IEEE Transactions on Cybernetics, Volume 55, Issue 1, January 2025,pp. 14-23
DOI:10.1109/tcyb.2024.3489967

Uncovering Reward Goals in Distributed Drone Swarms using Physics-Informed Multi-Agent Inverse Reinforcement Learning

Adolfo Perrusquía, *Member, IEEE* and Weisi Guo, *Senior Member, IEEE*,

***Abstract*—The cooperative nature of drone swarms poses risks in the smooth operation of services and the security of national facilities. The control objective of the swarm is, in most cases, occluded due to the complex behaviours observed in each drone. It is paramount to understand which is the control objective of the swarm, whilst understanding better how they communicate with each other to achieve the desired task. To solve these issues, this paper proposes a physics-informed multi-agent inverse reinforcement learning (PI-MAIRL) that i) infers the control objective function or reward function from observational data, and ii) uncover the network topology by exploiting a physics-informed model of the dynamics of each drone. The combined contribution enables to understand better the behaviour of the swarm, whilst enabling the inference of its objective for experience inference and imitation learning. A physically uncoupled swarm scenario is considered in this study. The incorporation of the physics-informed element allows to obtain an algorithm that is computationally more efficient than model-free IRL algorithms. Convergence of the proposed approach is verified using Lyapunov recursions on a global Riccati equation. Simulation studies are carried out to show the benefits and challenges of the approach.**

***Index Terms*—drone swarms, multi-agent inverse reinforcement learning, physics-informed, network topology, reward function, imitation learning.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) also known as drones have been used in a diverse number of beneficial applications including package delivery, search and rescue [1], and surveillance [2], [3]. The application power of drones is amplified when they work collaboratively to develop a task. Here a large group of multiple drones flying together as a coordinated entity is known as drone swarm. In this case, drone swarms are capable to highly improve the aforementioned applications by maximizing the area of coverage and reducing time [4]. However, the cooperative nature of drone swarms has magnified the threat space in national infrastructure security and people protection [5]–[7].

The state-of-the-art is interested in how we can ensure that the usage of drones is regulated to ensure smooth operation of services [8], [9]. This is done by detecting and interpreting the observed behaviour of the drone or drones or by inferring its intention [10].

A. Related work

The most simple approaches for behaviour prediction are based on the definition of geofence areas [11], [12] and the use of expert machine learning techniques [13] to classify the behaviour of the drone in either malicious or non-malicious [14]. Here, expert rules [15] are designed based on specific values of low-dimensional features, e.g., height and velocity constraints, and behavioural patterns in either velocity or accelerations measurements. The challenge of these methods is an inherent cognitive bias that increases the number of false positives in counter-drone system technologies.

Other approaches has adopted deep learning techniques to classify and predict intention based on proxy definitions of intention ranging from trajectory to reward goals[1] [16], [17]. These methods have been successfully applied in trajectory data of a single autonomous platforms and pedestrians [18]–[20], however the application of these techniques in drone swarms is still an open challenge. In this particular case, trajectory intention can classify independent trajectories [13], [21] but it cannot uncover the relationship between drones and what they are aiming to perform [22]. Alternatively, intended outcomes [23] have been used to understand decisions in reinforcement learning (RL) environments from the state-transition probability [18], [24], [25]. Flight physics has been incorporated into deep learning approaches as regularisation term to enhance the robustness and stability of the predictions [26], [27]. However, flight physics of drone swarms cannot be easily determined due to the diverse communication topology that yields to diverse swarm configurations that do not show a clear physical pattern [28].

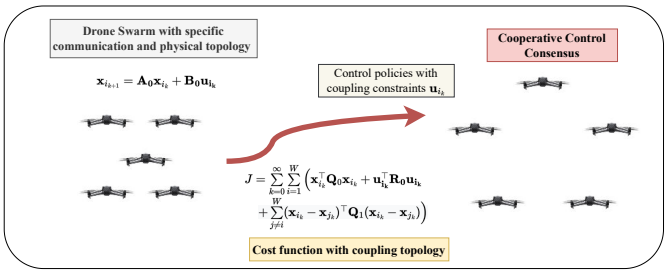


Fig. 1. Drone swarm modelling for cooperative control/consensus

A. Perrusquía and Weisi Guo are with the School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford, UK (e-mail: adolfo.perrusquia-guzman@cranfield.ac.uk).

[1]Trajectory intention is associated to the mission profile exhibited by the drone, whilst reward goal is associated to the control objective (reward function) that the drone aims to optimize.

On the other hand, reward goals [17] provide an indicator of what and why we observed a particular behaviour in the drones such that we can understand better the network topology and assign the level of risk [29]. Previous studies have used imitation learning [30], [31] and inverse reinforcement learning (IRL) approaches to uncover the hidden reward goal from observational data [32]–[34] in optimal and zero-sum game problems [35], [36]. These approaches usually assume a specific reward function structure to infer the reward weights of both the states and input policy to inject a desired behaviour. Here, a fixed input policy weight is used to infer the state reward weight to ensure stability of the IRL [37], [38]. However, multiple reward weights are obtained due to the inverse nature of the problem and rank deficiency of the collected data. This plays an important issue in intention inference problems since we obtain a different reward goal that does not necessarily model the real control objective. This is more acute in multi-agent problems where an accurate inference of the collective reward goal is required to effectively understand the observed multi-agent behaviour.

B. Contributions

The following gaps are identified in the current state-of-the-art: i) IRL approaches find a family of reward goals that generate the same control policy, but cannot uncover the real control objective intention, ii) IRL suffers of contextual understanding of the network topology, and iii) the computational complexity of the overall IRL algorithm increases as the number of agents increases. To deal with the aforementioned gaps, this paper proposes a physics-informed multi-agent IRL (PI-MAIRL) to uncover the reward goals of drone swarms. The approach models the multi-agent problem as a global system that facilitates the inference of the reward function. Here, the state-transition matrix of each drone is used as a physics-informed structure that reduces the computational requirements of the algorithm. This simultaneously allows to infer the swarm topology from the data. Simulations experiments using different drone swarms are given to show the benefits and challenges of the approach.

The main contributions of this work are: 1) a PI-MAIRL is proposed to uncover the exact reward function from observational data of the swarm trajectories, 2) the approach reduces the computational complexity of model-free IRL approaches by incorporating a physics-informed structure associated to the state-transition of each drone, 3) the network topology is inferred from the reward function which provides a better understanding of the global reward goal through the weights values.

The outline of the paper is as follows. Section II provides the drone swarm modelling and its optimal control design. Section III provides the main contribution of the work by developing the PI-MAIRL for physically uncoupled drone swarms. Section IV gives the simulation studies and the conclusions and future work are given in Section V.

C. Notations

Throughout this paper, $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times m}$, $\mathbb{C}^{n\times n}$ denote the spaces of natural numbers, real numbers, real n-vectors, real $n\times m$-matrices, and complex $n\times m$-matrices, respectively; $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ denotes an identity matrix of $n\times n$; $\otimes$ and $\bar{\otimes}$ denote the Kronecker and symmetric Kronecker product, vec$(\boldsymbol{A})$ and vech$(\boldsymbol{A})$ are the vectorization and half vectorization of matrix $\boldsymbol{A}$, mat$(\boldsymbol{x})$ is the matrization of the vector $\boldsymbol{x}$, where $x\in\mathbb{R}$ is a scalar, $\boldsymbol{x}\in\mathbb{R}^n$ is a vector, and $\boldsymbol{X}\in\mathbb{R}^{n\times m}$ is a matrix with $n,m\in\mathbb{N}$.

A graph topology $\mathcal{G}=(\mathcal{V}_\mathcal{G},\mathcal{E}_\mathcal{G})$ is defined by a finite set $\mathcal{V}_\mathcal{G}=\{v_1,\cdots,v_W\}$ of W vertices and a set of edges $\mathcal{E}_\mathcal{G}\subseteq\mathcal{V}_\mathcal{G}\times\mathcal{V}_\mathcal{G}$ that model interagent information-exchange links weighted by a connectivity weight $\alpha_{ij}>0$ if $(v_j,v_i)\in\mathcal{E}_\mathcal{G}$, otherwise $\alpha_{ij}=0$. The set of neighbors of vertex v_i is $\mathcal{N}_i=\{v_j:\alpha_{ij}>0\}$. We assume a simple graph without repeated edges. The degree matrix $\boldsymbol{D}\in\mathbb{R}^{W\times W}$ of the graph $\mathcal{G}$ is a diagonal matrix with entries $d_i=\sum_{j\in\mathcal{N}_i}\alpha_{ij}$. The Laplacian matrix $\boldsymbol{\mathcal{L}}$ is defined as $\boldsymbol{\mathcal{L}}=\boldsymbol{D}-\boldsymbol{\mathcal{A}}\in\mathbb{R}^{W\times W}$ where $\boldsymbol{\mathcal{A}}=\boldsymbol{\mathcal{A}}^\top=[\alpha_{ij}]\in\mathbb{R}^{W\times W}$ is the adjacency matrix [39].

II. Drone Swarm Modelling

Fig. 1 shows a drone swarm application for either cooperative control or consensus. The diagram consists in a number of drones coupled by a communication topology. The main purpose of the drone swarm is to optimize a cost index J to achieve a specific task, e.g., cooperative control or consensus. Here, we are interested to infer the hidden reward function or utility function that drives the drone swarm to exhibit a specific performance.

To this end, consider that each drone $i=1,2,...,W$ of the swarm has the following local dynamic model

$$\boldsymbol{x}_{i_{k+1}}=\boldsymbol{A}_0\boldsymbol{x}_{i_k}+\boldsymbol{B}_0\boldsymbol{u}_{i_k}, \tag{1}$$

where $\boldsymbol{x}_{i_k}\in\mathbb{R}^n$ defines the state of drone i, $\boldsymbol{u}_{i_k}\in\mathbb{R}^m$ denotes the local control input of the drone i, $\boldsymbol{A}_0\in\mathbb{R}^{n\times n}$ and $\boldsymbol{B}_0\in\mathbb{R}^{n\times m}$ defines the dynamics of drone i. The control policies for each drone are obtained by minimizing the following cost index

$$\begin{aligned}J(\boldsymbol{x}_k,\boldsymbol{u}_k)=\sum_{k=0}^{\infty}\Bigg[&\sum_{i=1}^{W}\boldsymbol{x}_{i_k}^\top\boldsymbol{Q}_0\boldsymbol{x}_{i_k}+\boldsymbol{u}_{i_k}^\top\boldsymbol{R}_0\boldsymbol{u}_{i_k}\\&+\sum_{i=1}^{W}\sum_{j\neq i}^{W}(\boldsymbol{x}_{i_k}-\boldsymbol{x}_{j_k})^\top\boldsymbol{Q}_1(\boldsymbol{x}_{i_k}-\boldsymbol{x}_{j_k})\Bigg]\end{aligned} \tag{2}$$

where the terms $\boldsymbol{x}_k=[\boldsymbol{x}_{1_k}^\top,\cdots,\boldsymbol{x}_{W_k}^\top]^\top\in\mathbb{R}^N$ and $\boldsymbol{u}_k=[\boldsymbol{u}_{1_k}^\top,\cdots,\boldsymbol{u}_{W_k}^\top]^\top\in\mathbb{R}^M$ are the lumped or global state and control input vectors, $\boldsymbol{Q}_0=\boldsymbol{Q}_0^\top\succeq 0\in\mathbb{R}^{n\times n}$ and $\boldsymbol{Q}_1=\boldsymbol{Q}_1^\top\succeq 0\in\mathbb{R}^{n\times n}$ are positive semi-definite weight matrices and $\boldsymbol{R}_0=\boldsymbol{R}_0^\top\succ 0\in\mathbb{R}^{m\times m}$ is a positive definite weight matrix. Here $N:=nW$ and $M:=mW$.

Assumption 1: It is assumed that both $\boldsymbol{Q}_0$ and $\boldsymbol{Q}_1$ are diagonal positive semi-definite unknown matrices. Matrix $\boldsymbol{R}_0$ is known or assumed to be unitary, i.e., $\boldsymbol{R}_0=\boldsymbol{I}_m$.

Both (1) and (2) can be written in a compact form as follows

$$\boldsymbol{x}_{k+1}=\boldsymbol{A}\boldsymbol{x}_k+\boldsymbol{B}\boldsymbol{u}_k \tag{3}$$

$$J(\boldsymbol{x}_k,\boldsymbol{u}_k)=\sum_{k=0}^{\infty}\left(\boldsymbol{x}_k^\top\boldsymbol{Q}\boldsymbol{x}_k+\boldsymbol{u}_k^\top\boldsymbol{R}\boldsymbol{u}_k\right), \tag{4}$$

where $\boldsymbol{A} = \boldsymbol{I}_W \otimes \boldsymbol{A}_0 \in \mathbb{R}^{N\times N}$, $\boldsymbol{B} = \boldsymbol{I}_W \otimes \boldsymbol{B}_0 \in \mathbb{R}^{N\times M}$, $\boldsymbol{R} = \boldsymbol{R}^\top = \boldsymbol{I}_W \otimes \boldsymbol{R}_0 \succ 0 \in \mathbb{R}^{M\times M}$ and $\boldsymbol{Q} = \boldsymbol{Q}^\top = (\boldsymbol{I}_W \otimes \boldsymbol{Q}_0 + \boldsymbol{\mathcal{L}} \otimes \boldsymbol{Q}_1) \succeq 0 \in \mathbb{R}^{N\times N}$. Then, the optimisation problem is simplified to

$$\begin{aligned} \min_{\boldsymbol{u}_k\in\mathbb{R}^M} J(\boldsymbol{x}_k,\boldsymbol{u}_k) = \quad & \min_{\boldsymbol{u}_k\in\mathbb{R}^M} \sum_{k=0}^{\infty} \left(\boldsymbol{x}_k^\top \boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_k^\top \boldsymbol{R}\boldsymbol{u}_k\right) \\ \text{subject to} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \end{aligned} \tag{5}$$

The unique solution of (5) can be written quadratically in the global state as

$$J^*(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k\in\mathbb{R}^M} J(\boldsymbol{x}_k,\boldsymbol{u}_k) = \boldsymbol{x}_k^\top \boldsymbol{P}\boldsymbol{x}_k \tag{6}$$

where $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0 \in \mathbb{R}^{N\times N}$ is a kernel matrix that is solution of the following discrete-time algebraic Riccati equation (DARE)

$$\boldsymbol{A}^\top\boldsymbol{P}\boldsymbol{A}+\boldsymbol{Q}-\boldsymbol{A}^\top\boldsymbol{P}\boldsymbol{B}(\boldsymbol{R}+\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{A}=\boldsymbol{P}. \tag{7}$$

Thus, the optimal control policies are given by

$$\boldsymbol{u}_k = -\boldsymbol{K}\boldsymbol{x}_k = -(\boldsymbol{R}+\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{A}\boldsymbol{x}_k, \tag{8}$$

where $\boldsymbol{K} \in \mathbb{R}^{M\times N}$ is the global control gain.

III. Physics-Informed Multi-Agent Inverse Reinforcement Learning(PI-MAIRL)

Fig. 2 depicts the high-level diagram of the proposed PI-MAIRL. The diagram has four main elements:

1) A hybrid model-free/model-based off-policy RL algorithm that computes the kernel matrix $\boldsymbol{P}_j$ and control gain $\boldsymbol{\mathcal{K}}_{j+1}$ associated to the dataset $\boldsymbol{\mathcal{X}}$ and an iterative global reward weight $\boldsymbol{Q}_l$. The approach initially infers from the data the optimal control solution for some initial reward weights. In addition, the collected data are used combined with a physics informed structure to estimate the input dynamics $\boldsymbol{B}$. This allows to transform the model-free RL algorithm into a model-based RL approach that reduces the computational resources of the overall algorithm.
2) A physics-informed state transition model that provides information of the unforced dynamics of each drone. This model allows the smooth transition between model-free and model-based approaches that helps to reduce the computational requirements for large-scale systems.
3) A model-based policy error MAIRL algorithm that is fed by the outputs of the RL algorithm and physics-informed model to compute the improved weight matrix $\boldsymbol{Q}_{l+1}$. The algorithm is based on an enhanced inverse optimal control architecture that integrates a projection mapping to ensure convergence to the exact reward weights.
4) A simple algorithm to infer the network topology based on the global weight $\boldsymbol{Q}_l$. This provides an additional dimension of explainability to the algorithm that allows to understand how each drone in the swarm communicates with the others.

The elements of the proposed PI-MAIRL are described in the following sections.

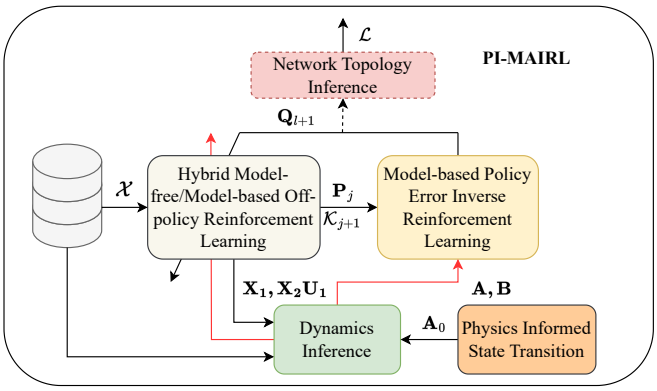


Fig. 2. PI-MAIRL high-level diagram

A. Off-policy Model-free RL

Consider a dataset $\boldsymbol{\mathcal{X}} = \{\boldsymbol{X}, \boldsymbol{U}\}$ constructed from the global trajectories (3) of the swarm given by

$$\begin{aligned} \boldsymbol{X} &= \begin{bmatrix} \boldsymbol{x}_0 & \cdots & \boldsymbol{x}_{T-1} \end{bmatrix} \in \mathbb{R}^{N\times T}, \\ \boldsymbol{U} &= \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_T \end{bmatrix} \in \mathbb{R}^{M\times T}, \end{aligned} \tag{9}$$

where $T \in \mathbb{N}$ is the length of the collected data.

The global dynamics can be written in terms of an iterative global control policy $\boldsymbol{v}_k^j = -\boldsymbol{\mathcal{K}}_j\boldsymbol{x}_k \in \mathbb{R}^M$ based on an iterative control gain $\boldsymbol{\mathcal{K}}_j \in \mathbb{R}^{M\times N}$ as

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \pm \boldsymbol{B}\boldsymbol{v}_k^j \\ &= (\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)\boldsymbol{x}_k + \boldsymbol{B}(\boldsymbol{u}_k - \boldsymbol{v}_k^j). \end{aligned} \tag{10}$$

The DARE (7) can be written in terms of the following iterative matrices

$$(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)^\top\boldsymbol{P}_j(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j) + \boldsymbol{Q}_l + \boldsymbol{\mathcal{K}}_j^\top\boldsymbol{R}\boldsymbol{\mathcal{K}}_j - \boldsymbol{P}_j = \boldsymbol{0}_N, \tag{11}$$

where $\boldsymbol{Q}_l = \boldsymbol{Q}_l^\top \succeq 0 \in \mathbb{R}^{N\times N}$ is an iterative weight matrix, $\boldsymbol{P}_j = \boldsymbol{P}_j^\top \succ 0 \in \mathbb{R}^{N\times N}$ is the corresponding kernel matrix in iteration j. Notice that when $\boldsymbol{\mathcal{K}}_j = \boldsymbol{K}$, then the DARE (7) is recovered. The Hamilton-Jacobi-Bellman (HJB) equation of (10) is

$$\begin{aligned} &\boldsymbol{x}_k^\top(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)^\top\boldsymbol{P}_j(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)\boldsymbol{x}_k - \boldsymbol{x}_k^\top\boldsymbol{P}_j\boldsymbol{x}_k \\ &\quad + 2\boldsymbol{x}_k^\top(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)^\top\boldsymbol{P}_j\boldsymbol{B}\widetilde{\boldsymbol{u}}_k^j + \widetilde{\boldsymbol{u}}_k^{j^\top}\boldsymbol{B}^\top\boldsymbol{P}_j\boldsymbol{B}\widetilde{\boldsymbol{u}}_k^j \\ &\quad + \boldsymbol{x}_k^\top\boldsymbol{Q}_l\boldsymbol{x}_k + \boldsymbol{v}_k^{j^\top}\boldsymbol{R}\boldsymbol{v}_k^j = \boldsymbol{0}_N, \end{aligned} \tag{12}$$

where $\widetilde{\boldsymbol{u}}_k^j = \boldsymbol{u}_k - \boldsymbol{v}_k^j \in \mathbb{R}^M$ is the global policy error [40]. Notice that (12) is equivalent to the DARE (11) when $\widetilde{\boldsymbol{u}}_k^j \to \boldsymbol{0}_M$. To ensure the HJB (12) matches with the DARE (11), we need to compensate the terms with $\widetilde{\boldsymbol{u}}_k^j$ as

$$\begin{aligned} &\boldsymbol{x}_k^\top(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)^\top\boldsymbol{P}_j(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)\boldsymbol{x}_k - \boldsymbol{x}_k^\top\boldsymbol{P}_j\boldsymbol{x}_k \\ &\quad + 2\boldsymbol{x}_k^\top(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)^\top\boldsymbol{P}_j\boldsymbol{B}\widetilde{\boldsymbol{u}}_k^j + \widetilde{\boldsymbol{u}}_k^{j^\top}\boldsymbol{B}^\top\boldsymbol{P}_j\boldsymbol{B}\widetilde{\boldsymbol{u}}_k^j \\ &= 2\boldsymbol{x}_k^\top(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{\mathcal{K}}_j)^\top\boldsymbol{P}_j\boldsymbol{B}\widetilde{\boldsymbol{u}}_k^j + \widetilde{\boldsymbol{u}}_k^{j^\top}\boldsymbol{B}^\top\boldsymbol{P}_j\boldsymbol{B}\widetilde{\boldsymbol{u}}_k^j \\ &\quad - \boldsymbol{x}_k^\top\boldsymbol{Q}_l\boldsymbol{x}_k - \boldsymbol{v}_k^{j^\top}\boldsymbol{R}\boldsymbol{v}_k^j. \end{aligned} \tag{13}$$

The off-policy RL [41] can be obtained by rewriting (13) as the Bellman equation

$$\begin{aligned}\boldsymbol{x}_k^\top \boldsymbol{P}_j \boldsymbol{x}_k - \boldsymbol{x}_{k+1}^\top \boldsymbol{P}_j \boldsymbol{x}_{k+1} + 2\boldsymbol{x}_k^\top \boldsymbol{A}^\top \boldsymbol{P}_j \boldsymbol{B} \widetilde{\boldsymbol{u}}_k^j \\ + \widetilde{\boldsymbol{v}}_k^{j^\top} \boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{B} \widetilde{\boldsymbol{u}}_k^j = \xi_k, \end{aligned} \tag{14}$$

where $\widetilde{\boldsymbol{v}}_k^j = \boldsymbol{u}_k + \boldsymbol{v}_k^j \in \mathbb{R}^M$ is an enhanced control policy and $\xi_k = \boldsymbol{x}_k^\top \boldsymbol{Q}_l \boldsymbol{x}_k + \boldsymbol{v}_k^{j^\top} \boldsymbol{R} \boldsymbol{v}_k^j$. A system of equations is constructed from $\kappa \geq \iota = \frac{1}{2}[(N+M)^2 + N + M]$ samples collected from the swarm trajectories,

$$\boldsymbol{\Phi} \boldsymbol{\theta}_j = \boldsymbol{\Xi}, \tag{15}$$

where $\boldsymbol{\theta}_j = [\boldsymbol{\theta}_1^\top, \boldsymbol{\theta}_2^\top, \boldsymbol{\theta}_3^\top]^\top \in \mathbb{R}^\iota$ is the vector of parameter estimates, $\boldsymbol{\Phi} = [\boldsymbol{\Phi}_{xx}, 2\boldsymbol{\Phi}_{xu}, \boldsymbol{\Phi}_{uu}] \in \mathbb{R}^{\kappa \times \iota}$, and $\boldsymbol{\Xi} = [\xi_k, \cdots, \xi_\kappa]^\top \in \mathbb{R}^\kappa$, with $\boldsymbol{\theta}_1 = \text{vech}^\top(\boldsymbol{P}_j) \in \mathbb{R}^{\frac{1}{2}N(N+1)}$, $\boldsymbol{\theta}_2 = \text{vec}^\top(\boldsymbol{A}^\top \boldsymbol{P}_j \boldsymbol{B}) \in \mathbb{R}^{NM}$, $\boldsymbol{\theta}_3 = \text{vech}^\top(\boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{B}) \in \mathbb{R}^{\frac{1}{2}M(M+1)}$ given by

$$\begin{aligned}\boldsymbol{\Phi}_{xx} &= \left[\boldsymbol{x}_t^\top \bar{\otimes} \boldsymbol{x}_t^\top \Big|_{k+1}^{k} \quad \cdots \quad \boldsymbol{x}_t^\top \bar{\otimes} \boldsymbol{x}_t^\top \Big|_{\kappa+1}^{\kappa}\right] \in \mathbb{R}^{\kappa \times \frac{1}{2}N(N+1)} \\ \boldsymbol{\Phi}_{xu} &= \left[\widetilde{\boldsymbol{u}}_k^{\top^j} \otimes \boldsymbol{x}_k^\top \quad \cdots \quad \widetilde{\boldsymbol{u}}_\kappa^{\top^j} \otimes \boldsymbol{x}_\kappa^\top\right] \in \mathbb{R}^{\kappa \times NM} \\ \boldsymbol{\Phi}_{uu} &= \left[\widetilde{\boldsymbol{u}}_k^{\top^j} \bar{\otimes} \widetilde{\boldsymbol{v}}_k^{j^\top} \quad \cdots \quad \widetilde{\boldsymbol{u}}_\kappa^{\top^j} \bar{\otimes} \widetilde{\boldsymbol{v}}_\kappa^{j^\top}\right] \in \mathbb{R}^{\kappa \times \frac{1}{2}M(M+1)}.\end{aligned}$$

Hence, in each episode the parameters are updated using a batch least-squares (BLS) rule

$$\boldsymbol{\theta}_j = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\Xi}. \tag{16}$$

After the BLS rule is computed then the policy is updated by using

$$\boldsymbol{\mathcal{K}}_{j+1} = (\boldsymbol{R} + [\text{mat}(\boldsymbol{\theta}_3)]_M)^{-1} [\text{mat}(\boldsymbol{\theta}_2)]_{N \times M}^\top. \tag{17}$$

The BLS update rule (16) and the gain computation (17) defines the model-free RL algorithm.

B. Physics-Informed Structures

Physics-informed structures can be incorporated in the inference problem to reduce the computational complexity of the IRL. To this end, we exploit the special structure of matrix $\boldsymbol{A}$ [42], [43]. To motivate this property, consider the continuous time matrix of the dynamics of drone i

$$\boldsymbol{A}_c = \begin{bmatrix} \boldsymbol{0}_{n/2} & \boldsymbol{I}_{n/2} \\ \boldsymbol{0}_{n/2} & \boldsymbol{0}_{n/2} \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

The discrete version of $\boldsymbol{A}_c$ yields the matrix $\boldsymbol{A}_0$ given by

$$\boldsymbol{A}_0 = \begin{bmatrix} \boldsymbol{I}_{n/2} & t_s \boldsymbol{I}_{n/2} \\ \boldsymbol{0}_{n/2} & \boldsymbol{I}_{n/2} \end{bmatrix},$$

where $t_s > 0$ is the sampling time. This sampling time is inferred from the collected data. Then, we can observe that $\boldsymbol{A}_0$ is an upper-triangular and regular (invertible) matrix. Moreover, the global matrix $\boldsymbol{A}$ is also a block-diagonal matrix with an upper-triangular structure. This provides a physics-informed structure that facilitates the design of a model-based policy error MAIRL algorithm. From the collected trajectories $\boldsymbol{X}$ and $\boldsymbol{U}$ we estimate $\boldsymbol{B}$ in the first episode of the model-free RL as

$$\boldsymbol{B} = (\boldsymbol{X}_2 - \boldsymbol{A}\boldsymbol{X}_1)\boldsymbol{U}_1^+, \tag{18}$$

where $\boldsymbol{X}_1 = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_{\iota-1}] \in \mathbb{R}^{N \times (\iota-1)}$, $\boldsymbol{X}_2 = [\boldsymbol{x}_2 \cdots \boldsymbol{x}_\iota] \in \mathbb{R}^{N \times (\iota-1)}$, $\boldsymbol{U}_1 = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_{\iota-1}] \in \mathbb{R}^{M \times (\iota-1)}$, and $\boldsymbol{U}_1^+$ denotes the Moore-Penrose pseudo-inverse of $\boldsymbol{U}_1$.

With this physical information about the swarm, we are able to modify the model-free RL algorithm in (16) into a model-based RL that solves the DARE (7). This simplifies the computational resources of the algorithm, whilst it constraints the learning manifold of the overall proposed approach.

C. Model-Based Policy Error Inverse Reinforcement Learning

In this section we design a policy error architecture that updates the kernel matrix $\boldsymbol{P}_j$ in the direction of the real kernel matrix $\boldsymbol{P}$. To do so, define the gain error

$$\boldsymbol{E}_K := \boldsymbol{K} - \boldsymbol{\mathcal{K}}_{j+1} \in \mathbb{R}^{M \times N}. \tag{19}$$

Consider the minimization of the index $E = \text{tr}\{\boldsymbol{E}_K^\top \boldsymbol{E}_K\}$. We are interested in minimizing E by updating the kernel matrix in the direction that $\boldsymbol{\mathcal{K}}_j$ approaches to $\boldsymbol{K}$. Consider a one-step update rule given by

$$\boldsymbol{\mathcal{P}} = \boldsymbol{P}_j - \gamma \frac{\partial E}{\partial \boldsymbol{P}_j}, \tag{20}$$

where $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{P}}^\top \succ 0 \in \mathbb{R}^{N \times N}$ is an updated kernel matrix and $\gamma > 0$ is a learning rate. Let $\boldsymbol{L}_j = \boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{B}$. Then by the chain rule we have that

$$\begin{aligned}\frac{\partial E}{\partial \boldsymbol{P}_j} = &- \boldsymbol{E}_k^\top \left(\frac{\partial \boldsymbol{L}_j^{-1}}{\partial \boldsymbol{P}_j} \boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{A} + \boldsymbol{L}_j^{-1} \frac{\boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{A}}{\partial \boldsymbol{P}_j} \right) \\ &- \left(\frac{\partial \boldsymbol{L}_j^{-1}}{\partial \boldsymbol{P}_j} \boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{A} + \boldsymbol{L}_j^{-1} \frac{\partial \boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{A}}{\partial \boldsymbol{P}_j} \right)^\top \boldsymbol{E}_K.\end{aligned}$$

Recall the following matrix derivation identity

$$\frac{\partial \boldsymbol{L}_j^{-1}}{\partial \boldsymbol{P}} = -\boldsymbol{L}_j^{-1} \frac{\partial \boldsymbol{L}_j}{\partial \boldsymbol{P}_j} \boldsymbol{L}_j^{-1}.$$

This implies that $\partial \boldsymbol{L}_j / \partial \boldsymbol{P}_j = \partial [\boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{B}] / \partial \boldsymbol{P}_j$ is an $M \times M$ matrix. The partial derivative of $\boldsymbol{L}_j$ with respect to $\boldsymbol{P}_j$ is usually computed by vectorizing the matrix and compute its gradient as

$$\frac{\partial [(\boldsymbol{B}^\top \otimes \boldsymbol{B}^\top) \text{vec}(\boldsymbol{P}_j)]}{\partial \text{vec}(\boldsymbol{P}_j)} = \boldsymbol{B}^\top \otimes \boldsymbol{B}^\top \in \mathbb{R}^{M^2 \times N^2}.$$

We need an $M \times M$ matrix in order to compute the new kernel matrix $\boldsymbol{\mathcal{P}}$. To this end, we sum the elements of each row and create a new column vector as follows

$$b_p = \sum_{q=1}^{N^2} (\boldsymbol{B}^\top \otimes \boldsymbol{B}^\top)_{pq} \in \mathbb{R},$$

where $p = 1, ..., M^2$ and $q = 1, ..., N^2$. Let denote $\boldsymbol{b} = [b_1, \cdots, b_{M^2}]^\top \in \mathbb{R}^{M^2}$. Then, a new matrix $\boldsymbol{\mathcal{B}}$ is constructed as $\boldsymbol{\mathcal{B}} = \text{mat}(\boldsymbol{b}) \in \mathbb{R}^{M \times M}$.

A similar approach is used to compute the derivative of $\partial [\boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{A}] / \partial \boldsymbol{P}_j$ which produces a new matrix $\boldsymbol{\mathcal{C}} = \text{mat}(\boldsymbol{c}) \in \mathbb{R}^{M \times N}$ with $\boldsymbol{c} = [c_1, ..., c_{NM}] \in \mathbb{R}^{NM}$ and

$$c_r = \sum_{q=1}^{N^2} (\boldsymbol{A}^\top \otimes \boldsymbol{B}^\top)_{rq} \in \mathbb{R}, \quad r = 1, ..., NM.$$

Remark 1: The obtained matrices $\mathcal{B}$ and $\mathcal{C}$ are valid for one single agent (drone in this case). However, for multi-agent problems, these matrices cannot be used since the sum of rows produce wrong update directions for the kernel matrix $\mathcal{P}$ derived by the block-diagonal structures of both $\boldsymbol{A}$ and $\boldsymbol{B}$. This makes hard to know how we can manipulate the high dimensional matrices $\boldsymbol{B}^\top \otimes \boldsymbol{B}^\top$ and $\boldsymbol{A}^\top \otimes \boldsymbol{B}^\top$ into an appropriate lower dimensional representation.

To deal with the challenge of Remark 1, we observed that: i) $\|\boldsymbol{B}^\top \otimes \boldsymbol{B}^\top\| = \|\boldsymbol{B}^\top \boldsymbol{B}\|$ which means that the Kronecker product distributes the values of $\boldsymbol{B}^\top \boldsymbol{B}$ in a higher dimensional space, ii) the control inputs in $\boldsymbol{u}$ are not coupled (linearly combined) by matrix $\boldsymbol{B}$ such that $\boldsymbol{B}^\top \boldsymbol{B}$ is always a positive-definite diagonal matrix, and iii) the kernel matrix $\boldsymbol{P}_j$ is sparse due to the uncoupled physics between agents.

Assumption 2: To maintain the sparsity of the kernel matrix $\boldsymbol{P}_j$ and exploit the topology of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we approximate the partial derivatives with the following expressions

$$\frac{\partial \boldsymbol{L}_j^{-1}}{\partial \boldsymbol{P}_j} \approx -\boldsymbol{L}_j^{-1}\boldsymbol{B}^\top \boldsymbol{B}\boldsymbol{L}_j^{-1}, \;\; \frac{\partial \boldsymbol{B}^\top \boldsymbol{P}_j \boldsymbol{A}}{\partial \boldsymbol{P}_j} \approx \boldsymbol{B}^\top \boldsymbol{A}.$$

Assumption 2 ensures stability of the proposed algorithm, whilst preserving the sparse topology of the multi-agent problem. Here, the loss of information is attenuated by Assumption 1 which ensures sparsity of the kernel matrix, whilst the most important information is captured in the diagonal terms of both $\boldsymbol{Q}_l$ and $\boldsymbol{P}_j$. Therefore, the new improved kernel matrix $\mathcal{P}$ is

$$\begin{aligned}\mathcal{P} = \boldsymbol{P}_j + \gamma\Big(&\boldsymbol{E}_K^\top \boldsymbol{L}_j^{-1}\boldsymbol{B}^\top(\boldsymbol{A} - \boldsymbol{B}\mathcal{K}_{j+1}) \\ &+(\boldsymbol{A} - \boldsymbol{B}\mathcal{K}_{j+1})^\top \boldsymbol{B}\boldsymbol{L}_j^{-1}\boldsymbol{E}_K\Big).\end{aligned} \tag{21}$$

Then, a new improved kernel gain is computed as

$$\mathfrak{K} = (\boldsymbol{R} + \boldsymbol{B}^\top \mathcal{P}\boldsymbol{B})^{-1}\boldsymbol{B}^\top \mathcal{P}\boldsymbol{A}. \tag{22}$$

We compute an estimate matrix $\widehat{\boldsymbol{Q}} = [\widehat{Q}_{uv}]_{1\le u,v\le N} \in \mathbb{R}^{N\times N}$ of matrix $\boldsymbol{Q}_{j+1}$ using the DARE (7) as,

$$\widehat{\boldsymbol{Q}} = \mathcal{P} - \boldsymbol{A}^\top \mathcal{P}\boldsymbol{A} + \mathfrak{K}^\top(\boldsymbol{R} + \boldsymbol{B}^\top \mathcal{P}\boldsymbol{B})\mathfrak{K}. \tag{23}$$

The estimated matrix $\widehat{\boldsymbol{Q}}$ is a symmetric matrix that does not follow Assumption 1 and can be a negative definite matrix. To overcome this issue, we exploit the topology of $\boldsymbol{Q}$ to construct an appropriate weight update $\boldsymbol{Q}_{l+1}$ as follows

$$\boldsymbol{Q}_{l+1} = \mathcal{H}[\widehat{\boldsymbol{Q}}] := \text{diag}(\boldsymbol{q}_1) + (\boldsymbol{1}_W - \boldsymbol{I}_W) \otimes \text{diag}(\boldsymbol{q}_2), \tag{24}$$

where $\mathcal{H}[\cdot]$ is an operator that recovers the diagonal elements of each $n \times n$ block matrix of $\widehat{\boldsymbol{Q}}$, $\boldsymbol{q}_1 = [\widehat{Q}_{11}, \cdots, \widehat{Q}_{NN}]^\top \in \mathbb{R}^N$ has the diagonal elements of matrix $\widehat{\boldsymbol{Q}}$ and $\boldsymbol{q}_2 = [\widehat{Q}_{n+1,n+1}, \cdots, \widehat{Q}_{2n,2n}]^\top \in \mathbb{R}^n$ has the diagonal elements of the second $n \times n$ block matrix of $\widehat{\boldsymbol{Q}}$. Algorithm 1 summarizes the proposed PI-MAIRL.

The following theorem establishes the convergence of the proposed PI-MAIRL to uncover the reward goal of drone swarms.

Theorem 1: Consider an optimal global control policy $\boldsymbol{K}$ based on diagonal reward weights. Assume that the observational data is rich enough such that $\boldsymbol{\Phi}$ has complete rank in each RL episode. Then, the global reward weight $\boldsymbol{Q}_l$ and kernel matrix $\boldsymbol{P}_j$ updated as in Algorithm 1 converges to their exact values in the limit.

Algorithm 1 PI-MAIRL algorithm

Require: Dataset $\mathcal{X}$, state transition matrix $\boldsymbol{A}_0$ and reward weight $\boldsymbol{R}$. Initial weight matrix $\boldsymbol{Q}_{j=0}$ and learning rate $\gamma > 0$. Stabilizing control policy $\boldsymbol{u}_0 = -\mathcal{K}_0\boldsymbol{x}$. Set $j = 0$, $l = 0$, and a small thresholds $\epsilon_K > 0$ and $\epsilon_Q > 0$.

while $\|\boldsymbol{Q}_{l+1} - \boldsymbol{Q}_l\| > \varepsilon_Q$ **do**
  if $j = 0$ and $\|\mathcal{K}_{j+1} - \mathcal{K}_j\| > \epsilon_K$ **then**
    Collect κ samples to construct $\boldsymbol{\Phi}$ and $\boldsymbol{\Xi}$.
    if rank$(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}) = \iota$ **then**
      Compute the BLS rule (16) and update control policy $\mathcal{K}_{j+1}$ using (17).
      Estimate $\boldsymbol{B}$ using (18).
      Set $j \leftarrow j + 1$
    else if $j \neq 0$ and $\|\mathcal{K}_{j+1} - \mathcal{K}_j\| > \epsilon_K$ **then**
      Solve DARE (7) and compute control gain $\mathcal{K}_{j+1}$ using (8)
      Set $j \leftarrow j + 1$
    else if $j \neq 0$ and $\|\mathcal{K}_{j+1} - \mathcal{K}_j\| \le \epsilon_K$ **then**
      Update kernel matrix $\mathcal{P}$ and control gain $\mathfrak{K}$ using (21) and (22).
      Compute $\widehat{\boldsymbol{Q}}$ using (23) and update weight matrix $\boldsymbol{Q}_{l+1}$ using (24).
      Set $l \leftarrow l + 1$ and $\|\mathcal{K}_{j+1} - \mathcal{K}_j\| > \varepsilon_K$.
    end if
  end if
end while

Proof: Assume that we have κ sample trajectories of the swarm such that rank$(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}) = \iota$. Then, in each IRL episode l, the estimated $\boldsymbol{Q}_{l+1}$ satisfies the following DARE

$$\boldsymbol{Q}_{l+1} = \mathcal{H}[\mathcal{P} - \boldsymbol{A}^\top \mathcal{P}\boldsymbol{A} + \boldsymbol{A}^\top \mathcal{P}\boldsymbol{B}(\boldsymbol{R} + \boldsymbol{B}^\top \mathcal{P}\boldsymbol{B})^{-1}\boldsymbol{B}^\top \mathcal{P}\boldsymbol{A}].$$

In addition, the weight matrix $\boldsymbol{Q}_{l+1}$ satisfies the following DARE in each RL episode $j + 1$

$$\begin{aligned}\boldsymbol{Q}_{l+1} =&\boldsymbol{P}_{j+1} - \boldsymbol{A}^\top \boldsymbol{P}_{j+1}\boldsymbol{A} \\ &+ \boldsymbol{A}^\top \boldsymbol{P}_{j+1}\boldsymbol{B}(\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}_{j+1}\boldsymbol{B})^{-1}\boldsymbol{B}^\top \boldsymbol{P}_{j+1}\boldsymbol{A}.\end{aligned}$$

Recall that $\mathcal{P} = \boldsymbol{P}_j - \gamma\frac{\partial E}{\partial \boldsymbol{P}_j}$ and is updated in the direction that $\boldsymbol{E}_K$ is minimized. Here, by forcing $\boldsymbol{Q}_{l+1}$ to have a specific structure enables that the gain error $\boldsymbol{E}_K$ is minimized, i.e., $\lim_{j\to\infty} \mathcal{K}_j = \boldsymbol{K}$. Then and without loss of generality, the DARE of the IRL is reduced to

$$\boldsymbol{Q}_{l+1} = \mathcal{H}[\boldsymbol{P}_j - \boldsymbol{A}^\top \boldsymbol{P}_j\boldsymbol{A} + \boldsymbol{A}^\top \boldsymbol{P}_j\boldsymbol{B}\boldsymbol{L}_j^{-1}\boldsymbol{B}^\top \boldsymbol{P}_j\boldsymbol{A}].$$

Equalling both the RL and IRL parts give

$$\begin{aligned}\boldsymbol{P}_{j+1} - &\boldsymbol{A}^\top \boldsymbol{P}_{j+1}\boldsymbol{A} + \boldsymbol{A}^\top \boldsymbol{P}_{j+1}\boldsymbol{B}\boldsymbol{L}_{j+1}^{-1}\boldsymbol{B}^\top \boldsymbol{P}_{j+1}\boldsymbol{A} \\ &= \mathcal{H}[\boldsymbol{P}_j - \boldsymbol{A}^\top \boldsymbol{P}_j\boldsymbol{A} + \boldsymbol{A}^\top \boldsymbol{P}_j\boldsymbol{B}\boldsymbol{L}_j^{-1}\boldsymbol{B}^\top \boldsymbol{P}_j\boldsymbol{A}].\end{aligned}$$

The above equality holds if and only if $\boldsymbol{P}_j = \boldsymbol{P}_{j+1}$. In addition, since we incorporate the operator $\mathcal{H}[\cdot]$ then the multiple solution problem of IRL models is avoided. In consequence, since the weight matrix $\boldsymbol{Q}_l$ has a specific structure then $\boldsymbol{P}_i$ converges to an unique kernel matrix that matches with the optimal kernel matrix $\boldsymbol{P}$, i.e., $\lim_{j\to\infty} \boldsymbol{P}_j = \boldsymbol{P}$ and hence,

this implies that the global reward weight $\boldsymbol{Q}_l$ converges to the exact $\boldsymbol{Q}$, i.e., $\lim_{l\to\infty}\boldsymbol{Q}_l=\boldsymbol{Q}$. This completes the proof. ■

Remark 2: In contrast to previous approaches using IRL, the proposed approach can uncover the exact reward function by forcing the reward weights to have a specific structure. Previous methods can effectively handle the IRL challenge for a single agent, however for multi-agents the approach becomes unstable due to the sparsity of the optimal control solution. Previous work [16] used reward function values as constraint in the IRL algorithm to uncover the exact reward function. However for multi-agent this assumption is prohibitive and faces the same sparsity challenge.

Remark 3: The computational complexity of the proposed PI-MAIRL is given by the off-policy model-free RL part with $\mathcal{O}(j\iota^2(\kappa+\iota))$. The BLS (16) *requires to compute ι parameters each j episode, which clearly increases as the number of agents W increases.*

D. Network Topology Inference

We can infer the network topology from the global weight matrix $\boldsymbol{Q}$. Due to the topology of the graph, we know that the off-diagonal elements of the Laplacian $\boldsymbol{\mathcal{L}}$ are either 0 or -1 such that for each $n\times n$ off-diagonal block matrix of $\boldsymbol{Q}$ we have either an $n\times n$ zero matrix or $-\boldsymbol{Q}_1$, that is,

$$\widehat{\mathcal{L}}_{ij}=\begin{cases}0 & \text{if} \quad \boldsymbol{Q}_{I,J}=\boldsymbol{0}_n,\\ 0 & \text{if} \quad I=J,\\ -1 & \text{if} \quad \text{otherwise},\end{cases} \tag{25}$$

where $I=(1:n)+n(i-1)$ and $J=(1:n)+n(j-1)$ for $i,j=1,...,W$. This produces a symmetric $n\times n$ matrix with diagonal of zeros. From matrix $\widehat{\boldsymbol{\mathcal{L}}}=[\widehat{\mathcal{L}}_{ij}]$ we can obtain both $\boldsymbol{Q}_1$, $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{Q}_0$. The weight matrix $\boldsymbol{Q}_1$ can be easily obtained by finding the indices i and j where $\widehat{\mathcal{L}}_{ij}=-1$, i.e., $\boldsymbol{Q}_1=-\widehat{\boldsymbol{\mathcal{L}}}_{I,J}$ with $I\neq J$. To obtain the Laplacian matrix, we first compute the diagonal terms as the sum of the elements of each row $i=1,...,W$ of $\widehat{\boldsymbol{\mathcal{L}}}$, i.e.,

$$\mathcal{D}_i=-\sum_{j=1}^{W}\widehat{\mathcal{L}}_{ij}.$$

Then, the Laplacian is given by

$$\boldsymbol{\mathcal{L}}=\widehat{\boldsymbol{\mathcal{L}}}+\boldsymbol{\mathcal{D}}, \tag{26}$$

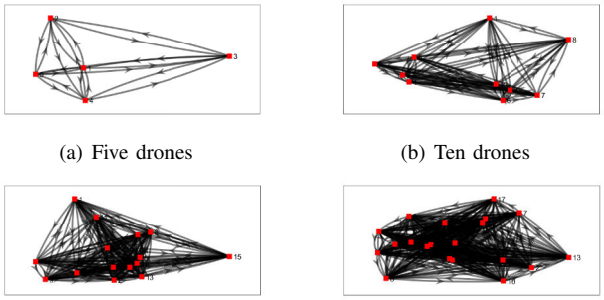
where $\boldsymbol{\mathcal{D}}=\text{diag}\{\mathcal{D}_1,...,\mathcal{D}_W\}$. Then $\boldsymbol{Q}_0$ can be computed by considering the first $n\times n$ block of $\boldsymbol{Q}$ as

$$\boldsymbol{Q}_0=\boldsymbol{Q}_{I,J}-\mathcal{D}_1\boldsymbol{Q}_1,\ I=1:n,\ J=1:n. \tag{27}$$

IV. Simulation Studies

In this section we consider a set of different drone swarms with physically decoupled dynamics (see Fig. 3). It is assumed that each drone communicates equally with the other drones. All computations are executed on an Intel(R) Core(TM) i9-10885H 2.40 GHz, 32-GB RAM, with MATLAB 2023b.

Each drone has the same dynamics with an undirected graph topology. We use a hovering continuous-time linear model and discretize it using a sample time $t_s=0.05$ seconds. The linear model for each drone is



(a) Five drones (b) Ten drones

(c) Fifteen drones (d) Twenty drones

Fig. 3. Network Topologies considered in this research

$$\boldsymbol{A}_0=\begin{bmatrix}\boldsymbol{I}_3 & t_s\boldsymbol{I}_3\\ \boldsymbol{0}_3 & \boldsymbol{I}_3\end{bmatrix},\quad \boldsymbol{B}_0=\begin{bmatrix}0.0123 & 0 & 0\\ 0 & -0.0123 & 0\\ 0 & 0 & 0.0027\\ 0.4905 & 0 & 0\\ 0 & -0.4905 & 0\\ 0 & 0 & 0.1071\end{bmatrix}.$$

The weight matrices of the cost or reward function are set to $\boldsymbol{Q}_0=\text{diag}\{1,1,1,5,5,5\}$, $\boldsymbol{Q}_1=\text{diag}\{1,1,1,0,0,0\}$, and $\boldsymbol{R}_0=\boldsymbol{I}_3$. We test different learning rates and choose the one that provides the best performance in terms of accuracy and time. We use an initial reward weight of $\boldsymbol{Q}_{l=0}=\boldsymbol{I}_N$. The final learning rate is $\gamma=50$.

A. Single drone

We first test the proposed approach using one single drone to provide numerical results of both the kernel matrix and reward weights. The optimal kernel matrix and control gain for the proposed expert weights

$$\boldsymbol{K}=\begin{bmatrix}0.5856 & 0 & 0 & 1.3543 & 0 & 0\\ 0 & -0.5856 & 0 & 0 & -1.3543 & 0\\ 0 & 0 & 0.878 & 0 & 0 & 2.162\end{bmatrix}$$

$$\boldsymbol{P}=\begin{bmatrix}46.2521 & 0 & 0 & 2.3252 & 0 & 0\\ 0 & 46.2521 & 0 & 0 & 2.3252 & 0\\ 0 & 0 & 49.2495 & 0 & 0 & 9.4067\\ 2.3252 & 0 & 0 & 7.8191 & 0 & 0\\ 0 & 2.3252 & 0 & 0 & 7.8191 & 0\\ 0 & 0 & 9.4067 & 0 & 0 & 25.4286\end{bmatrix}.$$

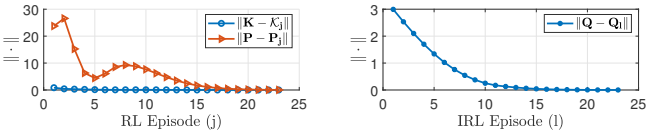
We stop the proposed algorithm when $\|\boldsymbol{Q}_{l+1}-\boldsymbol{Q}_l\|\leq 0.001$. Fig. 4 shows the convergence results of the proposed PI-IRL. The algorithm converges in $l=23$ and $j=30$ episodes to the following values

$$\boldsymbol{\mathcal{K}}_{39}=\begin{bmatrix}0.586 & 0 & 0 & 1.3543 & 0 & 0\\ 0 & -0.586 & 0 & 0 & -1.3543 & 0\\ 0 & 0 & 0.8777 & 0 & 0 & 2.1621\end{bmatrix}$$

$$\boldsymbol{P}_{39}=\begin{bmatrix}46.289 & 0 & 0 & 2.327 & 0 & 0\\ 0 & 46.289 & 0 & 0 & 2.327 & 0\\ 0 & 0 & 49.234 & 0 & 0 & 9.4036\\ 2.327 & 0 & 0 & 7.8191 & 0 & 0\\ 0 & 2.327 & 0 & 0 & 7.8191 & 0\\ 0 & 0 & 9.4036 & 0 & 0 & 25.4293\end{bmatrix}$$

$$\boldsymbol{Q}_{23}=\text{diag}([1.0009, 1.0009, 1.0009, 4.9999, 4.9999, 5.0004]).$$

The obtained results effectively demonstrates that the PI-IRL can uncover the reward function from the observed data. Here, we can accelerate convergence of the approach by increasing the stopping threshold, however this lead to degradation of the control policy.



(a) Model-free RL convergence (b) Model-based IRL convergence

Fig. 4. Convergence results for the single drone case

We tested the inferred reward function in a learner agent for imitation learning [44], [45]. Fig. 5 shows the evolution of the states of both the drone and the learner agent.

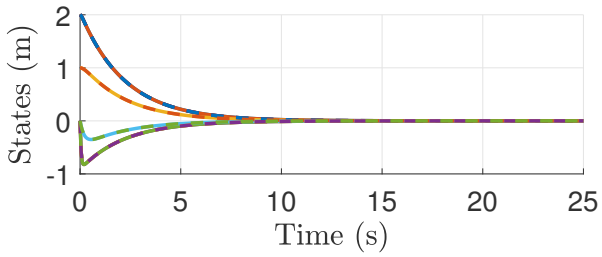


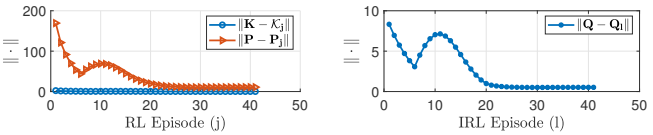
Fig. 5. Single drone case. Learned policy for imitation learning. Solid lines are the expert trajectories and the dotted lines are the learner trajectories.

Since the learned reward and policy are almost equivalent, then the learner agent achieves effectively the same expert trajectory profile.

Remark 4: The proposed approach is limited to the detection capabilities of current sensing technologies. This means that current sensors (e.g., radars, cameras) can only capture the data of certain number of drones, but they will struggle to capture the data of a large drone swarm. In this paper, we will assume that we can collect the trajectories of each drone effectively.

B. Ten Drones

Consider a drone swarm composed by ten drones. The convergence results of the proposed PI-MAIRL are shown in Fig. 6. Here, the RL and IRL algorithms converge in $j = 51$ (RL episodes where $\|\boldsymbol{\mathcal{K}}_{j+1}-\boldsymbol{\mathcal{K}}_j\| \leq \varepsilon_K$) and $l = 41$ episodes, respectively.



(a) Model-free RL convergence (b) Model-based IRL convergence

Fig. 6. Convergence results for the ten drones case

We tested the inferred reward function in a learner swarm of ten drones for imitation learning. Fig. 7 shows the evolution of the states of both the real and the learner drone swarm.

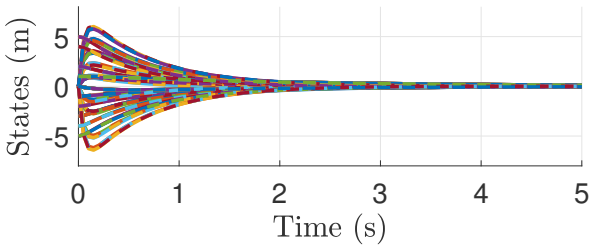
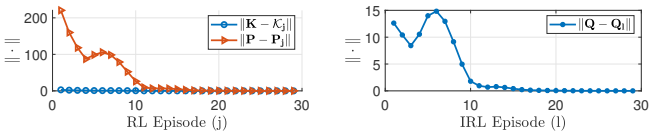


Fig. 7. Ten drones case. Learned policy for imitation learning. Solid lines are the expert trajectories and the dotted lines are the learner trajectories.

In this particular scenario we observed that the proposed learning rate is large in some directions of the optimization of $\boldsymbol{\mathcal{P}}$. Further work, will analyse the use of adaptive learning rates (similar to the ones used in RMSprop and Adam) to improve the kernel update in flat dimensions.

C. Fifteen Drones

Consider a drone swarm composed by fifteen drones. The convergence results of the proposed PI-MAIRL are shown in Fig. 8. The results shows convergence of the kernel matrix $\boldsymbol{P}_j$, control gain $\boldsymbol{\mathcal{K}}_j$, and the weight matrix $\boldsymbol{Q}_l$ similarly to the ten drones case. In this scenario, the collection of data for the construction of $\boldsymbol{\Phi}$ plays a critical issue. Here, the number of parameters to estimate grows to 9,180 which means that we need to have a rich and large dataset $\boldsymbol{\mathcal{X}}$ to construct a full rank matrix $\boldsymbol{\Phi}$, i.e., $\text{rank}(\boldsymbol{\Phi}) = 9,180$.



(a) Model-free RL convergence (b) Model-based IRL convergence

Fig. 8. Convergence results for the five drones case

In addition, the transformation from the raw data $\boldsymbol{\mathcal{X}}$ into the appropriate representation in $\boldsymbol{\Phi}$ requires a considerable large computation time. This is because we need to compute the Kronecker product between large vectors of dimensions N and M and do this operation at least ι times. This is usually not a problem in single-agent IRL algorithms since the data transformation is simple, but in the multi-agent case plays a critical problem. This means that the algorithm takes too much time in only transforming the data without computing the BLS update rule of the model-free RL algorithm. This is a critical issue in fully data-driven approaches which needs to process new data in the subsequent episodes. The proposed approach aims to reduce this issue by only computing one-single model-free RL episode and changed into a model-based RL approach that avoids the data transformation.

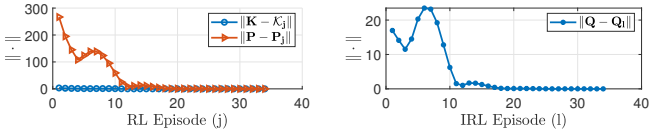
D. Twenty Drones

Consider a drone swarm composed by twenty drones. The convergence results of the proposed PI-MAIRL are shown in Fig. 8. This case is similar to the fifteen drones case, which

TABLE I
COMPARISON RESULTS ACROSS DIVERSE SWARMS

	$\|\boldsymbol{K}-\boldsymbol{\mathcal{K}}_j\|$		$\|\boldsymbol{P}-\boldsymbol{P}_j\|$		$\|\boldsymbol{Q}-\boldsymbol{Q}_l\|$		Time (seconds)	
Swarm	**pMAIRL**	**PI-MAIRL**	**pMAIRL**	**PI-MAIRL**	**pMAIRL**	**PI-MAIRL**	**pMAIRL**	**PI-MAIRL**
One drone	4.51×10^{-4}	0.0014	0.0369	0.0806	8.87×10^{-4}	0.0023	0.3930	0.1792
Two drones	0.0196	7.3686×10^{-4}	0.1876	0.0403	0.0084	0.0020	2.87	0.3142
Three drones	0.0331	6.9646×10^{-4}	0.4056	0.0380	0.0181	0.0022	13.4885	0.6280
Four drones	0.045	6.1138×10^{-4}	0.7649	0.0332	0.0345	0.0021	48.1067	1.2688
Five drones	0.0564	5.1313×10^{-4}	1.4487	0.0276	0.0651	0.0020	164.1314	3.0434
Ten drones	0.1563	3.1686×10^{-4}	10.8158	0.0160	0.5197	0.0016	15,135.4753	220.0121
Fifteen drones	0.4571	3.0327×10^{-4}	92.823	0.0137	1.0375	0.0018	60,785.8453	1,755.036
Twenty drones	1.1232	6.3553×10^{-5}	153.6587	0.0030	2.3498	2.8183×10^{-4}	115,872.3865	12,335.9679

demonstrate convergence of each element of the PI-MAIRL, but requires a large amount of data and its time consuming due to the data transformation. We need to collect 16,290 samples to construct matrix $\boldsymbol{\Phi}$, however the richness of the data plays a major role in large swarms, since the collection of linear independent samples become harder. In consequence, the algorithm will take more iterations to construct a full rank matrix $\boldsymbol{\Phi}$. This issue is solved by adding more frequencies into the probing noise signal, however since the data transformation process is too slow then it is difficult to verify if the probing noise effectively excites all the modes of the multi-agent platform.



(a) Model-free RL convergence (b) Model-based IRL convergence

Fig. 9. Convergence results for the five drones case

E. Comparative Studies

The computation time issue in distributed autonomous systems was addressed by [46]. Here, a dynamic mode decomposition approach is applied to consider only the elements of $\boldsymbol{\Phi}$ associated to the largest singular values which contains the most important information of the swarm dynamics. This algorithm reduces the computational complexity of the BLS rule of model-free RL algorithms, however it still suffers of the computation time issue of the data transformation. Furthermore, the low-dimensional representation of $\boldsymbol{\Phi}$ cannot be applied directly for reward inference purposes using IRL algorithms. This is because the new system is under other coordinates that we do not have direct measurements. IRL has been effectively applied in multi-player games [47], however the nature of the distributed problem makes IRL algorithms to be computationally unsuitable due to the calculation of two different BLS update rules and two different data transformations. In view of this and to conduct a fair comparison, we use the model-free RL algorithm proposed in [46] combined with the proposed model-based IRL approach, that is, we do not consider the model-based RL element. This will allow to show the issues of data-driven IRL approaches under distributed autonomous systems. We denote this algorithm as partial data-based MAIRL (pMAIRL).

We use the spectral error norm results and computation time as metrics to compare the pMAIRL and the proposed PI-MAIRL. Table I summarizes the obtained results across different swarm cases. The comparison results show that as the number of drones in the swarm increases then the spectral error norm increases for the pMAIRL. This is expected since the dimensionality of the global system increases by W^2. For example, for the ten drones case, the spectral error norm of the kernel matrix is 10.8158, however the number of elements in $\boldsymbol{P}$ is $3,600$ such that small deviations in each term are accumulated and produce a larger spectral error norm. This error can be attenuated by reducing the thresholds ε_Q and ε_K in Algorithm 1. On the other hand, the proposed PI-MAIRL decreases the spectral error norm as the number of drones increases. Here, by transforming the problem into a model-based RL approach, it adds constraints into the learning phase of the RL algorithm to compute the gain $\boldsymbol{\mathcal{K}}_{j+1}$ and the kernel matrix $\boldsymbol{P}_j$. This reduces the approximation error of data-driven approaches and avoid multiple-solutions even for large drone swarms.

The computation time is highly decreased by incorporating the physics-informed element. It is clear that the pMAIRL takes more time to converge to a near optimal solution when the number of drones increases. Here, the data transformation conducted by the model-free RL algorithm takes more time compared with the computation of the BLS update rule (16). Notice that the pMAIRL uses a model-based IRL algorithm to reduce the computational resources. The IRL loop computes in a simple step the next weight matrix $\boldsymbol{Q}_{l+1}$ without collecting data. If the physics-informed element is not given, then the IRL will need to be designed as a model-free architecture that faces the same computational issues compared with the model-free RL architecture, i.e., it will increase almost twice the computation time. This is an open challenge in the current state-of-the-art that we will address as future work.

The PI-MAIRL faces the same data transformation issue, however it notably reduces the computation time by switching from model-free RL into a model-based RL. This avoids the data-transformation step of model-free RL and IRL approaches and constraints the learning manifold of the overall algorithm for accurate inference results.

F. Large Swarms

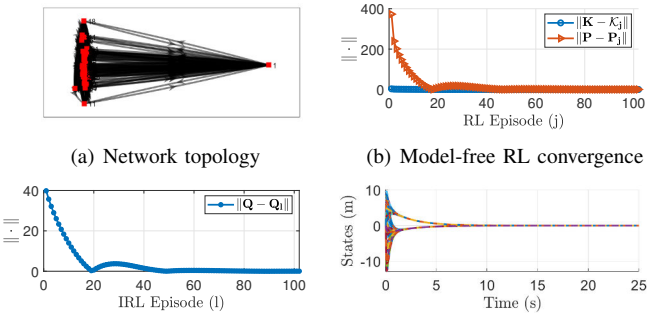
In this paper, we consider that each drone has 6 states and 3 inputs. This increases the computational demands for big

swarms. For example, if we consider 50 drones then we need to estimate 101,475 parameters and construct a square matrix of linear independent samples $\mathbf{\Phi} \in \mathbb{R}^{101,475 \times 101,475}$ which is prohibitive due to computational constraints. To reduce this problem we can exploit the decoupled nature of the drone linear model and divide the dynamics into 3 two-dimensional subsystems.

For instance, consider just the position in the x-axis whose dynamics is

$$\boldsymbol{A}_0 = \begin{bmatrix} 1 & 0.05 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{B}_0 = \begin{bmatrix} 0.0123 \\ 0.4905 \end{bmatrix}.$$

In this case, for 50 drones we need to estimate just 11,325 parameters which is just 11.16% of the full state system. We applied the proposed approach using this sub-system and consider 50 drones. For this case, we use $\gamma = 10$. The results are shown in Fig. 10.



(a) Network topology

(b) Model-free RL convergence

(c) Model-based IRL convergence

(d) Learned policy for imitation learning

Fig. 10. Results for the 50 drones case

The results converge to the following values $\|\boldsymbol{K} - \boldsymbol{\mathcal{K}}_j\| = 0.0013$, $\|\boldsymbol{P} - \boldsymbol{P}_j\| = 0.2201$, and $\|\boldsymbol{Q} - \boldsymbol{Q}_l\| = 0.0421$. These results effectively demonstrate that the proposed approach can deal for a large number of drones. However, the computational time notably increases as the number of drones increases. For this case, the algorithm takes 3997.48 seconds (approximately 1 hour 11 min) to converge. This means that when the full state is considered then the computation time grows up to more than 10 hours. This is more sensitive in the model-free version where the computational time increases to more than a day for large swarms under two-dimensional drone models. Further work will study how to exploit the sparse dynamics of the distributed system to reduce the computational demands of the proposed approach.

V. Conclusions

This paper reports a physics-informed multi-agent inverse reinforcement learning (PI-MAIRL) to uncover the reward goals of drone swarms. The drone swarm is modelled as a global linear system and the communication between each drone is reflected in the reward function of the global optimal control problem. The proposed architecture uses a hybrid model-free/model-based reinforcement learning algorithm to iteratively find the control gain and kernel matrix that solves a discrete algebraic Riccati equation. Prior knowledge of the state-transition matrix of a single drone is used as a physics-informed structure that enables the construction of a model-based policy error inverse reinforcement learning algorithm that notably reduces the computation time of the overall architecture. The network topology can be inferred from the final reward weights. Simulations studies are conducted under different drone swarm to show the benefits of the proposed methodology and its disadvantages. The results show that the proposed approach can effectively uncover the reward goals for different drone swarms, but the computation time highly increases as more agents are considered.

Future work will consider three main research areas: 1) reduce the computational complexity of the proposed approach due to the sparse topology of the kernel matrix, 2) consider multi-agent systems that are connected both in communication and physically, and 3) analyse how the proposed methodology can be expanded into reward functions with different structures.

References

[1] C. Gruffeille, A. Perrusquía, A. Tsourdos, and W. Guo, "Disaster area coverage optimisation using reinforcement learning," in *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024.

[2] M. El Debeiki, S. Al-Rubaye, A. Perrusquía, C. Conrad, and J. A. Flores-Campos, "An advanced path planning and uav relay system: Enhancing connectivity in rural environments," *Future Internet*, vol. 16, no. 3, p. 89, 2024.

[3] 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.

[4] E. Bildik, A. Perrusquía, A. Tsourdos, and G. Inalham, "Swarm decoys deployment for missile deceive using multi-agent reinforcement learning," in *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024.

[5] G. Lykou, D. Moustakas, and D. Gritzalis, "Defending airports from uas: A survey on cyber-attacks and counter-drone sensing technologies," *Sensors*, vol. 20, no. 12, p. 3537, 2020.

[6] A. M. Ali, A. Perrusquía, W. Guo, and A. Tsourdos, "Flight plan optimisation of unmanned aerial vehicles with minimised radar observability using action shaping proximal policy optimisation," *Drones*, vol. 8, no. 10, p. 546, 2024.

[7] P. Wendt, A. Voltes-Dorta, and P. Suau-Sanchez, "Estimating the costs for the airport operator and airlines of a drone-related shutdown: an application to frankfurt international airport," *Journal of Transportation Security*, vol. 13, pp. 93–116, 2020.

[8] J.-P. Yaacoub, H. Noura, O. Salman, and A. Chehab, "Security analysis of drones systems: Attacks, limitations, and recommendations," *Internet of Things*, vol. 11, p. 100218, 2020.

[9] A. Singh, G. Triulzi, and C. L. Magee, "Technological improvement rate predictions for all technologies: Use of patent data and an extended domain description," *Research Policy*, vol. 50, no. 9, p. 104294, 2021.

[10] I. Guvenc, F. Koohifar, S. Singh, M. L. Sichitiu, and D. Matolak, "Detection, tracking, and interdiction for amateur drones," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 75–81, 2018.

[11] J. Liang, B. I. Ahmad, M. Jahangir, and S. Godsill, "Detection of malicious intent in non-cooperative drone surveillance," in *2021 Sensor Signal Processing for Defence Conference (SSPD)*. IEEE, 2021, pp. 1–5.

[12] Q. Fu, X. Liang, J. Zhang, and X. Fan, "Intent inference based trajectory prediction and smooth for uas in low-altitude airspace with geofence," *CMC-COMPUTERS MATERIALS & CONTINUA*, vol. 63, no. 1, pp. 417–444, 2020.

[13] G. He, X. Li, Y. Lv, B. Gao, and H. Chen, "Probabilistic intention prediction and trajectory generation based on dynamic bayesian networks," in *2019 Chinese Automation Congress (CAC)*. IEEE, 2019, pp. 2646–2651.

[14] G. Singh, A. Perrusquía, and W. Guo, "A two-stages unsupervised/ supervised statistical learning approach for drone behaviour prediction," in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2023, pp. 1–6.

[15] Y. Cho, J. Kim, and J. Kim, "Intent inference of ship collision avoidance behavior under maritime traffic rules," *Ieee Access*, vol. 9, pp. 5598–5608, 2021.

[16] A. Perrusquía, W. Guo, B. Fraser, and Z. Wei, "Uncovering drone intentions using control physics informed machine learning," *Communications Engineering*, vol. 3, no. 1, p. 36, 2024.

[17] K. Mangalam, Y. An, H. Girase, and J. Malik, "From goals, waypoints & paths to long term human trajectory forecasting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 233–15 242.

[18] J. Liang, B. I. Ahmad, and S. Godsill, "Simultaneous intent prediction and state estimation using an intent-driven intrinsic coordinate model," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[19] H. Zhang, Y. Yan, S. Li, Y. Hu, and H. Liu, "UAV behavior-intention estimation method based on 4-D flight-trajectory prediction," *Sustainability*, vol. 13, no. 22, p. 12528, 2021.

[20] T. Su, Y. Meng, and Y. Xu, "Pedestrian trajectory prediction via spatial interaction transformer network," in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*. IEEE, 2021, pp. 154–159.

[21] 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 *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. IEEE, 2023, pp. 1–6.

[22] R. Mousavifard, K. Alipour, M. A. Najafqolian, and P. Zarafshan, "Formation control of multi-quadrotors based on deep q-learning," in *2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM)*. IEEE, 2022, pp. 172–177.

[23] H. Yau, C. Russell, and S. Hadfield, "What did you think would happen? explaining agent behaviour through intended outcomes," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[24] W. Yu and A. Perrusquia, *Human-Robot Interaction Control Using Reinforcement Learning*. John Wiley & Sons, 2021.

[25] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 13 178–13 188.

[26] C. Legaard, T. Schranz, G. Schweiger, J. Drgoňa, B. Falay, C. Gomes, A. Iosifidis, M. Abkar, and P. Larsen, "Constructing neural network based models for simulating dynamical systems," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–34, 2023.

[27] A. Perrusquía and W. Guo, "Reservoir computing for drone trajectory intent prediction: A physics informed approach," *IEEE Transactions on Cybernetics*, 2024.

[28] A. Perrusquía and W. Guo, "A closed-loop output error approach for physics-informed trajectory inference using online data," *IEEE Transactions on Cybernetics*, vol. 53, no. 3, pp. 1379–1391, 2022.

[29] 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.

[30] 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.

[31] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[32] W. Xue, B. Lian, J. Fan, T. Chai, and F. L. Lewis, "Inverse reinforcement learning for trajectory imitation using static output feedback control," *IEEE Transactions on Cybernetics*, 2023.

[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 Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 570–10 581, 2021.

[34] J. Sun, L. Yu, P. Dong, B. Lu, and B. Zhou, "Adversarial inverse reinforcement learning with self-attention dynamics model," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1880–1886, 2021.

[35] A. Y. Ng, S. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[36] B. Lian, W. Xue, Y. Xie, F. L. Lewis, and A. Davoudi, "Off-policy inverse q-learning for discrete-time antagonistic unknown systems," *Automatica*, vol. 155, p. 111171, 2023.

[37] 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.

[38] B. Lian, Y. Kartal, F. L. Lewis, D. G. Mikulski, G. R. Hudas, Y. Wan, and A. Davoudi, "Anomaly detection and correction of optimizing autonomous systems with inverse reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 53, no. 7, pp. 4555–4566, 2023.

[39] K. G. Vamvoudakis, H. Modares, B. Kiumarsi, and F. L. Lewis, "Game theory-based control system algorithms with real-time reinforcement learning: How to solve multiplayer games online," *IEEE Control Systems Magazine*, vol. 37, no. 1, pp. 33–52, 2017.

[40] A. Perrusquía and W. Guo, "Drone's objective inference using policy error inverse reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[41] 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 transactions on neural networks and learning systems*, vol. 30, no. 12, pp. 3735–3747, 2019.

[42] P. J. Baddoo, B. Herrmann, B. J. McKeon, J. Nathan Kutz, and S. L. Brunton, "Physics-informed dynamic mode decomposition," *Proceedings of the Royal Society A*, vol. 479, no. 2271, p. 20220576, 2023.

[43] M. Zou, A. Perrusquía, and W. Guo, "Explaining data-driven control in autonomous systems: a reinforcement learning case study," in *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2024, pp. 73–78.

[44] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[45] A. Perrusquía and W. Guo, "Optimal control of nonlinear systems using experience inference human-behavior learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 1, pp. 90–102, 2023.

[46] V. S. Donge, B. Lian, F. L. Lewis, and A. Davoudi, "Accelerated reinforcement learning via dynamic mode decomposition," *IEEE Transactions on Control of Network Systems*, vol. 10, no. 4, pp. 2022–2034, 2023.

[47] B. Lian, V. S. Donge, F. L. Lewis, T. Chai, and A. Davoudi, "Data-driven inverse reinforcement learning control for linear multiplayer games," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

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. His main research of interest focuses on reinforcement learning, inverse reinforcement learning, data-driven control nonlinear control, machine learning and system identification.

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.

Uncovering reward goals in distributed drone swarms using physics-informed multiagent inverse reinforcement learning

Perrusquía, Adolfo

2025-01-01

Perrusquía A, Guo W. (2025) Uncovering reward goals in distributed drone swarms using physics-informed multiagent inverse reinforcement learning. IEEE Transactions on Cybernetics, Volume 55, Issue 1, January 2025, pp. 14-23

https://doi.org/10.1109/tcyb.2024.3489967