Mixed-Initiative Multiagent Apprenticeship Learning for Human Training of Robot Teams – Supplementary Material

Esmaeil Seraj
Georgia Institute of Technology
eseraj3@gatech.edu

Jerry Xiong
Georgia Institute of Technology
jxiong60@gatech.edu

Mariah Scrum
University of California Berkeley
mariahschrum@berkeley.edu

Matthew Gombolay
Georgia Institute of Technology
matthew.gombolay@cc.gatech.edu

1 Algorithm Details and Pseudocode

In Algorithm 1, we define MixTURE's training process.

Algorithm 1 MixTURE Training

For each agent j for each class c
  ▷ obtain expert trajectories $\tau_E^{(c_j)}$
  ▷ initialize policies $\pi_\phi^{(c_j)}$ and discriminators $D_\theta^{(c_j)}$.

while not converged **do**
  Collect trajectories $\tau = \{(\bar{o}_t, \bar{a}_t)\}_{t=1}^{T}$ by executing policies $\pi_\phi^{(c_j)}$ for each agent j, class c,
  storing communications z_{ij} between each pair of agents i, j
  Predict rewards $r^{(c_j)}(\bar{o}_t, \bar{a}_t) \leftarrow \log(D)^{(c_j)}(\bar{o}_t, \bar{a}_t)$
  Update $\pi_\phi^{(c_j)}$ using objective (4)
  Train $\mathcal{D}_\theta^{(c_j)}$ to classify expert trajectories τ_E from collected trajectories τ using loss (1)
end while

We note that we publicly provide our codebase (including MixTURE implementation, the environment implementations, the expert heuristics, and the baselines) at https://github.com/CORE-Robotics-Lab/MixTURE.

2 Environment Details

2.1 Predator-Prey (PP)

The objective within this homogeneous environment is for $\mathcal{N}$ predator agents with limited vision to find a stationary prey and move to its location. The agents in this domain are homogeneous in their state, observation, and action spaces and thus, all agents are of the same *class*. All agents are able to sense/observe the environment and each agent's observation is a concatenated array of the state vectors of all grids within the agent's Field of View (FOV). The predator agents' action-space is of dimension five, including cardinal movements and a null action, and is the same for all agents. A higher-performing algorithm in this domain is defined as one that minimizes the average number

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

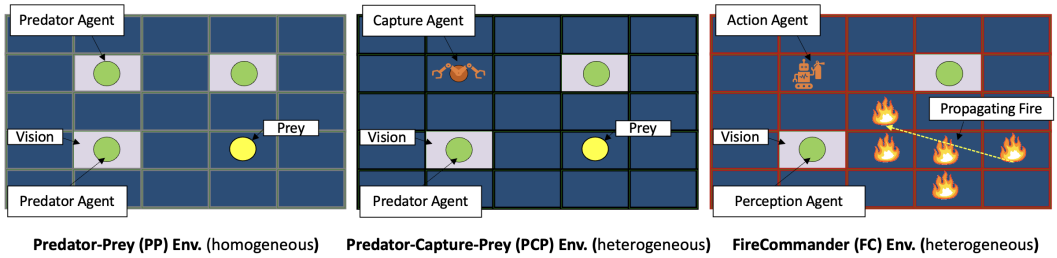


Figure 1: The Predator-Prey (PP), Predator-Capture-Prey (PCP), and FireCommander (FC) domains utilized for evaluating MixTURE and the baselines. Following prior work, all three domains have been employed to collect synthetic expert demonstration dataset and evaluating the methods. We also created a modified version of the FireCommander domain (Figure 2) as a strategic game with a user interface to collect human demonstration data. Please refer to Section 3 for more details.

of steps taken by agents to complete an episode. Details of the PP environment setup and problem dimensions in synthetic dataset are defined in Table 1.

2.2 Predator-Capture-Prey (PCP)

This environment is a *heterogeneous* extension of the PP domain, first introduced in prior work [4]. In this domain, we have two classes of agents: *predator* agents and *capture* agents. The first class of agent, called the *predator* agents, have the goal of discovering the prey and moving to its location with have an action-space of dimension five, including cardinal movements and a null (stay) action (same as in the PP domain). *Predator* agents have an observation space similar to the agents in PP domain. The second class of agents, called the *capture* agents, have the objective of locating the prey and capturing it. Capture agents differ from the predator agents in both their observation and their action spaces. Capture agents do not receive **any** observation inputs from the environment (i.e., no scanning sensors) and have an additional action of *capture-prey* in their action-space. This additional action must be used at a prey's location to capture the prey. Note that this domain is an explicit example of the perception-action composite teams described in our problem formulation. An episode is deemed successful once all agents have completed their class-specific objectives. Again, a better-performing algorithm in this domain is defined as one that minimizes the average number of steps taken by agents to complete an episode. Details of the PCP environment setup and problem dimensions in synthetic dataset are defined in Table 1.

Table 1: Environment Configuration Details (Synthetic Dataset)

Env.	(Dim.)(FOV)	# predator/perception	# capture/action	# prey/initial fire	max steps
PP	(5×5)(1×1)	3	-	1	20
PP	(10×10)(3×3)	6	-	1	80
PP	(20×20)(5×5)	10	-	1	80
PCP	(5×5)(1×1)	2	1	1	40
PCP	(10×10)(3×3)	3	3	1	80
PCP	(20×20)(5×5)	6	4	1	80
FC	(5×5)(3×3)	2	1	1	80
FC	(10×10)(3×3)	3	3	1	80
FC	(20×20)(5×5)	6	4	1	80

2.3 FireCommander (FC)

We also evaluate the performance of MixTURE and the baselines in a complex heterogeneous cooperative multi-agent environment, called FireCommander (FC), recently introduced by [5, 4]. FireCommander can be categorized as a strategic game, in which a composite team of robots (i.e., UAVs) must collaboratively find hidden areas of propagating wildfire and extinguish the fire in such areas as fast as possible. The robot team in FC is composed of two classes of agents: (1) *perception* agents, which can only sense the environment and detect areas of fire and, (2) *action* agents, which can only manipulate the environment by extinguishing a firespot which has already been detected by

perception agents. Neither *perception*, nor *action* agents are capable of accomplishing the task on their own, and therefore must communicate and collaborate. We employed this domain for evaluating MixTURE and baselines both on synthetic dataset (Figure 1) and the human expert demonstration dataset (Figure 2). We created a modified version of the FC domain for the human-subject user study as a strategic game with a user interface to collect human demonstration data. Details of the FC environment setup and problem dimensions in synthetic dataset and human expert demonstration datasets are defined in Table 1 and Table 2, respectively.

Table 2: Environment Configuration Details (Human Expert Demonstration Dataset)

Env.	(Dim.)(FOV)	# perception	# action	# initial fire	max steps
FC	(8×8)(3×3)	3	2	1	until win or lose (see Sec. 3)
FC	(10×10)(3×3)	2	2	5	until win or lose (see Sec. 3)
FC	(20×20)(5×5)	4	6	10	until win or lose (see Sec. 3)

Under the notations in our problem formulation, we have $\mathcal{C} = \{\mathrm{P}, \mathrm{A}\}$ where, P and A represent the *perception* and *action* agent classes. Also, the action-space of the two classes are defined as follows: $\mathcal{A}^{(\mathrm{P})} = \{1, 2, \cdots, 5\}$, representing the four primitive motions and stay (no-op action), and $\mathcal{A}^{(\mathrm{A})} = \{1, \cdots, 6\}$, representing the four primitive motions, stay no-op action, and an extra action which corresponds to extinguishing fire by dousing water (when on top of a fire). Note that the dimensions of the action space here are similar to the PCP. Agents of class P are equipped with fire detection sensors and can observe the environment. Agents of class A, do not receive any observation from the environment. Fire is hidden from all agents initially and must be found first before the agents can extinguish it. Fire spreads and expands, regardless of the collective actions of the team (i.e., restless environment). An episode of the game is successful only if all the active firespots within the map are discovered and extinguished. Once again, a better-performing algorithm in this domain is defined as one that minimizes the average number of steps taken by agents to complete an episode.

3 Human-Subject User Study Details: Environment and Procedures

Here we present further details regarding our experiment setup and environment design for collecting human demonstrations (i.e., human expert demonstration dataset) for teaching collaborative policies to multi-robot teams as well as evaluating the effects of demonstrating both environment-action and communication-action strategies on the human expert's performance and system usability. We designed a version of the FireCommander domain, previously introduced by [4], suitable for our experiments. **The following subsections present the detailed instructions and information provided to the human subjects during the experiments**[1]. Participants were compensated $25 for participation in the studies, which took 60-75 minutes.

3.1 Introduction to the Experiment

FireCommander is a strategic game, just like any other strategic games that you might have experienced before, such as: the soccer game series (e.g., FIFA or Pro Evolution Soccer (PES)), The World of Warcraft, The Age of Empires, or any other multi-character game that you can name. The goal in such games is to come up with a strategy to coordinate a team of characters (i.e., agents, players, robots, warriors, wizards, etc.) in the best possible way to win the game. The goal in the FireCommander game is to use a group of robots to put out a propagating wildfire.

3.2 Teaming: Necessity of Communication for Efficient Collaboration

An essential facet of a team is communication. Communication enables a shared mental model between members of a group and allows them to coordinate their actions in order to collaborate efficiently. Communication also allows for group members to adapt to uncertainties and changes in the environment and tasks.

Communication specifically is a necessity for collaboration when team members are operating under partial observability (i.e., each member only sees their local surrounding and does not have access to

[1] Instructions presented exactly as they were given to the human subjects. 'You' refers to the subject.

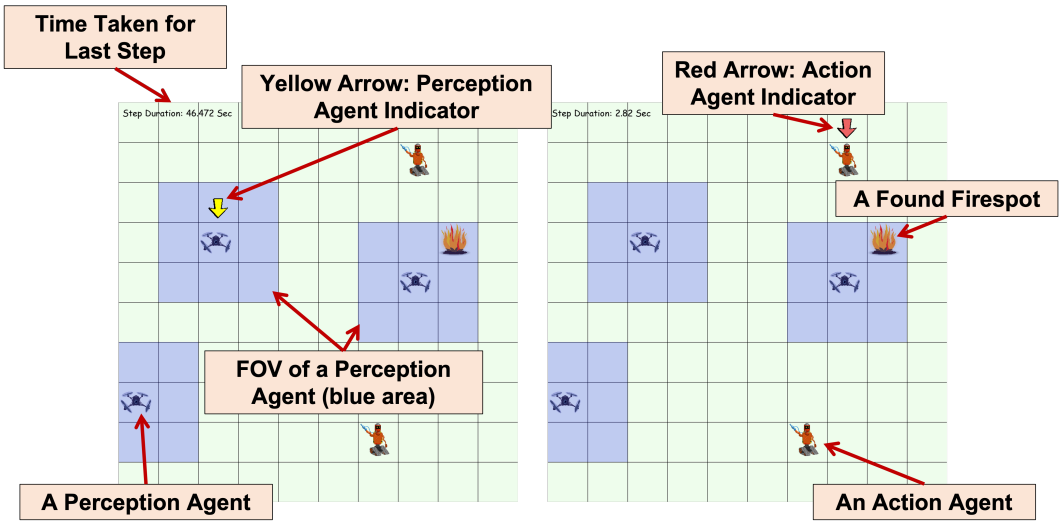


Figure 2: Designed FireCommander interface for the human-subject user experiment.

other members' observation or the full state of the environment). Pay attention to the Figure 3 as an example of the partial observability. When you play a strategic video-game, you are handling a team, while the communication between the members of this team is happening inside your brain (e.g., top row in the Figure 3). In other words, your brain acts like a centralized super-computer that takes in all the information from all the characters (i.e., sees everything) and creates the **shared mental model** of the game objective for the team. Nevertheless, when we create a robotic team and deploy them in real world for some task such as disaster response, this communication step cannot be bypassed, since we usually do not have access to centralized super-computers and agents are operating under partial observability (e.g., bottom row of the Figure 3). In such settings, team members (i.e., agents, robots, humans) must talk to each other to make other teammates aware of what they see, or what they just did, or what they are about to do, so that those other teammates can coordinate their actions better such that in turn, the efficiency is maximized.

Therefore, when we program (or teach) such robot teams, we not only need to show them how to do things, but we also need to show them how to talk to each other.

3.3 Introduction to Learning from Demonstration

Simply put, Learning from Demonstration (LfD), is a machine learning method in which you teach a robot how to do something by showing it. For instance, to teach a robot to play ping-pong, you can literally move its arm and show it where to go.

The FireCommander, though a fun game, is not just a game. In fact, we are trying to use LfD here to teach a team of robots how to efficiently put out a moving-spreading wildfire. While playing this game, we are collecting your data (i.e., the actions you take at each step of the game) to teach this team of robots how its done. We will use your data to teach the robot team how to coordinate as well as how to communicate and talk to each other, using some LfD techniques. So, when playing the game, try to strategize and come up with teaming plans to maximize the efficiency of your solution to the game. Note that, efficiency here means **fewest steps taken to finish the game as fast as possible**.

3.4 General Game Objectives and Logistics

In this game, you will have to use a group of Quadcopter robots (a.k.a the Perception agents) and a group of Ground robots (a.k.a the Action agents) to put out a propagating wildfire as fast and efficient as possible. But here are the game logistics:

- Fire is initially hidden from you. You need to search around to find the firespots.
- Only Perception agents can see the fire; but they have limited field of view (FOV), only within the blue shaded area around them.

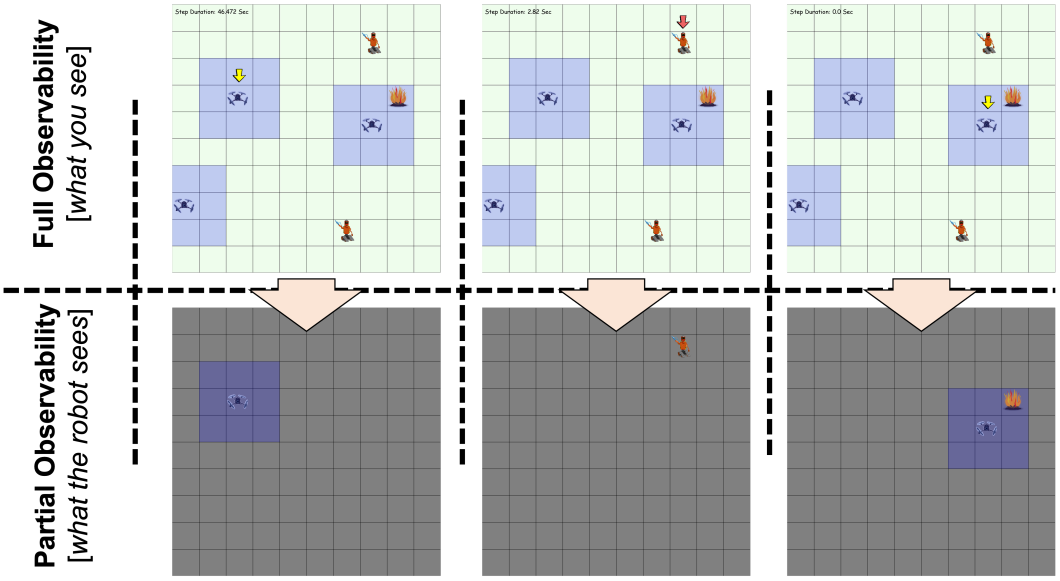


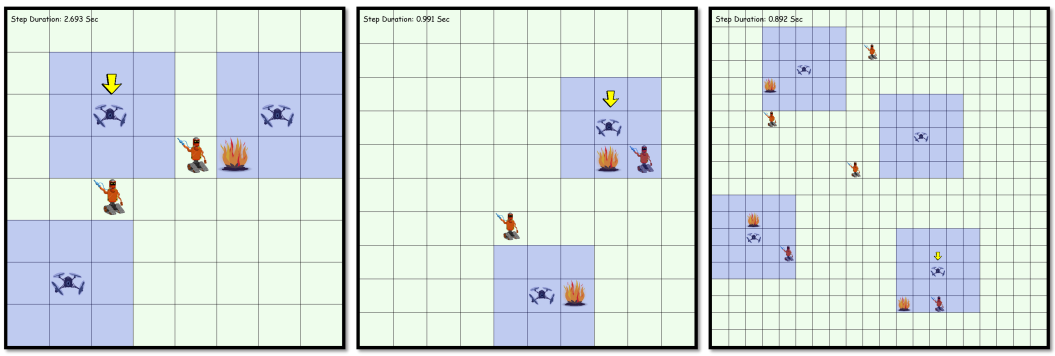
Figure 3: Environment partial observability in the FireCommander domain. When a user plays a strategic video-game, they are handling a team, while the communication between the members of this team is happening inside their brain (e.g., top row). Nevertheless, when we create a robotic team and deploy them in real world for some task such as disaster response, this communication step cannot be bypassed, since we usually do not have access to centralized super-computers and agents are operating under partial observability (e.g., bottom row).

- Only Action agents can put out a firespot; but they do not receive any environment observations (lack of sensory information).
- A firespot keeps propagating and growing in the background on wall-clock time, regardless of what you do, unless you put it out.
- Once all active firespots are put out the game ends and you'll receive a score.
- The faster you put out the fire, the higher you will score.
- Overall, use the quadcopters to find fires, then use ground robots to put out the found fires, and do this as fast and as efficient (fewest # steps taken to finish the game) as possible.
- **Note** that, to put out a firespot, you must first find it. You cannot put a firespot out before finding it.
- **Note** that, when you find a fire, its location will be known even if you move away the Perception agent. However, if that fire propagates, you will not see the new grid on fire, unless you have a Perception agent monitoring that area.
- **Note** that, to put out a firespot, the Action agent must be on top of it (i.e., agents will not be damaged by the fire)
- **Note** that, the game has an enforced cut off threshold on the score. When your **score drops below** 50, the game will end automatically.
- **Note** that, your score is initialized to a 100 and will change based on your strategy and wall-clock time. The score function details are described below.
- **Note** that, there are three levels to this game: Easy, Moderate, and Hard (see Fig. 4). You will be practicing on the easy mode for at least one round and until you feel comfortable, then you will the actual rounds from easy to harder levels.

3.5 Understanding the Game Score Function

Your `score` is initialized to 100 at the start of each round and then:

- `score` $-=$ $f_p^s \times$(fixed wall-clock time), where f_p^s is the fire propagation rate for scenario, s, (i.e., easy, moderate, or hard)



Easy Scenario (1 initial fire) **Easy Scenario** (5 initial fires) **Easy Scenario** (10 initial fires)

Figure 4: Instances of the designed FireCommander game environment at different levels for the human-subject user experiment.

- `score` $-= 0.2\times$(per existing firespot)
- `score` $+= 0.2\times$(per found firespot)
- `score` $+= 0.5\times$(per killed firespot)
- `score` $-= 0.1\times$(per usage of fire extinguisher)

3.6 Task Description for Each Condition

We utilize a 3×2 within-subjects design; three levels of difficulty varying across two abstractions:

1. ***noComm* condition**: Only demonstrating environment actions for each robot at each time step.
2. ***withComm* condition**: Demonstrating both environment actions and communication actions for each agent at each time step.

We implemented both conditions in the FC interface introduced above. The environment configuration and details for each level of difficulty is presented in Table 2. The details of each mode along with user instructions (as were presented to the human subjects during the experiment) for each condition are presented below.

3.6.1 Maneuvering the Game in noComm Condition

In this mode, the human subject will be only demonstrating the environment actions for each robot at each time step. The data collected in this condition are used to train our MixTURE architecture. Note that the previous sentence was not presented to the human subjects and instead, the users were only given the following instructions to maneuver this mode. Here are the details:

- At each step of the game, the current agent (i.e., agent that you will be handling) is shown with a downward arrow on top.
- After performing a task, the arrow indicator will move on top of the next agent (turn rotates).
- You will move/handle robots in this way, one-by-one, until the game ends.
- To move the robots around us the arrow keys: **Up** ($\uparrow$), **Down** ($\downarrow$), **Right** ($\rightarrow$), **Left** ($\leftarrow$)
- To remain still (move to next agent w/o doing anything), press the **Enter**
- To put out a fire, use the right or left **Ctrl** key

3.6.2 Maneuvering the Game in withComm Condition)

In this mode, the human subject will be demonstrating both an environment-actions and a communication-action for each robot at each time step. The data collected in this condition are used to train the MA-GAIL baseline [6]. Note that the previous sentence was not presented to the human subjects and instead, the users were only given the following instructions (starting from the

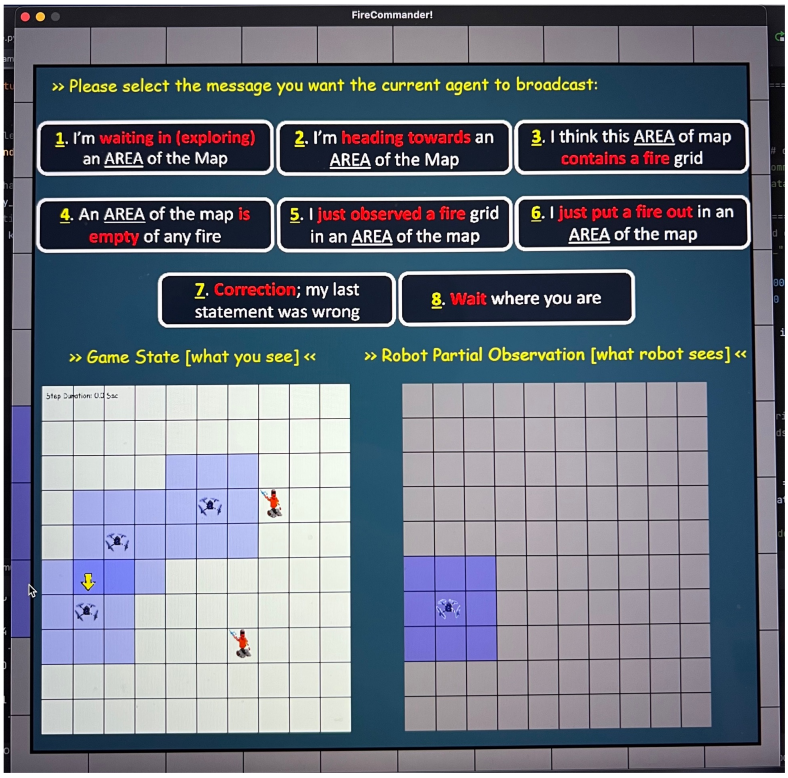


Figure 5: Inspired by prior work [2, 1], the messages in the communication menu designed for this condition include a wide range of options from anticipatory (i.e., sharing information by anticipating one another's needs rather than answering to requests or prompts) and deliberative information sharing (i.e., prioritizes information about the next goal to be accomplished during a task).

next paragraph) to maneuver this mode. We note that, the messages in the communication menu designed for this condition include a wide range of options from anticipatory (i.e., sharing information by anticipating one another's needs rather than answering to requests or prompts) and deliberative information sharing (i.e., prioritizes information about the next goal to be accomplished during a task), the design of which was inspired by prior work [2, 1]. The dimensionality of this message-space is 26, which is equal to the dimensions of the message space in our expert heuristic design, as well as the MARL-only baseline with learned communication action.

Everything in this mode is just the same as described in *noComm* mode, except that for each robot there is one extra step where you need to select a message from a predefined list to be broadcasted by the current robot to other teammates. Here are the details:

- In this mode, for each agent, first a communication menu will pop up. After message selection is done, the communication menu will be gone and you can move the robot.
- For each agent, you can choose a message from the menu shown below in Fig. 5 using keyboard numbers.
- Choosing options 1 – 6 would need you to specify your intended area (i.e., domain quadrants by choosing a number 1 – 4).
- Message option 7 indicates a previously incorrect statement by an agent, and message option 8 is a Null message (i.e., no communication).
- Note that, for reference, you can see your game screen (i.e., location of your robots and the fire (if any found)) on the bottom of the communication menu, at all times, alongside with robot's current partial observation (i.e., what you see v.s. what the robot sees). You should select the communication message based on what robot sees.
- Note that, if you press a wrong key (like the arrow keys), by default, message option 8 (i.e., "Null; nothing important") will be selected.

- Note that, in this case the fire is still propagating on wall-clock time in the background. Try to become familiar with the message options early on and come up with a communication strategy during practice so you won't have to read all the options every time. The communication menu is fixed.
- Important: as stated before, we will use LfD to teach robots how to communicate from your data. Therefore, it is highly important that you accurately select a message that reflects your strategy. Without messages being accurately selected, the robots that will be trained on your data will be inefficient and unsuccessful. Note that consistency is key.

4 Expert Heuristic Design for Collecting the Synthetic Dataset

We created expert heuristics (e.g, models of human experts / Oz-of-Wizard) to collect the synthetic expert demonstration dataset and evaluate MixTURE and the introduced baselines. The following presents our expert heuristic design details for each of the three environments.

4.1 Predator-Prey (PP) Expert Heuristic Design Details

The heuristic expert behavior for Predator-Prey involves each agent simply moving towards the nearest known prey location. If no prey locations are known, we use a greedy exploration behavior in which each agent attempts to reveal as many unexplored tiles as possible in the next turn. However, to reduce backtracking, we penalize exploring tiles which have unexplored neighbors. Thus, the expert behavior prioritizes exploring corner and edge tiles, and tiles on the boundary of the revealed areas. Table 3 shows the performances achieved by our heuristics in PP, PCP, and FC.

Table 3: Expert Heuristic (e.g, models of human experts / Oz-of-Wizard) Performances (# steps)

Dimensionality	Predator-Prey	Predator-Capture-Prey	FireCommander
5×5	8.573 ± 2.175	9.677 ± 2.628	14.439 ± 8.712
10×10	12.221 ± 3.017	14.763 ± 3.858	16.160 ± 8.247
20×20	24.915 ± 5.512	27.701 ± 6.617	24.213 ± 13.721

4.2 Predator-Capture-Prey (PCP) Expert Heuristic Design Details

Since capture agents find themselves unable to move off of prey (game logistic), they can effectively see within a (1×1) vision radius by simply checking whether their last attempted move was successful. Thus, we utilize the same logic as the Predator-Prey heuristic, but treat capture agents as predators with a single-cell vision radius. See Table 3 for our heuristics' performance.

4.3 FireCommander (FC) Expert Heuristic Design Details

Instead of tracking revealed and unrevealed tiles, we maintain a probabilistic belief over each grid cell estimating the likelihood that a particular cell contains a fire. These beliefs are updated based on observations from perception agents. To take into account fire spreading dynamics, at each timestep, we increase the estimated fire probabilities according to a Gaussian filter applied to the current belief. The perception agent behavior is similar to that of the Predator-Prey heuristic, but the score of each cell v_{xy} is instead equal to the current entropy of the corresponding belief. Action agents simply attempt to extinguish the closest known fire. If no fires are known, they instead follow the nearest perception agent. Table 3 shows the performance achieved by our heuristic.

4.4 Communication Heuristic Design Details

For the communication heuristic, we used a baseline approach in which each agent generates a one-hot encoded representation (with the same size as the message-space available to the human participants in our user study) as a function of their last k observations. To embed the observations into a one-hot vector, we simply perform K-means clustering over the observations in the demonstration dataset for each environment, and map each observation to the index of the nearest cluster center. For $k > 1$, we instead perform this procedure over the last k observation vectors, concatenated. We find $k = 2$ to have the best performance, which is in accordance with prior work [1].

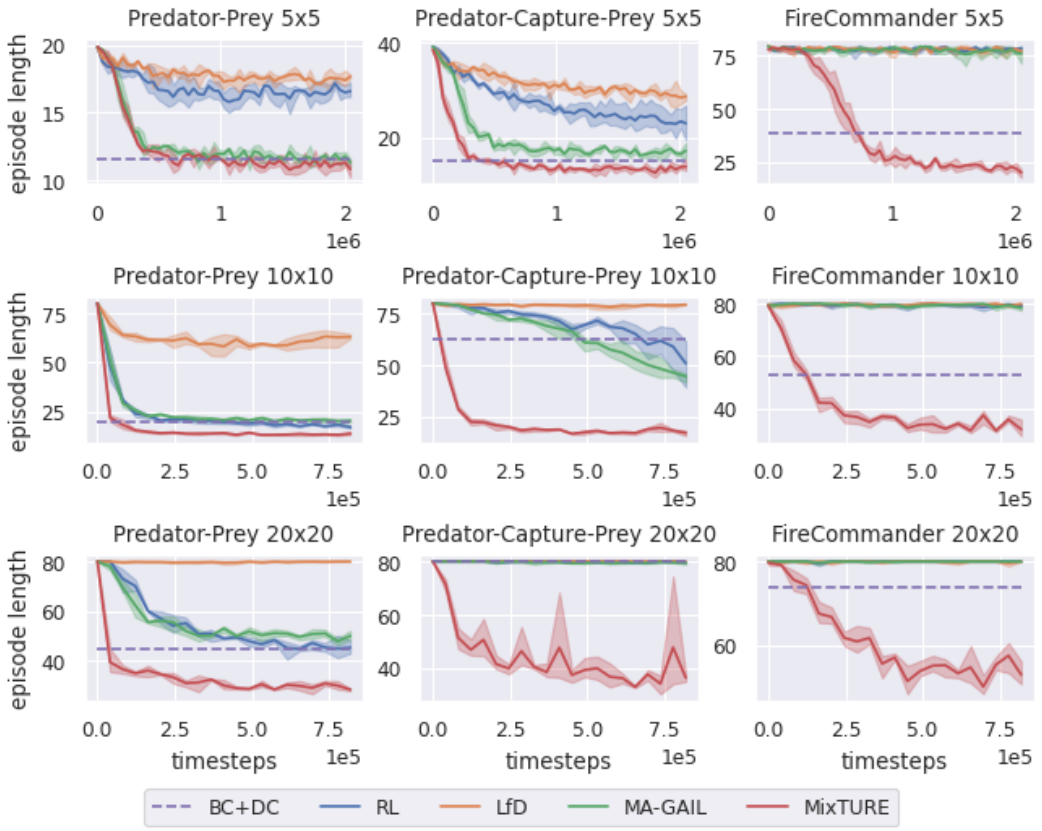


Figure 6: Full evaluation results for MixTURE and the baselines in the all difficult levels (i.e., easy, moderate, and hard) of the three environments (i.e., PP, PCP, and FC).

5 Ablation Studies and Supplementary Results

5.1 Scalability (Figure 6)

We evaluated the MixTURE against all baselines in all three domains introduced in Section 4 and under three different difficulty levels: (1) easy (5×5 domain, 3 robots), (2) medium (10×10 domain, 6 robots), and (3) hard (20×20 domain, 10 robots). More environment configuration details are provided in Table 1. Figure 6 shows the training and evaluation results for MixTURE and the baselines in all three levels of difficulty, for PP, PCP, and FC domains. Each epoch on the x-axis represents $40K$ data samples.

As shown in Figure 6, MixTURE outperforms all non-communicative, communicative with expert heuristic, and communicative with differentiable communication channel baselines in learning collaborative teaming policies. Under the similar training schemes, not only MixTURE outperforms the baselines at the convergence, it also shows significantly lower sample complexity, particularly with increased domain complexity (i.e., from PP domain on the left column to the FC domain on the right, and from an easy (5×5 domain, 3 robots) scenario in top row to a hard (20×20 domain, 10 robots) scenario in bottom row). We note that the medium and hard cases of the FC domain was reported to be very challenging in prior work [4] even for sophisticated MARL methods with differentiable communication channels. As we can see from the results provided, by combining knowledge acquired from the expert demonstration as well as the introduced MIM-based end-to-end communication, MixTURE can learn a high-quality policy for a heterogeneous team of robots in this challenging domain and successfully scale to increased domain dimensionality as well as different number of agents and team compositions. Results confirm MixTURE's strong ability to learn highly complex collaborative policies in heterogeneous, partially observable, and dynamic environments.

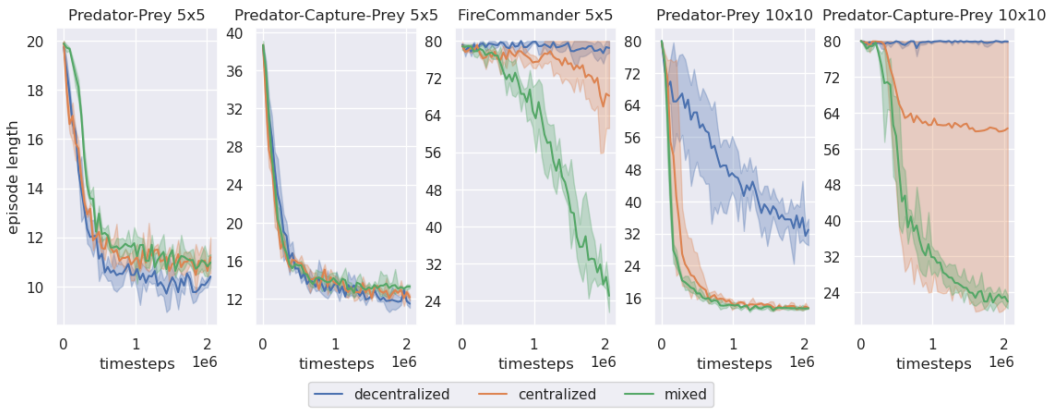


Figure 7: MixTURE ablation comparing training curves with several discriminator architectures

5.2 Effects of Discriminator Architecture (Figure 7)

MA-GAIL [6] introduces three discriminator architectures, which they denote as centralized, decentralized, and zero-sum, intended for fully cooperative, mixed cooperative-competitive, and zero-sum games, respectively. In the decentralized architecture, each agent receives their own reward signal as a function of their current local observation and action. The centralized discriminator instead takes in the joint observations and actions of all agents and outputs a single shared reward. Although the centralized discriminator was intended for fully cooperative tasks, Jeon et al. [3] find the centralized discriminator to have poor scalability as the number of agents increases and recommend using the decentralized discriminator instead.

We examine an alternative discriminator architecture which can access global observations while being restricted to only utilize local actions: $D^i : O^1 \times O^2 \times \cdots \times O^n \times A^i \rightarrow \mathbb{R}$. Intuitively, to enable cooperation among heterogeneous, communicating agents, we find that it may be necessary to utilize information from other agents in the discriminator in order to provide an informative learning signal. For example, if a particular agent does not have its own local observations and instead needs to learn to act solely based on information from other agents, a decentralized discriminator which only has access to the agent's local observation and action may not be sufficient. Since this architecture utilizes observations on a global scale with actions a local scale, we call our proposed architecture the *mixed* discriminator architecture.

We conduct an ablation study comparing performance of the decentralized, centralized, and mixed discriminators. To isolate the impact of the discriminator architecture, we train without behavioral cloning ($\lambda_{\text{BC}} = 0$). Training curves are depicted in Figure 7. The mixed discriminator architecture is competitive with or better than the two baseline architectures in all environments. As demonstrated, our proposed *mixed* discriminator architecture shows a more consistently good performance across different domains and dimensionalities as compared to fully centralized and decentralized architectures proposed by [6].

5.3 Effects of MIM Objective in Reverse Model (Figure 8)

We conduct an ablation study comparing MixTURE performance both with (tuned λ_{MIM}) and without ($\lambda_{\text{MIM}=0}$) the MIM loss on the easy (5×5) domains. Training curves for each method are depicted in Figure 8. As shown, MIM greatly improves sample efficiency on FireCommander and does not degrade performance on Predator-Prey or Predator-Capture-Prey. Our investigations confirm that reducing the message entropy via MI maximization in our proposed reverse model (i.e., message reconstruction module) is particularly helpful for domains with increased task complexity where agents rely more heavily on communication for reasoning and decision-making.

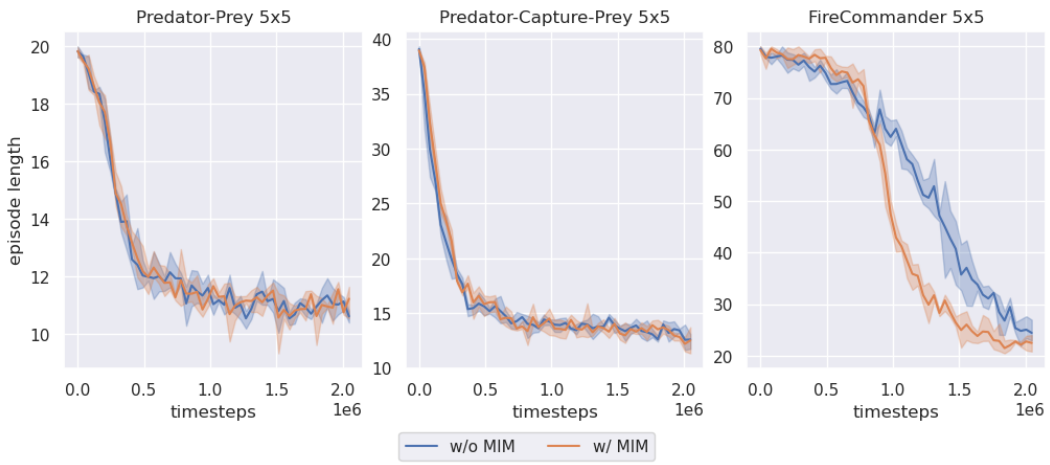


Figure 8: MixTURE ablation comparing training curves with and without MIM

5.4 Effects of Offline Behavioral Cloning Loss (Figure 9)

Here, we analyze the performance impact of different approaches for integrating behavioral cloning into our method. We compare three approaches: (1) online BC (auxiliary objective, ours), (2) offline BC (pretraining), and (3) no BC (baseline). In the latter case, the learning signal only comes from the discriminator's reward signal. We also compare against the naive approach of pretraining with behavioral cloning offline in order to initialize the policy network, before continuing training online, again using only the discriminator's reward signal. In online BC approach, we add an auxiliary behavioral cloning term to the loss during online training. In other words, the minibatch of transitions sampled from the demonstration dataset is not only used to update the discriminator, but also update the current policy via maximum likelihood.

As depicted in Figure 9, although the approach which simply initializes via behavioral cloning does result in a performance improvement compared to no behavioral cloning at all, the performance improves by at most only a small margin during online training, and even degrades over time in the 10×10 FireCommander domain. Meanwhile, the online BC (auxiliary objective) approach consistently learns a behavior which not only outperforms the initial performance of the pretrained approach, but also has the best final performance out of all methods.

5.5 Additional Human Subjects Study Results

We further analyzed the results of the human subjects study to determine if demographic information and pre-study metrics significantly correlated with our dependent measures. We conducted a Spearman's correlation in which our independent variables were: (1) experience with video games, (2) experience with strategic video games, (3) prior experience with Fire Commander, and (4) age, and our dependent variables were performance, SUS, and workload. We did not find significant correlations between any of these metrics. Based upon the results of a Kruskal-Wallis test, we additionally found no significance between sex and our dependent variables.

6 Hyperparameters and Compute Resources

Hardware Specifics – Most of our experiments were conducted on an NVIDIA Quadro RTX 8000 with approximately 50GB of Video Memory Capacity. On this machine, 56 total CPUs were available, however, we usually used 12 – 24 CPU cores for our experiments. We also conducted some experiments on an 8-Core CPU (Ryzen 7 5800X) desktop computer.

Hyperparameters – In the following (Table 4) we present the training hyperparameters in our implementations and experiments across all environment modalities and difficulty levels. We note that, the attention-based communication technically makes the time complexity $O(N^2)$ in the number of agents, but practical implementation and optimization involve minimal additional inference layers.

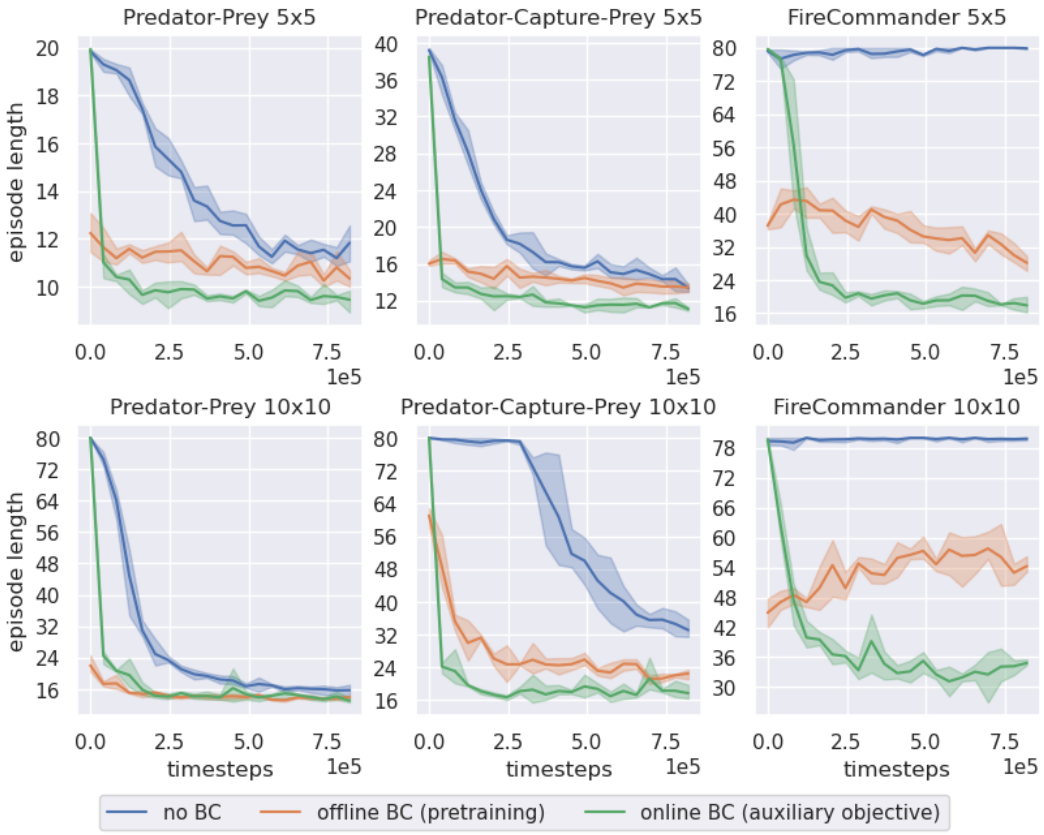


Figure 9: MixTURE ablation comparing training curves without, with BC pretraining, and with BC loss applied online.

Therefore, the overall additional computational cost imposed by the MIM reverse model is small, especially, in high-dimensional environments, policy and value networks bear the majority of the computation load.

Tuning Loss Weights – The initial two components of our objective function (Equation 2) – the GAIL loss and the PPO loss – are fundamental to our GAIL-based MA-LfD architecture. The remaining two parts – offline BC loss and MIM loss – can be regarded as dynamic components, both of which are studied in our supplementary results in Sections 5.3 and 5.4. Our experiments showed that: (1) the addition of the offline BC loss always helps improve model performance, and (2) the MIM loss seems more effective in intricate or larger domains with greater agent counts, yet adjusting its weight for substantial gains can be slightly challenging. For other loss-part weights we mostly stayed consistent with prior work and did not observe any significant trends of sensitivity to weight balance.

References

[1] Abhizna Butchibabu. Anticipatory communication strategies for human robot team coordination. PhD thesis, Massachusetts Institute of Technology, 2016.

[2] Abhizna Butchibabu, Christopher Sparano-Huiban, Liz Sonenberg, and Julie Shah. Implicit coordination strategies for effective team communication. Human factors, 58(4):595–610, 2016.

[3] Wonseok Jeon, Paul Barde, Derek Nowrouzezahrai, and Joelle Pineau. Scalable multi-agent inverse reinforcement learning via actor-attention-critic. arXiv preprint arXiv:2002.10525, 2020.

[4] Esmaeil Seraj, Zheyuan Wang, Rohan Paleja, Daniel Martin, Matthew Sklar, Anirudh Patel, and Matthew Gombolay. Learning efficient diverse communication for cooperative heteroge-

Table 4: Hyperparameters

Name	Value
hidden layer dimensionality	64 (easy), 256 (moderate/hard)
rollout steps	4096
T-BPTT segment length	8
segments per minibatch	8 (easy), 32 (moderate/hard)
total minibatch size	64 (easy), 256 (moderate/hard)
PPO clipping ϵ	0.2
PPO epochs	3
learning rate	$[10^{-4}, 10^{-3}]$
discount factor	0.99
GAE lambda	0.5
MIM coefficient (λ_{MIM})	0.1 (easy), 0.01 (moderate/hard)
BC coefficient (λ_{BC})	$[10^{-1.5}, 10^{0}]$
discriminator learning rate	10^{-5}
max gradient norm	5.0

neous teaming. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, pages 1173–1182, 2022.

[5] Esmaeil Seraj, Xiyang Wu, and Matthew Gombolay. Firecommander: An interactive, probabilistic multi-agent environment for heterogeneous robot teams. arXiv preprint arXiv:2011.00165, 2020.

[6] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. Advances in neural information processing systems, 31, 2018.