Dec-AIRL: Decentralized Adversarial IRL for Human-Robot Teaming

Prasanth Sengadu Suresh
THINC lab, School of Computing
University of Georgia
Athens, GA, USA
ps32611@uga.edu

Yikang Gui
THINC lab, School of Computing
University of Georgia
Athens, GA, USA
Yikang.Gui@uga.edu

Prashant Doshi
THINC lab, School of Computing
University of Georgia
Athens, GA, USA
pdoshi@uga.edu

ABSTRACT

We present a new method for inverse reinforcement learning (IRL) that allows an agent to learn from expert demonstrations and then spontaneously collaborate with a human on the same task. We generalize adversarial IRL (AIRL) to work in a decentralized setting using a decentralized Markov decision process (Dec-MDP) as the underlying model. We posit that a Dec-MDP is a better-suited model for pragmatic multi-agent IRL compared to the multi-agent Markov decision process (MMDP) or the Markov game, which have been utilized thus far. This is because the latter models require an agent to know the global state of the environment, which is impractical in the real world as it may include agent-specific attributes (e.g. joint angles) that may not be directly observable by the other agents. We test our method on two domains: a formative simulated patient assistance scenario and a summative real-world use-inspired domain of sorting onions on a line conveyor. Our method (Dec-AIRL) significantly improves on the previous techniques in both domains. These results indicate that a decentralized multi-agent IRL formalism promotes effective teaming in human-robot collaborative tasks.

KEYWORDS

Cobots; Human-Robot Teaming; Learning from observations; IRL

ACM Reference Format:
Prasanth Sengadu Suresh, Yikang Gui, and Prashant Doshi. 2023. Dec-AIRL: Decentralized Adversarial IRL for Human-Robot Teaming. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Inverse reinforcement learning (IRL) [28] is the paradigm of using demonstrations provided by expert agent(s) to learn complex task preferences that are often non-trivial to program. It aims to learn a reward function that provides an accurate representation of the expert's preferences. While there is much work on IRL applied to single agent scenarios, IRL for multi-agent tasks is less explored [2]. One key application of multi-agent IRL is in human-robot collaborative settings where the demonstration is provided by a human-human team, which may then be used by a robot to learn how to perform the task and actively collaborate with a human in the shared environment. For instance, consider a produce processing shed where humans stand across from each other and sort produce on a conveyance system by removing the blemished produce. Is it possible for a cobot to observe and learn to seamlessly collaborate with the humans on the line?

Toward realizing this, a key challenge is to develop IRL methods to learn the reward function of a collaborative multi-agent team. In this regard, recent methods generalize the well-known adversarial IRL (AIRL) and generative adversarial imitation learning (GAIL) [8, 15] to a multi-agent setting. These include multi-agent AIRL [37, 40] and its extension to demonstrations with latent variables [13], as well as Co-GAIL [35] and MA-GAIL [32]. These methods generalize the Markov decision process (MDP) model of the expert's behavior to either the Markov game [20] or the MMDP [4]. However, both these models require each participating agent to perfectly know the *global state* of the system, which may not be practicable. We illustrate this limitation in the context of produce processing where the produce is, say onions: the sufficient global state also includes variables indicating whether an agent has identified the onion as being blemished or not. While the robot's sensors facing the human can assess the onions on the conveyor, this may not align with the human's inspection of the onion. As such, some of the variables in the global state that pertain to the other agent may not be perfectly determined, and this may lead to adverse interactions such as both agents reaching for the same onion.

Given this limitation of extant methods, we posit that a *decentralized MDP (Dec-MDP)* [12], which can be solved to produce a vector of policies (one for each agent) in which an agent's actions are conditioned on its own history of observations, is a more appropriate model. Therefore, we present a new generalization of AIRL to multi-agent settings, which we label as Dec-AIRL, that ascribes a Dec-MDP to model the multi-agent collaboration. Dec-AIRL follows the generator-discriminator architecture where the generator is modeled as performing RL using centralized training and decentralized execution (CTDE) [39]. It generates trajectories of states and actions for each agent, which are used by the discriminator to inversely learn the joint reward function.

We evaluate the performance of Dec-AIRL on two domains and compare its empirical learned behavior accuracy (LBA) with that of Co-GAIL and MA-AIRL. The expert trajectories for our simulated *assistive domain* comes from a trained RL model. For the realistic *produce-processing domain*, we deploy our learn-from-observation (LfO) pipeline (see Fig. 1) to record and process RGB-D camera streams of humans engaged in sorting batches of onions on a table. While Co-GAIL uses an MMDP to model the human-robot collaborative problem and assumes access to the global state and the latent features pertaining to the human behavior, MA-AIRL models it as a Markov game, which also maps the global state but to each

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

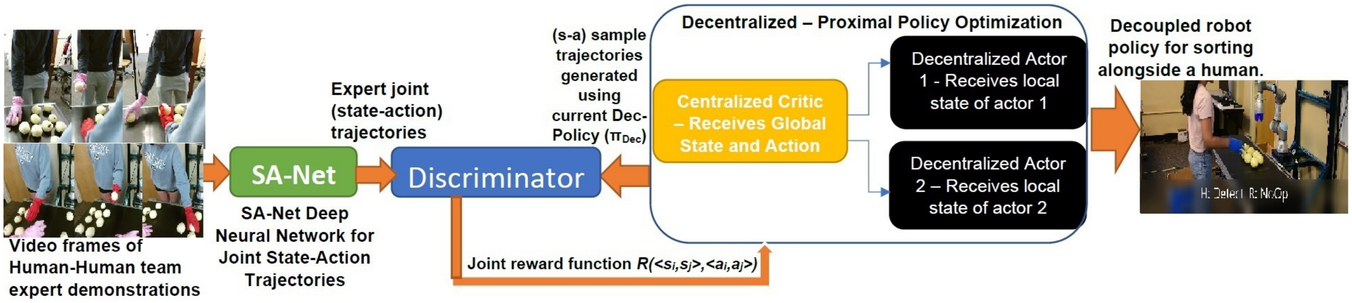

Video frames of Human-Human team expert demonstrations
SA-Net
SA-Net Deep Neural Network for Joint State-Action Trajectories
Expert joint (state-action) trajectories
Discriminator
(s-a) sample trajectories generated using current Dec-Policy (π_{Dec})
Decentralized – Proximal Policy Optimization
Centralized Critic – Receives Global State and Action
Decentralized Actor 1 - Receives local state of actor 1
Decentralized Actor 2 – Receives local state of actor 2
Decoupled robot policy for sorting alongside a human.
H: Detect R: NoOp
Joint reward function $R(<s_i,s_j>,<a_i,a_j>)$

Figure 1: The end-to-end pipeline of Dec-AIRL. Time-synchronized raw video frames of a 2-person team sorting onions on a conveyor is input to SA-Net [31], which classifies the state and action. Each agent's state-action pairs are obtained through the camera opposite to them observing them sort. These are concatenated to form the global state-action pairs, which is used as training data for Dec-AIRL. Dec-AIRL uses a centralized critic and decentralized actors to learn a vector of policies. The decoupled robot policy is used to sort alongside a human.

agent's action, and obtains a game-theoretic equilibrium as the convergence criterion. Our results show that Dec-AIRL significantly improves on the accuracy of both Co-GAIL and MA-AIRL mainly because of the challenge to each agent of inferring the true global state perfectly. On deploying the learned policy in simulation and on a physical cobot, we show that a collaborative robot (cobot) is able to successfully avoid adverse interactions much better than the baseline policies, as it collaborates with a human.

2 BACKGROUND

In a single agent scenario, IRL models the expert's environment as a Markov decision process (MDP) [24], which it solves optimally. Formally, the MDP of an expert is defined as a tuple $\langle S, A, T, R, \gamma, \rho_0 \rangle$, where S is the set of states defining the environment, A is the expert's set of possible actions, $T : S \times A \times S \rightarrow [0, 1]$ gives the transition probabilities from any given state to a next state for each action, $R : S \times A \rightarrow \mathbb{R}$ is the reward function modeling the expert's reward for performing an action from a state, $\gamma \in [0, 1)$ is the discount factor and ρ_0 is the initial state distribution. Typically, the learner is aware of the expert's S, A, γ, ρ_0, and T but not R. Model-free techniques do not require T either. A (stationary) policy for a MDP is a mapping from states to actions $\pi : S \rightarrow A$. Let $\mathcal{X}^E$ be the set of expert demonstrations and a complete trajectory X is given by, $X = (s_1, a_1, s_2, a_2, s_3, ..., s_\mathcal{T}, a_\mathcal{T})$; where $X \in \mathcal{X}^E$.

2.1 Review of Adversarial IRL

AIRL builds on the maximum causal entropy framework [41], which considers an entropy-regularized MDP. Forward RL aims to find the optimal policy π^* that maximizes the expected entropy-regularized discounted reward, under π, T, and ρ_0:

$$\pi^* = \arg\max_{\pi} E_{X\sim\pi}\left[\sum_{t=0}^{\mathcal{T}} \gamma^t \left(r(s_t, a_t) + H(\pi(.|s_t))\right)\right].$$

The distribution of the trajectories derived from the optimal policy $\pi^*(a|s)$ has been shown to take the form $\pi^*(a|s) \propto e^{Q^*_{soft}(s_t,a_t)}$ [14, 41], where:

$$Q^*_{soft}(s_t, a_t) = r_t(s, a) + E_{s_{t+1},...,s_\mathcal{T}\sim\pi}\left[\sum_{t'=t}^{\mathcal{T}} \gamma^t \left(r(s'_t, a'_t) + H(\pi(.|s'_t))\right)\right].$$

The maximum causal entropy framework parameterizes the reward functions $r_\theta(s, a)$ but fixes initial state distribution and the dynamics. The structure of the discriminator is adapted from GCL [7] where the discriminator $D_\theta(X)$ learns a function $f_\theta(X)$ [8] which at optimality approximates the expert policy's advantage function [1]. The softmax policy [11] is given as,

$$\pi(X) = e^{A^{soft}_{f_\theta}(s,a)} = e^{Q^{soft}_{f_\theta}(s,a) - V^{soft}_{f_\theta}(s)} \tag{1}$$

The discriminator takes the form, $D_\theta(X) = \frac{e^{f_\theta(X)}}{e^{f_\theta(X)} + \pi(X)}$ and the reward update rule is given as:

$$R_\theta(X) \leftarrow \log D_\theta(X) - \log(1 - D_\theta(X)). \tag{2}$$

AIRL minimizes the *reverse KL divergence* between the learner's and expert's marginal state-action distribution $KL(\rho_\pi(s, a)||\rho_{exp}$ $(s, a))$ since this results in mode-seeking behavior in contrast to behavior cloning methods that use forward KL divergence with a mode-covering behavior [10]. One key difference between the original formulation by Fu et al [8] and our approach is that we model the reward function as a function of both states and actions as the disentangled reward learning comes with some strong assumptions [9, 11, 34].

2.2 Proximal Policy Optimization

Proximal policy optimization (PPO) is the forward rollout method used by AIRL [8]. PPO improves upon trust-region policy optimization (TRPO) by using a lower-bound on the unclipped objective. More specifically, TRPO maximizes the policy objective: $L(\omega) = E_t[\lambda^t A^t_{\pi_\omega}]$; where A^t is the advantage function (generalized advantage estimate used in practice) at the current timestep and the importance sampling ratio λ^t is given as $\frac{\pi_\omega(a^t|s^t)}{\pi^{old}_\omega(a^t|s^t)}$; where π^{old}_ω is the policy from the previous iteration and ω are the weights for the policy. However, this may lead to an excessively large policy update [29], and PPO suggests the following policy objective:

$$L^{CLIP}(\omega) = E_t\left[min\left(\lambda^t A^t_{\pi_\omega}, clip(\lambda^t, 1 - \epsilon, 1 + \epsilon)A^t_{\pi_\omega}\right)\right]$$

where $L^{CLIP}(\omega)$ is the clipped surrogate objective that takes the minimum of the clipped and unclipped objective to maintain a pessimistic bound over the final objective. Clipping the importance sampling ratio removes the incentive for moving λ^t outside of the interval $[1-\epsilon, 1+\epsilon]$. $A^t_{\pi_\omega}$ is calculated with respect to the reward estimates $R_{\boldsymbol{\theta}}(X)$ from Eqn 2. We generalize this method to a decentralized formulation for use with Dec-AIRL.

3 DECENTRALIZED AIRL FOR HUMAN-ROBOT INTERACTION

This paper presents a novel method that addresses a significant challenge in human-robot interaction (HRI) scenarios with a shared workspace where the human's state is not perfectly observable by the robot. By introducing a decentralized approach combined with IRL, we intend to learn not only the task preferences but also conflict avoidance during interactions from joint expert demonstrations. To understand this HRI scenario, we start by introducing the problem setup, then proceed to explain our approach to the problem, and finally present the Dec-AIRL algorithm that learns a joint reward function, whose optimization in a Dec-MDP yields a vector of policies that simulate the observed behaviors of the agents.

3.1 Interaction Model

We model the observed two-agent interaction as a Dec-MDP [3] that allows for multiple agents to act in a decentralized way while still permitting cooperation. A two-agent Dec-MDP is defined using the following tuple:

$$\mathcal{DM} \triangleq \langle S, A, T, R \rangle$$

where the global state, $S = S_i \times S_j$. Here, S_i and S_j are the locally observed states of the two agents i and j, which when combined yield the complete global state of the system; $A = A_i \times A_j$ is the set of joint actions of the two agents; $T : S \times A \times S \longrightarrow [0, 1]$ is the transition function of the multi-agent system; and $R : S \times A \longrightarrow \mathbb{R}$ is the common reward function. [1] Note that the latter is unknown, whereas rest of the elements of the model are usually known. As such, the agents know their own local state only, any observable parameters relevant to the interaction, and can act individually while optimizing a task-centric common reward [22]. The solution to a Dec-MDP is a vector of policies, $\boldsymbol{\pi}^* = \langle \pi^*_i, \pi^*_j \rangle$, where $\pi^*_i : S_i \longrightarrow A_i$ and analogously for π^*_j. If the interactions between the two agents are sparse, and can occur at some joint states only, we may leverage this domain structure to simplify the model. Let $S_I \in S$ be the set of states where an interaction may occur, and $S_{NI} = S/S_I$ be the remaining states. Then, we may specify the transition and reward functions as:

$$T(s, \boldsymbol{a}, s') = \begin{cases} Pr(s'|s, \boldsymbol{a}) & \text{if } s \in S_I \\ Pr(s'_i|s_i, a_i) \times Pr(s'_j|s_j, a_j) & \text{if } s \in S_{NI} \end{cases}$$

$$R(s, \boldsymbol{a}, s') = \begin{cases} R(s, \boldsymbol{a}, s') \text{ if } s \in S_I \\ R_i(s_i, a_i, s'_i) + R_j(s_j, a_j, s'_j) & \text{if } s \in S_{NI} \end{cases}$$

where $s \in S$, $s' \in S$ and $\boldsymbol{a} \in A$. Thus, we may exploit the transition and reward independence at the non-interaction states.

While our Dec-MDP model characterizes the agents as having local full-observability, there may be scenarios where each agent may be uncertain about its own current partial view. In robotics, this may result from hardware inaccuracies (e.g. sensor noise) or environmental limitations. If no attribute of the system state is hidden from the agents and the combination of agents' observations results in the global state, we may adopt a *jointly fully-observable* Dec-MDP model [3]. Finally, in cases where certain features of the system are never perfectly revealed to the agents the appropriate way to model the environment would be as a Dec-POMDP.

3.2 IRL from Team Demonstrations

Let the joint experts' demonstration be given as:

$$\mathcal{X}^E = (\langle s^0_i, s^0_j \rangle, \langle a^0_i, a^0_j \rangle, \langle s^1_i, s^1_j \rangle, \langle a^1_i, a^1_j \rangle, ..., \langle s^T_i, s^T_j \rangle, \langle a^T_i, a^T_j \rangle).$$

As we mentioned in Section 2, AIRL makes use of state-action samples drawn from both the expert (given) and the currently learned data, which is used by the discriminator to compute a binary cross entropy (BCE) loss at each iteration. Let the expert's demonstrations be $\mathcal{X}^E$ (as shown above) and trajectories drawn from the currently learned decentralized policy $\hat{\boldsymbol{\pi}}$ be denoted as $\hat{\mathcal{X}}$. Samples are drawn from $\mathcal{X}^E$ and $\hat{\mathcal{X}}$ at every iteration and used to minimize the distance between the expert's and learned state-action marginal distributions. The reward is updated using the discriminator's confusion [11] and the updated reward is sent to the generator. The latter uses a decentralized generalization of PPO [29], which we call Dec-PPO, which follows the same CTDE architecture as MA-PPO [39].

Dec-PPO generalizes PPO from Section 2.2. The centralized critic network updates the value function as a squared-error loss:

$$L^{VF}_t(\omega) = \left((V^{\pi_\omega}(s^t) - \hat{V}^{targ}_t)^2 \right)]$$

where $\hat{V}^{targ}_t$ is the discounted reward-to-go obtained during the corresponding episode at timestep t and $V^{\pi_\omega}(S^t)$ is the predicted value of global state s^t (Section 3.1). Let,

$$\lambda^t_i = \frac{\pi_{\omega,i}(a^t_i|s^t_i)}{\pi^{old}_{\omega,i}(a^t_i|s^t_i)} \text{ and } \lambda^t_j = \frac{\pi_{\omega,j}(a^t_j|s^t_j)}{\pi^{old}_{\omega,j}(a^t_j|s^t_j)}$$

Then the policy loss of the decentralized actor networks for the agents i and j respectively is given as:

$$L^{CLIP}_i(\omega) = E_{(s^t_i, a^t_i) \sim \pi_{\omega,i}} \left[min \left(\lambda_i \, A^{\pi_{\omega,i}}, clip(\lambda_i, 1-\epsilon, 1+\epsilon) A^{\pi_{\omega,i}} \right) \right]$$
$$L^{CLIP}_j(\omega) = E_{(s^t_j, a^t_j) \sim \pi_{\omega,j}} \left[min \left(\lambda_j \, A^{\pi_{\omega,j}}, clip(\lambda_j, 1-\epsilon, 1+\epsilon) A^{\pi_{\omega,j}} \right) \right].$$

The policy entropy loss is added to the clipped surrogate objective to promote exploration. This is given as:

$$L^{ENT}_i(\omega) = \sigma H \left[\pi_{\omega,i}(s^t_i) \right] \text{ and } L^{ENT}_j(\omega) = \sigma H \left[\pi_{\omega,j}(s^t_j) \right]$$

where H is the policy entropy and σ is the entropy hyperparameter. The total loss for the two actors is then given as:

$$L_i(\omega) = L^{CLIP}_i(\omega) + L^{ENT}_i(\omega)$$
$$L_j(\omega) = L^{CLIP}_j(\omega) + L^{ENT}_j(\omega)$$

At the end of training, the discriminator returns the learned common reward function while the generator provides the converged vector of policies.

[1] Thus, our Dec-MDP is a locally fully observable model whose local states when combined yield the fully observable global state, per Goldman and Zilberstein's categorization of such decentralized models [12].

```
Algorithm 1: Dec-AIRL
    Input: $\mathcal{DM}$ sans $R$ and $T$; Exp trajs $\mathcal{X}^E$.
    Output: Learned common reward function $R$
 1  Initialize decentralized policy $\boldsymbol{\pi}$, Discriminator $D_{\boldsymbol{\theta}}$.
 2  for $iter \leftarrow 0$ to $train\_iters$ do
 3      Generate joint trajectories $\hat{\mathcal{X}}$ using $\boldsymbol{\pi}$
 4      Sample joint $(s, a)$ pairs $\hat{\mathcal{Y}}$, $\mathcal{Y}^E$ from $\hat{\mathcal{X}}$, $\mathcal{X}^E$, respectively.
 5      for $ep \leftarrow 0$ to $discriminator\_epochs$ do
 6          Train discriminator $D_{\boldsymbol{\theta}}$ via BCE to classify expert
             data $\mathcal{Y}^E$ from learned policy data $\hat{\mathcal{Y}}$.
 7      Update reward
         $\hat{R} \leftarrow \log D_{\boldsymbol{\theta}}(S, A, S') - \log(1 - D_{\boldsymbol{\theta}}(S, A, S'))$
 8      for $ep \leftarrow 0$ to $generator\_epochs$ do
 9          Train generator $G(\hat{R}) \leftarrow$ Dec-PPO.
10      Get updated policy $\boldsymbol{\pi}^{\boldsymbol{L}} \leftarrow G(\hat{R})$.
11  return $\hat{R}$, $\boldsymbol{\pi}^{\boldsymbol{L}}$
```

3.3 Algorithm

The algorithm for Dec-AIRL (Algorithm 1) takes the model $\mathcal{DM}$ without the reward function and transition function (as Dec-AIRL is model free), the expert trajectories $\mathcal{X}^E$ as input. The aim is to learn the common reward function for the task R. It starts by generating a random decentralized policy vector $\hat{\boldsymbol{\pi}}$ (line 1), and a discriminator $D_{\boldsymbol{\theta}}$ with random weights is initialized. The algorithm begins its iterative updates until the end of training iterations (line 2). In each iteration, joint trajectories $\hat{\mathcal{X}}$ of the agents are generated using the current policy vector $\hat{\boldsymbol{\pi}}$, followed by picking some state-action pairs $\hat{\mathcal{Y}}$ from $\hat{\mathcal{X}}$ and some state-action pairs $\mathcal{Y}^E$ from $\mathcal{X}^E$ (line 4). Using these samples, the discriminator is trained to differentiate between the expert and learned sample data using BCE loss (line 6). From the trained discriminator, the updated reward $\hat{R}$ is then extracted in line 7. Next, the generator $G(\hat{R})$ is trained with a centralized critic and decentralized actors based Dec-PPO to generate the forward rollout vector of policies. Finally, the learned reward function $\hat{R}$ and converged policy $\boldsymbol{\pi}^L$ are returned by the algorithm.

3.4 Human-Robot Collaborative Task Execution

Notice that in Algorithm 1 Dec-AIRL yields the learned reward function as well as the learned policy vector, $\boldsymbol{\pi}^L$. From this vector, the policy of the agent being replaced by the robot is then used to control the robot in the collaboration. The environment state at each time step is perfectly observed through the robot's sensors and the corresponding action from the decentralized robot policy is performed by the robot. This execution continues until the task is completed or a goal state is reached. To illustrate, in a simulated assistive domain, the robot's policy would map the state obtained from the simulation to the target robot's joint torques. Whereas for the real-world produce sorting domain, discrete state values consisting of the onion location, blemished or not, and the location of the human's gloved hand is mapped to a high-level action, which is then performed by the robot. Section 4 explains the specifics in more detail.

4 EXPERIMENTS

We implemented Algorithm 1 in Python by extending an existing AIRL implementation [36]. Our codebase is freely available on GitHub at https://github.com/prasuchit/Dec-AIRL. We evaluate Dec-AIRL using two domains, one of which is a simulation while the other uses human agents and a physical collaborative robot.

4.1 Simulated Patient Assistance

Our formative evaluation domain is available within the simulated open-source *assistive-gym* [6], which consists of multiple environments that involve a cobot assisting a patient with basic needs such as bathing, scratching, feeding, and dressing. These environments are built on top of OpenAI's gym testbed [5], and available in an MMDP format in version 0.1 and a Dec-MDP format in the latest version. For the purpose of this paper, we chose an environment that allows both the human and robot to act at every time step, and consists of both necessary and adverse interactions.

Setup. A Sawyer robotic arm (7 DOF cobot from Hahn Robotics previously Rethink Robotics) must safely and successfully feed a human, while the human also moves towards the spoon to receive it, as shown in Fig. 2. Each agent – robot and human – follow a learned policy to successfully complete the task because the start state of both agents is not fixed and they do not follow predetermined paths to reach their respective goals. Therefore, an agent's action at every time step may be contingent on the other agent's action.

The task leads to several interactions between the agents: a *necessary interaction* is where the robot brings the spoonful of food in contact with the human's mouth and the human moves towards the robot to cooperate. On the other hand, the robot touching the human anywhere other than the mouth, either with the spoon or its arm, is considered an *adverse interaction.* The resulting penalty is proportional to the force of contact. Figure 2c shows an example of an adverse interaction where the robot touches the human near the eye with the spoon while following a poorly learned policy.

The robot and human **state** is composed of the following values respective to the agent: *spoon: pose and orientation, human head: pose and orientation, spoon distance from human mouth, body joint angles, and the spoon force on the human.* The dimension of **actions** for the human and robot are 4 and 7 respectively based on the number of mobile joints. The state S in the Dec-MDP is a concatenation of the human's and robot's states.

Performance evaluation. For Dec-AIRL, we used 256 hidden nodes for the 2 hidden layers in both the g and h networks within the discriminator, a batchsize of 128, learning rate for both the discriminator and the actors as 0.0003, discount factor and generalized advantage estimator (λ) both as 0.95, max gradient normalization as 0.5, and a random seed between 1 and 100. For Dec-PPO, we used the same hyperparameters as mentioned in gSDE [25] because it has been tuned and tested on multiple pybullet environments. The default hyperparameters were used for Co-GAIL and MA-AIRL.

The expert trajectories were obtained from a trained RL model. We compared the performance of Dec-AIRL with two known baselines, MA-AIRL and Co-GAIL. All three algorithms were provided 10^5 steps of expert data $\mathcal{X}^E$ with each episode lasting a maximum of 200 time steps though usually less, and trained for 10^9 iterations.

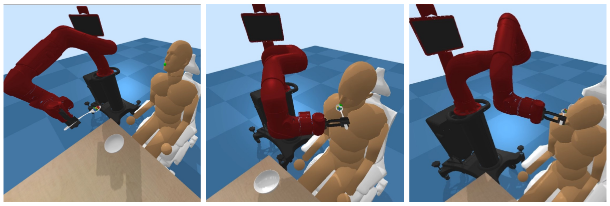

(a) (b) (c)

Figure 2: (a) Sawyer picking up the food from the table. (b) Feeding it to the human in the chair while the human also moves towards the spoon to cooperate. (c) The cobot encountering an adverse interaction by touching the human near the eye.

Method	# steps	# adverse interactions	Total reward
Expert	24.5	4	148.6
MA-AIRL	131.2	12	108
Co-GAIL	164.5	19	91.4
Dec-AIRL	**35**	**7**	**143.2**

Table 1: Dec-AIRL improves on both baseline methods significantly and is closest to the expert's performance.

The forward RL step in MA-AIRL and Co-GAIL requires to be run on perfectly observed global state-action pairs, while the Dec-PPO in Dec-AIRL was run such that data that was observable to each agent only was provided to it. All three learned policies were deployed in a setting where the human state was perturbed using a small offset of 0.1 to depict a realistic perception through sensors like RGB-D cameras.

Table 1 compares the methods on the number of steps to complete the feeding, number of adverse interactions, and reward accrued per episode, averaged across 100 episodes. The results show that Co-GAIL and MA-AIRL converge to a joint policy that performs poorly in interacting states as compared to Dec-AIRL and this difference is statistically significant (Wilcoxon rank-sum test on reward accrued and number of adverse interactions, all p-values < 0.0005). Both MA-AIRL and Co-GAIL yield policies that prescribe actions conditioned on the global state. In other words, due to the choice of the underlying model (MMDP or Markov game), the robot is assumed to have access to the human's joint angles and by extension the accurate pose and orientation of the human head; it must also assess the force experienced by the human due to the spoon. However, these may not be observed perfectly in practice by the robot leading to suboptimal actions. As Dec-AIRL yields a decentralized policy for the robot, which depends only on the parameters that it can observe, its performance is better aligned with the situation. Therefore, not only does Dec-AIRL generate less adverse interactions, it also accrues a better reward per episode compared to the baselines. Figure 3 shows the comparison between the Dec-AIRL policy and the baseline policies when the robot is close to the human's mouth.

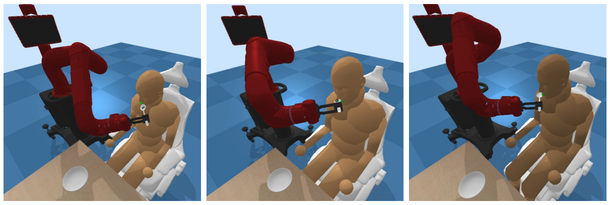

(a) (b) (c)

Figure 3: Comparison of the three learned policies implemented on an environment where the human's pose is not perfectly observed. (a) Co-GAIL policy and (b) MA-AIRL policy reach close to the mouth but fail to accurately feed the human and also drop some food, while (c) Dec-AIRL policy takes a slightly longer path to the mouth than the expert but manages to feed the human correctly while avoiding adverse interactions.

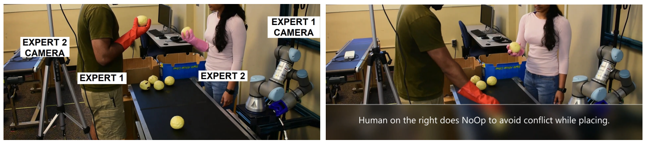

EXPERT 2 CAMERA
EXPERT 1 CAMERA
EXPERT 1
EXPERT 2
Human on the right does NoOp to avoid conflict while placing.
(a) (b)

Figure 4: Demonstration setup: Snapshots of a human-human (expert) demonstration where the two agents stand on opposite sides of a conveyor and sort faux onions. Observe that the camera placed behind expert 1 observes expert 2's state and actions, and vice versa. (a) Both agents inspecting their respective onions during the sort. (b) Expert 2 does a No-Op action (pauses) to avoid conflict in a shared workspace while placing on conveyor.

Notice how with the baseline policies, the cobot reaches close but is not able to reach the mouth accurately. Meanwhile, with Dec-AIRL policy, the cobot is able to feed the human correctly, spills less food in the process, and avoids adverse interactions.

4.2 Human-Robot Line Sorting

Our summative evaluation domain is a use-inspired human-robot line sorting task where the physical cobot UR3e (from Universal Robots) is tasked with sorting onions along with a human on a conveyor belt after observing a human-human team perform the task. A key challenge to a successful sort in the shared workspace is for both agents to avoid reaching for an onion on the table simultaneously or placing a good onion back on the table in tandem after inspection — we refer to these situations as adverse *interactions*.

We model the collaborative onion-sorting domain as a discrete Dec-MDP sans the reward function, which is learned. The factored local **state** for each agent is captured by 4 key variables yielding a total of 96 local states: *Onion_location*:{unknown, on conveyor, hover location, in front of face}; *EndEffector_location*:{on conveyor, hover location, in front of face, in bin}; *Prediction*:{unknown, good, bad} and *Interaction*:{true, false}. The value 'hover location' is a region of space just on top of the conveyor and 'in bin' indicates just inside a bin that holds discarded onions. The interaction variable is activated for states where the other agent also reaches out simultaneously to pick or place an onion on the table. An example global state where the robot's target onion is on the table, its end-effector is in the hover location, the onion prediction is bad, both the human's onion and hand are in front of the face, the prediction is good, and the agents are not in an interaction state, would be represented as ((on conveyor, hover location, bad, false),(in front, in front, good, false)). Prior to detection, each onion's location and status is unknown.

Each sorter performs one of six abstract **actions** in the Dec-MDP: *Detect*, which shifts the sorter's focus to a new onion on the table and also gives a preliminary prediction, *Pick up the onion* which is self explanatory; *Inspect* the onion by rotating it and checking for blemishes; *Place onion on the conveyor* after it is picked and found to be unblemished; otherwise *Place onion in the bin* after it is picked; *No-Operation* to wait for one timestep in the same state. Both the agents have the same set of local state variables and local actions. [2] The Dec-MDP model is implemented as an environment in MA-Gym [17] for convenient dissemination.

Setup. Two depth cameras alongside each agent observe and record the other agent (human or robot) standing opposite (see Fig. 4). The time-synchronized, fused RGB-D frames from these cameras are quantized into appropriate state variables by SA-Net [31] while a trained YOLOv5 deep neural network model [27] is used to detect and classify the onions as blemished or not, *which is not perfect.* These state recognitions are shown in red text on the frames in Fig. 5. For Dec-AIRL, we used the same hyperparameters as mentioned in Section 4.1 except for the size of hidden nodes in the g and h networks, which we modified to 128 hidden nodes for the 2 hidden layers because the Dec-MDP model of agent sorting is smaller and discrete compared to the assistive domain.

Performance evaluation. We recorded 10 rounds of sorting by 4 different teams of 2 humans each. Each round is a sort of a batch of 10 onions. This yielded 40 trajectories with an average of 650 frames per trajectory, containing a total of about 25,000 joint state-action pairs. Dec-AIRL's performance is again compared with the two baselines: Co-GAIL [35] and MA-AIRL [40]. To obtain the global state on which the MMDP policy conditions its actions, the RGB-D frames from the two cameras are time synchronized through ROS at each time step. All three methods were run for 10^5 iterations, which resulted in near convergence. We compare their performance using the empirical *learned behavior accuracy* (LBA), a previously-used metric [2] that gives the proportion of prescribed actions from the learned policy for an agent which matches those of the optimal policy. We extend this metric to both agents. Figure 6 shows the LBAs for increasing numbers of input state-action pairs, which yields agent behaviors of varying quality. Observe that, (*i*) the LBA of all three methods improves as more training data is available, as we may expect; and (*ii*) Dec-AIRL's inversely learned behavior for the agents improves significantly on that learned by MA-AIRL and Co-GAIL for various sets of input trajectories. One reason for this improvement is that MA-AIRL and Co-GAIL learn policies which prescribe actions that led to more adverse interactions compared to the actions from Dec-AIRL. The best policies yielded 153 interactions for Co-GAIL and 96 interactions for MA-AIRL compared to 27 for Dec-AIRL across 2,304 states of the sort. A key reason for this is that the human's blemished onion assessment, which is available during training as part of the global state, cannot be observed during execution. It may lead the cobot to pick an onion thinking that the human will discard its picked onion into the bin. However, if the noisy assessment has *mislabeled* the onion being inspected by the human as blemished, the human actually places it back on the table leading to a collision (for illustration, see Fig. 7).

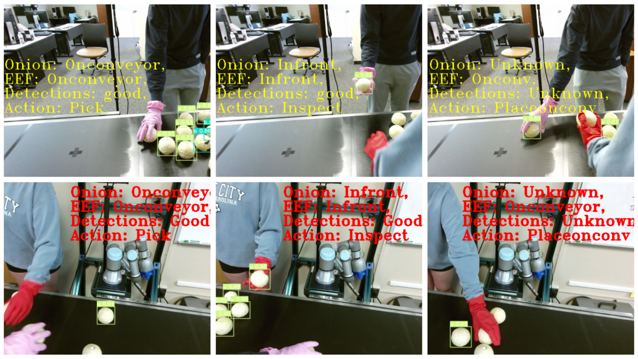


Figure 5: Some of the image frames that compose the joint state-action trajectories $\mathcal{X}^E$ via SA-Net [31]. The annotations on the images are the state-action predictions and the bounding boxes are generated by YOLOv5. The first frame captures when the onion is on the conveyor and the human expert is about to pick. The second captures the human inspecting and finding a good onion. The third is when the human is placing the onion back on the conveyor.

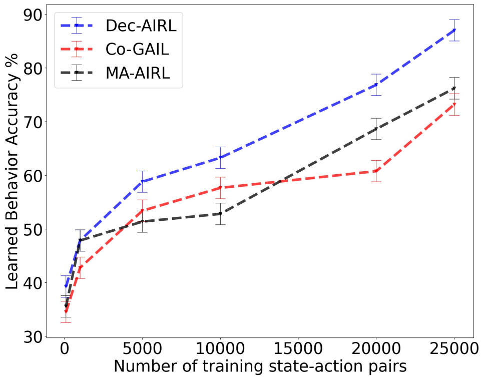
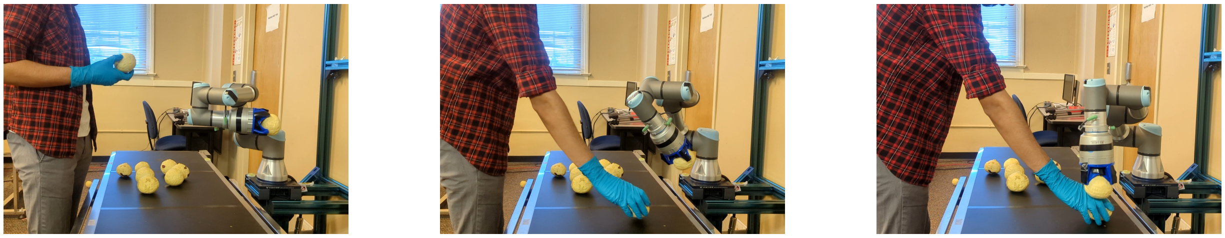


Figure 6: The empirical LBA comparison between Dec-AIRL, MA-AIRL and Co-GAIL with varying levels of training data. 25000 states from SA-Net trajectories were used to obtain the robot action from these policies and compared to the ground truth robot action.

[2] Note that our Dec-MDP model does not provide low-level control of the cobot's motion, which allows its variables to remain discrete. We follow a pipeline programming architecture in which the Dec-MDP's abstract actions map to motion planners in configuration space and plan in real time.



(a) (b) (c)

Figure 7: Adverse interaction from a MA-AIRL learned policy. (a) Both the cobot and human are engaged in inspection. (b) The human decides to place the onion back on the conveyor because he assesses it as a good onion. Meanwhile, the cobot is also placing its onion back because it wrongly estimated the global state as it cannot see the onion from the human's perspective. This particular onion had a blemish not visible while on the conveyor, which led the cobot to (erroneously) reason that the human will discard the onion. (c) Both agents place their onion on the conveyor thereby encountering an adverse interaction.

Table 2: Average precision and recall from 5 cobot-human sorting sessions with 10 onions sorted per session using policies learned via MA-AIRL and Dec-AIRL. The cobot-human team's performance with Dec-AIRL policy improves on MA-AIRL performance in terms of recall. TP - True positive, FP - False positive, TN - True negative, FN - False negative.

Method	(TP,FP,TN,FN)	Precision	Recall
Human-human	(4,1,5,0)	0.8	1.0
Human-cobot (MA-AIRL)	(3,2,3,2)	0.6	0.6
Human-cobot (Dec-AIRL)	**(3,2,4,1)**	**0.6**	**0.75**

As MA-AIRL exhibits a higher LBA compared to Co-GAIL, we selected MA-AIRL as the baseline for comparison with Dec-AIRL on the physical sorting. We deployed the best performing decentralized policy of one of the agents to control a physical cobot UR3e to guide collaboration with a human to sort faux onions. We ran 5 rounds of cobot-human sorting where lab members (not involved in this paper) played the role of the human sorter. Our evaluation focuses on how well the cobot sorts in the team and whether it avoids adverse interactions; the human sorters follow a fixed behavior in sorting the onions (see the supplementary file for more details). The cobot performs the sort by receiving the bounding boxes from YOLO v5, on which techniques such as central orthogonal projection, direct linear, and affine transforms are used to obtain the coordinates of the onions in its 3D workspace. The cobot and the human each pick up the onions and place them either in the bin or back on the conveyor table after inspection. We measure the team's sorting performance using the domain-specific metrics of precision and recall where *precision = TP/(TP + FP)* and *recall = TP/(TP + FN)*. Here, true positive (TP) is the count of blemished onions in the bin, false positive (FP) is the number of good onions in bin, true negative (TN) is the number of good onions remaining on table, and false negative (FN) is the count of blemished onions remaining on table. From Table 2, we note that Dec-AIRL leads to the cobot-human team exhibiting recall that is better than that of MA-AIRL but still somewhat far out from the precision and recall of the human-human team, which indicates room for improvement. Figure 8 shows some frames from the cobot-human collaborative sort when the cobot is using the Dec-AIRL-learned policy.

5 RELATED WORK

Recently, few IRL techniques have targeted the problem of multi-agent coordination/cooperation. They model the problem either as a game-theoretic equilibrium, solve an MMDP with the assumption that all agents have access to the global state, or just consider the other agent(s) as part of the transition noise in the environment [23].

Multi-agent imitation learning. Imitation learning techniques that deal with multi-agent tasks include MAGAIL [32], CoDAIL [21], MA-DAAC [16], and Bayesian MA-DAAC [38]. These techniques extend GAIL to a multi-agent setup. MAGAIL and CoDAIL model the problem as a Markov game with Nash equilibrium as the convergence criteria. MA-DAAC aims to improve the scalability of multi-agent GAIL by representing the generator with an actor-attention-critic instead of the regular actor-critic and using a centralized or decentralized discriminator. All of these methods evaluate their approach on multi-agent particle environments (a simulated platform by OpenAI containing small multi-agent domains that do not involve adverse interactions) in both cooperative and competitive setups. Recently, Co-GAIL [35] extended InfoGAIL [19] to model human-robot interaction scenarios with multiple human strategies. They model it as an MMDP, where all agents have access to the global state, and assume that the latent feature, the human strategy, is shared between agents. Co-GAIL is implemented on iGibson's simulated human-robot domains and the expert trajectories are recorded using Amazon turk. BTIL [30] uses a task model and agent model and infers the time-varying mental states of team members. It learns decentralized team policies from demonstrations of sub-optimal teamwork. It also jointly learns their transition models. BTIL was evaluated on a single movers-and-packers domain, which is a relatively small and simulated platform. Although, the paper claims to learn a decentralized policy, their policy is a mapping of the global task state to agent action, which is not a fully decentralized representation. *None of these techniques utilize a decentralized MDP as the underlying model of the task*, which is better suited for modeling problems where an agent cannot perceive the global state.

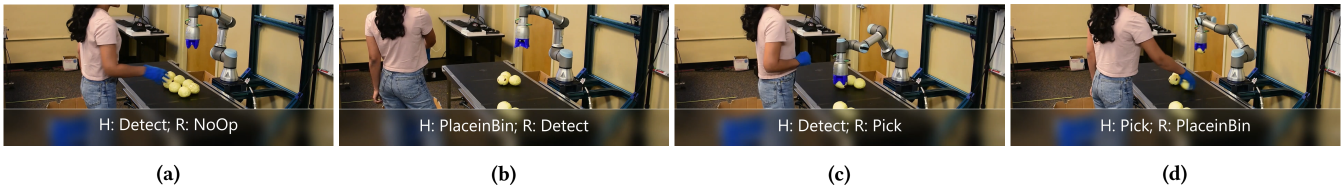


Figure 8: Frames from collaborative sort with the cobot using a Dec-AIRL learned policy. (a) The cobot recognizes that the human is about to pick and does No-Op to avoid an adverse interaction. (b) The cobot pauses to choose an onion (detect), while the human places her onion in the bad onion bin. (c) The human pauses to choose an onion (detect) on the conveyor, while the robot picks the next onion. (d) The human picks the chosen onion while the robot places its onion in the bad bin.

They also do not evaluate the method on domains that require a human to actively interact with a robot.

Multi-agent inverse RL. In the IRL context, MA-AIRL [40], ME-AIRL [37], and MA-AIRL-Latent [13] extend AIRL to a multi-agent setup. MA-AIRL models the problem under logistic stochastic best response equilibrium (also known as the quantal response) and tests the method on multi-agent particle environments. MA-AIRL-Latent aims to improve the sample efficiency of MA-AIRL when the agents have shared latent variables, and evaluates on highway trajectories from the highD data set [18] and trajectories of aircraft in terminal airspace from non-public Federal Aviation Administration (FAA) data (also simulated domains). ME-AIRL uses an analytical method to connect the actor-critic model with AIRL and GAIL and measures the performance of the method on a simple single-state continuous game for 2 agents with one action dimension per agent. Older techniques such as Joint-Action-IRL [23] and BA-IRL [33] consider the other agent to be part of the environment as transition noise and model the IRL problem as a single-agent scenario. Joint-Action-IRL implements the method on a hand-finishing task where the human's role is to refinish the surface of a box attached to an ABB industrial robot and the robot moves to aid the process. BA-IRL evaluates their method on a ROS/Gazebo simulation of two Turtlebots trying to sort balls on a table using attached manipulators. An early technique GSSG-IRL [26] models decentralized, non-cooperative multiple agents using IRL, but formulates the problem as a Nash equilibrium based general-sum stochastic game between the agents and tests the technique on a simple Gridworld domain only. The key difference between GSSG-IRL and Dec-AIRL is that in our case, *the agents are cooperative and maximize a common system reward instead of working in self-interest.* Additionally, we test our method on both a simulated continuous domain and a use-inspired discrete domain with human-robot collaboration scenarios that model adverse interactions.

6 CONCLUDING REMARKS

Motivated by the paradigm of learn-from-observation to learn human-robot collaborations, we presented a novel multi-agent IRL method that learns from human-human demonstrations to allow a cobot to work collaboratively with a human while avoiding adverse interactions within a shared workspace. We highlighted the importance of using a decentralized MDP to model the learning problem as compared to the previous game-theoretic approaches or using a multi-agent MDP model. Results show that the Dec-AIRL outperforms the previous multi-agent MDP based imitation learning method and a Markov game based IRL technique. Dec-AIRL also performs satisfactorily when compared with a human-human sorting team. We have also shown generalizability of our technique across two domains, one being continuous and simulated and the other being discrete and realistic. Thus, Dec-AIRL could pave the way for facilitating future cobot deployments in human-robot collaborative scenarios involving shared workspaces.

An avenue for future work is to learn type-contingent reward functions where the human teammate may be of varying types engaging in differing ways of completing the task based on her type. A type-contingent reward function will let the cobot adapt its behavior to the different types of teammates. Another possible extension of this work would be to model the human (demonstrators and/or co-worker(s)) as bounded rational instead of fully rational. This could be done by solving the model attributed to them approximately (to their level of reasoning) or by maintaining a model of the human that also factors into the policy training. Additionally, Dec-AIRL models a simultaneous action multi-agent scenario though the human and cobot complete some of their actions at different rates. As such, the Dec-AIRL may need to be generalized to reason with time-varying actions.

In certain cases, the expert (human) demonstrations may not be perfectly observable, or environmental obstacles may cause occlusions. Existing IRL methods use expectation-maximization or marginalization to deal with these problems. Alternatively, we could use human-in-the-loop techniques to receive clarifications or more demonstrations for the uncertain portions. Finally, there may be cases where the human teammate may exhibit divergent behavior than what was observed in the expert demonstrations. In such cases, one approach is to start with the policy learned using the IRL reward function as a starting point and perform online RL to accommodate these new changes progressively.

Some additional domains where Dec-AIRL could be applied include collaborative assembly tasks (e.g. furniture assembly), independent tasks within a shared workspace like an automation line, and emergency search-and-rescue scenarios. As such, Dec-AIRL can be extended and utilized in many other avenues to solve a variety of problems in human-robot collaboration in shared workspaces.

ACKNOWLEDGMENTS

This research is supported in part by NSF grant #IIS-1830421 and a Phase 1 grant from the GA Research Alliance. We thank the demonstrators and other experimenters for their assistance.

REFERENCES

[1] Rushdi Alsaleh and Tarek Sayed. 2021. Markov-game modeling of cyclist-pedestrian interactions in shared spaces: A multi-agent adversarial inverse reinforcement learning approach. *Transportation research part C: emerging technologies* 128 (2021), 103191.

[2] Saurabh Arora and Prashant Doshi. 2021. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence* 297 (2021), 103500.

[3] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The Complexity of Decentralized Control of Markov Decision Processes. *Math. Oper. Res.* 27, 4 (Nov. 2002), 819–840.

[4] Craig Boutilier. 1996. Planning, learning and coordination in multiagent decision processes. In *TARK*, Vol. 96. Citeseer, Unknown, None, 195–210.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540

[6] Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C. Karen Liu, and Charles C. Kemp. 2020. Assistive Gym: A Physics Simulation Framework for Assistive Robotics. *IEEE International Conference on Robotics and Automation (ICRA)* 1, None (2020), 0–0.

[7] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. *arXiv preprint* arXiv:1603.00448 (2016), 0.

[8] Justin Fu, Katie Luo, and Sergey Levine. 2018. Learning Robust Rewards with Adverserial Inverse Reinforcement Learning. In *International Conference on Learning Representations*. unknown, None, 1–10. https://openreview.net/forum?id=rkHywl-A-

[9] Sinong Geng, Houssam Nassif, Carlos Manzanares, Max Reppen, and Ronnie Sircar. 2020. Deep PQR: Solving inverse reinforcement learning using anchor actions. In *International Conference on Machine Learning*. PMLR, unknown, unknown, 3431–3441.

[10] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. 2020. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*. PMLR, unknown, unknown, 1259–1277.

[11] Adam Gleave and Sam Toyer. 2022. A Primer on Maximum Causal Entropy Inverse Reinforcement Learning. *arXiv preprint arXiv:2203.11409* 1 (2022), arxiv.

[12] Claudia V. Goldman and Shlomo Zilberstein. 2003. Optimizing Information Exchange in Cooperative Multi-agent Systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (Melbourne, Australia) *(AAMAS '03)*. ACM, New York, NY, USA, 137–144.

[13] Nate Gruver, Jiaming Song, Mykel J Kochenderfer, and Stefano Ermon. 2020. Multi-agent adversarial inverse reinforcement learning with latent variables. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. None, None, 1855–1857.

[14] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*. PMLR, unknown, unknown, 1352–1361.

[15] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *Advances in neural information processing systems* 29 (2016), None.

[16] Wonseok Jeon, Paul Barde, Derek Nowrouzezahrai, and Joelle Pineau. 2020. Scalable multi-agent inverse reinforcement learning via actor-attention-critic. *arXiv preprint arXiv:2002.10525* None, None (2020), None.

[17] Anurag Koul. 2019. ma-gym: Collection of multi-agent environments based on OpenAI gym. https://github.com/koulanurag/ma-gym.

[18] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. 2018. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, None, None, 2118–2125.

[19] Xiaomin Lin, Peter A Beling, and Randy Cogill. 2017. Multiagent inverse reinforcement learning for two-person zero-sum games. *IEEE Transactions on Games* 10, 1 (2017), 56–68.

[20] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, Vol. 157. None, None, 157–163.

[21] Minghuan Liu, Ming Zhou, Weinan Zhang, Yuzheng Zhuang, Jun Wang, Wulong Liu, and Yong Yu. 2020. Multi-agent interactions modeling with correlated policies. *arXiv preprint arXiv:2001.03415* None, None (2020), None.

[22] Francisco S Melo and Manuela Veloso. 2011. Decentralized MDPs with sparse interactions. *Artificial Intelligence* 175, 11 (2011), 1757–1789.

[23] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. 2015. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, None, None, 189–196.

[24] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.

[25] Antonin Raffin, Jens Kober, and Freek Stulp. 2022. Smooth exploration for robotic reinforcement learning. In *Conference on Robot Learning*. PMLR, None, None, 1634–1644.

[26] Tummalapalli Sudhamsh Reddy, Vamsikrishna Gopikrishna, Gergely Zaruba, and Manfred Huber. 2012. Inverse reinforcement learning for decentralized non-cooperative multiagent systems. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, None, None, 1930–1935.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. None, None, 779–788.

[28] Stuart Russell. 1998. Learning Agents for Uncertain Environments (Extended Abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (Madison, Wisconsin, USA) *(COLT' 98)*. ACM, New York, NY, USA, 101–103.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* None, None (2017), None.

[30] Sangwon Seo and Vaibhav V. Unhelkar. 2022. Semi-Supervised Imitation Learning of Team Policies from Suboptimal Demonstrations. *IJCAI* abs/2205.02959 (2022), 2492–2500. https://doi.org/10.24963/ijcai.2022/346

[31] Nihal Soans, Ehsan Asali, Yi Hong, and Prashant Doshi. 2020. Sa-net: Robust state-action recognition for learning from observations. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, None, None, 2153–2159.

[32] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. 2018. Multi-agent generative adversarial imitation learning. *Advances in neural information processing systems* 31 (2018), None.

[33] Maulesh Trivedi and Prashant Doshi. 2018. Inverse learning of robot behavior for collaborative planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, None, None, 1–9.

[34] David Venuto. 2020. *Robust Adversarial Inverse Reinforcement Learning with Temporally Extended Actions*. McGill University (Canada), None.

[35] Chen Wang, Claudia Pérez-D'Arpino, Danfei Xu, Li Fei-Fei, Karen Liu, and Silvio Savarese. 2022. Co-GAIL: Learning Diverse Strategies for Human-Robot Collaboration. In *Conference on Robot Learning*. PMLR, unknown, unknown, 1279–1290.

[36] Toshk Watanabe. 2020. A PyTorch implementation of GAIL and AIRL based on PPO. https://github.com/ku2482/gail-airl-ppo.pytorch.

[37] Ermo Wei, Drew Wicke, and Sean Luke. 2019. Multiagent Adversarial Inverse Reinforcement Learning.. In *AAMAS*. None, None, 2265–2266.

[38] Fan Yang, Alina Vereshchaka, Changyou Chen, and Wen Dong. 2020. Bayesian multi-type mean field multi-agent imitation learning. *Advances in Neural Information Processing Systems* 33 (2020), 2469–2478.

[39] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. *arXiv preprint arXiv:2103.01955* 1, None (2021), None.

[40] Lantao Yu, Jiaming Song, and Stefano Ermon. 2019. Multi-agent adversarial inverse reinforcement learning. In *International Conference on Machine Learning*. PMLR, None, None, 7194–7201.

[41] Brian Ziebart. 2010. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Ph.D. Dissertation. Carnegie Mellon University.