theta}}\}_{k=1}^{N}$ in order to compute the rest unknown $\{\frac{\delta \mathbf{u}_k}{\delta \boldsymbol{\theta}}\}_{k=1}^{N-1}$ by virtue of (15). Note that $\mathcal{Z}$ is the optimal solution which satisfies the dynamic equation in (1) and hence the augmented trajectory $\bar{\mathcal{Z}}$ satisfies the dynamic equation in (16). Taking the variation of the latter, one has

$$\delta \mathbf{y}^+ := \begin{bmatrix} \delta \boldsymbol{\theta}^+ \\ \delta \mathbf{x}^+ \end{bmatrix} = \bar{\mathbf{f}}_{\mathbf{y}} \delta \mathbf{y} + \bar{\mathbf{f}}_{\mathbf{u}} \delta \mathbf{u} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{f}_{\boldsymbol{\theta}} & \mathbf{f}_{\mathbf{x}} \end{bmatrix} \begin{bmatrix} \delta \boldsymbol{\theta} \\ \delta \mathbf{x} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_{\mathbf{u}} \end{bmatrix} \delta \mathbf{u},$$

by which we can establish the relationship among $\frac{\delta \mathbf{x}^+}{\delta \boldsymbol{\theta}}$, $\frac{\delta \boldsymbol{\theta}^+}{\delta \boldsymbol{\theta}}$, $\frac{\delta \mathbf{x}}{\delta \boldsymbol{\theta}}$ and $\frac{\delta \mathbf{u}}{\delta \boldsymbol{\theta}}$. Therefore, repeatedly evaluating $\frac{\delta \mathbf{x}}{\delta \boldsymbol{\theta}}$ and $\frac{\delta \mathbf{u}}{\delta \boldsymbol{\theta}}$ according to (15) and (18) for $k \in \mathcal{I}_N$, one can obtain $\frac{\mathrm{d}\bar{\mathcal{Z}}}{\mathrm{d}\boldsymbol{\theta}}$. ■

B. Proof of Theorem III.4

Proof. The result can be established by following a similar process to that of Theorem III.3. Suppose we use the active-set method to find the optimal solution of the augmented system (19), note that due to the absence of inequality constraints, the set identification step can be omitted. From the KKT condition

$$\begin{bmatrix} \bar{Q}^{\diamond}_{\mathbf{uu}} & (\bar{\mathbf{h}}^{\diamond}_{\mathbf{u}})^\top \\ \bar{\mathbf{h}}^{\diamond}_{\mathbf{u}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta \mathbf{u} \\ \boldsymbol{\gamma}^{\diamond} \end{bmatrix} = - \begin{bmatrix} \bar{Q}^{\diamond}_{\mathbf{uy}} \\ \bar{\mathbf{h}}^{\diamond}_{\mathbf{y}} \end{bmatrix} \delta \mathbf{y} - \begin{bmatrix} \bar{Q}^{\diamond}_{\mathbf{u}} \\ \mathbf{0} \end{bmatrix}, \tag{29}$$

one can obtain the backward recursion

$$\begin{aligned} \delta \mathbf{u} &= \left[(\bar{Q}^{\diamond}_{\mathbf{uu}})^{-1} [\mathbf{I} - (\bar{\mathbf{h}}^{\diamond}_{\mathbf{u}})^\top \mathbf{A}] \quad \mathbf{A}^\top \right] \left(\begin{bmatrix} \bar{Q}^{\diamond}_{\mathbf{uy}} \\ \bar{\mathbf{h}}^{\diamond}_{\mathbf{y}} \end{bmatrix} \delta \mathbf{y} + \begin{bmatrix} \bar{Q}^{\diamond}_{\mathbf{u}} \\ \mathbf{0} \end{bmatrix} \right), \\ &=: \bar{\mathbf{k}}^{\diamond} + \bar{\mathbf{K}}^{\diamond} \delta \mathbf{y}, \end{aligned}$$

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

where $\bar{\mathbf{k}}^{\diamond}$ and $\bar{\mathbf{K}}^{\diamond}$ are used to update $(\bar{V}^{\diamond}_{\mathbf{yy}})^{+}$ and $(\bar{V}^{\diamond}_{\mathbf{y}})^{+}$ similar to what was done in (11). The forward recursion can be obtained exactly in the same way as that of Theorem III.3. ■

C. Proof of Theorem IV.1

Proof. The first statement follows from [40, Theorem 10.3]. The second statement follows from the fact that $\boldsymbol{\theta}_t + \mathbf{c}$ with $\mathbf{c} \in \mathrm{Null}(\mathbf{J})$ still satisfies (26) if $\mathrm{Rank}(\mathbf{J}) < m_{\boldsymbol{\theta}}$. ■

D. Proof of Corollary IV.5

Proof. By Assumption IV.4-1), it follows from (5) that

$$V_{\mathbf{x}}(\mathbf{x}^{**}) = \hat{Q}_{\mathbf{x}}(\mathbf{x}^{**}, \mathbf{u}^{**}) = \boldsymbol{\phi}_{\mathbf{x}}^{\top}\boldsymbol{\theta} + \mathbf{f}_{\mathbf{x}}^{\top} V_{\mathbf{x}}^{+} + \mathbf{c}_{\mathbf{x}}, \tag{30}$$

where the last equality resulted from Assumption IV.4-3). Furthermore, it can be found that $V_{\mathbf{x}}$ is always linear in $\boldsymbol{\theta}$.

Stacking (30) for $k = 0, \ldots, m$, one can obtain

$$V_{\mathbf{x},1:m} = \boldsymbol{\phi}_{\mathbf{x},1:m}^{\top}\boldsymbol{\theta} + \mathsf{D}(\{\mathbf{f}_{\mathbf{x}}^{\top}\}_{k=1}^{m}) V_{\mathbf{x},2:m+1} + \mathbf{c}_{\mathbf{x},1:m}.$$

By some mathematical manipulations, one has

$$\mathbf{A}_{1:m} V_{\mathbf{x},1:m+1} + \boldsymbol{\phi}_{\mathbf{x},1:m}^{\top}\boldsymbol{\theta} - \mathbf{E}_{m+1} V_{\mathbf{x},m+2} + \mathbf{c}_{\mathbf{x},1:m} = \mathbf{0}.$$

It follows from Assumption IV.4-1) that

$$\begin{aligned} \boldsymbol{\epsilon}^{\mathcal{S}} &= \mathrm{col}(\{\hat{Q}_{\mathbf{u}}(\mathbf{x}^{**}, \mathbf{u}^{**})\}_{k=0}^{m}) \\ &= \mathrm{col}(\{\boldsymbol{\phi}_{\mathbf{u}}^{\top}\boldsymbol{\theta} + \mathbf{f}_{\mathbf{u}}^{\top} V_{\mathbf{x}}^{+} + \mathbf{c}_{\mathbf{u}})\}_{k=0}^{m}) \\ &= \boldsymbol{\phi}_{\mathbf{u},1:m}^{\top}\boldsymbol{\theta} + \mathbf{B}_{0:m} V_{\mathbf{x},1:m+1} + \mathbf{c}_{\mathbf{u},1:m} \\ &= \mathbf{J}_{\mathrm{lin},1}\boldsymbol{\theta} + \mathbf{J}_{\mathrm{lin},2} V_{\mathbf{x},m+2} + \mathbf{J}_{\mathrm{lin},3}. \end{aligned} \tag{31}$$

If $\boldsymbol{\epsilon}^{\mathcal{S}} = \mathbf{0}$ and $\mathrm{Rank}(\mathbf{J}_{\mathrm{lin},1:2}) = m_{\boldsymbol{\theta}} + m_{\mathbf{x}}$, one has $[\boldsymbol{\theta}^{\top}, V_{\mathbf{x},m+2}^{\top}]^{\top} = -(\mathbf{J}_{\mathrm{lin},1:2}^{\top}\mathbf{J}_{\mathrm{lin},1:2})^{-1}\mathbf{J}_{\mathrm{lin},1:2}^{\top}\mathbf{J}_{\mathrm{lin},3}$. As $\mu \to 0$, (30) and (31) reduce to the non-perturbed version of Bellman principle of optimality differentiated w.r.t. the state and control, respectively, which are the equilibrium conditions for the constrained IOC problem. ■

REFERENCES

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] 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.

[3] B. Hambly, R. Xu, and H. Yang, "Recent advances in reinforcement learning in finance," *Mathematical Finance*, vol. 33, no. 3, pp. 437–503, 2023.

[4] M. Zhang, P. I. Gómez, Q. Xu, and T. Dragicevic, "Review of online learning for control and diagnostics of power converters and drives: Algorithms, implementations and applications," *Renewable and Sustainable Energy Reviews*, p. 113627, 2023.

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *arXiv preprint arXiv:1606.06565*, 2016.

[6] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the 21st International Conference on Machine Learning*, 2004, p. 1.

[7] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[8] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 729–736.

[9] W. Jin, Z. Wang, Z. Yang, and S. Mou, "Pontryagin differentiable programming: An end-to-end learning and control framework," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7979–7992, 2020.

[10] W. Jin, S. Mou, and G. J. Pappas, "Safe pontryagin differentiable programming," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 034–16 050, 2021.

[11] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[12] W. Jin, T. D. Murphey, D. Kulić, N. Ezer, and S. Mou, "Learning from sparse demonstrations," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 645–664, 2022.

[13] H. Von Stackelberg, *Market structure and equilibrium*. Springer Science & Business Media, 2010.

[14] J. Bracken and J. T. McGill, "Mathematical programs with optimization problems in the constraints," *Operations Research*, vol. 21, no. 1, pp. 37–44, 1973.

[15] L. S. Pontryagin, *Mathematical theory of optimal processes*. Routledge, 2018.

[16] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[17] M. Patyerson and A. V. G. I. Rao, "a matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software*, vol. 41, no. 1, pp. 1–41, 2014.

[18] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[19] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[20] J. De O. Pantoja, "Differential dynamic programming and newton's method," *International Journal of Control*, vol. 47, no. 5, pp. 1539–1553, 1988.

[21] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5674–5680.

[22] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 695–702.

[23] Y. Aoyama, G. Boutselis, A. Patel, and E. A. Theodorou, "Constrained differential dynamic programming revisited," *arXiv preprint arXiv:2005.00985*, 2020.

[24] A. Pavlov, I. Shames, and C. Manzie, "Interior point differential dynamic programming," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 6, pp. 2720–2727, 2021.

[25] T. C. Lin and J. S. Arora, "Differential dynamic programming technique for constrained optimal control: Part 1: theoretical development," *Computational Mechanics*, vol. 9, no. 1, pp. 27–40, 1991.

[26] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7674–7679.

[27] S. El Kazdadi, J. Carpentier, and J. Ponce, "Equality constrained differential dynamic programming," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[28] W. Jallet *et al.*, "Constrained differential dynamic programming: A primal-dual augmented lagrangian approach," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[29] M. Kobilarov, D.-N. Ta, and F. Dellaert, "Differential dynamic programming for optimal estimation," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

[30] F. Bailly, J. Carpentier, and P. Souères, "Optimal estimation of the centroidal dynamics of legged robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[31] A. Oshin *et al.*, "Parameterized differential dynamic programming," *arXiv preprint arXiv:2204.03727*, 2022.

[32] A. Oshin, H. Almubarak, and E. A. Theodorou, "Differentiable robust model predictive control," *Robotics: Systems and Science*, 2023.

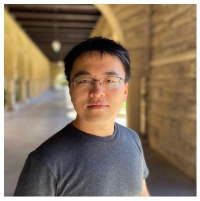
[33] A. Keshavarz, Y. Wang, and S. Boyd, "Imputing a convex objective function," in *2011 IEEE International Symposium on Intelligent Control*. IEEE, 2011, pp. 613–619.

[34] P. Englert, N. A. Vien, and M. Toussaint, "Inverse KKT: Learning cost functions of manipulation tasks from demonstrations," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[35] M. Johnson, N. Aghasadeghi, and T. Bretl, "Inverse optimal control for deterministic continuous-time nonlinear systems," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 2906–2913.
[36] T. L. Molloy, J. J. Ford, and T. Perez, "Finite-horizon inverse optimal control for discrete-time nonlinear systems," *Automatica*, vol. 87, pp. 442–446, 2018.
[37] W. Jin, D. Kulić, S. Mou, and S. Hirche, "Inverse optimal control from incomplete trajectory observations," *The International Journal of Robotics Research*, vol. 40, no. 6-7, pp. 848–865, 2021.
[38] T. L. Molloy, J. J. Ford, and T. Perez, "Online inverse optimal control for control-constrained discrete-time systems on finite and infinite horizons," *Automatica*, vol. 120, p. 109109, 2020.
[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
[40] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.
[41] A. Forsgren, P. E. Gill, and M. H. Wright, "Interior methods for nonlinear optimization," *SIAM Review*, vol. 44, no. 4, pp. 525–597, 2002.
[42] K. Jittorntrum, "An implicit function theorem," *Journal of Optimization Theory and Applications*, vol. 25, no. 4, pp. 575–577, Aug. 1978.
[43] S. Ghadimi and M. Wang, "Approximation methods for bilevel programming," *arXiv preprint arXiv:1802.02246*, 2018.
[44] J. R. Magnus and H. Neudecker, *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.
[45] W. Xue, P. Kolaric, J. Fan, B. Lian, T. Chai, and F. L. Lewis, "Inverse reinforcement learning in tracking control based on inverse optimal control," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 570–10 581, 2021.
[46] A. Pinosky, I. Abraham, A. Broad, B. Argall, and T. D. Murphey, "Hybrid control for combining model-based and model-free reinforcement learning," *The International Journal of Robotics Research*, vol. 42, no. 6, pp. 337–355, 2023.
[47] M. Lutter and J. Peters, "Combining physics and deep learning to learn continuous-time dynamics models," *The International Journal of Robotics Research*, vol. 42, no. 3, pp. 83–107, 2023.
[48] R. E. Allen and M. Pavone, "A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance," *Robotics and Autonomous Systems*, vol. 115, pp. 174–193, 2019.
[49] P. Sharma, B. Sundaralingam, V. Blukis, C. Paxton, T. Hermans, A. Torralba, J. Andreas, and D. Fox, "Correcting robot plans with natural language feedback," *arXiv preprint arXiv:2204.05186*, 2022.

Kun Cao is a faculty member at Tongji University, Shanghai, China. He received the B.Eng. degree in mechanical engineering from Tianjin University, Tianjin, China, in 2016, and the Ph.D. degree in electrical and electronic engineering from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2021. He was the 2022 Wallenberg-NTU Presidential Postdoctoral Fellow with the School of Electrical and Electronic Engineering at Nanyang Technological University and the School of Electrical Engineering and Computer Science at KTH Royal Institute of Technology in Sweden. His research interests include individual and swarm intelligence.

Xinhang Xu received the B.Eng. degree in Automation from the South China University of Technology, Guangzhou, China, and the M.Sc. degree in Electrical and Electronic Engineering from Nanyang Technological University, Singapore, in 2023. He is currently pursuing the Ph.D. degree in Electrical and Electronic Engineering at Nanyang Technological University, Singapore. His research interests include intelligent sensing, social navigation, aerial manipulation, and tethered robotics.



Wanxin Jin is an Assistant Professor in the School for Engineering of Matter, Transport, and Energy at Arizona State University. From 2021 to 2023, he was a postdoctoral fellow at the GRASP Lab at the University of Pennsylvania. He received his Ph.D. from Purdue University in 2021. At ASU, Dr. Jin leads the Intelligent Robotics and Interactive Systems Lab, where his research focuses on contact-rich manipulation and robot learning with human feedback.

Karl H. Johansson is Swedish Research Council Distinguished Professor in Electrical Engineering and Computer Science at KTH Royal Institute of Technology in Sweden and Founding Director of Digital Futures. He earned his MSc degree in Electrical Engineering and PhD in Automatic Control from Lund University. He has held visiting positions at UC Berkeley, Caltech, NTU and other institutions. His research interests focus on networked control systems and cyber-physical systems with applications in transportation, energy, and automation networks. For his scientific contributions, he has received numerous best paper awards and various other distinctions from IEEE, IFAC, and other organizations. He has been awarded Distinguished Professor by the Swedish Research Council, Wallenberg Scholar by the Knut and Alice Wallenberg Foundation, Future Research Leader by the Swedish Foundation for Strategic Research. He has also received the triennial IFAC Young Author Prize, IEEE CSS Distinguished Lecturer, IFAC Outstanding Service Award, and IEEE CSS Hendrik W. Bode Lecture Prize. His extensive service to the academic community includes being President of the European Control Association, IEEE CSS Vice President Diversity, Outreach & Development, and Member of IEEE CSS Board of Governors and IFAC Council. He has served on the editorial boards of Automatica, IEEE TAC, IEEE TCNS and many other journals. He has also been a member of the Swedish Scientific Council for Natural Sciences and Engineering Sciences. He is Fellow of both the IEEE and the Royal Swedish Academy of Engineering Sciences.

Lihua Xie received the Ph.D. degree in electrical engineering from the University of Newcastle, Australia, in 1992. Since 1992, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where he is currently President's Chair in Control Engineering and Director, Center for Advanced Robotics Technology Innovation. He has served as the Head of Division of Control and Instrumentation and Co-Director, Delta-NTU Corporate Lab for Cyber-Physical Systems. He held teaching appointments in the Department of Automatic Control, Nanjing University of Science and Technology from 1986 to 1989.

Dr Xie's research interests include robust control and estimation, networked control systems, multi-agent networks, and unmanned systems. He has published 11 books and numerous papers in the areas, and holds 25 patents/technical disclosures. He was listed as a highly cited researcher by Thomson Routers and Clarivate Analytics. He is an Editor-in-Chief for Unmanned Systems and Associate Editor for IEEE Control System Magazine. He has served as Editor for IET Book Series in Control and Associate Editor for a number of journals including IEEE Transactions on Automatic Control, Automatica, IEEE Transactions on Control Systems Technology, IEEE Transactions on Network Control Systems, etc. He was an IEEE Distinguished Lecturer (Jan 2012 – Dec 2014) and the General Chair of the 62nd IEEE Conference on Decision and Control (CDC 2023). He is currently Vice-President of IEEE Control System Society. Dr Xie is Fellow of Academy of Engineering Singapore, IEEE, IFAC, and CAA.