

Interpreting the Observed Behaviour of a Class of Autonomous Linear Systems using Explainable Inverse Reinforcement Learning

Adolfo Perrusquía, *Senior Member, IEEE*, Mengbang Zou, Weisi Guo, *Senior Member, IEEE*,

Abstract—One of the main challenges faced by the society is how to verify safety of autonomous systems. As the level of autonomy grows, it becomes critical to understand why an autonomous system exhibits a particular behaviour and what we need to do to ensure it does not pose risks to people or the surrounding environment. To this end, we need to infer, from observational data, the necessary evidence or decision-making factors to effectively interpret the autonomous system's behaviour. Previous approaches in the literature have used explainable models to provide simple input-output mappings to interpret the observer behaviour. However, autonomous systems do not pose simple input-output relationships which compromises the veracity of the interpretations. To alleviate this issue, this paper proposes a novel explainable inverse reinforcement learning (EXIRL) that provides the evidence to interpret the behaviour of a class of autonomous linear systems. The approach uses a combined model-free and model-based mechanism that improves the learning phase of traditional inverse learning algorithms to accurately infer the evidence from the data. The inferred evidence is used to design counterfactual explanations to interpret the observed behaviour and provide the mechanisms to modify it into a desired one. Simulation studies using a DJI Phantom 4 and a quadrotor model are provided to show the benefits and challenges of the proposed work.

Index Terms—Explainable Inverse Reinforcement Learning, Counterfactual explanations, Evidence space, Autonomous systems, Interpretations.

I. INTRODUCTION

The deployment of safe autonomous platforms is critical to ensure the smooth operation of services in transport and smart living sectors [1], [2]. The observed behaviour of autonomous platforms needs to be interpretable to determine if the autonomous system is safe or pose any risk to the surrounding environment and people. More broadly, the challenge of inferring and verifying the behaviour of autonomous algorithms extends beyond physical platforms to optimisation and decision-making processes embedded in critical digital infrastructure. In particular, emerging 6G telecommunication networks increasingly rely on autonomous control layers to manage resource allocation, network slicing, and service orchestration, making the verification and interpretability of these algorithms equally important for ensuring reliable and safe operation.

Expert rules are usually defined to classify the behaviour of autonomous platforms as either normal or anomalous

[3]–[5]. However, these rules have a simplistic nature that cannot provide solid evidence (decision-making factors) to interpret the behaviour of the autonomous system effectively. Other approaches use deep learning models to identify high-dimensional features from trajectory data to classify the behaviour of autonomous platforms [6], [7]. These kind of behaviour classification algorithms can discriminate trajectory data based on shared-features, however they cannot interpret the factors behind an exhibited performance. Therefore, other methodologies have been studied to address this challenge.

Reinforcement Learning (RL) and Adaptive Dynamic Programming (ADP) algorithms have been widely used in autonomous systems to learn and adapt its decisions from observational data. Model-free RL/ADP architectures (e.g., Q-learning [8] and actor-critic models [9]) are commonly used in the literature due to their ability to learn optimal control policies from data. A complete survey of RL/ADP algorithms is given in [10], [11].

Inverse RL (IRL) algorithms [12], [13] are used to infer both the policy and reward function of RL/ADP algorithms from observational data. These approaches assume some partial knowledge of the reward function and/or specific structure to ensure convergence of both the policy and reward parameters. IRL algorithms have been used in safety critical control applications [14], [15] to inject an expert model into a learner model using either the inferred reward function or policy. However, IRL methods are computational expensive and they cannot provide by themselves the complete evidence to interpret the observed behaviour of an autonomous system. This is because IRL methods just focus on inferring policies and equivalent rewards, without considering the other learning elements within a RL/ADP algorithm (e.g., the inferred state-transition model, the real optimal control objective and its solution). This is, as best of our knowledge, an open gap that we aim to address in this work.

To solve the aforementioned challenges, this paper reports a novel explainable IRL (EXIRL) algorithm that enables the interpretation of autonomous behaviour from observational data. Here, the benefit of the behaviour interpretation is that allows to uncover why the autonomous system exhibits a particular behaviour and the mechanisms to inject a desired performance. This is done by uncovering each learning element of the decision-making process to build the evidence to design counterfactual explanations. Specifically, we infer the control policy, reward function, Q-function, and the state-transition matrices. The algorithm combines a model-free RL

A. Perrusquía, Mengbang Zou, and Weisi Guo are with the Faculty of Engineering and Applied Sciences, Cranfield University, Bedford, UK (e-mail: adolfo.perrusquia-guzman@cranfield.ac.uk).

algorithm for the optimal control inference with a model-based inverse optimal control for reward inference. The combined power of both approaches regularise the learning manifold of the algorithm to obtain unique solutions for the inferred elements. Convergence of the proposed approach is assessed using Lyapunov recursions. Simulation experiments are conducted under a DJI Phantom 4 drone and a quadcopter to demonstrate the feasibility of the approach compared with previous IRL methods.

A. Literature review

The interpretation problem [16], [17] has been addressed by explainable artificial intelligence (XAI) tools either using perturbation or gradient-based methods [18]–[21]. These tools need access to the algorithm design within the autonomous platform. Counterfactual explanations [22] have been used to interpret autonomous decisions from observational data by providing clear common sense statements of why a decision has been applied [23]. The challenge with counterfactual explanations is to uncover all the potential factors that influence a specific observed behaviour.

Imitation learning (IL) [24], [25] and inverse reinforcement learning (IRL) [13] have been used as a kind of explainable tool to infer the policy and/or reward function associated to the observed behaviour [26], [27] by exploiting their adversarial learning structure [28]–[30]. However, the obtained policies or reward functions are not easy to explain or interpret [31]. Another class of IRL algorithms [32]–[34] based on ADP methods have been used to infer the reward function from observational data. The key idea of these methods is to train an RL/ADP algorithm under iterative reward function weights until the difference between the reward weights of two consecutive iterations is less or equal than a threshold or, in the best scenario, when it converges to zero. The IRL approaches used in the literature [34], [35] assume partial knowledge of the reward structure or fix some of its weights. This causes that equivalent reward functions are obtained as output that generate the same control policy, but it does not match with the real reward function design. In [14] an anomaly detector algorithm was proposed based on an IRL framework. The key idea is to identify differences either in the objective or intention (policy) of an expert and learner systems and subsequently infer the expert behaviour into the learner system. This approach is computational expensive since it has two nested loops and it requires large amount of data to infer equivalent objective functions.

Alternatively, intended outcomes [36] have been explored to interpret autonomous decisions based on the intended goal of the autonomous platform. The key idea of this approach is to determine why a particular policy is obtained by understanding the inference of the state-transition function from the data. This tool can be used to interpret decisions and anomalies based on the obtained policy and data.

Based on the literature the open challenges are: (1) IRL/IL is limited to learn equivalent rewards which cannot capture the real control objective, (2) IRL/IL methods are computational expensive and requires large amount of data, (3) IRL/IL is

not capable to infer other learning elements that are necessary for the construction of the evidence for interpretations, and (4) counterfactual explanations need to have the complete evidence to provide, in a clear common sense, interpretations of the observed behaviour.

B. Contributions, outline and notations

The main contributions of this work are fourfold:

- 1) The proposed EXIRL algorithm is capable to infer the complete evidence from observational data to interpret the autonomous system behaviour.
- 2) A state-transition inference model is designed to determine what state-transition model is inferred by the RL algorithm. This allows to better understand how the RL is learning instead of using independent system identification models.
- 3) In contrast to previous IRL approaches, the EXIRL requires less data and computational resources by transforming the IRL algorithm into an inverse optimal control (IOC) based on the inferred state-transition matrices.
- 4) Counterfactual explanations are designed to provide interpretations of the observed behaviour and to provide the mechanisms to inject a desired one based on the collected evidence.

The outline of the paper is as follows. Section II provides the problem statement. Section III establishes the preliminaries and the LQR control problem. Section IV describes the proposed explainable inverse reinforcement learning. Section V reports the simulations studies using two different autonomous systems, and the conclusions are presented in Section VI.

Throughout this paper, $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n \times m}$ denote the spaces of natural numbers, real numbers, real n -vectors, and real $n \times m$ -matrices, respectively; $I_n \in \mathbb{R}^{n \times n}$ denotes an $n \times n$ identity matrix; $\mathbf{1}_n$ and $\mathbf{0}_n$ define $n \times n$ matrices of ones and zeros; $\otimes$ and $\odot$ denote the Kronecker and symmetric Kronecker product; $\text{vec}(\mathbf{A})$ and $\text{vech}(\mathbf{A})$ define the matrix vectorization and half-vectorization, $\text{mat}(\mathbf{x})$ define the matricization operator, $\|\mathbf{x}\|$ and $\|\mathbf{A}\|$ define the Euclidean and induced norms of a vector and matrix, respectively, where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $n, m \in \mathbb{N}$.

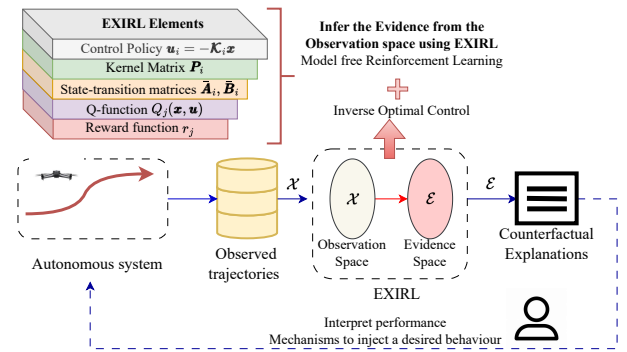


Fig. 1. High-level diagram of the proposed approach for behaviour interpretation of autonomous systems

II. PROBLEM STATEMENT

Fig. 1 shows the high-level diagram of the proposed methodology for interpreting the behaviour of autonomous systems. In this diagram, an autonomous platform is driven by an optimal linear quadratic regulator (LQR) controller or reinforcement learning (RL) controller. Here, we would like to understand why the autonomous platform is exhibiting a particular trajectory profile and/or closed-loop behaviour and how this behaviour can be modified into a desired one. To this end, the trajectories of the autonomous platform are stored in the dataset $\mathcal{X}$ denoted as the observation space. This observation space is mapped into an evidence space $\mathcal{E}$ which provides the necessary information to interpret the observed behaviour of the autonomous platform. This mapping between the observation space $\mathcal{X}$ and the evidence space $\mathcal{E}$ is performed by an explainable inverse reinforcement learning (EXIRL) architecture that infers: (1) the control policy, (2) the reward function, (3) the state-transition matrices, (4) the solution of the LQR problem (kernel matrix), and (5) the Q-function, as evidence to interpret the observed behaviour. The outputs of the EXIRL architecture are used to build counterfactual explanations that helps to provide simple causal interpretations of (i) why the autonomous system exhibits a particular behaviour, (ii) what it is inferring from the data and (3) which are the mechanisms to achieve a desired closed-loop performance.

III. PRELIMINARIES: LINEAR QUADRATIC REGULATOR (LQR) CONTROL PROBLEM

Consider an autonomous platform whose dynamics can be described by the following linear time-invariant system,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad (1)$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the state vector and $\mathbf{u}(t) \in \mathbb{R}^m$ defines the control inputs. The matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times m}$ define the state transition matrices of the autonomous platform model and $\mathbf{x}_0 \in \mathbb{R}^n$ is an initial condition.

Assumption 1: In this paper, we use the following structure for matrices $\mathbf{A}$ and $\mathbf{B}$ to model the dynamics of the autonomous platform,

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{n/2} & \mathbf{I}_{n/2} \\ \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0}_{n/2 \times m} \\ \mathbf{B}_1 \end{bmatrix}, \quad (2)$$

where $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{(n/2) \times (n/2)}$ and $\mathbf{B}_1 \in \mathbb{R}^{n/2 \times m}$ contains the main parameters of the autonomous platform.

Assumption 1 is a common state space model of many linearised Euler-Lagrange models of autonomous platforms including robot manipulators [37], [38], ground vehicles [39], and drones [40]. For example,

- 1) Rigid arm [38]: $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix}$, where g is the gravity, l is the length of the arm, and m is the mass.
- 2) Mass-spring-damper system [41]: $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$, where k is the stiffness coefficient, b is the damping coefficient, and m is the mass.

- 3) Quadcopter dynamic model in hover condition [40]:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0}_{3 \times 1} & 0 & 0 \\ g & -g & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix}, \quad \text{where } g \text{ is the gravity and } m \text{ is the mass.}$$

Assume an optimal linear quadratic regulator (LQR) control design that fulfils the following optimization problem, [42], [43],

$$\min_{\mathbf{u} \in U} \int_t^\infty (\mathbf{x}^\top(\tau) \mathbf{Q} \mathbf{x}(\tau) + \mathbf{u}^\top(\tau) \mathbf{R} \mathbf{u}(\tau)) d\tau \quad (3)$$

subject to $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$, $\mathbf{x}(t_0) = \mathbf{x}_0$,

where $U \subset \mathbb{R}^m$ is a set of admissible control policies, $\mathbf{Q} = \mathbf{Q}^\top \succeq 0 \in \mathbb{R}^{n \times n}$ and $\mathbf{R} = \mathbf{R}^\top \succ 0 \in \mathbb{R}^{m \times m}$ are the weight matrices of the objective function. The optimal control policy that fulfils (3) is

$$\mathbf{u}^*(t) = -\mathbf{R}^{-1} \mathbf{B}^\top \mathbf{P} \mathbf{x}(t) = -\mathbf{K} \mathbf{x}(t), \quad (4)$$

where $\mathbf{K} \in \mathbb{R}^{m \times n}$ is the control gain and $\mathbf{P} = \mathbf{P}^\top \succ 0 \in \mathbb{R}^{n \times n}$ is a kernel matrix that verifies the following algebraic Riccati equation (ARE) [44]

$$\mathbf{A}^\top \mathbf{P} + \mathbf{P} \mathbf{A} + \mathbf{Q} - \mathbf{P} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^\top \mathbf{P} = \mathbf{0}_n. \quad (5)$$

The solution of the optimisation problem (3) can be written as the optimal value function $V^*(\mathbf{x})$ given by

$$V^*(\mathbf{x}(t)) = \min_{\mathbf{u} \in U} \int_t^\infty r(\mathbf{x}(\tau), \mathbf{u}(\tau)) d\tau = \mathbf{x}^\top(t) \mathbf{P} \mathbf{x}(t), \quad (6)$$

where $r = r(\mathbf{x}(t), \mathbf{u}(t)) := \mathbf{x}^\top(t) \mathbf{Q} \mathbf{x}(t) + \mathbf{u}^\top(t) \mathbf{R} \mathbf{u}(t)$ is the reward function.

The state-action value function also known as Q-function is usually used to evaluate the state-action pair. The Q-function is given by

$$Q(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix}^\top \begin{bmatrix} \mathbf{Q}_{xx} & \mathbf{Q}_{xu} \\ \mathbf{Q}_{xu}^\top & \mathbf{Q}_{uu} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix}, \quad (7)$$

where $\mathbf{Q}_{xx} = \mathbf{A}^\top \mathbf{P} + \mathbf{P} \mathbf{A} + \mathbf{Q} + \mathbf{P} \in \mathbb{R}^{n \times n}$, $\mathbf{Q}_{xu} = \mathbf{P} \mathbf{B} \in \mathbb{R}^{n \times m}$ and $\mathbf{Q}_{uu} = \mathbf{R} \in \mathbb{R}^{m \times m}$. When the Q-function is evaluated in the optimal control policy $\mathbf{u}^*(t)$ yields the optimal Q-function that satisfies $Q^*(\mathbf{x}(t), \mathbf{u}^*(t)) = V^*(\mathbf{x}(t)) = \mathbf{x}^\top(t) \mathbf{P} \mathbf{x}(t)$.

The previous optimal control derivation determines the required elements for its design. Specifically, the kernel matrix $\mathbf{P}$ of the ARE (5), the reward function $r(\mathbf{x}, \mathbf{u})$, and the state-transition matrices $\mathbf{A}, \mathbf{B}$. These elements are usually known by the designer. However, these elements are hidden for external observers which require to infer them from the observed data. Here, in this paper we aim to contribute in the inference of these hidden attributes to explain the performance of the autonomous platform in clear common sense mechanisms. In the next section, the proposed explainable inverse reinforcement learning (EXIRL) algorithm for the inference of these learning elements is provided.

IV. EXPLAINABLE INVERSE REINFORCEMENT LEARNING (EXIRL)

The high-level diagram of the proposed EXIRL is depicted on Fig. 2. Real-time or previous collected data is used to construct the observation space $\mathcal{X}$ which is subsequently used by an off-policy model-free reinforcement learning (RL) algorithm to obtain the kernel matrix P_i associated to the iterative weight matrices Q_j and R_j . The outputs of the RL model feed a state-transition inference model that allows to uncover which model is learned by the RL agent. Both the outputs of the RL and state-transition inference models, and the collected data are used to feed an inverse optimal control (IOC) model that updates the iterative reward weight matrices Q_{j+1} and R_{j+1} . These improved weights are used by the RL algorithm and start over again the process until both the iterative reward weights and kernel matrix P_i converge. The outputs of the EXIRL are used to design counterfactual explanations that interpret the observed behaviour of the autonomous platform and the mechanisms to modify its behaviour into a desired closed-loop performance.

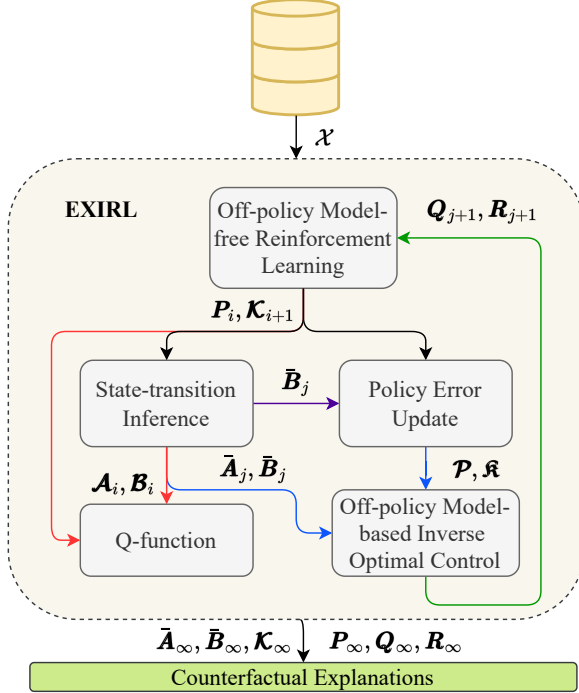


Fig. 2. EXIRL high-level diagram. From the observation space $\mathcal{X}$ and weight matrices Q_j and R_j , it is extracted the associated kernel matrix P_i and the control gain $\mathcal{K}_{i+1}$ using an off-policy model-free reinforcement learning. These matrices are subsequently used by a state-transition inference algorithm and a policy error update to obtain the state-transition matrices $\bar{A}_j, \bar{B}_j$ and improved kernel and gain matrices $\mathcal{P}, \mathcal{K}$. The output of these modules are used by an inverse optimal control algorithm to infer new weight matrices Q_{j+1} and R_{j+1} that updates the off-policy RL algorithm as well as the associated Q-function. This process continues until convergence and the evidence space $\mathcal{E} = \{Q_\infty, R_\infty, \bar{A}_\infty, \bar{B}_\infty, P_\infty, \mathcal{K}_\infty\}$ is constructed. The evidence space $\mathcal{E}$ is used to construct counterfactual explanations to interpret the observed behaviour.

A. Model-free Reinforcement Learning

Consider that we have an observation space $\mathcal{X} = \{X, U, r\}$ constructed from the observed trajectories of an autonomous

platform (1). The matrices within the dataset are given by

$$\begin{aligned} X &= [x(t - (\mathcal{T} - 1)T) \quad \cdots \quad x(t - T) \quad x(t)] \in \mathbb{R}^{n \times \mathcal{T}}, \\ U &= [u(t - (\mathcal{T} - 1)T) \quad \cdots \quad u(t - T) \quad u(t)] \in \mathbb{R}^{m \times \mathcal{T}}, \\ r &= [r(t - (\mathcal{T} - 1)T) \quad \cdots \quad r(t - T) \quad r(t)] \in \mathbb{R}^{1 \times \mathcal{T}}, \end{aligned}$$

where $\mathcal{T} > 0$ is the length of the data trajectories and $T > 0$ is the sample time. The EXIRL algorithm is used to obtain the reward function from observational data that describes the design of the optimal control problem. To this end, consider the following equivalent version of (1)

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu_i(t) + B(u(t) - u_i(t)), \\ &= (A - BK_i)x(t) + B(u(t) - u_i(t)), \end{aligned} \quad (8)$$

where $u_i(t) = -\mathcal{K}_i x(t) \in \mathbb{R}^m$ defines an iterative control policy that is updated in each RL iteration i , and $\mathcal{K}_i \in \mathbb{R}^{m \times n}$ is the associated iterative control gain. Consider the ARE (5) written in terms of the following iterative matrices,

$$\begin{aligned} A^\top P_i + P_i A + Q_j - \mathcal{K}_i^\top R_j \mathcal{K}_i &= 0_n, \\ (A - BK_i)^\top P_i + P_i (A - BK_i) + Q_j + \mathcal{K}_i^\top R_j \mathcal{K}_i &= 0_n, \end{aligned} \quad (9)$$

where $P_i = P_i^\top \succ 0 \in \mathbb{R}^{n \times n}$ defines an iterative kernel matrix that is updated in each iteration i of the RL part, $Q_j = Q_j^\top \succeq 0 \in \mathbb{R}^{n \times n}$ and $R_j = R_j^\top \succ 0 \in \mathbb{R}^{m \times m}$ are iterative matrices updated in each IOC iteration j . In this paper, we used the batch least-squares (BLS) policy iteration algorithm proposed in [14] to compute the optimal kernel matrix P_i and the optimal control gain $\mathcal{K}_{i+1}$. The key idea is to construct a system of equations $\Sigma \theta_i = \xi$, from $\kappa \geq \iota = \frac{1}{2}n(n+1) + nm$ samples; and $\theta_i = [\text{vech}^\top(P_i), \text{vec}^\top(\mathcal{K}_{i+1})]^\top \in \mathbb{R}^\iota$, $\sigma_i(t) = -x^\top(t)Q_j x(t) - u_i^\top(t)R_j u_i(t)$, $\Sigma = [\Phi, -2I_{ux}] \in \mathbb{R}^{\kappa \times \iota}$, and

$$\begin{aligned} \Phi &= \left[x^\top(\tau) \bar{\otimes} x^\top(\tau) \Big|_t^{t+T}, \dots, x^\top(\tau) \bar{\otimes} x^\top(\tau) \Big|_{t+(\kappa-1)T}^{t+\kappa T} \right]^\top \\ I_{ux} &= \left[\int_t^{t+T} r_u^\top(\tau) \otimes x^\top(\tau) d\tau, \dots, \int_{t+(\kappa-1)T}^{t+\kappa T} r_u^\top(\tau) \otimes x^\top(\tau) d\tau \right]^\top \\ \xi &= \left[\int_t^{t+T} \sigma_i(\tau) d\tau, \dots, \int_{t+(\kappa-1)T}^{t+\kappa T} \sigma_i(\tau) d\tau \right]^\top, \end{aligned}$$

where $\tilde{u}_i = u - u_i \in \mathbb{R}^m$ is the policy error and $r_u(t) = R_j \tilde{u}_i(t)$. Then, the solution of the model-free RL is given by the solution of BLS rule $\theta_i = (\Sigma^\top \Sigma)^{-1} \Sigma^\top \xi$.

Remark 1: The collection of at least ι samples is achieved under the fulfilment of a persistent of excitation (PE) condition [10]. This is done by adding a probing noise in the control $u(t)$, e.g., random noise or sinusoidal signal with different frequencies [45].

B. State-transition matrices inference

Previous model-free IRL architectures have two learning loops to infer the reward function weights: i) an inner RL loop that computes the control gain and the kernel solution of an ARE associated to iterative reward weights, and ii) an outer loop that updates the reward weight matrices based on

the RL outputs. This process is time consuming and requires rich trajectory data throughout the complete experiment. To overcome this issue, this paper proposes to infer the state-transition model of the autonomous platform as additional evidence to interpret the observed behaviour.

Remark 2: This paper aims to obtain the state-transition model that the RL algorithm infers from the data. This is different from identifying the system model from the data which is independent to the RL algorithm.

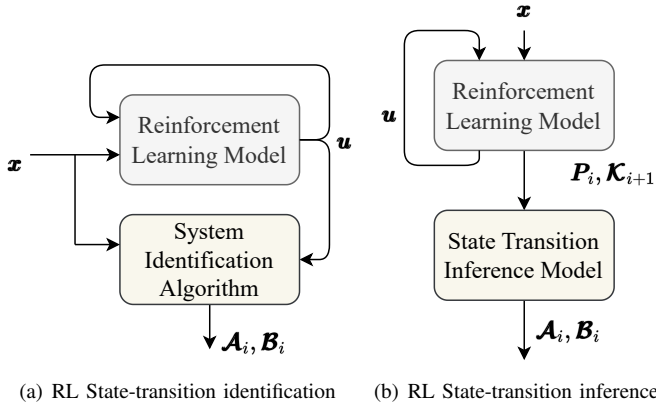


Fig. 3. Difference between state-transition identification and inference

To motivate better Remark 2, consider the high level diagrams of Fig. 3. Fig. 3(a) shows a simple RL state-transition identification diagram where the estimation of the state-transition matrices is obtained from an independent system identification module. In this approach we can adopt well-known identification techniques such as least-squares (LS), dynamic mode decomposition (DMD) or gradient methods [46]. However, this specific diagram cannot interpret what the RL is learning from the data. Moreover, the identification algorithm can perform the estimation of the state-transition matrices even the RL algorithm is not working properly, such that biased interpretations can be obtained. This means that the precision of the state-transition identification depends mainly on the quality of the data.

Fig. 3(b) shows the RL state-transition inference diagram that uses directly the kernel matrix P_i of the RL to infer the state-transition model. This inference model adds a new explainable dimension to the overall framework to understand what the RL agent has inferred from the data. Identification models cannot add this explainable dimension due to its independent design. However, the inference of the state-transition matrices is not straightforward because the ARE (5) is non-linear both in A and B , whilst P, Q, R are symmetric. This means that we have an ill-posed problem that can have infinite solutions. To overcome this challenge, we use Assumption 1 to add constraints in the state-transition matrices such that their inference from the RL outputs can be performed effectively.

To do so, the ARE (5) is written as the following Lyapunov equation,

$$\mathcal{A}_i^\top P_i + P_i \mathcal{A}_i = T_i, \quad (10)$$

where $\mathcal{A}_i \in \mathbb{R}^n$ is an estimation of matrix A in step i and $T_i = \mathcal{K}_{i+1}^\top R_j \mathcal{K}_{i+1} - Q_j$. Let P_i be written as

$P_i = [p_1^\top \dots p_n^\top]^\top$, where $p_k \in \mathbb{R}^{1 \times n}$ is the k -th row of the kernel matrix P_i with $k \in \{1, \dots, n\}$. Then, (10) can be expressed as

$$\mathcal{N} \text{vec}(\mathcal{A}_i) = \text{vec}(T_i), \quad (11)$$

where $\mathcal{N} = (I_n \otimes P_i + W_i) \in \mathbb{R}^{n^2 \times n^2}$ and $W_i = [I_n \otimes p_1^\top \dots I_n \otimes p_n^\top]^\top \in \mathbb{R}^{n^2 \times n^2}$. By construction $\mathcal{N}$ has $\text{rank}(\mathcal{N}) < n^2$. Notice that many elements of A are zero and hence, they are not required to be inferred. This special structure can be regarded as a physics-informed constraint that forces the inference algorithm to estimate the real values. To this end, let write $\mathcal{N} = [\eta_1, \dots, \eta_{n^2}]$, where $\eta_q \in \mathbb{R}^{n^2}$ with $q = 1, \dots, n^2$. Consider the next auxiliary matrix,

$$N = \begin{bmatrix} 0_{n/2} & I_{n/2} \\ 1_{n/2} & 1_{n/2} \end{bmatrix}, \quad n = \text{vec}(N) \in \mathbb{R}^{n^2}. \quad (12)$$

Let $Z = \{z_1, \dots, z_k\}$ be the set of $k = \frac{1}{2}n(n-1)$ positional indices of zeros in n . A submatrix $\widehat{\mathcal{N}}$ is constructed from $\mathcal{N}$ by deleting the columns η_q associated to each index in Z as,

$$\widehat{\mathcal{N}} = [\eta_l] \in \mathbb{R}^{n^2 \times (\frac{1}{2}n(n+1))}, \quad l \notin Z. \quad (13)$$

The terms $\widehat{A}_i$ of $\mathcal{A}_i$ associated to the non-zero terms of n are estimated by

$$\text{vec}(\widehat{A}_i) = \widehat{\mathcal{N}}^{-1} \text{vec}(T_i). \quad (14)$$

The matrix $\mathcal{A}_i$ is recovered by expanding the estimated vector $\text{vec}(\widehat{A}_i)$ with zeros in the corresponding indices Z .

Remark 3: In this case $\text{rank}(\widehat{\mathcal{N}}) \leq \frac{1}{2}n(n+1)$. However, since the number of variables to estimate are reduced, then the solution that minimizes $\|\widehat{\mathcal{N}} \text{vec}(\widehat{A}_i) - \text{vec}(T_i)\|$ is given by the real values of A , i.e., $\text{vec}(\mathcal{A}_i) = \text{vec}(A)$.

The estimation of $\mathcal{B}_i$ of B is obtained with the improved gain as,

$$\begin{aligned} \mathcal{K}_{i+1}^\top &= P_i \mathcal{B}_i R_j^{-1}, \\ \mathcal{B}_i &= P_i^{-1} \mathcal{K}_{i+1}^\top R_j. \end{aligned} \quad (15)$$

When $\|\mathcal{K}_{i+1} - \mathcal{K}_i\| \leq \epsilon_K$ for a small $\epsilon_K > 0$, then the matrix $\mathcal{A}_i$ and $\mathcal{B}_i$ are fixed to apply the model-based inverse optimal control (IOC). We denote these fixed matrices as $\overline{A}_j$ and $\overline{B}_j$, respectively, where j denotes the IOC step.

Remark 4: It is important to emphasize, that the inferred matrices $\overline{A}_j$ and $\overline{B}_j$ are for interpretation purposes, i.e., we want to understand what the RL is learning from the data in each iteration and to identify the mechanisms to improve the closed-loop performance of the autonomous platform. These matrices could change depending on the quality of the data and the RL design.

Remark 5: In contrast to traditional system-identification-based approaches, which focus on estimating system dynamics in order to design controllers, observers, or filters, the proposed method is purely inference-oriented. Rather than aiming at control synthesis, it provides an explainable interface that monitors and interprets the autonomous system's behavior directly from observed data.

C. Q-function

We use the Q-function to evaluate the evolution of the state-control pair and determine how good is the learned control policy in comparison with the real one of the dataset. The Q-function in iteration j can be constructed from the outputs of the RL and state-transition inference results as follows,

$$Q_j(x(t), u_i(t)) = \begin{bmatrix} x(t) \\ u_i(t) \end{bmatrix}^\top \begin{bmatrix} L_i + P_i & P_i \bar{B}_i \\ \bar{B}_i^\top P_i & R_j \end{bmatrix} \begin{bmatrix} x(t) \\ u_i(t) \end{bmatrix}, \quad (16)$$

where $L_i = \bar{A}_i^\top P_i + P_i \bar{A}_i + Q_j$.

D. Model-based Inverse Optimal Control

A policy error approach [47] is used to improve the kernel and control gain matrices. We will omit the dependency of t for visualisation purposes. Consider the minimization of the following input error function $E_i = \tilde{u}_i^\top \tilde{u}_i$. The function E is minimized by adjusting the P_i matrix close to the real kernel matrix P . A simple one-step gradient descent algorithm is used as follows,

$$\begin{aligned} \rho &= \varrho_i - \alpha \frac{\partial E}{\partial \varrho_i}, \\ &= \varrho_i - \alpha \left(\tilde{u}_i^\top (x^\top \otimes R_j^{-1} \bar{B}_j^\top) + (x^\top \otimes R_j^{-1} \bar{B}_j^\top)^\top \tilde{u}_i \right), \end{aligned} \quad (17)$$

where $\alpha \in (0, 1)$ is the learning rate, $\rho = \text{vec}(\mathcal{P})$, $\varrho_i = \text{vec}(P_i)$ and $\mathcal{P} = \mathcal{P}^\top \succ 0 \in \mathbb{R}^{n \times n}$ is a new improved kernel matrix that moves in the direction that minimizes the policy error $\tilde{u}_i$, i.e., in the direction of P . The new matrix $\mathcal{P}$ is recovered by the matricization of ρ .

From $\mathcal{P}$ we can obtain a new improved control gain $\mathcal{K} \in \mathbb{R}^{m \times n}$ that is closer to the real gain K as,

$$\mathfrak{K} = \mathcal{K}_{i+1} P_i^{-1} \mathcal{P}. \quad (18)$$

Notice that when $\tilde{u}_i$ approaches to zero as $i \rightarrow \infty$ then $\mathcal{P} \rightarrow P_\infty$ and hence, $\mathfrak{K} \rightarrow \mathcal{K}_\infty$. The ARE (5) under the new improved kernel matrix $\mathcal{P}$ and control gain $\mathfrak{K}$ verifies

$$Q_{j+1} - \mathfrak{K}^\top R_{j+1} \mathfrak{K} = -\bar{A}_j^\top \mathcal{P} - \mathcal{P} \bar{A}_j. \quad (19)$$

Define $\psi_j = [\text{vec}^\top(Q_{j+1}), \text{vec}^\top(R_{j+1})]^\top \in \mathbb{R}^{n^2+m^2}$ such that (19) can be written as follows

$$\begin{bmatrix} I_{n^2} & -(\mathfrak{K} \otimes \mathfrak{K})^\top \end{bmatrix} \psi_j = -\text{vec}(\bar{A}_j^\top \mathcal{P} + \mathcal{P} \bar{A}_j). \quad (20)$$

The main drawback of (20) is that we need to estimate $n^2 + m^2$ parameters but the rank of $\text{rank}([I_{n^2}, -(\mathcal{K}_{i+1} \otimes \mathcal{K}_{i+1})^\top]) \leq n^2$ which implies that infinite solutions can be obtained that can destabilize the algorithm. To solve this problem, the reward function data r of $\mathcal{X}$ is used to force the IOC to seek a unique solution for both Q_{j+1} and R_{j+1} . Here, we write each reward function value of r in terms of Q_{j+1} and R_{j+1} as

$$\begin{bmatrix} r(t) \\ \vdots \\ r(\mathcal{T}) \end{bmatrix} = \begin{bmatrix} (x(t) \otimes x(t))^\top & (u(t) \otimes u(t))^\top \\ \vdots & \vdots \\ (x(\mathcal{T}) \otimes x(\mathcal{T}))^\top & (u(\mathcal{T}) \otimes u(\mathcal{T}))^\top \end{bmatrix} \psi_j. \quad (21)$$

Define $\mathfrak{X} \in \mathbb{R}^{\mathcal{T} \times n^2}$ and $\mathfrak{U} \in \mathbb{R}^{\mathcal{T} \times m^2}$ as

$$\mathfrak{X} = \begin{bmatrix} (x(t) \otimes x(t))^\top \\ \vdots \\ (x(\mathcal{T}) \otimes x(\mathcal{T}))^\top \end{bmatrix}, \quad \mathfrak{U} = \begin{bmatrix} (u(t) \otimes u(t))^\top \\ \vdots \\ (u(\mathcal{T}) \otimes u(\mathcal{T}))^\top \end{bmatrix}.$$

So, the rewards in (21) is expressed in matrix form as

$$[\mathfrak{X} \quad \mathfrak{U}] \psi_j = r^\top, \quad (22)$$

Then, combining (20) with (22) gives the following system,

$$\Omega \psi_j = \Pi, \quad (23)$$

where

$$\begin{aligned} \Omega &= \begin{bmatrix} I_{n^2} & -(\mathfrak{K} \otimes \mathfrak{K})^\top \\ \mathfrak{X} & \mathfrak{U} \end{bmatrix} \in \mathbb{R}^{(n^2+\mathcal{T}) \times (n^2+m^2)}, \\ \Pi &= \begin{bmatrix} -\text{vec}(\bar{A}_j^\top \mathcal{P} + \mathcal{P} \bar{A}_j) \\ r^\top \end{bmatrix} \in \mathbb{R}^{(n^2+\mathcal{T})}. \end{aligned}$$

Notice that the dimensions of Ω and Π can be reduced by considering at least m^2 independent samples of the states, control, and reward trajectories. The system (23) is solved using the least-squares (LS) rule

$$\psi_j = (\Omega^\top \Omega)^{-1} \Omega^\top \Pi. \quad (24)$$

When the reward weights converge in the limit, then the evidence space is constructed by the outputs of the EXIRL algorithm, i.e., the evidence space is given by $\mathcal{E} = \{Q_\infty, R_\infty, P_\infty, \bar{A}_\infty, \bar{B}_\infty, \mathcal{K}_\infty\}$ which will be used to construct counterfactual explanations. Algorithm 1 depicts the pseudo-code of the proposed EXIRL algorithm.

Algorithm 1 Explainable Inverse Reinforcement Learning (EXIRL) algorithm

Require: Initial weight matrices Q_0 and R_0 and observation space $\mathcal{X}$. Stabilizing control policy $u_0 = -\mathcal{K}_0 x$. Set $i = 0$, $j = 0$, and a small thresholds $\epsilon_K, \epsilon_Q, \epsilon_R > 0$.

- 1: **Policy Evaluation.** Compute P_i and $\mathcal{K}_{i+1}$ using BLS policy iteration algorithm [14].
 - 2: **State-Transition Inference.** Compute $\mathcal{A}_i$ and $\mathcal{B}_i$ using (14) and (15), respectively.
 - 3: Set $i \leftarrow i + 1$. Repeat steps 1-2 until $\|\mathcal{K}_{i+1} - \mathcal{K}_i\| \leq \epsilon_K$.
 - 4: Set $\bar{A}_j \leftarrow \mathcal{A}_i$ and $\bar{B}_j \leftarrow \mathcal{B}_i$.
 - 5: **Kernel and Policy improvement.** Compute $\mathcal{P}$ and $\mathfrak{K}$ using (17) and (18), respectively.
 - 6: **Inverse Optimal Control.** Compute the next weight matrices Q_{j+1} and R_{j+1} using (24).
 - 7: Set $j \leftarrow j + 1$. Repeat steps 1-6 until $\|Q_{j+1} - Q_j\| \leq \epsilon_Q$ and $\|R_{j+1} - R_j\| \leq \epsilon_R$.
 - 8: Construct the evidence space from the EXIRL outputs as $\mathcal{E} = \{Q_\infty, R_\infty, P_\infty, \bar{A}_\infty, \bar{B}_\infty, \mathcal{K}_\infty\}$.
-

The following theorem establishes the convergence in the limit of the proposed EXIRL algorithm.

Theorem 1: Consider an observation space $\mathcal{X} = \{X, U, r\}$ constructed from the trajectories of an autonomous platform

and initial reward weights¹ $\mathbf{Q}_0 \succ 0$ and $\mathbf{R}_0 \succ 0$. Assume that the trajectories of $\mathcal{X}$ are rich enough such that,

$$\begin{aligned} \text{rank}(\Sigma) &= \frac{1}{2}n(n+1) + nm, \\ \text{rank}(\Omega) &= n^2 + m^2, \end{aligned} \quad (25)$$

are full rank in almost all their trajectories length. Then $\mathbf{P}_i$, $\mathbf{K}_{i+1}$, $\mathbf{Q}_{j+1}$ and $\mathbf{R}_{j+1}$ converge to their real values $\mathbf{P}$, $\mathbf{K}$, $\mathbf{Q}$ and $\mathbf{R}$, respectively, as the number of iterations $i, j \rightarrow \infty$.

Proof: If (25) is satisfied, then each $\mathbf{P}_i$, $\mathbf{Q}_j$ and $\mathbf{R}_j$ satisfy the ARE (5). For each iteration i of the model-free RL part we have that the following ARE holds for each $\mathbf{P}_i$ and $\mathbf{K}_{i+1}$,

$$-\mathbf{Q}_j = \mathbf{A}^\top \mathbf{P}_i + \mathbf{P}_i \mathbf{A} - \mathbf{K}_{i+1}^\top \mathbf{R}_j \mathbf{K}_{i+1}. \quad (26)$$

In addition, from the model-based IOC we have that the following ARE is satisfied

$$-\mathbf{Q}_{j+1} = \mathbf{A}^\top \mathbf{P} + \mathbf{P} \mathbf{A} - \mathbf{K}^\top \mathbf{R}_{j+1} \mathbf{K}. \quad (27)$$

Let define

$$\mathbf{M}_i = \text{mat} \left(\tilde{\mathbf{u}}_i^\top (\mathbf{x}^\top \otimes \mathbf{R}_j^{-1} \bar{\mathbf{B}}_j^\top) + (\mathbf{x}^\top \otimes \mathbf{R}_j^{-1} \bar{\mathbf{B}}_j^\top)^\top \tilde{\mathbf{u}}_i \right).$$

Then (27) can be written as

$$\begin{aligned} -\mathbf{Q}_{j+1} &= \mathbf{A}^\top (\mathbf{P}_i - \alpha \mathbf{M}_i) + (\mathbf{P}_i - \alpha \mathbf{M}_i) \mathbf{A} \\ &\quad - (\mathbf{P}_i - \alpha \mathbf{M}_i) \mathbf{P}_i^{-1} \mathbf{K}_{i+1}^\top \mathbf{R}_{j+1} \mathbf{K}_{i+1} \mathbf{P}_i^{-1} (\mathbf{P}_i - \alpha \mathbf{M}_i) \\ &= \mathbf{A}^\top \mathbf{P}_i + \mathbf{P}_i \mathbf{A} - \mathbf{K}_{i+1}^\top \mathbf{R}_{j+1} \mathbf{K}_{i+1} - \alpha \mathbf{A}^\top \mathbf{M}_i - \alpha \mathbf{M}_i \mathbf{A} \\ &\quad + \alpha \mathbf{M}_i \mathbf{P}_i^{-1} \mathbf{K}_{i+1}^\top \mathbf{R}_{j+1} \mathbf{K}_{i+1} + \alpha \mathbf{K}_{i+1}^\top \mathbf{R}_{j+1} \mathbf{K}_{i+1} \mathbf{P}_i^{-1} \mathbf{M}_i \\ &\quad - \alpha^2 \mathbf{M}_i \mathbf{P}_i^{-1} \mathbf{K}_{i+1}^\top \mathbf{R}_{j+1} \mathbf{K}_{i+1} \mathbf{P}_i^{-1} \mathbf{M}_i \end{aligned} \quad (28)$$

We can observe that (28) has a similar form compared to (26) with additional terms weighted by powers of the learning rate α . These terms update the weight matrix $\mathbf{Q}_{j+1}$ in the direction that minimizes the input error $\tilde{\mathbf{u}}_i$. Since a one step gradient descent rule is used, then by selecting a proper learning rate α it follows that $0 \leq E_{i+1} < E_i$, such that in the limit $\lim_{i \rightarrow \infty} E_i = 0$ and $\lim_{i \rightarrow \infty} \tilde{\mathbf{u}}_i = \mathbf{0}_{m \times 1}$. In consequence it follows that $\lim_{i \rightarrow \infty} \mathbf{M}_i = \mathbf{0}_n$. This means that in each iteration i the elements of matrix $\mathbf{M}_i$ and the input error $\tilde{\mathbf{u}}_i$ decrease and tend to zero such that $\lim_{i \rightarrow \infty} (\mathbf{P}_i - \alpha \mathbf{M}_i) = \mathbf{P}_\infty$ and $\lim_{i \rightarrow \infty} \mathbf{K}_i = \mathbf{K}$. With these results (28) is rewritten as,

$$-\mathbf{Q}_{j+1} = \mathbf{A}^\top \mathbf{P}_\infty + \mathbf{P}_\infty \mathbf{A} - \mathbf{K}^\top \mathbf{R}_{j+1} \mathbf{K}. \quad (29)$$

We need to analyse how the reward constraint affects the reward update in the IOC step. To this end, for each trajectory $1, \dots, \mathcal{T}$, in the observation space $\mathcal{X}$ we have that in each iteration j the following system of equations is constructed

$$\begin{aligned} \text{vec}(\mathbf{Q}_{j+1}) - (\mathbf{K}^\top \otimes \mathbf{K}^\top) \text{vec}(\mathbf{R}_{j+1}) &= -\text{vec}(\mathbf{A}^\top \mathbf{P} + \mathbf{P} \mathbf{A}) \\ (\mathbf{x}^\top(1) \otimes \mathbf{x}^\top(1)) \text{vec}(\mathbf{Q}_{j+1}) + (\mathbf{u}^\top(1) \otimes \mathbf{u}^\top(1)) \text{vec}(\mathbf{R}_{j+1}) &= r(1) \\ &\vdots \\ (\mathbf{x}^\top(\mathcal{T}) \otimes \mathbf{x}^\top(\mathcal{T})) \text{vec}(\mathbf{Q}_{j+1}) + (\mathbf{u}^\top(\mathcal{T}) \otimes \mathbf{u}^\top(\mathcal{T})) \text{vec}(\mathbf{R}_{j+1}) &= r(\mathcal{T}) \end{aligned} \quad (30)$$

Here, the reward values $\mathbf{r}$ add constraints in the estimation process such that to find a solution for (30), then the BLS rule (24) produces equivalent matrices $\mathbf{Q}_{j+1}$ and $\mathbf{R}_{j+1}$ that yield to the same reward values. These new weight matrices have associated a new control gain $\mathbf{K}_{j+1}$ which is closer

to the real $\mathbf{K}$, i.e., $\mathbf{u}_j(t) \leq \mathbf{u}_{j+1}(t) \leq \dots \leq \mathbf{u}^*(t)$ and the estimation process starts over again. This approach is equivalent to policy iteration algorithms used in dynamic programming [10]. To better describe this process, consider the following ARE associated to the reward values $\mathbf{r}$

$$-\mathbf{Q} = \mathbf{A}^\top \mathbf{P} + \mathbf{P} \mathbf{A} - \mathbf{K}^\top \mathbf{R} \mathbf{K}. \quad (31)$$

The difference between the ARE (31) and the ARE (29) produces the following algebraic error equation,

$$\mathbf{Q}_{j+1} - \mathbf{Q} = \mathbf{A}^\top \Delta \mathbf{P}_j - \Delta \mathbf{P}_j \mathbf{A} + \mathbf{K}^\top (\mathbf{R}_{j+1} - \mathbf{R}) \mathbf{K}, \quad (32)$$

where $\Delta \mathbf{P}_j = \mathbf{P}_\infty - \mathbf{P}$. Notice that $\Delta \mathbf{P}_j$ can be either positive or negative semi-definite depending in how the weight matrices $\mathbf{Q}_0$ and $\mathbf{R}_0$ are initialised. From (32) we can observe that $\mathbf{Q}_{j+1}$ and $\mathbf{R}_{j+1}$ are updated in the directions of the real weight matrices $\mathbf{Q}$ and $\mathbf{R}$ but in opposite direction, that is, when $\mathbf{Q}_{j+1} - \mathbf{Q} \succeq 0$ then $\mathbf{R}_{j+1} - \mathbf{R} \preceq 0$ and conversely, when $\mathbf{Q}_{j+1} - \mathbf{Q} \preceq 0$ then $\mathbf{R}_{j+1} - \mathbf{R} \succeq 0$. So, without loss of generality, we have that

$$\mathbf{Q}_\infty = \lim_{j \rightarrow \infty} \mathbf{Q}_j = \mathbf{Q} + \lim_{j \rightarrow \infty} (\mathbf{K}^\top (\mathbf{R}_j - \mathbf{R}) \mathbf{K} + \mathbf{E}_j), \quad (33)$$

where $\mathbf{E}_j = \mathbf{A}^\top \Delta \mathbf{P}_j + \Delta \mathbf{P}_j \mathbf{A}$. Then, in each iteration j the BLS rule (24) is applied such that equivalent matrices are obtained and by the policy iteration theorem [48] we have that $\lim_{j \rightarrow \infty} \mathbf{Q}_j = \mathbf{Q}$ and $\lim_{j \rightarrow \infty} \mathbf{R}_j = \mathbf{R}$. In consequence, $\mathbf{P}_\infty$ converges into the exact kernel matrix $\mathbf{P}$ which implies that $\Delta \mathbf{P}_\infty = \mathbf{0}_n$ and $\mathbf{E}_\infty = \mathbf{0}_n$. This completes the proof. ■

Remark 6: The initial reward weights $\mathbf{Q}_0$ and $\mathbf{R}_0$ can be set to small weight matrices, e.g., $\mathbf{Q}_0 = \gamma_Q \mathbf{I}_n$ and $\mathbf{R}_0 = \gamma_R \mathbf{I}_m$ for some small $\gamma_Q, \gamma_R > 0$ to ensure stability and fast convergence results. On the other hand, for large weight matrices the EXIRL algorithm can require more iterations to converge or, in the worst case, it can produce numerical and stability issues.

Remark 7: The involved computation for Algorithm 1 is dominated by the policy evaluation and inverse optimal control parts. If one does the calculations of the BLS policy iteration algorithm [14], the computation has growth $\iota^2(\iota + \kappa)$, and for the inverse optimal control (24) the computation grows $(n^2 + m^2)^2(2n^2 + m^2 + \mathcal{T})$. Thus, the complexity is $\mathcal{O}[\iota^2(\iota + \kappa) + (n^2 + m^2)^2(2n^2 + m^2 + \mathcal{T})]$ with the term $(n^2 + m^2)^2(2n^2 + m^2 + \mathcal{T})$ dominating.

E. Counterfactual Explanations

Counterfactuals explanations [22] are explainable tools that aim to interpret autonomous decisions. The idea is based on establishing idealistic scenarios of what would had been happened if a different action or condition had been applied.

To construct counterfactuals it is required to obtain the necessary evidence to explain, in a clear common sense, why a specific behaviour was observed and how this behaviour can be modified into a desired one. These evidence is obtained from the EXIRL algorithm and given by the evidence space $\mathcal{E}$. Fig. 1 shows the learning elements that are inferred from the data using the EXIRL to construct counterfactuals.

Notice that the inferred elements in the evidence space $\mathcal{E}$ are arranged hierarchically in terms of its simplicity of being

¹The initial reward weights must ensure that the ARE (5) has a solution.

inferred from the data and its dependence of previous learning elements. In this case, the control policy can be easily inferred by using both the states and control input values. The reward function weights are harder to be inferred since they require to learn additional learning elements and constraints. Notice that the state-transition matrices depend on both the policy and kernel matrix P_i since we are inferring what state-transition model is learned by the RL agent.

After the evidence space $\mathcal{E}$ is inferred from the data, then we can build counterfactuals of the form:

- If the real part of the eigenvalues of $\lambda(\bar{A}_j - \bar{B}_j \mathcal{K}_i)$ had had negative real part with zero imaginary part, then the observed behaviour should exhibit an exponential behaviour without oscillations.
- If the reward weight Q_j had weighted more the position states rather than the velocity states, then the convergence of the closed-loop trajectories should be accelerated.
- If the pair of inferred state-transition matrices $(\bar{A}_j, \bar{B}_j)$ had been controllable, then the control policy $\mathcal{K}_i$ should stabilize the autonomous system.
- If the observation space $\mathcal{X}$ had been enough rich throughout the complete length of the trajectories, then the inference of the learning elements should be accurate.

V. SIMULATION STUDIES

This section provides the results and comparisons of the proposed EXIRL algorithm. The simulations are conducted in Matlab 2024a. We use the number of steps (sampling steps) ι of the simulation instead of the RL and IOC iterations i, j , since these ones differ in length as we converge close to the real gain, kernel and weight matrices. We use a sampling time of $T = 0.002$ seconds which matches with real on-board sensors of autonomous systems.

A. DJI Phantom 4

Consider a DJI Phantom 4 safe hovering model used in [49]. The model of the drone is given by $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ 0.72 \end{bmatrix}$. Consider that we have a dataset $\mathcal{X} = \{X, U, r\}$ collected from an optimal LQR controller with reward weights $\text{vec}(Q) = [4, 2, 2, 4]^\top$ and $R = 3$. The respective kernel P and gain K of the ARE (5) are

$$P = \begin{bmatrix} 5.3817 & 4.8111 \\ 4.8111 & 8.8786 \end{bmatrix}, \quad K = [1.1547 \quad 2.1309].$$

The initial reward weights are given by $Q_0 = 0.1I_2$ and $R_0 = 0.1$ with an initial stabilizing policy of $\mathcal{K}_0 = [1.4142, 2.4348]$. The learning rate α of the policy error algorithm (17) is set to 0.5 to ensure fast convergence of the EXIRL algorithm. This hyperparameter is manually tuned based on the initial reward weights. A threshold of $\epsilon_K = 0.001$ is used in the RL loop, while in the IOC loop no thresholds (ϵ_Q, ϵ_R) are imposed as well as early stopping conditions, permitting the algorithm to execute all proposed time steps, $\iota = 5,000$.

Fig. 4 shows the convergence results using the proposed EXIRL. The results show convergence of the learning matrices

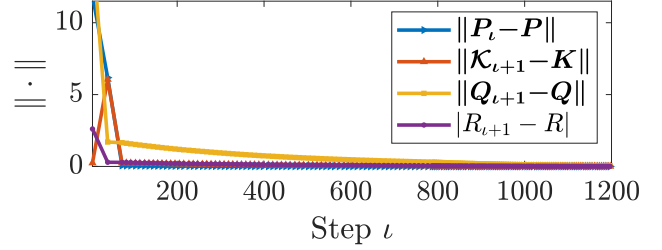


Fig. 4. EXIRL convergence results in a LQR control problem

in approximately 1,000 steps. Here, the matrices converge to the following values,

$$P_{1000} = \begin{bmatrix} 5.3817 & 4.8111 \\ 4.8111 & 8.8786 \end{bmatrix}, \quad \mathcal{K}_{1000} = [1.1547 \quad 2.1309],$$

$$Q_{1000} = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}, \quad R_{1000} = 3.$$

Fig. 5 shows the results of the state transition matrices inference. The top plot of Fig. 5 shows how the RL learns and infers the state-transition matrices from the data. Recall that the RL algorithm needs to estimate the kernel matrix P_t and control gain $\mathcal{K}_{t+1}$ by solving a BLS update rule. Additionally, these matrices need to be consistent with the ARE (9) to obtain the optimal solution of the LQR control problem under the reward weights Q_t and R_t . We can observe from Fig. 4 that both the matrix P_t and $\mathcal{K}_{t+1}$ are not the optimal since the state-transition matrices were not inferred correctly as shown in the first 40 steps. However, as more data is collected, the estimation of the state-transition matrices becomes more accurate. This shows that the RL algorithm aims to maintain the consistency of the ARE (9) by adapting both the inference of the state-transition matrices and the estimation of the optimal solution of the LQR problem. Each step the IOC phase of the EXIRL algorithm is executed, then new reward weights are computed and the RL training starts again. Notice that the results of the state-transition matrices inference in the second RL training (after 40 steps) shows better results since the new kernel matrix P_{t+1} is close to the previous kernel matrix P_t and the updates are smaller.

The bottom plot of Fig. 5 shows the state-transition matrices obtained from the RL algorithm that are used to feed the IOC algorithm. The results show that in each IOC iteration j , the state transition matrices remain unchanged which demonstrates that the RL is working properly.

Notice that the reward values r give a constraint to the reward weights learning, such that the multiple solution problem obtained by traditional IRL algorithms is solved.

1) *Counterfactual Explanations:* With the evidence collected by the EXIRL algorithm we are able to design counterfactuals. For example, for the LQR control we have that $\lambda(\bar{A}_\infty - \bar{B}_\infty \mathcal{K}_\infty) = -0.7671 \pm 0.4928i$. This allows to identify that the drone will exhibit a damped sinusoid behaviour such that we can establish mechanisms to achieve a desired-closed loop behaviour. Here, we can construct policy counterfactuals of the form

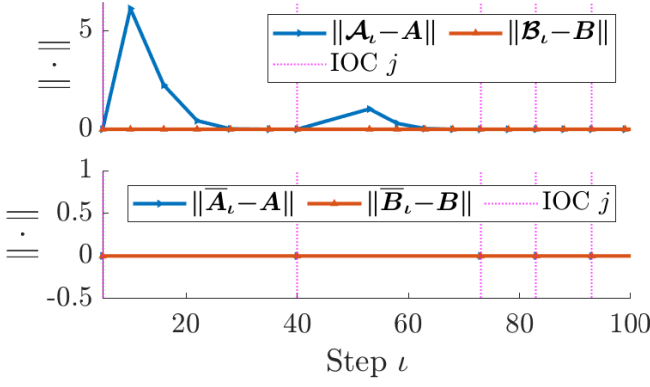


Fig. 5. Inferred State-transition matrices convergence results. Top figure: $\mathcal{A}_t$ and $\mathcal{B}_t$. Bottom figure: $\bar{\mathcal{A}}_t$ and $\bar{\mathcal{B}}_t$.

- “If the characteristic polynomial $p(\lambda) = \lambda^2 + (a+b)\lambda + ab$ had had real roots with $a, b > 0$, then the closed-loop trajectories should exhibit an exponential stable performance without overshoots.”.
- “If the magnitude of the real part of $\lambda(\bar{\mathcal{A}}_\infty - \bar{\mathcal{B}}_\infty \mathcal{K}_\infty)$ had been larger, then the trajectories of the system should converge faster.”

As an example consider two cases of characteristic polynomials: C1) $p_1(\lambda) = \lambda^2 + 7\lambda + 10$, and C2) $p_2(\lambda) = \lambda^2 + 2\lambda + 10$. The eigenvalues associated to each case are $\lambda_{1,2} = \{-5, -2\}$ and $\lambda_{1,2} = \{-1 \pm 3i\}$, respectively. C1 fulfils the previous counterfactuals, whilst C2 violates them. Fig. 6 shows how the behaviour of the autonomous platform is modified by following the proposed counterfactuals.

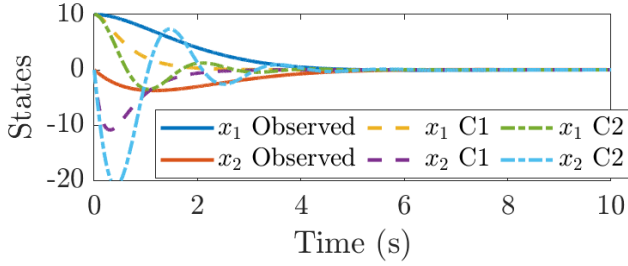


Fig. 6. Behaviour modification by designing policy counterfactuals

As expected, the policy counterfactuals are able to explain that the real trajectories exhibit a damped sinusoid behaviour, which can be modified by changing the location of the poles. Here, high imaginary part produces large oscillations that can be attenuated by increasing the magnitude of the real part. Since the proposed EXIRL infers both the state-transition dynamics and the control gain, then we have the complete evidence to formulate policy counterfactuals.

We can also design reward counterfactuals based on the reward weights values as follows,

- “If the weight Q had weighted more the velocity states rather than the position states, then the control input should be smoother with low amplitude.”.
- “If the weight Q had large values for a fixed weight R , then the control amplitude should be large.”

Consider again two cases for the reward weights: C1) $Q = \text{diag}\{10, 1\}$ and $R = 1$, and C2) $Q = \text{diag}\{1, 10\}$ and $R = 1$. The optimal control policies for each case are $K = [3.1623, 3.1279]$ and $K = [1, 3.5746]$. C1 satisfies the second counterfactual, whilst C2 satisfies the first counterfactual. Fig. 7 shows how by modifying the reward function modifies the observed behaviour.

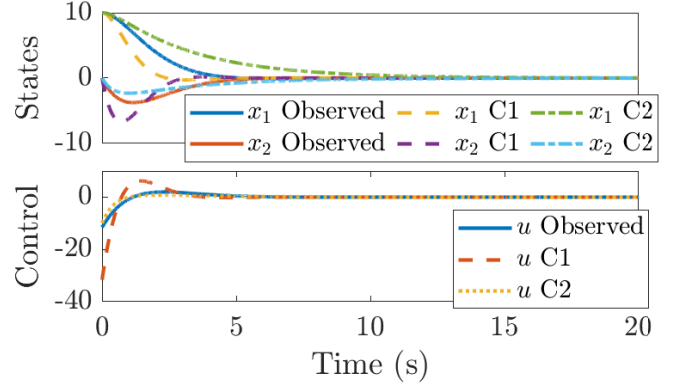


Fig. 7. Behaviour modification by designing reward counterfactuals. Top plot: Closed-loop behaviour comparisons. Bottom plot: control policy u comparisons.

Fig. 7 shows that C1 weights more the positions rather than the velocities, in consequence this accelerates trajectory convergence with overshoots and large control input values. Conversely, C2 weights more the velocities rather than the positions. This produces slow convergence, whilst decreasing the amplitude of the control policy. Therefore, the combination of both policy and reward counterfactuals creates the baseline to interpret the behaviour of autonomous platforms.

2) *Sensitivity Analysis:* We perform a sensitivity analysis of the proposed EXIRL algorithm. Here, the learning rate α of the policy error algorithm is the only hyperparameter to tune in this approach. However, the initial conditions of the reward weight matrices Q_0 and R_0 are crucial elements that modifies the convergence time of the proposed approach. To verify this, we test two different initial conditions: (1) $Q_0 = 0.1I_2$ and $R_0 = 0.1$, and (2) $Q_0 = \begin{bmatrix} 20 & 5 \\ 5 & 20 \end{bmatrix}$ and $R_0 = 5$. These matrices are selected to verify the results of Theorem 1 and Remark 6.

For case (1), it follows that P_0 is a lower bound of the real kernel matrix P such that for all t we have that $P_t - P \preceq 0$ as it is observed in the top plot of Fig. 8. The bottom plot of Fig. 8 shows how the difference between the inferred reward weights and the real ones changes in each step t . We can observe that Q_t updates in the opposite direction to R_t as stated in Theorem 1. In this case, the EXIRL requires approximately 3.60 seconds to converge to the real reward values which is equivalent to 3,131 steps.

Case (2) verifies the claim of Remark 6 as observed in Fig. 9. The results demonstrate that in this case, the initial reward weights produce a kernel matrix P_t that is an upper bound to the real kernel matrix P as shown in the top plot of Fig. 9. Furthermore, the bottom plot of Fig. 9 demonstrates that Q_t

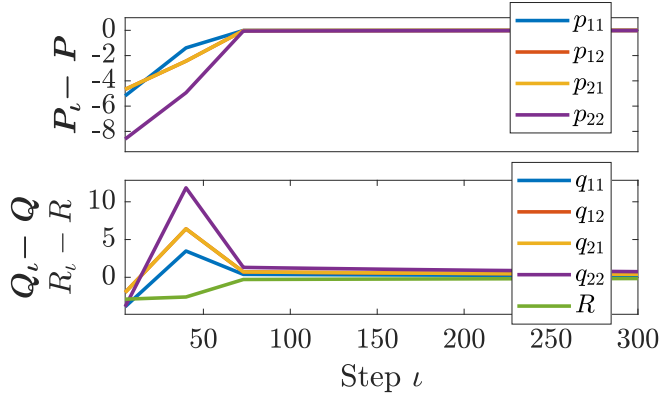


Fig. 8. Verification of the theoretical results in Theorem 1 for $\mathbf{Q}_0 = 0.1\mathbf{I}_2$ and $R_0 = 0.1$. Top plot: $\mathbf{P}_t - \mathbf{P}$. Bottom plot: $\mathbf{Q}_t - \mathbf{Q}$ & $R_t - R$.

updates in the opposite direction of R_t as stated in Theorem 1. In this case, the EXIRL algorithm requires approximately 3.86 seconds to converge to the real reward values which is equivalent to 3,442 steps.

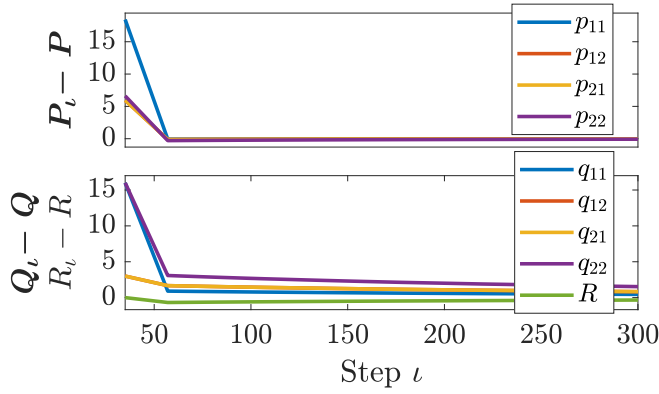


Fig. 9. Verification of the theoretical results in Theorem 1: $\mathbf{Q}_0 = \begin{bmatrix} 20 & 5 \\ 5 & 20 \end{bmatrix}$ and $R_0 = 5$. Top plot: $\mathbf{P}_t - \mathbf{P}$. Bottom plot: $\mathbf{Q}_t - \mathbf{Q}$ & $R_t - R$.

We further test different learning rates to show how it affects the convergence of the EXIRL algorithm under different reward weights. We test three different learning rates $\alpha = \{0.5, 0.25, 0.1\}$ under the aforementioned initial reward weights and compare the kernel error $\|\mathbf{P}_t - \mathbf{P}\|$. Fig. 10 shows that for small initial reward weights, the convergence of the kernel matrix is accelerated using a learning rate of $\alpha = 0.5$, and it shows a slow convergence performance for smaller learning rates. On the other hand, the convergence is slower for relative large initial reward weights even for relative large learning rates. Notice for this particular case, for all learning rates the EXIRL algorithm decreases from a large kernel error into a small value in less number of steps and after that, the updates are shorter due to the policy error. A large learning rate is required to accelerate convergence of the kernel matrix by taking larger updates in the direction of the gradient of the policy error.

3) *Comparisons:* We compare the EXIRL results against the IRL algorithm proposed in [33] which is the baseline algorithm of new IRL developments in robust control [50],

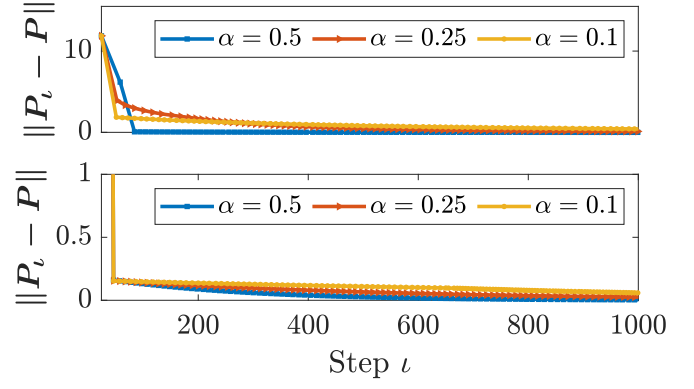


Fig. 10. Sensitivity analysis of the learning rate under different initial reward weights. Top plot: $\mathbf{Q}_0 = 0.1\mathbf{I}_2$ and $R_0 = 0.1$. Bottom plot: $\mathbf{Q}_0 = \begin{bmatrix} 20 & 5 \\ 5 & 20 \end{bmatrix}$ and $R_0 = 5$.

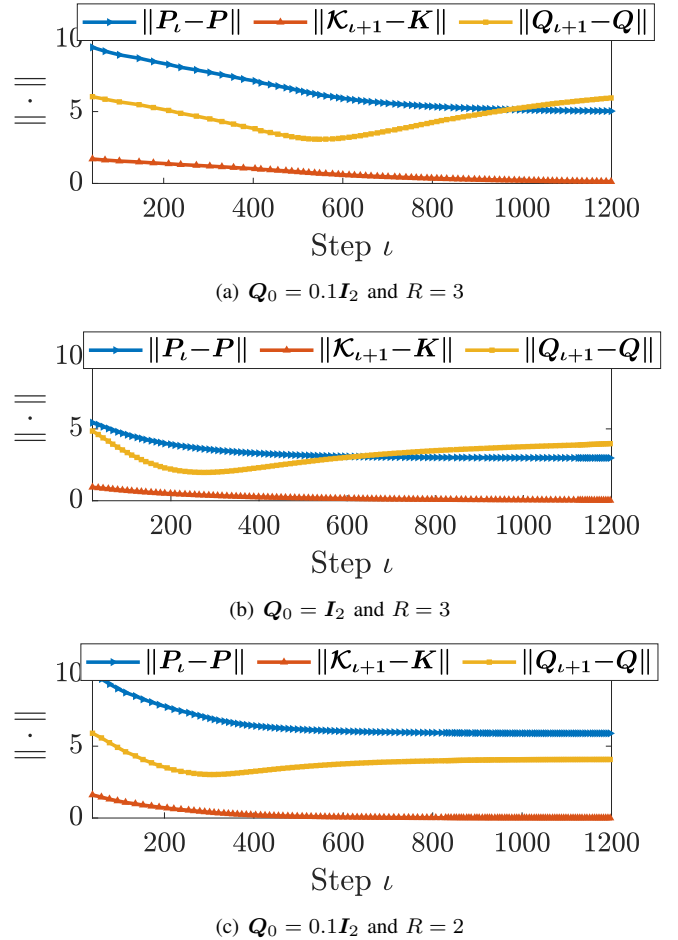


Fig. 11. IRL algorithm [33] convergence results in a LQR control problem

[51], distributed agents [7], and recently in Markov-jump systems [52]). In this particular algorithm, the reward values $\mathbf{r}$ are not available in the dataset and hence, we need to fix the value of R . We consider three cases: 1) $\mathbf{Q}_0 = 0.1\mathbf{I}_2$ and $R = 3$, 2) $\mathbf{Q}_0 = \mathbf{I}_2$ and $R_0 = 3$, and 3) $\mathbf{Q}_0 = 0.1\mathbf{I}_2$ and $R_0 = 2$. These cases are chosen to show that even using the exact weight matrix R , the IRL algorithm obtains non-unique

solutions for Q_i . The results are shown in Fig. 11.

Notice that in all cases the weight matrix Q_i does not converge to its exact value, however almost the same control gains K_i are obtained. The respective matrices obtained after 1200 steps are

$$\begin{aligned} Q_{1200}^1 &= \begin{bmatrix} 3.1988 & 6.1338 \\ 6.1338 & 4.7523 \end{bmatrix}, \quad K_{1200}^1 = \begin{bmatrix} 1.0326 & 2.1101 \end{bmatrix} \\ Q_{1200}^2 &= \begin{bmatrix} 3.8307 & 4.8088 \\ 4.8088 & 4.1898 \end{bmatrix}, \quad K_{1200}^2 = \begin{bmatrix} 1.136 & 2.1297 \end{bmatrix} \\ Q_{1200}^3 &= \begin{bmatrix} 2.6655 & 4.5535 \\ 4.5535 & 2.6679 \end{bmatrix}, \quad K_{1200}^3 = \begin{bmatrix} 1.1544 & 2.1309 \end{bmatrix}. \end{aligned}$$

In this case we obtain negative definite matrices Q_i which produce almost the same control policy $K_i \approx K$. This could yield to biased interpretations of the observed behaviour with unstable control objective for other mission profiles. To assess this, consider, we plot the reward and Q-function obtained from the previous cases. Fig. 12 summarizes the reward function and Q-function results. The reward function for all cases show negative values, whilst the optimal one remains positive throughout the complete trajectory. This clearly means that for other missions, the reward function could be negative yielding to biased interpretations. On the other hand, the Q-function provides a good indicator if a followed control policy is good or not. Similarly to the reward plot, negative values of the Q-function means that the learned values are not optimal even the control gain is the optimal one.

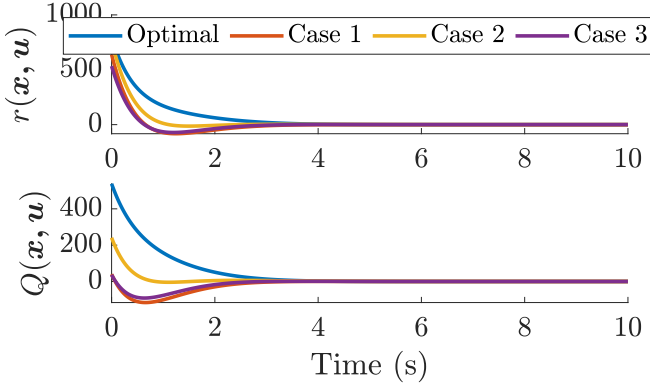


Fig. 12. Reward function and Q-function values. Top plot: Reward function. Bottom plot: Q-function

B. Quadrotor model

Consider a quadrotor linear model defined by the following linear matrices

$$A = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}, \quad B = \begin{bmatrix} \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 1} \\ g & 0 & 0 \\ 0 & -g & 0 \\ 0 & 0 & 1/m \end{bmatrix},$$

where $g = 9.81 \text{ m/s}^2$ and $m = 0.467 \text{ kg}$. Here, the state vector is given by $\mathbf{x} = [x, y, z, u, v, w]^\top$ which contains the linear positions and velocities of the quadrotor, and $\mathbf{u} = [\theta, \phi, \mu]^\top$ defines the control input composed by the Euler angles pitch and roll, and the total thrust force.

Algorithm 2 Modified Explainable Inverse Reinforcement Learning (mEXIRL) algorithm

Require: Initial weight matrices Q_0 and R_0 and observation space $\mathcal{X}$. Stabilizing control policy $\mathbf{u}_0 = -K_0 \mathbf{x}$. Set $i = 0$, $j = 0$, and a small thresholds $\epsilon_K, \epsilon_Q, \epsilon_R > 0$.

- 1: **Policy Evaluation.** Compute P_i and K_{i+1} using BLS policy iteration algorithm [14].
- 2: **State-Transition Inference.** Compute $\mathcal{A}_j$ and $\mathcal{B}_j$ using (14) and (15), respectively.
- 3: Set $i \leftarrow i + 1$. Repeat steps 1-2 until $\|K_{i+1} - K_i\| \leq \epsilon_K$.
- 4: Set $\bar{\mathcal{A}}_j \leftarrow \mathcal{A}_i$ and $\bar{\mathcal{B}}_j \leftarrow \mathcal{B}_i$.
- 5: **Kernel and Policy improvement.** Compute $\mathcal{P}$ and $\mathfrak{K}$ using (17) and (18), respectively.
- 6: **Inverse Optimal Control.** Compute auxiliary matrices $\mathcal{Q}$, $\mathfrak{Q}$, and $\mathfrak{R}$ using (34) and (35).
- 7: Compute Q_{j+1} and R_{j+1} using (36).
- 8: Set $j \leftarrow j + 1$. Repeat steps 1-7 until $\|Q_{j+1} - Q_j\| \leq \epsilon_Q$ and $\|R_{j+1} - R_j\| \leq \epsilon_R$.
- 9: Construct the evidence space from the EXIRL outputs as $\mathcal{E} = \{Q_\infty, R_\infty, P_\infty, \bar{\mathcal{A}}_\infty, \bar{\mathcal{B}}_\infty, K_\infty\}$.

For this particular system, Algorithm 1 faces challenges due to the uncoupled dynamics observed in the quadrotor model. This causes instability in the inference of the reward weights. To alleviate this issue we will consider only diagonal weight matrices for both Q and R , which is consistent to the uncoupled dynamic claim. We define one auxiliary diagonal matrix $\mathcal{Q} \in \mathbb{R}^{n \times n}$ which is computed using the standard IOC update rule [53]

$$\mathcal{Q} = -\text{diag} \left(\bar{\mathcal{A}}_j^\top \mathcal{P} - \mathcal{P} \bar{\mathcal{A}}_j + \mathfrak{K}^\top R_j \mathfrak{K} \right). \quad (34)$$

This matrix is used to constraint the update space of Q_{j+1} in terms of R_j . Since we are forcing $\mathcal{Q}$ to have a diagonal structure, then we are losing information that prevent to recover the real weight matrix. To compensate this loss of information, we propose a modified EXIRL (mEXIRL) algorithm that uses the reward constraint $\mathbf{r}$ to compute residual weight matrices $\mathfrak{Q} \in \mathbb{R}^{n \times n}$ and $\mathfrak{R} \in \mathbb{R}^{m \times m}$ as

$$\begin{aligned} \begin{bmatrix} \text{vec}(\mathfrak{Q}) \\ \text{vec}(\mathfrak{R}) \end{bmatrix} &= [\mathcal{X}, \mathcal{U}]^+ (\mathbf{r}^\top - \hat{\mathbf{r}}^\top) \\ &= [\mathcal{X}, \mathcal{U}]^+ (\mathbf{r}^\top - \mathcal{X} \text{vec}(\mathcal{Q}) - \mathcal{U} \text{vec}(R_j)) \end{aligned} \quad (35)$$

where $\hat{\mathbf{r}}^\top = \mathcal{X} \text{vec}(\mathcal{Q}) - \mathcal{U} \text{vec}(R_j) \in \mathbb{R}^T$. Therefore, the new updated weight matrices are computed as

$$\begin{aligned} Q_{j+1} &= \mathcal{Q} + \text{diag}(\mathfrak{Q}) \\ R_{j+1} &= R_j + \text{diag}(\mathfrak{R}). \end{aligned} \quad (36)$$

Notice that when the difference between the reward values $\mathbf{r}$ and the estimated reward values $\hat{\mathbf{r}}$ is zero, then the residual matrices $\mathfrak{Q}$ and $\mathfrak{R}$ are zero, in consequence $R_{j+1} = R_j$ and $Q_{j+1} = \mathcal{Q} = Q_j$ as stated in (27). This modification allows to avoid instability issues derived by the estimation of matrix R_{j+1} which affect the admissibility of the control $\mathbf{u}_j$. Algorithm 2 summarizes the proposed mEXIRL algorithm.

To show the effectiveness of Algorithm 2 consider a dataset $\mathcal{X} = \{\mathbf{X}, \mathbf{U}, \mathbf{r}\}$ collected from an optimal LQR controller

with reward weights $\mathbf{Q} = \text{diag}\{1, 1, 1, 5, 5, 5\}$ and $\mathbf{R} = 5\mathbf{I}_3$. The optimal kernel matrix and control gain are

$$\mathbf{P} \approx \begin{bmatrix} 2.336 & 0 & 0 & 0.228 & 0 & 0 \\ 0 & 2.336 & 0 & 0 & 0.228 & 0 \\ 0 & 0 & 2.662 & 0 & 0 & 1.044 \\ 0.228 & 0 & 0 & 0.532 & 0 & 0 \\ 0 & 0.228 & 0 & 0 & 0.532 & 0 \\ 0 & 0 & 1.044 & 0 & 0 & 2.78 \end{bmatrix}$$

$$\mathbf{K} \approx \begin{bmatrix} 0.447 & 0 & 0 & 1.045 & 0 & 0 \\ 0 & -0.447 & 0 & 0 & -1.045 & 0 \\ 0 & 0 & 0.447 & 0 & 0 & 1.191 \end{bmatrix}.$$

The initial reward weights are set to $\mathbf{Q}_0 = \mathbf{I}_6$ and $\mathbf{R}_0 = 8\mathbf{I}_3$ with an initial stabilizing policy of

$$\mathbf{K}_0 \approx \begin{bmatrix} 0.354 & 0 & 0 & 0.444 & 0 & 0 \\ 0 & -0.354 & 0 & 0 & -0.444 & 0 \\ 0 & 0 & 0.354 & 0 & 0 & 0.675 \end{bmatrix}.$$

Fig. 13 shows the convergence results of the proposed mEXIRL. We can observe that the approach is capable to infer accurately the inner elements of the LQR controller. Notice that the use of the auxiliary variables stabilize the EXIRL algorithm such that the algorithm converges to the real-values in approximately 3,000 steps. Similar to the DJI drone case, the RL model finds in each step consistent solutions for (9) by adapting the inference of the state-transition matrices with respect to the computed kernel matrix $\mathbf{P}_t$ and control gain $\mathbf{K}_{t+1}$. This process is performed by the RL algorithm until it finds the solution of the LQR control problem under the current reward weights $\mathbf{Q}_t$ and $\mathbf{R}_t$. Then the IOC step updates the reward weights and the RL step starts again. Notice that as the reward weights get closer to their real values, then the state-transition inference error is smaller.

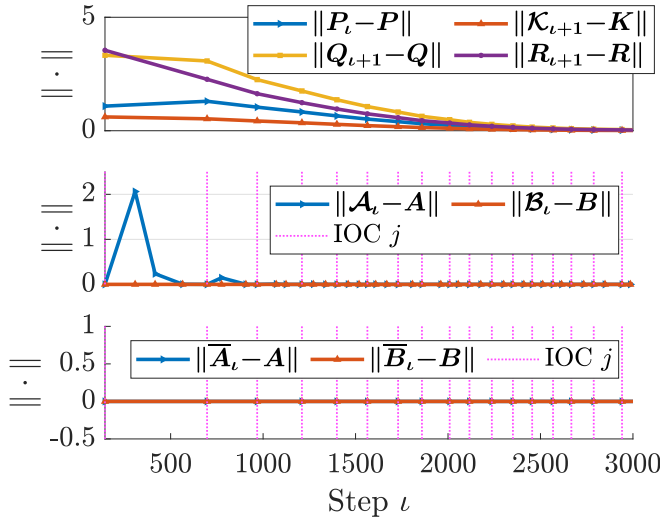


Fig. 13. mEXIRL results for the quadrotor model. Top plot: Kernel, gain, and weight matrices convergence. Middle plot: State-transition matrices in the RL phase. Bottom plot: State-transition matrices in the IOC phase.

Notice that if the reward values $\mathbf{r}$ are not available, then the IOC part of Algorithm 2 is reduced to only compute (34). This yields to equivalent weight matrices $\mathbf{Q}_{t+1}$ and $\mathbf{R}_{t+1}$ that

produces the desired control gain $\mathbf{K}$, however the inferred kernel matrix $\mathbf{P}_t$ and reward weight matrices $\mathbf{Q}_{t+1}, \mathbf{R}_{t+1}$ do not converge to the real $\mathbf{P}, \mathbf{Q}$, and $\mathbf{R}$ matrices.

1) *Counterfactual Explanations*: Similar counterfactual explanations can be designed for this particular system. In this case, the eigenvalues of the closed-loop system are $\lambda \approx \{-0.448, -9.8, -9.8, -0.45, -2.09, -0.46\}$. Notice that the closed-loop system is exponentially stable with one eigenvalue with algebraic multiplicity two. Since the proposed mEXIRL uncovers each learning element from the data then we can use the Ackerman formula to obtain a desired control gain $\mathbf{K}_d$ and design counterfactuals of the form

- “If the control gain $\mathbf{K}$ had been set as $\mathbf{K}_d$, then the eigenvalues of $\lambda = \{\lambda_1, \dots, \lambda_6\}$ should be different with algebraic multiplicity one”.
- “If the values of the weight matrix $\mathbf{Q}$ had weighted differently the velocity terms, then the eigenvalues of $\lambda = \{\lambda_1, \dots, \lambda_6\}$ should be different with algebraic multiplicity one”.

These counterfactual explanations can be easily verified as follows. Consider that we want to assign the poles of the system in the following different locations $\lambda = \{-0.447, -9.799, -3.012, -0.318, -10.738, -0.408\}$ which yields the following desired control gain $\mathbf{K}_d$

$$\mathbf{K}_d \approx \begin{bmatrix} 0.447 & 0 & 0 & 1.045 & 0 & 0 \\ 0 & -0.447 & 0 & 0 & -1.136 & 0 \\ 0 & 0 & 0.447 & 0 & 0 & 1.555 \end{bmatrix}$$

These poles are obtained by using the following reward weights $\mathbf{Q}_d = \text{diag}\{1, 1, 1, 5, 6, 10\}$ and $\mathbf{R}_d = 5\mathbf{I}_3$.

Remark 8: The Ackerman formula produces different control gains that give the same poles location depending in how the poles are ordered. In this paper, we use one of the potential solutions of the Ackerman formula.

We collect a dataset $\mathcal{X} = \{\mathbf{X}, \mathbf{U}, \mathbf{r}\}$ of trajectories of the quadrotor under the desired control gain to infer the reward weights $\mathbf{Q}$ and $\mathbf{R}$ using the mEXIRL. Fig. 14 shows the inference results of the reward weights, kernel matrix, and control gain.

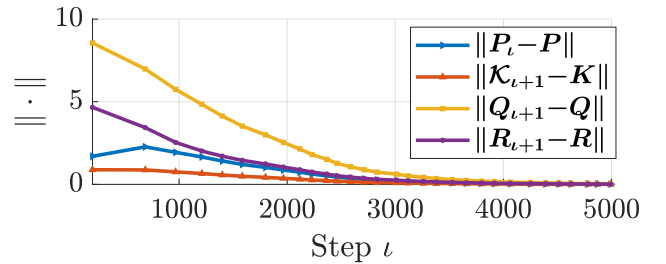


Fig. 14. mEXIRL results to verify counterfactuals

Here, the mEXIRL converges to the same desired kernel, gain and reward weight values and achieve the desired closed-loop performance observed in Fig. 15.

Notice that the overall framework is capable to obtain the evidence to interpret the observed behaviour of an autonomous

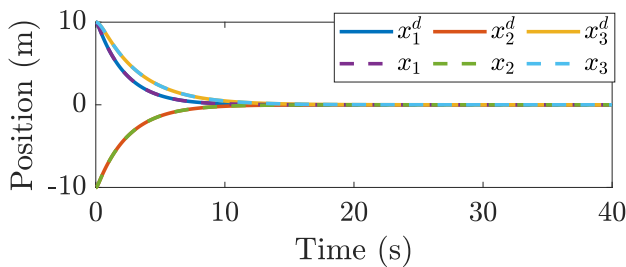


Fig. 15. Closed-loop performance under the desired poles

system and identify the mechanisms to inject a desired performance by modifying either the control gain or reward weight matrices.

VI. CONCLUSIONS

This paper proposed an explainable inverse reinforcement learning (EXIRL) framework to interpret the behaviour of autonomous systems from observational data. The method infers the control policy, reward function, state-transition model, kernel representation, and associated Q-function, enabling the construction of counterfactual explanations that reveal the mechanisms driving observed behaviours. The proposed approach combines model-free reinforcement learning with model-based inverse optimal control, allowing the learning manifold to be constrained and the underlying decision-making structure to be accurately recovered from data. Convergence properties of the EXIRL algorithm were analysed through Lyapunov recursions and validated through simulation experiments. Results demonstrate that the integration of model-free and model-based learning improves the inference of internal learning elements and enables interpretable explanations of autonomous behaviour.

Future work will extend the framework to nonlinear systems and non-quadratic reward functions, as well as to multi-agent settings with dynamic interaction topologies. These extensions are particularly relevant to autonomous decision-making mechanisms in emerging communication infrastructures, where explainable reinforcement learning could support operator monitoring of data-driven control processes such as traffic management and resource allocation in future multi-tenant radio access networks.

REFERENCES

- [1] B. Fraser, A. Perrusquía, D. Panagiotakopoulos, and W. Guo, "Hybrid deep neural networks for drone high level intent classification using non-cooperative radar data," in *Proc. 3rd Int. Conf. Elect., Comput., Commun. Mechatron. Eng. (ICECCME)*. IEEE, 2023, pp. 1–6.
- [2] Z. Chen, D. Guo, and Y. Lin, "A deep Gaussian process-based flight trajectory prediction approach and its application on conflict detection," *Algorithms*, vol. 13, no. 11, p. 293, 2020.
- [3] K. KN, D. Ignatyev, A. Tsourdos, and A. Perrusquia, "Advancing fault diagnosis in aircraft landing gear: An innovative two-tier machine learning approach with intelligent sensor data management," in *Proc. AIAA SCITECH 2024 Forum*, 2024, p. 0759.
- [4] H.-S. Shin, D. Turchi, S. He, and A. Tsourdos, "Behavior monitoring using learning techniques and regular-expressions-based pattern matching," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, 2018.
- [5] G. Singh, A. Perrusquía, and W. Guo, "A two-stages unsupervised/supervised statistical learning approach for drone behaviour prediction," in *Proc. 9th Int. Conf. Control, Decision Inf. Technol. (CoDIT)*, 2023.
- [6] B. Fraser, A. Perrusquía, D. Panagiotakopoulos, and W. Guo, "A deep mixture of experts network for drone trajectory intent classification and prediction using non-cooperative radar data," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*. IEEE, 2023, pp. 1–6.
- [7] A. Perrusquía, W. Guo, B. Fraser, and Z. Wei, "Uncovering drone intentions using control physics informed machine learning," *Commun. Eng.*, vol. 3, no. 1, p. 36, 2024.
- [8] K. G. Vamvoudakis, "Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach," *Syst. Control Lett.*, pp. 14–20, 2017.
- [9] K. Vamvoudakis and F. L. Lewis, "On-line actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, pp. 878–888, 2010.
- [10] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Syst. Mag.*, vol. 32, no. 6, pp. 76–105, 2012.
- [11] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2042–2062, 2017.
- [12] J. Kalantari, H. Nelson, and N. Chia, "The unreasonable effectiveness of inverse reinforcement learning in advancing cancer research," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 437–445.
- [13] S. Baee, E. Pakdamanian, I. Kim, L. Feng, V. Ordonez, and L. Barnes, "MEDIRL: Predicting the visual attention of drivers via maximum entropy deep inverse reinforcement learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 13 178–13 188.
- [14] B. Lian, Y. Kartal, F. L. Lewis, D. G. Mikulski, G. R. Hudak, Y. Wan, and A. Davoudi, "Anomaly detection and correction of optimizing autonomous systems with inverse reinforcement learning," *IEEE Trans. Cybern.*, vol. 53, no. 7, pp. 4555–4566, 2022.
- [15] A. Perrusquía and W. Guo, "Uncovering reward goals in distributed drone swarms using physics-informed multiagent inverse reinforcement learning," *IEEE Trans. Cybern.*, vol. 55, no. 1, 2025.
- [16] R. M. Annasamy and K. Sycara, "Towards better interpretability in deep Q-networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019.
- [17] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. Jimenez, "Towards interpretable reinforcement learning using attention augmented agents," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
- [18] M. T. Ribeiro, S. Singh, and C. Guestrin, "why should i trust you?" explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1135–1144.
- [19] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [20] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, 2016.
- [21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 618–626.
- [22] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harv. J. Tech.*, vol. 31, p. 841, 2017.
- [23] H. Andreas, M. Armgardt, and M. Gunther, "Counterfactuals for causal responsibility in legal contexts," *Artificial intelligence and law*, vol. 31, no. 1, pp. 115–132, 2023.
- [24] J. Ramírez, W. Yu, and A. Perrusquía, "Model-free reinforcement learning from expert demonstrations: a survey," *Artificial Intelligence Review*, vol. 55, no. 4, pp. 3213–3241, 2022.
- [25] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Adv. Neural Inf. Process. Syst.*, vol. 29, 2016.
- [26] N. Ab Aza, A. Shahmansoorian, and M. Davoudi, "From inverse optimal control to inverse reinforcement learning: A historical review," *Annual Reviews in Control*, vol. 50, pp. 119–138, 2020.
- [27] A. Perrusquía, "A complementary learning approach for expertise transference of human-optimized controllers," *Neural Networks*, vol. 145, pp. 33–41, 2022.
- [28] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *Int. Conf. Learn. Repres.*, 2017.
- [29] N. Baram, O. Anschel, I. Caspi, and S. Mannor, "End-to-end differentiable adversarial imitation learning," in *Int. Conf. Mach. Learn.* PMLR, 2017, pp. 390–399.
- [30] J. Sun, L. Yu, P. Dong, B. Lu, and B. Zhou, "Adversarial inverse reinforcement learning with self-attention dynamics model," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1880–1886, 2021.

- [31] T. Huber, M. Demmler, S. Mertes, and M. Olson, "GANterfactual-RL: Understanding reinforcement learning agents' strategies through visual counterfactual explanations," *Proc. 22nd Int. Conf. Autono. Agents Multiag. Syst. (AAMAS)*, 2023.
- [32] B. Lian, W. Xue, F. L. Lewis, and A. Davoudi, "Inverse value iteration and q -learning: Algorithms, stability, and robustness," *IEEE Trans. Neural Netw. Learn. Syst.*, 2024.
- [33] 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 Trans. Cybern.*, vol. 52, no. 10, pp. 10 570–10 581, 2021.
- [34] W. Xue, B. Lian, J. Fan, P. Kolaric, T. Chai, and F. L. Lewis, "Inverse reinforcement Q-learning through expert imitation for discrete-time systems," *IEEE Trans. Neural Netw. Learn. Syst.*, 2021.
- [35] B. Lian, V. S. Donge, F. L. Lewis, T. Chai, and A. Davoudi, "Data-driven inverse reinforcement learning control for linear multiplayer games," *IEEE Trans. Neural Netw. Learn. Syst.*, 2022.
- [36] H. Yau, C. Russell, and S. Hadfield, "What did you think would happen? explaining agent behaviour through intended outcomes," *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020.
- [37] A. Perrusquía, R. Garrido, and W. Yu, "Stable robot manipulator parameter identification: A closed-loop input error approach," *Automatica*, vol. 141, p. 110294, 2022.
- [38] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.
- [39] A. Perrusquía, M. Zou, and W. Guo, "Explainable data-driven q -learning control for a class of discrete-time linear autonomous systems," *Inf. Sci.*, vol. 682, p. 121283, 2024.
- [40] J. Liang, B. I. Ahmad, M. Jahangir, and S. Godsill, "Detection of malicious intent in non-cooperative drone surveillance," in *Proc. Sensor Signal Process. Defence Conf. (SSPD)*. IEEE, 2021, pp. 1–5.
- [41] W. Yu and A. Perrusquía, "Simplified stable admittance control using end-effector orientations," *Int. J. Social Rob.*, vol. 12, 2020.
- [42] A. Perrusquía, "Solution of the linear quadratic regulator problem of black box linear systems using reinforcement learning," *Inf. Sci.*, vol. 595, pp. 364–377, 2022.
- [43] Y. Yang, Z. Guo, H. Xiong, D.-W. Ding, Y. Yin, and D. C. Wunsch, "Data-driven robust control of discrete-time uncertain linear systems via off-policy reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3735–3747, 2019.
- [44] W. Yu and A. Perrusquía, *Human-Robot Interaction Control Using Reinforcement Learning*. John Wiley & Sons, 2021.
- [45] B. Kiumarsi, F. L. Lewis, and Z.-P. Jiang, " h_∞ control of linear discrete-time systems: Off-policy reinforcement learning," *Automatica*, vol. 78, pp. 144–152, 2017.
- [46] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
- [47] A. Perrusquía and W. Guo, "Drone's objective inference using policy error inverse reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023.
- [48] D. Vrabie and F. Lewis, "Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems," *Neural networks*, vol. 22, no. 3, pp. 237–246, 2009.
- [49] N. Niknejad and H. Modares, "Physics-informed data-driven safe and optimal control design," *IEEE Control Syst. Lett.*, vol. 8, 2023.
- [50] W. Xue, B. Lian, J. Fan, T. Chai, and F. L. Lewis, "Inverse reinforcement learning for trajectory imitation using static output feedback control," *IEEE Trans. Cybern.*, 2023.
- [51] W. Xue, B. Lian, Y. Kartal, J. Fan, T. Chai, and F. L. Lewis, "Model-free inverse h -infinity control for imitation learning," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [52] W. Xue, F. L. Lewis, and B. Lian, "Certified data-driven inverse reinforcement learning of markov jump systems," *Automatica*, vol. 176, p. 112239, 2025.
- [53] H. El-Hussieny and J.-H. Ryu, "Inverse discounted-based LQR algorithm for learning human movement behaviors," *Applied Intelligence*, vol. 49, no. 4, pp. 1489–1501, 2019.



Adolfo Perrusquía received the M.Sc. and Ph.D. degrees, both in Automatic Control from the Automatic Control Department at the CINVESTAV-IPN in 2016 and 2020, respectively. He is currently a lecturer at Cranfield University, and a RAEng alumni. He is a member of the IEEE Computational Intelligence Society and Associate Editor of the IEEE TNNLS and IEEE TASE. His main research of interest focuses on reinforcement learning, inverse reinforcement learning, data-driven control nonlinear control, machine learning and system identification.



Mengbang Zou received his Ph.D. in Aerospace School at Cranfield University in 2024 and M.Sc degree from Huazhong University of Science and Technology in Mechanical Science and Engineering in 2019. Now, He is a research fellow in Cranfield University. He is interested in complex system with nonlinear dynamics including complex system's resilience, synchronization and region of attraction. He is also interested in machine learning for dynamics.



Weisi Guo received the M.Eng. degree in engineering, and the M.A. and Ph.D. degrees in computer science from the University of Cambridge, Cambridge, U.K., in 2005, 2011, and 2011, respectively. He is a Chair Professor of Human Machine Intelligence with Cranfield University, Cranfield, U.K. He has published over 180 papers and is a PI on a number of molecular communication research grants.

Prof. Guo's research has won him several international awards. He was a Turing Fellow at the Alan Turing Institute.



DSPACE

<https://dspace.org/>

Interpreting the observed behavior of a class of autonomous linear systems using explainable inverse reinforcement learning

Perrusquía, Adolfo; Zou, Mengbang; Guo, Weisi

2026-12-31

Institute of Electrical and Electronics Engineers (IEEE)

<https://doi.org/10.1109/tnnls.2026.3699708>

Perrusquía A, Zou M, Guo W. (2026) Interpreting the observed behavior of a class of autonomous linear systems using explainable inverse reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems, Available online 8 June 2026