Learning Adversarial Policies for Swarm Leader Identification using a Probing Agent

Stergios E. Bachoumas and Panagiotis Artemiadis* *IEEE Senior Member*

***Abstract*— This study introduces a novel approach to swarm leader identification (SLI) in multi-agent robot systems by employing a physical adversary interacting with the swarm in the same environment. We develop a new simulation environment to study the SLI problem and train an adversary, which we term the prober, to solve the SLI problem using forceful interactions with the swarm as its guiding information source. The prober's policy is modeled using the simplified structure state space sequence (S5) model and trained with the Proximal Policy Optimization (PPO) algorithm. The prober only has access to the information on the relative positions of the other agents. We evaluate our approach through extensive simulations using two performance metrics and validate the sim-to-real transfer through robot experiments. Results on evaluating the performance in 10,000 different testing scenarios demonstrate that our method finds the leader's identity in the vast majority (95.7%) of the cases, regardless of the initial leader selection during training. The proposed system represents the first instance of learning-based automatic identification of leader agents in a swarm. This capability is crucial for enabling efficient and robust human-swarm interaction, understanding artificial swarm behaviors, and analyzing latent behaviors in biological swarms in nature.**

I. Introduction

Human-swarm interaction (HSI) is the growing research field investigating the complexities of the cooperation between humans and robotic swarms [1]. Research in the HSI field seeks to combine the strengths of swarm systems, such as scalability and robustness to mission failure, with human strengths, for example, intuition and adaptability to unknown situations.

In swarm robotics literature, it is often emphasized that the primary advantages of swarm systems stem from the large number of robots used and their typically low cost [1]. However, these same reasons pose a significant challenge in the HSI setting, particularly when a human operator tries to control the swarm and maintain sufficient situational awareness (SA) to complete the task. The swarm exhibits complex global behaviors as a system, such as flocking, which are often difficult for humans to interpret. Previous studies have shown that most robot-assisted tasks fail due to poor SA when the human operator is overwhelmed [2].

To address these complexities, the literature suggests that the most effective control strategy for swarms is the leader-follower paradigm [1]. However, in environments where the swarm coexists with potential adversaries or threats, the mission's success hinges upon the ability of the swarm to conceal the identity of its leader. We argue that in order to understand leader-based swarm systems, a systematic study of the swarm leader identification (SLI) problem is needed.

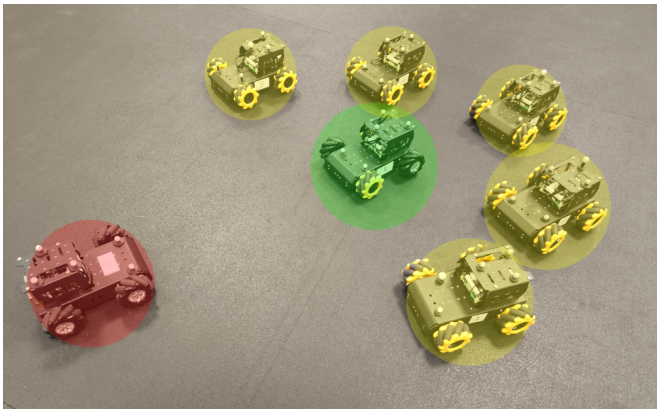

Fig. 1: A swarm of agents (yellow circles), led by the designated leader (green circle), advances collectively toward a common goal, while an adversary-prober (red circle) maneuvers within the swarm, strategically engaging with its members in an effort to pinpoint the leader's identity.

Previous work targeting the SLI problem has focused on observation-based models where the adversary is an artificial entity. For example, in [3], the authors used a DBSCAN-based probabilistic approach to identify the leader of the swarm. In [4], a Genetic Algorithm (GA) was designed to optimize the model for private flocking, i.e., flocking where the swarm is concealing the identity of the leader from a fictitious adversarial discriminator. Finally, in [5], the authors used a Graph Neural Network (GNN) to model a swarm that tries to conceal its leader from an artificial adversary, which was modeled as a Long short-term memory (LSTM) neural network.

All approaches above address the problem using full observability of the environment and without consideration for physical adversaries that interact with the swarm. In this work, we aim to address the SLI problem in situations where an adversary is attempting to identify the hidden leader of the swarm, and the environment is only partially observable, as illustrated in Fig. 1. To identify the leader, the adversary must interact with the swarm and use these interactions as its guiding information source to solve the problem. Specifically, we employ a Partially Observable Markov Decision Process (POMDP) framework to lead us to the identification of the leader. Unlike previous studies that focused on de-

*This material is based upon work supported by the National Science Foundation under Grant No. #2014264, as well as by the U.S. Air Force Office of Scientific Research (AFOSR) award FA9550-18-1-0221.

Stergios E. Bachoumas and Panagiotis Artemiadis are with the Mechanical Engineering Department, at the University of Delaware, Newark, DE 19716, USA. stevbach@udel.edu, partem@udel.edu

*Corresponding author

veloping policies using Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks [6], and more recently, transformer-based models [7], this work uses a variant of the Structured State Space Sequence (S4) model [8]. Specifically, we employ the Simplified Structured State Space Sequence (S5) model [9], which is particularly well-suited for addressing long-time horizon problems [10].

The proposed framework introduces an adversarial agent trained to interact with a swarm and identify its leader. We do not make any assumptions about the swarm's flocking behavior or dynamics. The method is evaluated across multiple simulated environments and through a real-time implementation with a robot swarm. This system represents the first instance of learning-based automatic identification of leader agents in a swarm, a capability that is crucial for enabling efficient and robust human-swarm interaction, understanding artificial swarm behaviors, and analyzing latent behaviors in biological swarms in nature.

The remainder of the paper is organized as follows. In Section II, we cover the essential preliminaries that lay the foundation for our work. Section III details the problem at hand and outlines our methodology for addressing it. In Sections IV and V, we present the results from both extensive simulations and a real-world robot experiment, demonstrating the effectiveness of our approach. Finally, in Section VI, we conclude with a discussion of our findings and suggest directions for future research.

II. Preliminaries

A. Swarm Flocking with Leaders

We use the term *swarm* to describe a collection of $N \in \mathbb{N}_{\geq 2}$ agents that share a common goal, where $\mathbb{N}_{\geq 2} \coloneqq \{2, 3, \dots\}$. Typically, members of a swarm interact using predefined behaviors that are locally simple but can lead to globally complex outcomes, such as flocking [11]. These behaviors emerge from force-based interactions that can be attractive or repulsive. In leader-based flocking, a subset of $N_L < N$ agents within the swarm is designated as leaders, who exert a stronger influence in the interactions described. These leaders are used to guide the swarm as a whole to a target location with increased efficiency and decreased fragmentation [11].

B. Swarm Leader Identification - SLI

Formally, consider a swarm of $N \in \mathbb{N}_{\geq 2}$ agents engaged in a task. As the task progresses over time $t \in [t_0, t_f]$, where t_0 and t_f represent the initial and final time instances, respectively, our goal is to identify the leader agents $N_L < N$. Unlike previous research on leader identification, our approach aims to use the interactions between an adversary and the swarm as a way to identify the leader. Therefore, this adversary, henceforth referred to as the *prober*, will be controlled in a way to identify the leader of the swarm by forcefully interacting with its agents. We define this variant of SLI problem the Interaction-based Swarm Leader Identification problem (iSLI).

C. Partially Observable Markov Decision Process - POMDP

In this work, we use the Partially Observable Markov Decision Process (POMDP) framework. Formally, a POMDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, O, R, P, \gamma \rangle$. Here, $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{Z}$ is the set of observations, $O : \mathcal{S} \to \mathcal{Z}$ is a function that gives an observation from a state, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that maps from a given state and action to a real value, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ defines the distribution of next-state transitions given a state and action, and γ is the discount factor. The learning agent's objective is to find the optimal policy π^* (a function that maps from a given observation to a distribution over actions) that maximizes the expected discounted sum of rewards, also called the return.

D. Sequential Deep Reinforcement Learning

Deep reinforcement learning combines deep neural networks with reinforcement learning principles to solve complex tasks by training an agent in environments with large state-action spaces. In the POMDP framework, such as our problem, a sequential approach is necessary. This is because, in general, the optimal policy π^* is a function of the complete history of observations $o_{0:t}$ and actions $a_{0:t}$, due to the partial observability of the environment.

We design the policy of the prober using the S5 architecture and train it using the Proximal Policy Optimization (PPO) algorithm [12]. PPO is a policy gradient algorithm that works in two phases, the rollout phase and the learning phase [10], [13]. More details about the approach can be found in [12].

III. Methodology

In this Section, we discuss the methodology used to train the prober and present a description of the designed environment, comprising the agents' dynamics and the model used for the prober's interaction with the members of the swarm.

A. Agent Dynamics

The prober and the N members of the swarm each follow the dynamics of a double integrator model shown below:

$$\dot{\mathbf{p}}_i(t) = \mathbf{v}_i(t) \tag{1}$$
$$\dot{\mathbf{v}}_i(t) = \mathbf{u}_i(t, \mathbf{P}(t), \mathbf{V}(t))$$
$$\dot{\mathbf{p}}_p(t) = \mathbf{v}_p(t) \tag{2}$$
$$\dot{\mathbf{v}}_p(t) = \mathbf{u}_p(t, \mathbf{P}(t))$$

Let $\mathbf{p}_i(t) = [x_i, y_i]^\top \in \mathbb{R}^2$, $\mathbf{v}_i(t) = [\dot{x}_i, \dot{y}_i]^\top \in \mathbb{R}^2$ and $\mathbf{u}_i(t) = [\ddot{x}_i, \ddot{y}_i]^\top \in \mathbb{R}^2$ for $i = 1, 2, \dots, N$ be the position, velocity and acceleration vectors of the i^{th} swarm agent, respectively. Likewise, let $\mathbf{p}_p(t) = [x_p, y_p]^\top \in \mathbb{R}^2$, $\mathbf{v}_p(t) = [\dot{x}_p, \dot{y}_p]^\top \in \mathbb{R}^2$ and $\mathbf{u}_p(t) = [\ddot{x}_p, \ddot{y}_p]^\top \in \mathbb{R}^2$ be the position, velocity and acceleration vectors of the prober, respectively. Time is denoted by $t \in \mathbb{R}_{\geq 0}$. We define two arrays: $\mathbf{P}(t) \coloneqq [\mathbf{p}_1, \dots, \mathbf{p}_N, \mathbf{p}_p]^\top \in \mathbb{R}^{(N+1)\times 2}$ is a concatenation of all position, and $\mathbf{V}(t) \coloneqq [\mathbf{v}_1, \dots, \mathbf{v}_N, \mathbf{v}_p]^\top \in \mathbb{R}^{(N+1)\times 2}$ is a concatenation of all velocity vectors.

B. Flocking Algorithm

There is a plethora of algorithms in the literature to model the flocking behavior of a swarm. In the seminal paper [11], the author designs an algorithm for free-space flocking using virtual agents. These are called γ-agents. They stably lead the rest of the swarm, called α-agents, to a certain goal location. The algorithm provably ensures a collision-free and fragmentation-free flocking behavior.

For our work, we modify this model to be a leader-based swarm by doing the following modifications: First, we introduce a single physical leader agent in the swarm to play the role of the γ-agent. Then, we assume that this leader is the only agent that has knowledge of the task and can be directly controlled. Hereinafter, we will also use the name α-agents to refer to the followers.

The task of the leader l is to guide the swarm to a certain goal location G while staying close to the α-agents. The forces exerted on the agents and the leader are defined as:

$$\mathbf{u}_i(\mathbf{P},\mathbf{V}) = w_\alpha \mathbf{u}_i^\alpha(\mathbf{P},\mathbf{V}) + w_l \mathbf{u}_i^l(\mathbf{P},\mathbf{V}) + \mathbf{u}_i^p(\mathbf{p}_i,\mathbf{p}_p) \quad (3)$$

$$\mathbf{u}_l(\mathbf{P},\mathbf{V}) = w_\alpha \mathbf{u}_l^\alpha(\mathbf{P},\mathbf{V}) + \mathbf{u}_l(\mathbf{P},\mathbf{p}_G) + \mathbf{u}_l^p(\mathbf{p}_l,\mathbf{p}_p) \quad (4)$$

In the first equation, $\mathbf{u}_i(\mathbf{P},\mathbf{V})$ represents the total force acting on the i^{th} α-agent. This force is composed of three influence terms: $w_\alpha \mathbf{u}_i^\alpha(\mathbf{P},\mathbf{V})$, which is exerted from other α-agents with weight $w_\alpha \in \mathbb{R}$; $\mathbf{u}_i^l(\mathbf{P},\mathbf{V})$, which is exerted from the leader with weight $w_l \in \mathbb{R}$; and $\mathbf{u}_i^p(\mathbf{p}_i,\mathbf{p}_p)$, which is exerted from the prober on the i^{th} agent. In the second equation, $\mathbf{u}_l(\mathbf{P},\mathbf{V})$ denotes the total force acting on the leader. This force consists of three components: $\mathbf{u}_l^\alpha(\mathbf{P},\mathbf{V})$, which is exerted from the α-agents; $\mathbf{u}_l(\mathbf{P},\mathbf{p}_G)$, which is the control input to the leader; and $\mathbf{u}_l^p(\mathbf{p}_l,\mathbf{p}_p)$, which is exerted from the prober on the leader. The two weights $w_\alpha, w_l \in \mathbb{R}$ can be used to change the behavior of the α-agents and are kept constant during a single instance of flocking.

The input $\mathbf{u}_l(\mathbf{P},\mathbf{p}_G)$ to the leader can be realized with different control schemes. In this work we employ the following simple control law:

$$\begin{aligned}\mathbf{u}_l(\mathbf{P},\mathbf{p}_G) = K_{lp}(\mathbf{p}_G - \mathbf{p}_l) + K_{li}\int_0^t (\mathbf{p}_G - \mathbf{p}_l(\tau))d\tau \\ + K_{l\alpha}(\bar{\mathbf{p}}_\alpha - \mathbf{p}_l)\end{aligned} \quad (5)$$

where K_{lp}, K_{li}, and $K_{l\alpha} \in \mathbb{R}$ are the gains of the leader controller chosen so that the leader drives the swarm to the goal while staying close to the average position of the swarm $\bar{\mathbf{p}}_\alpha = \frac{1}{N}\sum_{j=1}^N \mathbf{p}_j$.

C. Interaction-based Swarm Leader Identification - iSLI

1) iSLI Definition: In the Interaction-based Swarm Leader Identification (iSLI) problem, the prober has to identify the leader of the swarm using forceful interactions as its guiding information source. To interact with the swarm, the prober must first approach it and strategically place itself close to the leader agent so that it interacts with it more than the others. We hypothesize that interactions of the prober with the leader are going to be more informative when compared to interactions with the α-agents. This hypothesis is backed up by the fact that the leader plays a crucial role in the formation of the swarm, and all α-agents are influenced by the leader more than every other α-agents.

From the point of view of the prober, the iSLI problem is a decision-making problem with partial information about the environment since the identity of the leader and the dynamics of the swarm are completely unknown to the prober. Thus, the prober must learn a policy π_p, i.e., a way to act in the environment, so that it maximizes its interactions with the leader of the swarm.

2) iSLI as POMDP: To this end, we formulate the iSLI problem as a Partially Observable Markov Decision Process (POMDP) for the prober and design an environment to study the iSLI problem based on the Gym paradigm [14]. The environment is a two-dimensional arena where all entities are constricted to move inside a box $\mathcal{B} = [\underline{b}, \bar{b}]$, where $\underline{b}, \bar{b} \in \mathbb{R}$ are the lower and upper bounds of the two-dimensional box. In the center of the box we place the world frame $\mathcal{W}$. Time $t \in [t_0, t_f]$ is measured at a fixed time-rate Δt such that $t_{k+1} = t_k + \Delta t$ and $t_0, t_f \in \mathbb{N}$ are the initial and final time. We define the state of the environment $s_{t_k} = \{l, \mathbf{P}^{\mathcal{W}}(t_k), \mathbf{V}^{\mathcal{W}}(t_k)\}$ as the tuple containing the identity of the leader l and the arrays of positions $\mathbf{P}^{\mathcal{W}}(t_k) \in \mathbb{R}^{(N+1)\times 2}$ and velocities $\mathbf{V}^{\mathcal{W}}(t_k) \in \mathbb{R}^{(N+1)\times 2}$ of all entities expressed in the world frame $\mathcal{W}$ at time $t_k \in [t_0, t_f]$. The environment has 4 initial states that each has 25% probability of being selected. At each initial state, the swarm spawns in one of the four corners of the world, and the prober always spawns near the middle of the world.

We place a coordinate frame $\mathcal{P}$ on the prober and define the observation of the environment as $o_{t_k} = O(s_{t_k}) = \left[\mathbf{p}_1^{\mathcal{P}}(t_k), \mathbf{p}_2^{\mathcal{P}}(t_k), \ldots, \mathbf{p}_N^{\mathcal{P}}(t_k), \mathbf{p}_p^{\mathcal{W}}(t_k)\right]$, which is the concatenation of the positions of each swarm agent in the prober frame $\mathcal{P}$ and the position of the prober in the world frame $\mathcal{W}$ at time $t_k \in [t_0, t_f]$.

The action space is defined as the set of accelerations in $X^{\mathcal{W}}$ and $Y^{\mathcal{W}}$ directions of the world frame $\mathcal{W}$ ranging from $-a$ to a discretized with a step of Δa, i.e. the action at any time is $a_{t_k} \in \{-a, \ldots, a\}^2$. For this work, we use $a = 50$ and $\Delta a = 10$. This discretization of the input achieves satisfactory results, and when compared to a continuous acceleration profile, it more closely resembles a zero-order-hold (ZOH) implementation on a real robot.

For a swarm of N agents, and during a single episode, the prober stores its interactions with each swarm agent in the interaction vector $\mathbf{q}_{t_0:t_k} = [q^1_{t_0:t_k}, q^2_{t_0:t_k}, \cdots q^N_{t_0:t_k}] \in \mathbb{N}^N$, where $q^i_{t_0:t_k}$ is the total number of interactions that the prober had with the i^{th} swarm agent up to time instance t_k. The interactions are zeroed when the environment restarts, i.e., $\mathbf{q}_{t_0} = \mathbf{0}_N$, where $\mathbf{0}_N$ is the integer N-dimensional zero vector. We hypothesize that maximizing the number of interactions of the prober with the leader, i.e., $q^l_{t_0:t_k} = \max \mathbf{q}_{t_0:t_k}$, will eventually lead to successful leader identification.

For this work, we employ a distance-based method for counting valid interactions, as shown below:

$$q^i(t_k) = \begin{cases} 1 & \text{if } \|\mathbf{x}_p(t_k) - \mathbf{x}_i(t_k)\| \le R \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

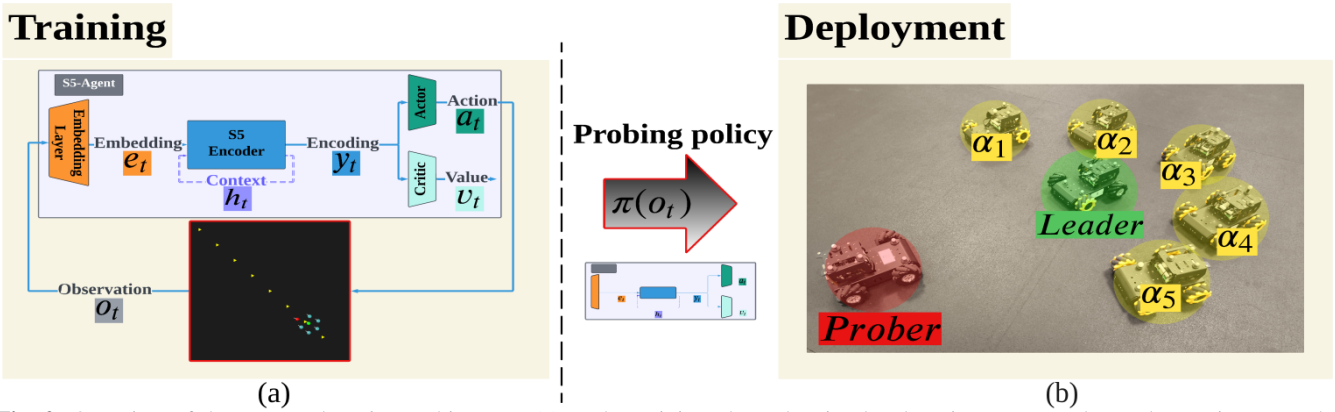

Fig. 2: Overview of the proposed *prober* architecture. (a) In the training phase the simulated environment sends an observation o_t to the Embedding layer, which produces an embedded representation e_t. The S5 Encoder processes this, maintaining internal context h_t and generating an encoding y_t. This encoding informs both the Actor, which outputs an action a_t, and the Critic, which estimates the state value v_t. Finally, action a_t is used to change the state of the Environment, and the process repeats. (b) After training we deploy the probing policy $\pi(o_t)$ to the prober robot (red circle) that is tasked with solving the iSLI problem in the real world, i.e., identifying the leader (green circle).

which essentially means that a valid interaction with an agent at a specific time instance t_k occurs if the distance of the prober to that agent is less than a predefined radius of interaction denoted as R. Using equation (5), interactions up to time instance t_k with the i^{th} agent are the following sum:

$$q^i_{t_0:t_k} = \sum_{\tau=t_0}^{t_k} q^i(\tau) \tag{6}$$

where every symbol is as defined above.

To model the interaction forces $\mathbf{u}^p_i(\mathbf{p}_i, \mathbf{p}_p)$ in (3) and $\mathbf{u}^p_l(\mathbf{p}_l, \mathbf{p}_p)$ in (4), we employ a neighborhood-based approach. We define the neighborhood of the prober as the set:

$$\mathcal{N}_p(t_k) := \{i \mid 0 < d_{ip}(t_k) \leq \bar{r}_p, \forall i \in \{1, \ldots, N\}\} \tag{7}$$

where $d_{ip}(t_k) = \|\mathbf{p}_i - \mathbf{p}_p\|$ is the ℓ_2 Euclidean distance between the i^{th} swarm agent and the prober p, and $\bar{r}_p$ is the sensing radius of the prober. Notice that the neighborhood is a set that changes with time t_k. The interaction forces for all agents are the same and calculated as:

$$\mathbf{u}^p_i(\mathbf{p}_i, \mathbf{p}_p) = [1_{i \in \mathcal{N}_p}] \frac{\mathbf{p}_i - \mathbf{p}_p}{\|\mathbf{p}_i - \mathbf{p}_p\|^2} \tag{8}$$

where $[1_{i \in \mathcal{N}_p}] = 1$ if the i^{th} swarm agent is a neighbor of the prober and $[1_{i \in \mathcal{N}_p}] = 0$ otherwise.

TABLE I: The reward scheme used for the iSLI problem.

Reward	Weight
$N \cdot Q_k(l) - 1$	1.0
$(\hat{\mathbf{v}}_p \cdot \hat{\mathbf{v}}_l) \cdot [1_{d_{lp} \leq R_r}]$	1.0
$\|\mathbf{x}_p - \mathbf{x}_l\|$	-0.1
$\|u_p(t-1) - u_p(t)\|_1 \cdot [1_{d_{lp} \leq R_r}]$	-0.001

3) Rewards: Finally, we design the reward signal to train the policy of the prober. We list all components of the reward function and their associated weights in Table I.

For the main reward component, at time t_k we convert the interaction vector $\mathbf{q}_{t_0:t_k}$ into a probability distribution Q_k using the standard (unit) *softmax* function $\sigma : \mathbb{R}^N \to (0,1)^N$ so that $Q_k = \sigma(\mathbf{q}_{t_0:t_k})$. Then we query the distribution for the leader's probability $Q_k(l)$ and calculate the following reward:

$$r_1(t_k) = N \cdot Q_k(l) - 1 \tag{9}$$

Intuitively, the prober gets a reward proportional to the probability it assigns to the leader, with a maximum reward of $N - 1$. In contrast, it gets a negative reward when the probability $Q_k(l) \leq \frac{1}{N}$ with a maximum penalty of -1. This reward pushes the prober to place itself near the leader so that it maximizes the probability $Q_k(l)$ and keeps it away from other agents so that it minimizes $Q_k(i), \forall i = \{1, \ldots, N\}$ and $i \neq l$.

The second main reward drives the prober to stay aligned with the leader of the swarm and is calculated using the dot product between the velocity vectors of the two entities. Crucially, the prober gets this reward only if it is within a certain radius of $R_r = 100$ units. The first auxiliary rewards is the distance to the leader that incentivizes the prober to go near the swarm at the beginning of the episode when it has not interacted with the swarm yet. The second auxiliary reward is an action-smoothing term that penalizes abrupt changes in the control input and is used to make an easier deployment of the policy to the real robot.

D. Probing Policy

As discussed in the POMDP formulation of the iSLI problem, the observation that the prober gets is only the relative positions of the swarm and its absolute position. This distinction in the formulation has significant implications for the strategies available to the prober. It also suggests that the swarm agents may have an informational advantage over the prober, potentially allowing for more sophisticated coordination and response strategies.

We construct our policy architecture based on the S5 framework. Figure 2 shows the individual components of the policy, i.e., the embedding layer, the S5 encoder and the action and value heads. Inspired from the generalist agent (Gato) [15], we use an embedding layer to discretize

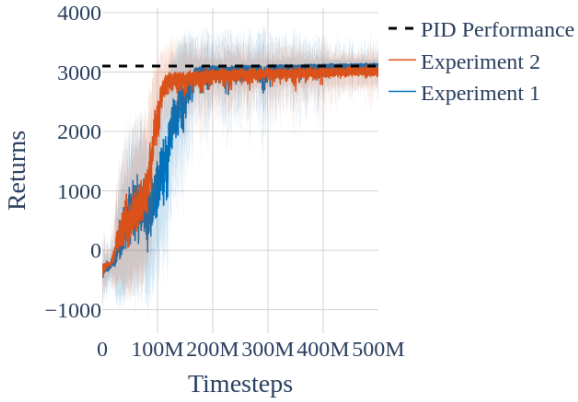

Fig. 3: Average return and standard deviation for two different seeds at the 6 agents environment. Comparison with hand-tuned PI controller (black dashed) that has knowledge of the environment's state and thus knows the identity of the leader.

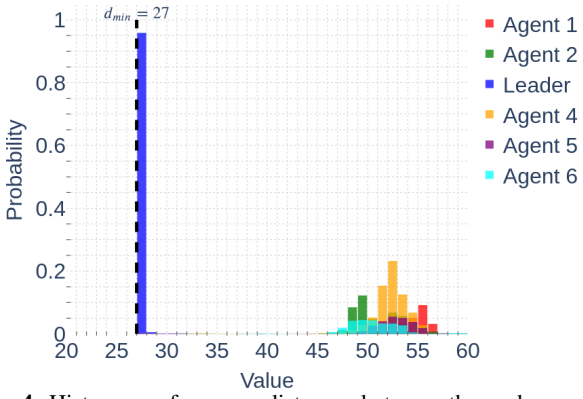

Fig. 4: Histogram of average distances between the prober and the unknown leader $L = 3$ for $10,000$ different validation simulations. The average distances for the last 512 steps (50% of the episode) are shown for all $10,000$ experiments.

the continuous observation $o_{t_k} \in [\underline{b}, \overline{b}]^{(N+1)\times 2}$ of the environment. The embedding layer is a trainable look-up table (LUT) that places the observation into $E = \{0, 1, \cdots 1023\}$ buckets and projects it to $n_e = 8$ dimensions. Then, the embedding $e_t \in \{0, 1, \cdots, 1023\}^{(N+1)\times n_e}$ is flattened and passed to the S5 encoder that keeps a track of the evolution of the environment inside its context (hidden state) $h_t \in \mathbb{R}^{n_p}$, where we choose $n_p = 256$.

For the actor and critic networks, we used hidden layer sizes 128, 64, 32, and $tanh$ activation functions. The critic network outputs the estimated value of the state, and the actor outputs a categorical distribution over the actions. Importantly, we use orthogonal initialization [16] for the weights of these networks.

The PPO parameters used for training the prober policy were: a discount factor γ of 0.99, GAE lambda λ of 0.95, and a total of 500 million timesteps. Each rollout consisted of 512 steps, and the learning rate varied linearly from $5 \times 10^{-4} \rightarrow 5 \times 10^{-6}$. The training process utilized 128 environments, and the model contained 1.1 million parameters.

IV. Simulations & Experiments

A. Numerical Simulations

The simulations were conducted on a computer equipped with a 12th Gen Intel® Core™ i9-12900K processor featuring 24 cores, as well as an NVIDIA® RTX A5500 GPU, running Linux Ubuntu 22.04 LTS. The framework is built using Python 3.10.12 and JAX, a high-performance scientific computing module [17]. Training one policy for $500M$ steps with the parameters mentioned before takes approximately three hours, which is a substantial reduction of training time compared to other methods that reported training times spanning several days.

We trained different models for varying numbers of swarm agents $N = 6, 10, 20, 30$. We report our results for the case $N = 6$ as it is the same number of agents used for the real robot experiment. After training, we evaluated our model across $10,000$ distinct evaluation episodes with different randomization seeds. To quantify the prober's performance in identifying the swarm's leader, we employed a specific procedure. For each evaluation episode, we first counted valid interactions using (6). Then, converted these interactions into a probability distribution using the *softmax* function. Finally, we calculated the percentage out of the $10,000$ episodes where the highest probability was assigned to the real leader. The results confirmed that the prober accurately identified the swarm's leader in the vast majority (95.7%) of episodes. To further quantify its performance, we employed two additional metrics: (i) the average returns accumulated during training and (ii) the distribution of distances between the prober and each agent during the latter half of each episode, across all evaluation episodes.

B. Robot Experiment

We validated our approach through real-world robot experiments. The experimental setup employed seven Hiwonder TurboPi [18] robots equipped with mecanum wheels. Six of these robots formed the swarm, while one acted as the prober, as seen in Fig. 2b. The arena the robots were moving in was 16'×14' in size and covered with rubber floor mats. We accurately track the robots' positions at $240Hz$ using an Optitrack Motion Capture System comprising eight cameras: six Primex 13 and two Primex 13W cameras. This setup allowed us to transition from simulation to a physical environment and test our algorithm's performance under real-world conditions.

For the deployment of the trained policy on the real prober robot, we developed a ROS2 [19] package. The output from the policy is in units of acceleration. We used a zero-order-hold (ZOH) to integrate this acceleration and convert it into commanded velocity for the robot. Thanks to the Just-In-Time (JIT) compilation capabilities of the JAX framework, we were able to execute the policy even at 80 Hz on the Raspberry Pi4 onboard the robot. However, for the experiments we run the policy at 10 Hz which provided sufficient responsiveness for our application.

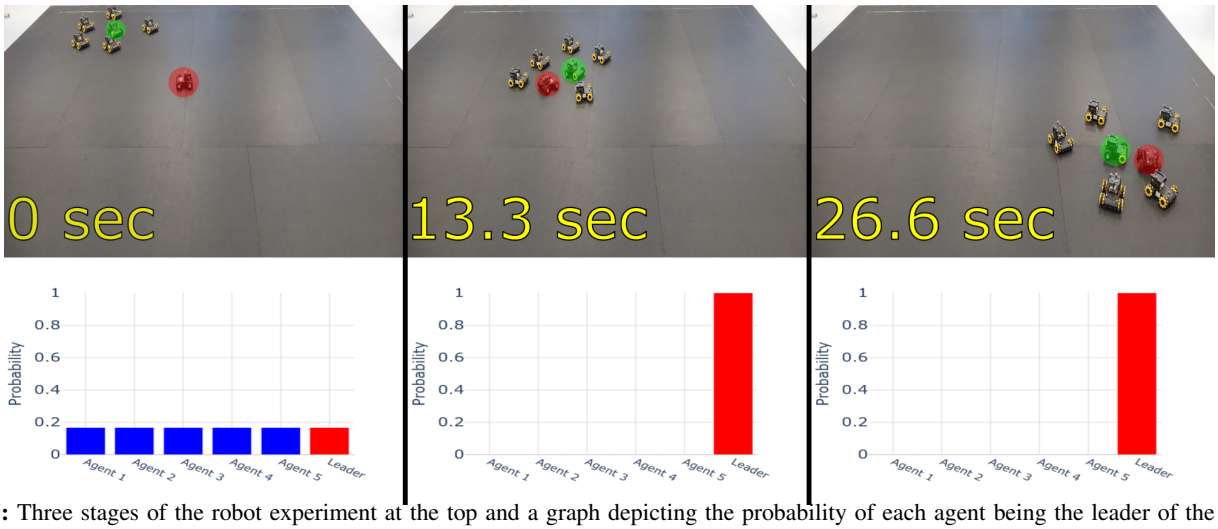

Fig. 5: Three stages of the robot experiment at the top and a graph depicting the probability of each agent being the leader of the swarm at the bottom. In the pictures, we highlight the prober with a red circle and the leader with a green circle. The swarm moves along the diagonal from the top left to the bottom right of the arena. From left to right, the experiment shows the swarm transitioning from its initial position to its final destination. As the swarm progresses, the prober advances toward the (unknown) leader and eventually comes and stays close to it.

V. Results and Discussion

A. Simulation results

Figure 3 shows the average returns of the proposed framework over all the timesteps during training, for two different randomization seeds (red and blue lines). The proposed method yields an average return of $3,000$. This performance is nearly equivalent to that of the hand-tuned PI controller used as a baseline, which achieves $3,100$. However, a key difference is that the PI controller knows the leader's identity.

The second metric used to evaluate our proposed method is the distribution of distances between the prober and the leader during the latter 50% of each episode, across $10,000$ different evaluation episodes. The 50% mark was chosen because the prober starts far from the swarm, so averaging distances from the start would not provide a reliable measure of success. Figure 4 presents the results, in histogram form, from these $10,000$ evaluation episodes with a swarm of $N = 6$ agents. Although the leader was assigned randomly during training, for consistency in evaluation, we set the leader as the third agent ($L = 3$) for all evaluation episodes. As shown in Fig. 4, the prober's average distance to the true leader is consistently minimized, demonstrating successful identification of the leader across almost all $10,000$ evaluation episodes.

B. Real robot results

Figure 5 illustrates the behavior of a prober robot as it attempts to identify a leader within a swarm during three key stages of an experiment. In the figure, the prober is highlighted with a red circle and the leader with a green circle. The top row of images shows the swarm's progression towards its final location. As the experiment unfolds, the prober advances towards the unknown leader, eventually positioning itself in close proximity. The bottom graph displays the probability distribution of each agent being the leader, calculated based on the prober's interactions. We define an interaction as an instance where the distance between the prober and another agent falls below 0.25 meters (twice the robot's length). These interaction counts are then converted into probabilities using the *softmax* function. At the beginning of the experiment, the prober has not yet interacted with any swarm members, resulting in a uniform probability distribution across all agents. By the intermediate stage at 13.3 seconds, the prober has successfully approached and identified the leader, as evidenced by the peaked probability distribution. Finally, the prober stays close to the leader for the remainder of the experiment until the swarm reaches its goal and the experiment ends after 26.6 seconds. This experiment demonstrates the effectiveness of our proposed leader identification algorithm in swarm robotics. The prober's ability to quickly converge on the leader showcases the potential for decentralized leadership recognition in robotic swarms, which could have significant implications for swarm coordination and task allocation in various applications, from search and rescue to environmental monitoring.

VI. Conclusion

In this study, we developed a method to identify the leader agent in a swarm by employing a physical adversary, which we named prober, that interacts with the swarm. The prober only has access to information on the relative positions of the other agents. The prober's policy was modeled using state-of-the-art sequential decision-making techniques and trained with the PPO algorithm. Results showed that our method could lead to the correct identification of the leader, regardless of the initial conditions. The proposed system represents the first instance of learning-based automatic identification of leader agents in a swarm, a capability that is crucial for enabling efficient and robust human-swarm interaction, understanding artificial swarm behaviors, and analyzing latent behaviors in biological swarms in nature.

REFERENCES

[1] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2015.

[2] Y. Liu and G. Nejat, "Robotic urban search and rescue: A survey from the control perspective," *Journal of Intelligent & Robotic Systems*, vol. 72, pp. 147–165, 2013.

[3] A. Singh and P. Artemiadis, "Automatic identification of the leader in a swarm using an optimized clustering and probabilistic approach," in *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2021, pp. 1–6.

[4] H. Zheng, J. Panerati, G. Beltrame, and A. Prorok, "An adversarial approach to private flocking in mobile robot teams," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1009–1016, 2020.

[5] A. Deka, W. Luo, H. Li, M. Lewis, and K. Sycara, "Hiding leader's identity in leader-follower navigation through multi-agent reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4769–4776.

[6] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[7] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[8] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

[9] J. T. Smith, A. Warrington, and S. W. Linderman, "Simplified state space layers for sequence modeling," *arXiv preprint arXiv:2208.04933*, 2022.

[10] C. Lu, Y. Schroecker, A. Gu, E. Parisotto, J. Foerster, S. Singh, and F. Behbahani, "Structured state space models for in-context reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[11] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[13] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Araújo, "Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms," *Journal of Machine Learning Research*, vol. 23, no. 274, pp. 1–18, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-1342.html

[14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[15] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[16] W. Hu, L. Xiao, and J. Pennington, "Provable benefit of orthogonal initialization in optimizing deep linear networks," *arXiv preprint arXiv:2001.05992*, 2020.

[17] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne *et al.*, "JAX: composable transformations of Python+ NumPy programs," 2018.

[18] Hiwonder Technology Co. (2024) Hiwonder TurboPi. [Online]. Available: https://www.hiwonder.com/products/turbopi?variant=40112905388119&srsltid=AfmBOoqyJKX4jsd38rZFgiIWLi5PP554z5pDaStMOCx1fUs09hnxTI4C

[19] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074