2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Learning to Imitate Spatial Organization in Multi-robot Systems

Ayomide O. Agunloye[1], Sarvapali D. Ramchurn[1], and Mohammad D. Soorati[1]

Abstract— Understanding collective behavior and how it evolves is important to ensure that robot swarms can be trusted in a shared environment. One way to understand the behavior of the swarm is through collective behavior reconstruction using prior demonstrations. Existing approaches often require access to the swarm controller which may not be available. We reconstruct collective behaviors in distinct swarm scenarios involving shared environments without using swarm controller information. We achieve this by transforming prior demonstrations into features that describe multi-agent interactions before behavior reconstruction with multi-agent generative adversarial imitation learning (MA-GAIL). We show that our approach outperforms existing algorithms in spatial organization, and can be used to observe and reconstruct a swarm's behavior for further analysis and testing, which might be impractical or undesirable on the original robot swarm.

I. INTRODUCTION

Swarm robotics and its applications are transitioning from the laboratory to the real world [1], [2], and it is expected to lead to the large-scale deployment of multiple robots in environments that are shared between robots and humans. For the robot-robot and human-robot interactions to be seamless, robot swarms must be safe and trustworthy [3]. Ensuring that a swarm is safe and trustworthy in shared environments requires precise and continuous knowledge of their collective behavior and how it evolves. In scenarios where swarm controllers are directly accessible, collective behavior can be modeled using the controllers [4]. However, in practical scenarios, swarm controllers may not be accessible for various reasons (e.g., inability to extract controllers from natural swarms). In robot swarms, access to controllers may be restricted or impractical due to various reasons such as encryption of controller information due to strategic or privacy concerns [5], [6]. To this end, understanding the collective behavior of a swarm and modeling its dynamics without using swarm controller information requires thorough research.

Collective behavior reconstruction and recognition are two established methods for modeling swarm dynamics or explaining collective behavior [7]. Collective behavior reconstruction can be model-based or data-driven. In model-based approaches, behavior reconstruction is achieved using a mathematical or regression model [8], [9]. In data-driven approaches, multi-agent interactions are statistically extracted or learned from prior demonstrations to reproduce observed behavior. Recently, data-driven approaches used imitation learning (IL) algorithms such as inverse reinforcement learning (IRL) and generative adversarial imitation learning (GAIL) for improved reconstruction accuracy in swarm scenarios modeled as multi-agent systems [10], [11], [12], [13], [14], [15]. Genetic programming and graphical neural networks have also been used for data-driven behavior reconstruction with swarm controllers extracted from video demonstrations [16] and swarm behavior prediction [17]. For recognition of collective behaviour, binary classification of observed behavior as defined or undefined collective behavior is a common approach [18], [19], [20]. This approach does not scale, as each swarm scenario requires a unique classifier. Multinomial classification addresses this and has been achieved in closely related swarming scenarios using predefined and learned multi-agent interaction quantifiers [21], [22], [23].

Existing data-driven behavior reconstruction approaches extract multi-agent interactions from expert demonstrations without capturing swarm-environment interactions. As a result, the recovered multi-agent interactions cannot accurately reconstruct or predict expert behavior. While this issue is addressed in [17], their approach relied on learned extraction of multi-agent interactions and it is difficult to explain how the robots interact with the environment.

In this work, we investigate the reconstruction of collective behavior in three spatial organization tasks involving shared environments without using swarm controller information. We consider three common swarm robotic scenarios: aggregation, homing, and obstacle avoidance. We model these scenarios as single-objective swarm scenarios where swarming agents interact with each other and the environment while completing the swarm objective. We generate informed and explainable multi-agent interactions through feature transformation of expert demonstrations and, achieve near-optimal behavior reconstruction using multi-agent GAIL. We show that even when learner robots are initialized from unforeseen states, they perform similarly to the expert robots in all investigated scenarios.

The main contributions of this paper are as follows. (1) We present an approach for reconstructing collective behaviors in shared environments without accessing robot controllers; and (2) We demonstrate the use of informed and explainable multi-agent interactions for improved learning representation in data-driven behavior reconstruction.

[1]Authors are with the School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1TR, United Kingdom. {a.o.agunloye, sdr1, m.soorati} @soton.ac.uk

II. RELATED WORKS

IRL has been extensively used in the literature for the reconstruction of collective behaviour as it recovers the underlying reward functions while reproducing expert behavior. Šošić et al. [11] reconstructed the behavior of a homogeneous swarm by assuming that all agents are interchangeable and share a central reward function, thereby reducing the problem to a single-agent IRL. Another study [12] reconstructed the collective behavior observed in a flock of pigeons by recovering individual reward functions for each pigeon. This individualistic approach exposed the multi-agent interactions in the flock and allowed the researchers to model the leader-follower hierarchy. In [24], a similar individualistic reward function approach was used to evolve the robot controller using IRL by manually specifying the desired goal location or the path. Other studies [13], [10] extended the IRL framework to multi-agent IRL to simultaneously recover the reward functions of several agents. Liu et al. [14], however, integrated GAIL with population-based training for collective behavior reconstruction. Besides IRL and GAIL, other machine learning methods have been used to reconstruct collective behavior. In [16], genetic programming was used to extract explainable controllers from video demonstrations of collective behavior with 8 predefined swarm interaction metrics defining the fitness measure. Zhou et al. [17] used graphical neural networks to imitate the behavior of expert robots and predict trajectories. Their approach considers swarm-environment interactions and modeled robots and environmental entities as graph nodes, but does not provide explainable interactions or controllers. Most of these works demonstrate behavior reconstruction in a single scenario or multiple similar scenarios (e.g., swarming and schooling). However, Yu et al. [15] used Adversarial Imitation Learning with parameter sharing (PS-AIRL) for behavior reconstruction in distinct swarming scenarios. Their approach focused on homogeneous biological swarms and did not consider swarm-environment interactions. They also require access to the original swarm controllers which is rarely available in practical scenarios. In contrast, our approach considers swarm-environment interactions and reconstructs expert behavior without accessing robot controllers. We also generate informed multi-agent interactions that can be used to explain swarm behavior.

III. BACKGROUND

We consider decentralized Partially Observable Markov Decision Processes (Dec-POMDP) [25] in which agents receive individual rewards for their actions. A Dec-POMDP is defined as an MDP comprising a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \Omega, \gamma \rangle$. $\mathcal{N}$ represents the set of agents in the Dec-POMDP, $\mathcal{S}$ is the global state space of the environment, $\mathcal{A}$ contains the shared action space of all agents in $\mathcal{N}$ and $\mathcal{O}$ represents the joint observation space of all agents in the environment. At each episodic time step t, each agent $i \in \mathcal{N} \equiv \{1, ..., n\}$ takes an action $a_i \in \mathcal{A}$ to form the joint action $\mathrm{a} \in \boldsymbol{\mathcal{A}} \equiv \mathcal{A}^n$ based on its partial observation of the environment $o_i \in \Omega$ as provided by the observation function $\mathcal{O}(s, a)$ using parameterized policy $\pi_i(a_i|o_i)$. The state transition function $\mathcal{T}(s\prime|s, \mathrm{a}) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathcal{S}$ provides the next global state, and the shared reward function $r(s, \mathrm{a}) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathcal{R}$ gives each agent an individual reward $r_i \in \mathcal{R}$. $\gamma \in [0, 1)$ denotes the reward discount factor.

GAIL achieves imitation learning by matching the occupancy measures ρ_{π_E} of the expert policy π_E in the learner domain through generative adversarial training [26]. The occupancy measure is the unnormalized distribution of an agent's trajectory as it navigates the environment using a policy π [27]. In GAIL, the generator is a policy network π that produces trajectories from a similar environment as the expert. The discriminator network D compares generated trajectories with expert demonstrations and attempts to distinguish them through binary classification.

The GAIL objective function can be written in terms of occupancy measures and expectations over expert and learner policies as [27]:

$$\psi^{\star}_{GA}(\rho_{\pi} - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\pi}[\log(D(s, a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s, a))] \quad (1)$$

where $\psi^{\star}_{GA}$ is the convex regularization imposed on the generator by D, $D(s, a)$ is the discriminator output, and $\log D(s, a)$ is the learning signal for the generator.

GAIL optimizes Equation 1 by finding its saddle point (π, D). At this point, D is unable to differentiate between trajectories from π and π_E. When π and D are represented by function approximators, GAIL fits a parameterized policy π_θ and a discriminator network D_w with weights w. The discriminator feedback serves a reward function that encourages the generator to minimize the dissimilarity between ρ_π and ρ_{π_E}.

In multi-agent systems, individual agents optimize separate reward functions that describe their behavior. As a result, multiple reward functions exist and optimality is only guaranteed through a set of stationary policies that provide a Nash equilibrium solution. Multi-agent GAIL addresses this by jointly optimizing the Nash equilibrium constraints with the objective function during occupancy measures matching [28].

IV. METHOD

In this section, we describe our approach to accurate collective behavior reconstruction. We formulate the problem as a collective behavior reconstruction problem in a shared Dec-POMDP environment. Expert demonstrations $\mathcal{D}$ contain the absolute position of all observable entities $\mathcal{M}$ in the environment. We transform $\mathcal{D}$ into informed multi-agent interactions before recovering policies that accurately reproduce expert behaviors using multi-agent GAIL (MA-GAIL) [28].

A. Expert Demonstrations Transformation

We transform each expert trajectory in $\mathcal{D}$ to a set of state representative features $\mathbf{f}_s$ describing the interaction between the expert and all other observable entities in the environment given a state $s \in \mathcal{D}$. We achieve the transformation by computing the cohesion between agent i and every other entity in $\mathcal{M}$. Thus, the state representative features for agent i in state s is:

$$\mathbf{f}_s^i = [-dist(i,j) | j \in \mathcal{M}, j \neq i] \tag{2}$$

where $dist(i,j)$ denotes the euclidean distance between agent i and entity j.

B. Policy Recovery with Multi-Agent GAIL

To recover stationary policies in the DEC-POMDP, we use MA-GAIL with n individual discriminators $\mathbf{D} = \{D_1, D_2, ..., D_n\}$ and match occupancy measures on transformed expert demonstrations. For the generator network π, we use Multi-Agent Proximal Policy Optimization (MAPPO) with parameter sharing in which all learners use a single policy network. This applies to our environment since our agents are homogeneous and have identical observation and action spaces [29]. Using individual discriminators ensures that each learner strictly matches the occupancy measures of a particular expert. However, this prevents generalization for homogeneous agents as learners receive poor feedback if they behave like any other expert. We address this through expert demonstration sharing and allow individual discriminators to compare trajectories from their learners with all expert demonstrations available. This ensures that learners are positively rewarded for demonstrating any valid expert behavior instead of the particular behavior from one expert.

The policy recovery algorithm is summarized in Algorithm 1. Given expert demonstrations $\mathcal{D}$, learners interact with the environment and generate rollout trajectories T_k. The discriminators are trained using feature transformed $\mathcal{D}$ and T_k. At each time step, learners receive individual reward feedback r_{π,D_n} with which the shared policy is improved. Compared to PS-AIRL [15], our algorithm uses n discriminators instead of one and transforms all input into state representative features before passing them to the discriminators. It also allows the discriminators to share the features for improved learning representation.

Algorithm 1 Policy recovery with MA-GAIL

Input: expert demonstrations $\mathcal{D}$
Randomly initialize generator π & discriminators $\mathbf{D}$
for $k = 1, 2, \ldots$ **do**
  Rollout learner $\boldsymbol{T} = \{T_1, T_2, ..., T_k\}$ using π
  for $n = 1, 2, \ldots, |\mathcal{N}|$ **do**
    Train D_n to classify $\mathbf{f}_s^n \forall s \in \mathcal{D}$ from $\mathbf{f}_s^n \forall s \in T_k$
  end for
  Generate r_{π,D_n} for each generator policy

$$r_{\pi,D_n} \leftarrow [\log(D_n(\mathbf{f}_s^n))] + [\log(1 - D_n(\mathbf{f}_s^n)]$$

  Update π using r_{π,D_n} with PS-MAPPO
end for

V. EXPERIMENT

We evaluate the performance of our proposed approach in three classical swarm robotic scenarios: aggregation, homing, and obstacle avoidance. We model these scenarios as cooperative and single-objective in a shared environment. We consider a swarm size of 3 and represent our swarming agents as uncrewed aerial vehicles (UAVs) with inaccessible controllers. To improve learning representation, we reduce the complexity of the shared environment and separate it into motion and control layers. The control layer is a discretized grid world representation of the shared environment with reduced state and action space. We compare the performance of the proposed approach with PS-AIRL [15] and behavior cloning (BC), where a direct mapping between expert states and actions is learned [30].

A. Swarm Scenarios

In aggregation, the objective of swarming UAVs is to maximize the intra-swarm cohesion. They achieve this by safely forming a cluster at any suitable zone in a shared environment. The shared environment includes two active UAVs hovering at fixed positions. Swarming UAVs can observe fixed-position UAVs if they are within perception range in both layers but can only interact with them in the motion layer. We model the individual reward at each time step in the control layer r_n as:

$$r_n = \begin{cases} n_{\text{agents}} \times \text{c} & \text{if } n_{\text{agents}} > 1 \\ -\text{c} & \text{otherwise} \end{cases} \tag{3}$$

where $n_{\text{agents}} = |\{c_n > t \forall n \in \mathcal{N}\}|$, and t is an environment specific aggregation threshold.

The objective in the homing scenario differs from aggregation in that the clustering zone—home position—is fixed and cannot be dynamically chosen by the robots. In this scenario, the UAVs must explore the environment and locate the home positions before the episode ends. Once a UAV finds a home position, it must remain there until all other UAVs have homed. We model r_n as the maximum cohesion between UAV n and any home position at a given time step.

In a new behavior that we refer to as obstacle avoidance, UAVs must navigate the shared environment without interacting with fixed-location inactive UAVs in the shared environment. This scenario differs from existing obstacle avoidance scenarios in that the UAVs can access the positions already occupied by the inactive UAVs in the control layers. However, they receive a large negative reward for doing this. The motivation for this behavior is that it is crucial to maintain a safe distance from unknown entities in a practical shared environment, even if they seem inactive. The UAVs also receive a small

negative reward for insufficient exploration. We model r_n as a large constant $-c$ when the cohesion between the UAV n and any fixed position UAV is maximized and 0 otherwise.

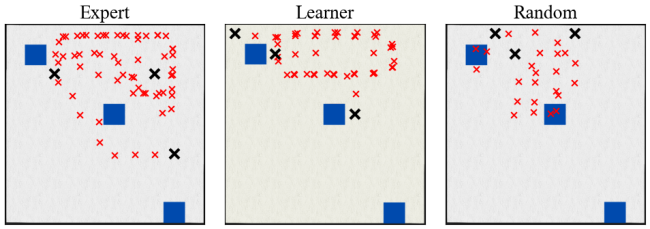

Fig. 1. Snapshot of the motion layer showing the obstacle avoidance behavior and position trace for Experts, Learners, and Random swarming UAVs between t=0 and t=300s. X represents UAVs positions at t=0. Blue boxes are inactive UAVs locations.

B. Simulation Environment

We implement the motion layer for the shared environment in Webots [31] using simulated Crazyflies 2.0 drones [32] as the swarm. The simulation boundary is a 3m by 3m continuous rectangular world. UAVs can move in all directions and can detect obstacles using onboard range sensors.

The control layer is a 10×10 grid world environment. Here, the action space in the is limited to $a \in \mathbb{R}^5$ corresponding only to the high-level control of the UAVs {stop, right, left, forward, and backward}. Low-level motion controls such as lift-off, turning, translation, and hovering are implemented deterministically in the motion layer. Agents in the grid world can observe the positions of other entities up to 6 grid positions in all directions. Given that only two fixed-position entities are in the aggregation scenario, the agent's observation space is $o \in \mathbb{R}^{10}$ in this scenario and $o \in \mathbb{R}^{12}$ in others. All episodes run for a fixed duration of 50 time steps in the control layer. This corresponds to an episode duration of about $300s$ in the motion layer. Figure 1 shows the position trace of the experts, learners, and random UAV behavior in the obstacle avoidance scenario between $t = 0$ and $t = 300s$ with small red crosses. The three larger crosses on each setup mark the initial positions.

C. Implementation Details

1) Expert Demonstrations: To generate expert demonstrations of collective behavior in each scenario, we train expert UAVs in the corresponding Dec-POMDP grid world using PS-MAPPO for $100,000$ training episodes. After training, we generate an expert demonstration data pool of $1,000$ trajectories in each scenario. We also generate noisy expert demonstrations by varying expert optimality $\epsilon \in [0, 1]$, where 0 implies optimal experts and 1 implies experts sampling actions at random. It should be noted that trajectories in the expert demonstrations data pool are randomly generated and may contain similar expert UAV behavior.

2) Learner UAVs: Learner UAVs interact with the environment for $10,000$ training episodes using expert demonstrations between 200 and 500. This expert demonstration range was chosen as it agrees with expert dataset sizes in existing works [17], [15]. Learner UAVs receive individual rewards from their discriminator for each episodic time step. The rewards and corresponding trajectories are stored in a shared buffer for training the PS-MAPPO policy at the end of each episode. After 50 training episodes, the discriminators are first initialized and trained using available learner and expert trajectories. They are then updated every 50th episode for $1,000$ training episodes and then every 500th episode. This update frequency ensures that the discriminators are properly initialized but do not change too quickly, thus allowing learner UAVs to understand reward patterns.

3) Network Training: All models were trained and evaluated on a single cluster node with a 64 cores 2.2 GHz Intel CPU and 256 GB of RAM. Expert policies training took about 7 hours per scenario, while learner policies training only took an hour per scenario. PS-MAPPO implementations for expert policy and MA-GAIL generator network used the default hyperparameters provided in the original paper [29]. The MA-GAIL discriminators were simple 2 layer multi-layer perceptron network (MLP) with 128 hidden units and rectified linear unit (relu) activations. These discriminators were trained in parallel using a learning rate of 1×10^{-5} so that their training does not influence the training time. PS-AIRL was implemented using the algorithm provided in the paper while BC was achieved using individual 3 layer MLPs with 128 hidden units and relu activation.

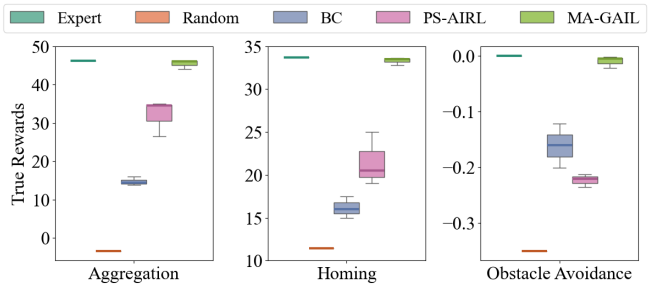

Fig. 2. Boxplots of true episode rewards obtained in 200 evaluation episodes by the proposed approach (MA-GAIL), BC, and PS-AIRL trained with 400 expert demonstrations in all scenarios.

VI. RESULTS

We demonstrate the advantage of our behavior reconstruction algorithm in two different ways. In the first setup, we initialize learner UAVs from starting positions present in $\mathcal{D}$ for every evaluation episode. Figure 2 shows the performance comparison between the proposed approach (MA-GAIL), PS-AIRL, and BC trained using 400 expert demonstrations over 200 evaluation episodes with unnormalized reward values in all scenarios. As the figure demonstrates, our approach closely reproduces

expert behavior in all scenarios compared to BC and PS-AIRL. This high performance across distinct swarm scenarios can be attributed to the transformed expert demonstrations, which sufficiently describe the multi-agent interactions in the shared environment. It can also be attributed to expert demonstration sharing, which increases the set of valid expert behaviors, thus allowing learners to easily reproduce expert behaviors regardless of the scenario. PS-AIRL outperformed BC in aggregation and homing scenarios but failed to maintain its superiority in the obstacle avoidance scenario. We attribute this to the sparsity of the reward function in the obstacle avoidance scenario, which forces the experts to conservatively explore a small area in the shared environment and avoid the large negative rewards. The abundance of sequential data from this region makes it easy for BC to clone expert actions when initialized from starting positions close to the area and outperform PS-AIRL. Conversely, the continuous reward function in aggregation and homing scenarios provides a normal distribution of state-action pairs in expert demonstrations making it difficult for BC to clone expert actions. It should be noted that evaluation results are from learners trained using 400 expert demonstrations as they represent the best-performance region for PS-AIRL and BC.

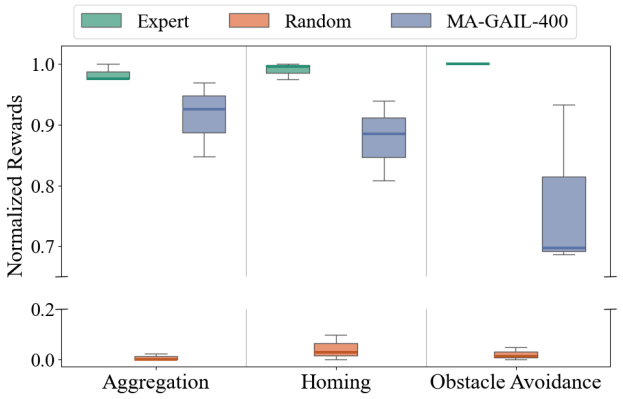

Fig. 3. Boxplots of normalized reward values for Expert, Random, and MA-GAIL-400 over 200 evaluation episodes initialized from random starting states in all scenarios.

In the second evaluation setup, learner UAVs are initialized randomly from unforeseen starting states at the beginning of each evaluation episode. Figure 3 presents the normalized reward values for the experts, random (suboptimal experts with $\epsilon = 1$), and learners trained with 400 expert demonstrations (MA-GAIL-400) over 200 different evaluation episodes in all scenarios. As the figure demonstrates, learners do not perfectly reproduce expert behaviors in all scenarios due to their non-familiarity with the initial states. This effect is, however, pronounced in the obstacle avoidance scenario where expert agents can safely navigate the fixed-position entities without interacting with them, even though it is risky

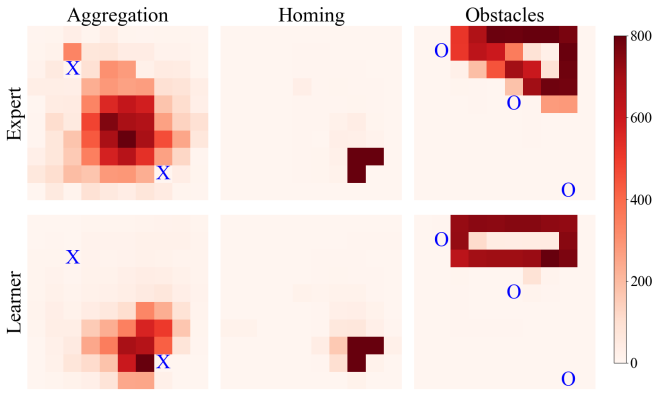

Fig. 4. Visualization of swarming UAVs positions for 10 evaluation episodes in all scenarios. Area coverage of expert (top) and learner (bottom). Active UAV locations are marked by X and inactive locations are shown as O.

due to the sparsity of the reward function. Learner UAVs, on the other hand, do not consistently reproduce this risky behavior when initialized from random starting states. This shows that while imitating the controllers of multi-robot systems generated with a sparse reward or cost function may be easy, accurately predicting how they will perform in unforeseen states still requires further research. It should be noted that modeling the obstacle avoidance scenario using a continuous cohesion-based reward function did not produce optimal expert policies in the control layer.

Figure 4 shows the area coverage of optimal experts and MA-GAIL-400 learner UAVs in 10 evaluation episodes in all scenarios. The fixed-position active UAVs are marked with 'X' in aggregation, while inactive UAVs are represented as 'O' in obstacle avoidance. We observe that learner UAVs do not directly reproduce particular expert behavior but unravel patterns in the demonstrations that allow them to maximize discriminator reward and mimic any expert. This is evident in the aggregation and obstacle avoidance scenarios where learner UAVs do not explore the shared environment as much as the experts, even though they are initialized from the same starting positions. Variations in expert and learner UAVs' absolute positions in Figure 4 result from learner UAVs matching the occupancy measures of features describing expert behaviors in the control layer and not their absolute positions. This is intuitive since the GAIL convex regularizer only penalizes the generator heavily when it maximizes dissimilarity between expert and learner occupancy measures, and expert demonstration transformation and sharing reduce how often this happens based on absolute positions.

Transforming expert demonstration to generate informed and explainable multi-agent interactions improves learning representation and facilitates accurate behavior reconstruction. Furthermore, using n individual discriminators while allowing them to share the transformed demonstrations reduces the complexity of

each discriminator network and guarantees learners will be intuitively rewarded for all valid expert behavior. Nevertheless, these improvements introduce certain limitations. First, cohesion may not sufficiently describe multi-agent interactions in complex swarm scenarios involving multiple collective behaviors, and the search for a suitable interaction quantifier in such scenarios might not be trivial. A straightforward approach to address this is to transform expert demonstrations using several interaction metrics as in [16]. However, this may introduce redundancy and increase the computation budget as the swarm size increases. Second, using n discriminators and sharing expert demonstrations between them can introduce scalability issues as n becomes very large. While we address this through distributed discriminator training in this work, several discriminators (e.g., $n > 100$) may be computationally expensive to train even in parallel. Finally, the challenge of collecting an adequate number of expert demonstrations remains, and we have not optimized our model to use minimal demonstrations. We intend to focus on these limitations in future works.

VII. CONCLUSION

In this work, we reconstructed collective behavior in shared environments without having access to the swarm controller. We achieve this by transforming expert demonstrations into state features that sufficiently describe the multi-agent interactions between entities in the shared environments. We investigate three distinct classical swarm robotics. To improve learning representation, we separate the shared environments into motion and control layers and model the control layers as DEC-POMDPs grid world environments. Our results in the control layer demonstrate the capability of our MA-GAIL approach to accurately reconstruct observed collective behaviors in spatial organisation (i.e., aggregation, goal homing, and obstacle avoidance) compared to existing reconstruction algorithms. We show that transforming expert demonstrations into shared state features that sufficiently describe multi-agent interactions improves behavior reconstruction accuracy in distinct or unrelated swarm scenarios. In the future, we will investigate behavior reconstruction and recognition in complex practical scenarios involving multiple collective behaviors. As cohesion does not sufficiently describe all multi-robot interactions in more complex scenarios, we will investigate the automated discovery of explainable interaction quantifiers to achieve this. Future work will also consider the real-world constraints in experiments with physical multi-robot systems.

References

[1] M. Dorigo, G. Theraulaz, and V. Trianni, "Swarm Robotics: Past, Present, and Future [Point of View]," Proc. IEEE, vol. 109, no. 7, pp. 1152–1165, Jul. 2021.

[2] A. R. Cheraghi, S. Shahzad, and K. Graffi, "Past, Present, and Future of Swarm Robotics," in Intell. Syst. Appl., ser. Lect. Notes Netw. Syst., K. Arai, Ed. Cham: Springer Int. Publ., 2022, pp. 190–233.

[3] M. D. Soorati, M. Naiseh, W. Hunt, K. Parnell, J. Clark, and S. D. Ramchurn, "Enabling trustworthiness in human-swarm systems through a digital twin," in Putting AI in the Crit. Loop, P. Dasgupta, J. Llinas, T. Gillespie, S. Fouse, W. Lawless, R. Mittu, and D. Sofge, Eds. Academic Press, 2024, pp. 93–125.

[4] A. Ligot and M. Birattari, "On Using Simulation to Predict the Performance of Robot Swarms," Sci. Data, vol. 9, no. 1, p. 788, Dec. 2022, number: 1 Publisher: Nature Publishing Group.

[5] W. Hunt, J. Ryan, A. O. Abioye, S. D. Ramchurn, and M. D. Soorati, "Demonstrating Performance Benefits of Human-Swarm Teaming," Mar. 2023, arXiv:2303.12390 [cs].

[6] L. Chen, S. Fu, L. Lin, Y. Luo, and W. Zhao, "Privacy-Preserving Swarm Learning Based on Homomorphic Encryption," in Algorithms Archit. Parallel Process., ser. Lect. Notes Comput. Sci., Y. Lai, T. Wang, M. Jiang, G. Xu, W. Liang, and A. Castiglione, Eds. Cham: Springer Int. Publ., 2022, pp. 509–523.

[7] M. Naiseh, M. D. Soorati, and S. Ramchurn, "Outlining the design space of explainable swarm (xswarm): Experts' perspective," in Distrib. Auton. Robot. Syst. Cham: Springer Nature Switzerland, 2024, pp. 28–41.

[8] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," ACM SIGGRAPH Comput. Graph., vol. 21, no. 4, pp. 25–34, Aug. 1987.

[9] M. Sinhuber, K. Van Der Vaart, Y. Feng, A. M. Reynolds, and N. T. Ouellette, "An equation of state for insect swarms," Sci. Rep., vol. 11, no. 1, p. 3773, Feb. 2021.

[10] D. Waelchli, P. Weber, and P. Koumoutsakos, "Discovering Individual Rewards in Collective Behavior through Inverse Multi-Agent Reinforcement Learning," May 2023, arXiv:2305.10548 [cs].

[11] A. Šošić, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koeppl, "Inverse Reinforcement Learning in Swarm Systems," Mar. 2017, arXiv:1602.05450 [cs, stat].

[12] R. Pinsler, M. Maag, O. Arenz, and G. Neumann, "Inverse Reinforcement Learning of Bird Flocking Behavior," IEEE Int. Conf. Robot. Autom. (ICRA) Swarms: Biol. Robot. Back Workshop, 2018.

[13] T. Costa, A. Laan, F. J. H. Heras, and G. G. de Polavieja, "Automated Discovery of Local Rules for Desired Collective-Level Behavior Through Reinforcement Learning," Front. Phys., vol. 8, 2020.

[14] S. Liu, X. Peng, and T. Wang, "PBT-GAIL: An Imitation Learning Framework in Swarm Systems," in Proc. Int. Conf. Auton. Unmanned Syst. (ICAUS), ser. Lect. Notes Electr. Eng., M. Wu, Y. Niu, M. Gu, and J. Cheng, Eds. Singapore: Springer, 2022, pp. 1884–1894.

[15] X. Yu, W. Wu, P. Feng, and Y. Tian, "Swarm Inverse Reinforcement Learning for Biological Systems," in IEEE Int. Conf. Bioinf. Biomed. (BIBM), Dec. 2021, pp. 274–279.

[16] K. Alharthi, Z. S. Abdallah, and S. Hauert, "Automatic Extraction of Understandable Controllers from Video Observations of Swarm Behaviors," in Swarm Intell., ser. Lect. Notes Comput. Sci., M. Dorigo, H. Hamann, M. López-Ibáñez, J. García-Nieto, A. Engelbrecht, C. Pinciroli, V. Strobel, and C. Camacho-Villalón, Eds. Cham: Springer Int. Publ., 2022, pp. 41–53.

[17] S. Zhou, M. J. Phielipp, J. A. Sefair, S. I. Walker, and H. B. Amor, "Clone Swarms: Learning to Predict and Control Multi-Robot Systems by Imitation," in IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS). Macau, China: IEEE, Nov. 2019, pp. 4092–4099.

[18] M. M. Khan, K. Kasmarik, and M. Barlow, "Autonomous detection of collective behaviours in swarms," Swarm Evol. Comput., vol. 57, p. 100715, Sep. 2020.

[19] N. Khattab, S. Abpeikar, K. Kasmarik, and M. Garratt, "Autonomous Recognition of Collective Motion Behaviours in Robotic Swarms from Video using a Deep Neural Network," in Int. Joint Conf. Neural Netw. (IJCNN), Jun. 2023, pp. 1–8, iSSN: 2161-4407.

[20] W. Li, M. Gauci, and R. Groß, "Turing learning: a metric-free approach to inferring behavior and its application to swarms," Swarm Intell., vol. 10, no. 3, pp. 211–243, Sep. 2016.

[21] M. Cenek and S. Dahl, "Towards Emergent Design: Analysis, Fitness and Heterogeneity of Agent Based Models Using Geometry of Behavioral Spaces Framework." in Proc. Artif. Life Conf. Cancun, Mexico: MIT Press, 2016, pp. 46–53.
[22] D. S. Brown and M. A. Goodrich, "Limited bandwidth recognition of collective behaviors in bio-inspired swarms." in Proc. Int. Conf. Auton. Agents Multi-agent Syst. (AAMAS), 2014, pp. 405–412.
[23] S. Abpeikar, K. Kasmarik, and M. Garratt, "Automatic Multi-Class Collective Motion Recognition Using a Decision Forest Extracted from Neural Networks," in IEEE Reg. 10 Symp. (TENSYMP), Sep. 2023, pp. 1–6, iSSN: 2642-6102.
[24] I. Gharbi, J. Kuckling, D. G. Ramos, and M. Birattari, "Show me What you want: Inverse Reinforcement Learning to Automatically Design Robot Swarms by Demonstration," in IEEE Int. Conf. Robot. Autom. (ICRA). London, United Kingdom: IEEE, May 2023, pp. 5063–5070.
[25] F. A. Oliehoek and C. Amato, A Concise Introduction to Decentralized POMDPs, ser. SpringerBriefs Intell. Syst. Cham: Springer Int. Publ., 2016.
[26] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, "A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications," IEEE Trans. Knowl. Data Eng., vol. 35, no. 4, pp. 3313–3332, Apr. 2023, conference Name: IEEE Trans. Knowl. Data Eng.
[27] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," in Adv. Neural Inf. Process. Syst., vol. 29. Curran Associates, Inc., 2016.
[28] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-Agent Generative Adversarial Imitation Learning," Jul. 2018, arXiv:1807.09936 [cs, stat].
[29] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games," Nov. 2022, arXiv:2103.01955 [cs].
[30] B. Zheng, S. Verma, J. Zhou, I. Tsang, and F. Chen, "Imitation Learning: Progress, Taxonomies and Challenges," Oct. 2022, arXiv:2106.12177 [cs].
[31] Webots, "Cyberbotics: Robotics simulation with Webots."
[32] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in Int. Conf. Methods Models Autom. Robot. (MMAR), Aug. 2017, pp. 37–42.