Proximal Policy Optimization for Coordination of Distributed Agents in a Cooperative Jamming Scenario

Robert T. Lattus and John M. Shea
robertlattus@ufl.edu, jshea@ece.ufl.edu
University of Florida, Gainesville, FL

***Abstract*—Drones and other mobile robots are playing an increasing role in modern warfare. Jamming is a key technology to protect certain areas from intrusion by adversaries of these types. In this paper, we consider protecting a sensitive area from mobile adversaries that are operating in a leader-follower configuration and coordinated over a wireless channel. The protected area contains agents equipped with low-power RF transmitters and highly directional antennas whose goal is to block the operations of the adversaries by disrupting their ability to communicate. The fact that such antennas concentrate the RF power over relatively small sectors makes evasive maneuvers by the adversaries more effective. We consider a scenario in which the agents cannot directly communicate, and thus they must decide the orientations of their jamming beams in a decentralized fashion. Furthermore, the adversaries' locations must be inferred from RF or RADAR measurements that are corrupted by noise. We cast this problem as a continuous-observation, partially-observable Markov decision process. Each agent is trained using multi-agent proximal policy optimization with centralized training, decentralized execution. We compare our results to deterministic and independent approaches, demonstrating the benefits of stochastic policies for distributed cooperation.**

I. Introduction

We consider the problem of protecting a region from intrusion by mobile adversaries operating in a leader-follower architecture and relying on RF communication with an external control center. As an example, the mobile adversaries may be drones used for reconnaissance/surveillance or as armaments. A team of agents equipped with radio frequency (RF) jammers attempts to disrupt the adversaries' operations by blocking their communications with the control center. In the scenario we consider, the jamming agents are at fixed locations and use steerable directional antennas to project RF energy toward the adversaries. These agents could use large jamming antennas mounted on trucks or permanently fixed to defensive fortifications. The main focus of this research is scenarios in which the jamming agents cannot have reliable, direct communication, and thus their decisions about where to point their antennas must be made in a distributed fashion.

Research was supported by AFOSR award number FA9550-19-1-0169. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsoring agency.

Jamming has been studied for many years as a way to disrupt wireless communications (*cf.* [1]). Wireless communication links are often critical to the control or tasking of unmanned autonomous vehicles, or drones, and thus jamming is a natural approach to disrupting such systems [2]–[6]. Of these, [2] and [3] use a differential-game approach to analyze a network of drones in the presence of a mobile jammer in a deterministic environment. In [4] trajectory and transmission power are considered for multiple unmanned aerial vehicle (UAV) jamming agents that are tasked with intercepting a rogue drone; the resulting joint control problem is decomposed into separate, discretized optimization problems. A similar scenario is considered in [5], but reinforcement learning (RL) is applied to jam a rogue drone while minimizing the interference to friendly communicators. In [6], multi-agent RL is applied for a drone team in tracking and jamming adversarial drones targeting protected locations.

In this paper, we consider scenarios that are similar to [5] and [6] in that multiple agents must jam a team of adversaries. However, we focus on scenarios in which the agents cannot directly coordinate their actions and must make distributed decisions. We show that using proximal policy optimization (PPO) for agent training can lead to emergent coordination in this distributed decision process with partial observability.

When the motion of an adversary can be modeled as Markovian, the resulting system is a multi-agent Markov decision process (MMDP or MAMDP) [7]. For a MAMDP with centralized control, the optimal performance can be achieved with perfect state knowledge. However, in this paper, the observations of each agent are subject to noise, and the agents make decisions in a decentralized fashion. Because of the noise, the MAMDP becomes partially observable, leading to a partially-observable Markov decision process (POMDP) formulation. The lack of coordination requires agents to perform dynamic task assignment

to choose which adversary to jam. In [8], it is noted that the use of a stochastic policy can lead to a weak form of cooperation in a multi-agent, distributed tracking task. We further develop this idea, applying stochastic policies to aid task assignment in this noisy environment with distributed control.

We use proximal policy optimization [9], a deep RL algorithm, to determine the actions in this multi-agent POMDP. PPO has been applied recently to multi-agent jamming and control scenarios. In [10], the authors use an independent-PPO (IPPO) strategy for multiple mobile agents for radar system jamming. They provide a method for each agent to iteratively search for an optimal model to provide distributed control. Despite some success, independent forms of multi-agent reinforcement learning (MARL) are commonly known to have issues with instability [11], [12]. Another similar work uses multi-agent PPO (MAPPO), but for motion control in autonomous defense against targeted missiles [13]. To accurately estimate the state, the algorithm uses a gated recurrent unit. For this paper, we consider the use of the lightweight particle filter for belief-state estimation under partial observation. The performance of MAPPO with stochastic policies is compared with the use of deterministic policies, PPO without belief state estimation, and IPPO.

II. System Model

We consider the problem of protecting a region of the real plane[1], $\mathcal{R} \subset \mathbb{R}^2$ from intrusion by $1 \leq i \leq M$ mobile adversaries. We examine only the time period when the adversaries are located in $\mathcal{R}$, and we denote the location of adversary i at time t by $\mathbf{a}_i^{(t)} \in \mathcal{R}$. An adversary's motion is assumed to follow a Markov model in $\mathcal{R}$. In each time interval, each adversary moves some fixed-length step in a direction that depends on its current location and direction of movement.

The region is protected by W identical agents at fixed locations $\mathbf{c}_w \in \mathcal{R},\ w = 1, \ldots, W$. Each agent is equipped with an RF transmitter and a steerable directional antenna to jam communications at the adversaries. We assume that each agent can measure a noisy angle estimate of the location of the adversaries in each time interval, and the agents use the observed sequence of angles to determine where to point their directional antennas in the next interval. Agents are trained based on continuous beam movements, but the training is anonymous to each agent in the sense that the agents are not allowed to exploit the location to develop a unique identity that can result in distributed decisions becoming joint decisions. In addition, we consider the scenario in which the agents cannot directly communicate to coordinate their actions.

For highly directional antennas, the transmission pattern consists of a single main lobe with a fixed beamwidth, θ, and several much smaller sidelobes. We approximate the transmission pattern by considering only the main lobe and assume that the jamming power is sufficient over $\mathcal{R}$ that an adversary's communications will be disrupted if the adversary lies within the angle covered by the main lobe of one of the agents. For this initial investigation, we discretize time and assume that in one time step, agents can angularly rotate their antennas in either direction (clockwise or counter-clockwise). We assume that each agent limits the direction of its beam angle to its front side, resulting in $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$, with $\theta = 0$ equating to the beam pointing directly ahead. Let $\phi_w(t)$ denote the orientation of agent w's antenna at time t. Treating coordinates as complex values, adversary i will be jammed if

$$\left| \arctan\left(\frac{\mathrm{Im}\left\{\mathbf{a}_i^{(t)}\right\} - \mathrm{Im}\left\{\mathbf{c}_w\right\}}{\mathrm{Re}\left\{\mathbf{a}_i^{(t)}\right\} - \mathrm{Re}\left\{\mathbf{c}_w\right\}} \right) - \phi_w^{(t)} \right| < \frac{\theta}{2}. \quad (1)$$

holds for at least one agent w.

The goal of the agents is to maximize the number of time intervals in which each of the adversaries are jammed by at least one agent. Agents must choose how to steer their beams before receiving the next noisy observation of the adversaries, all while not having communication access to their fellow agents. With multiple agents under central coordination, the agents can direct their beams together to maximize area coverage, as shown in Fig. 1. However, when the agents are decentralized in action choice, optimal beam positions become unclear. We will show that the agents can improve their joint performance through the use of stochastic policies.

III. POMDP Belief State Estimation

Here, we develop the techniques used to estimate the probability distribution of the vehicle's angular position based on the noisy measurements and Markovian movement. The distribution estimates need to be sufficiently accurate for jamming to be successful, a process made difficult by the noise present in sensing.

A. POMDP Description

We cast the overall jamming problem as a POMDP with state consisting of the locations of the adversaries and the beam angles of the agents. The system is partially observable, where at time t agent w knows only its current beam angle $\phi_w^{(t)}$ and has a noisy

[1]This could readily be extended to three dimensions.

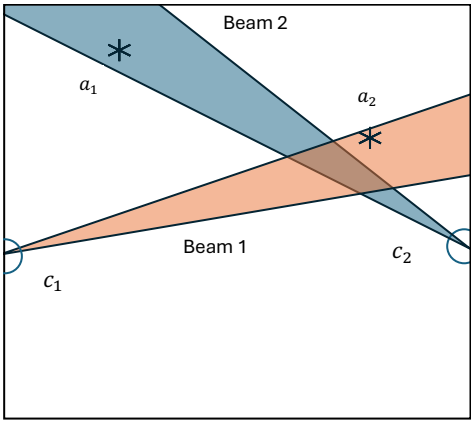

Fig. 1. System of two agents on opposite sides of the environment jamming two adversaries (indicated by $*$).

observation of the angle to each adversary i, $\{\psi_{w,i}^{(t)}\}_{i=1}^{M}$. The observed angles to the adversaries are corrupted by zero-mean, Gaussian noise, $\psi_{w,i}^{(t)} = \gamma_{w,i}^{(t)} + \mathcal{N}(0, \rho_n^2)$, where $\gamma_{w,i}^{(t)}$ denotes the true phase from agent w to adversary i at time t, and the resulting real value is interpreted module $(-\pi, \pi]$; see below.

The POMDP can be represented by a six-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}(s' \mid s, a), \mathcal{R}(s, a), \mathcal{O}, \mathcal{Z}(o \mid s', a))$, consisting of

$\mathcal{S}$: states — $\mathcal{A}$: actions
$\mathcal{R}$: rewards — $\mathcal{O}$: observations
$\mathcal{T}(s'|s, a)$: state transition function
$\mathcal{Z}(o|s', a)$: probability of observation function,

where s and a correspond to a state and action, respectively.

B. Particle Filter Estimation

Because the observations of the angles to the adversaries are noisy, each agent uses a particle filter to estimate the *a posteriori* probabilities (i.e., beliefs) of these angles. There is a separate particle filter to track the belief state of the angle to each adversary. Each particle filter utilizes a set of N particles and weights $\{p_j^{(t)}, y_j^{(t)}\}_{j=1}^{N}$, where the particle $p_j^{(t)}$ is a possible state for adversary j at timestep t and $y_j^{(t)}$ is the corresponding weight. At each step, the particle filter updates the particles by predicting the state transition for each particle and then updating the particle weights as follows:

(1) Initialize: At the outset of the scenario, an agent samples each particle from a uniform random variable $p_j^0 \sim \mathcal{U}(0, 2\pi)$ to represent initial estimates of the angle of the respective adversary. The corresponding weights are also initialized uniformly as $y_j^0 = 1/N$.

(2) Predict: In the predict step, each particle is updated according to the state transition model after an agent takes an action. For our purposes, the state transition function is modeled by adding process noise $\epsilon_j^{(t)} \sim \mathcal{N}(0, \rho_p^2)$ to the current measurement. Here, ρ_p^2 is the process noise variance.

$$\mathcal{T}(s'|s, a) = p_j^{(t)} + \epsilon_j^{(t)}$$
$$p_j^{(t+1)} \sim \mathcal{T}(s'|s, a)$$

(3) Observation Update: Having received a new state for each particle from the predict step, each agent w takes an observation from the environment $o_w^{(t+1)} = [\phi_w^{(t+1)}, \psi_{w,1}^{(t+1)}, \cdots, \psi_{w,M}^{(t+1)}]$. The adversary angles $\psi_{w,i}$ for adversaries $i = 1 \cdots M$ in the incoming observation are used in a likelihood function for each particle to update the particle weights. Because the noisy observations are angle measurements, we choose an angle-wrapped Gaussian model $G(\psi_{w,i}^{(t)} \mid p_j^{(t)}, \rho_m)$ for likelihood updates.

$$G(\psi_{w,i}^{(t)} \mid p_j^{(t)}, \rho_m) = \sum_{k=-\infty}^{\infty} \frac{1}{\sqrt{2\pi\rho_m^2}} \exp\left(-\frac{(\psi_{w,i}^{(t)} - p_j^{(t)} + 2\pi k)^2}{2\rho_m^2}\right).$$

Here, ρ_m^2 represents the measurement noise variance and the mean is the current particle state estimate. To ease computational complexity, we use $g(\psi_{w,i}^{(t)} \mid p_j^{(t)}, \rho_m)$, an approximate form of this wrapped Gaussian for the likelihood:

$$\exp\left(-\frac{1}{2\rho_m^2}\left(\Delta(\psi_{w,i}^{(t)} - p_j^{(t)})\right)^2\right) \propto G^{(t)}(\psi_{w,i}^{(t)} \mid p_j^{(t)}, \rho_m)$$
$$g(\psi_{w,i}^{(t)} \mid p_j^{(t)}, \rho_m) = \exp\left(-\frac{1}{2\rho_m^2}\left(\Delta(\psi_{w,i}^{(t)} - p_j^{(t)})\right)^2\right).$$

The wrap function is defined as $\Delta(\chi) = [(\chi + \pi) \bmod 2\pi - \pi]$ and serves to wrap the difference between the observed and particle angles around the unit circle.

With the approximate wrapped Gaussian likelihood, the weight update becomes:

$$Pr(\psi_{w,i}^{(t+1)} \mid p_j^{(t+1)}, a^{(t)}) \propto g(\psi_{w,i}^{(t+1)} \mid p_j^{(t+1)}, \rho_m)$$
$$y_j^{(t)} = Pr(\psi_{w,i}^{(t+1)} \mid p_j^{(t+1)}, a^{(t)}).$$

After each weight is updated, the weights are re-normalized

$$y_j^{(t+1)} \leftarrow \frac{y_j^{(t+1)}}{\sum_{k+1}^{N} y_k^{(t+1)}},$$

completing the observation update step in the particle filter.

(4) Resampling: The final step in the particle filter resamples each of the particles to prevent particle degeneration. To resample, we first create a new empty

particle set of size N. The current cumulative distribution of the weights is formed:

$$\mathcal{C}^l = \sum_{k=1}^{l} y_k^{(t+1)}.$$

Then, N random samples $r_n \sim U(0,1)$ are generated. Each random sample is compared with the current particle weights. If $\mathcal{C}^l \geq r_n$ is satisfied, then that particle is added to the new particle set, thus favoring the particles with the highest likelihoods. After each random sample has been evaluated, the particle weights are reset: $y_j^{(t+1)} = 1/N$.

At the outset of the scenario, each agent performs step (1) and then applies steps (2)–(4) at each iteration to update the belief state estimates. For compatibility with the actor and critic networks, we use the summary statistics of particle mean and variance to approximate the current belief state after the observation update. To extract the belief state summary statistics, an agent calculates the circular mean and variance of the particles. The circular mean is ensured to be bounded between $[0, 2\pi)$ and is calculated as:

$$p_{\text{mean}_0} = \operatorname{atan2}\left(\sum_{j=1}^{N} y_j^{(t)} \sin(p_j^{(t)}), \sum_{j=1}^{N} y_j^{(t)} \cos(p_j^{(t)})\right)$$

$$p_{\text{mean}} = (p_{\text{mean}_0} + 2\pi) \bmod 2\pi.$$

The variance is calculated in similarly circular manner:

$$R = \sqrt{\left(\sum_{j=1}^{N} y_j^{(t)} \sin(p_j^{(t)})\right)^2 + \left(\sum_{j=1}^{N} y_j^{(t)} \cos(p_j^{(t)})\right)^2}$$

$$p_{\text{var}} = 1 - \frac{R}{\sum_{j=1}^{N} y_j^{(t)}}.$$

These summary statistics are then passed into the networks individually as $b_i = p_{\text{mean}}$ and $\tilde{b}_i = p_{\text{var}}$ (see Fig. 2 and Fig. 3).

IV. Performance Evaluation

Next we consider the application of the concepts from Section III in a simulated environment similar to that in Fig. 1. The protected area is a square that is 100 units on each side where the agents are on the edges of the environments at coordinates $(0, 50)$ and $(100, 50)$. For the majority of the results reported, the agents use directional antennas with a beamwidth of 30° that can be oriented to cover a range of 180°, although results for other beamwidths are also reported.

The environment contains two or more adversaries who are coordinated through a leader-follower architecture, where the roles are pre-determined and remain fixed throughout each scenario. The motion of the leader $\mathbf{a}_1^{(t)}$ directly follows a Markov model. In each interval, the leader adversary can travel one of three directions: straight, left turn, or right turn. The straight action continues in the same direction as the previous action. The left and right actions are δ° counter-clockwise and δ° clockwise from the straight action, respectively. Each of the three actions has a step size of 1.0, and the adversaries are clipped at the boundaries so that they remain within the environment. The adversaries also repeat their previous action with probability ν. For larger values of ν, the adversaries' motion becomes more predictable, with long, straight lines or continuous turns. For smaller values of ν, the adversaries' motion becomes more evasive, as an adversary will oscillate more often between straight actions and turning.

The followers generally move to follow the leader but take some random actions to make jamming more difficult. With probability ξ, a follower will take an action from the same set of choices as the leader, but the action will be chosen to aim towards the point that is u_{tail} units in the negative direction of the velocity vector from $\mathbf{a}_1^{(t)}$. For the results reported here, $u_{tail} = 10$. With probability $1-\xi$, a follower randomly chooses from one of the three actions described above. This architecture effectively serves to allow the leader to direct the adversaries as a group to chosen sections of the environment, while still allowing randomness in the follower's actions that aid jamming evasion. We assume that jamming by the agents does not affect this local coordination between the adversaries. For the results presented in this paper, $\xi = 0.05$.

PPO is used for each agent to learn the optimal actions while operating in this environment. The goal of the agents is to cooperatively jam both adversaries. We use centralized training, decentralized execution (CTDE) with a single policy network for the agents and a joint critic that outputs a single value. The value loss for the joint critic is taken as the average of the individual returns of the agents. The structures of the actor and critic networks are shown in Fig. 2 and Fig. 3, respectively. Several configurations of actor and critic network were tested for optimality. These included architectures with combinations of 64, 128, or 256 neurons and either 1, 2, or 3 hidden layers for both the actor and critic. Testing demonstrated that strategies where the actor had fewer neurons than the critic performed at higher levels. Based on this analysis, we selected our actor network to contain

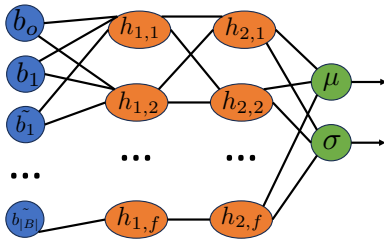

Fig. 2. The actor network receives belief state values $b_0, b_1, \tilde{b}_1 \cdots \tilde{b}_{|B|}$ where b_0 is the current agent's own beam angle, with the other inputs being the summary statistics of the belief. It has one hidden layer with f neurons, and outputs the policy mean μ and standard deviation σ.

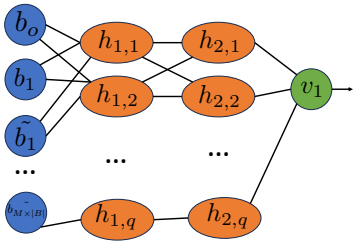

Fig. 3. The critic network receives the concatenated belief states of each of the M agents, each belief state of size $|B|$ and each agent's beam angle, with the rest of the inputs representing the belief state summary statistics. It has two hidden layers with q neurons, and outputs a single, joint value, v_1.

two hidden layers, both with 128 neurons, with the critic network containing two hidden layers, each with 256 neurons. The actor output contains the mean and standard deviation of a Gaussian policy for each agent, given the belief state summary statistic input taken from an agent's particle filters. The action drawn from this policy distribution is added to the current agent beam angle, leading to a continuous movement space. In implementation, a symmetrical translation was used for the agent on the right side of the environment to achieve state-space and action-space homogeneity. This allows each agent to operate with the same policy network while still deploying to their unique positions.

The state space for the POMDP is taken to consist of an agent's beam angle and the true angles from the agent to each of the adversaries. The state space contains $M+1$ continuous entries, with one entry for the self-angle and the rest for the angle from the agent to the corresponding adversary. However, because of the partial observability, an agent does not have access to the true adversary angles but only a noisy observation of them. In vector form, the observation for an agent w at timestep t is $o_w^{(t)} = [\phi_w^{(t)}, \gamma_{w,1}^{(t)} + \mathcal{N}(0, \rho_n^2), \ldots, \gamma_{w,M}^{(t)} + \mathcal{N}(0, \rho_n^2)]$, where ϕ_w is the agent's true angle, M is the number of adversaries operating in the environment, and $\gamma_{w,i}$ the true angle for adversary i. We do not assume knowledge of the adversaries' coordinates or the other agents' beam angles, nor do we try to estimate

TABLE I
REWARD TABLE FOR AN AGENT

Condition	Reward
$1 \leq i \leq M$ adversaries are jammed	i
No adversary is jammed	$-M$

TABLE II
PPO HYPERPARAMETERS

Hyperparameter	Value	Hyperparameter	Value
Discount	0.99	Actor Learn. Rate	5e-4
Hidden Neurons (actor)	128	Critic Learn. Rate	1e-3
Hidden Neurons (critic)	256	Episodes	5000
ϵ Clip	0.1	Max. Steps	400
Entropy Coefficient	1e-5	Optimization Epochs	2
Minibatch Size	25	Number of Particles	500
Process Noise ρ_p	0.03	Critic Loss Rate	0.5

these values.

We first present results for the scenario with two agents and two adversaries. The key criterion for the choice of rewards is if (1) is satisfied at each adversary, indicating that each adversary is jammed. A full summary of the reward given to an individual agent is presented in Table I. The PPO training parameters are listed in Table II. For these particular simulations, each of the agents' beam angles initialize at 0.0 radians, visually pointing directly at each other. The leader initializes at (50,60) and the follower initializes at (50,40) in the (x, y) coordinate system.

Learning curves are shown for different beamwidths with $\rho_n = 0.1$ in Fig. 4 to test different antenna strategies. On the x-axis is the episode number, simulated up to 15,000 to demonstrate continued convergence, and the y-axis is the cumulative average of the sum of the rewards across the two agents. The results demonstrate that the agents receive higher reward with larger beamwidths, while agents with smaller beamwidths receive noticeably less reward. This confirms intuition, as a larger beamwidth naturally would have more area to provide effective jamming radiation to the adversary. The performance is sensitive to the particular locations of the adversaries, which is correlated across time, resulting in the fluctuations in the learning curves.

Motivated by the need for cooperation by the agents to accomplish the goal of successfully jamming both the adversaries, we compare our PPO results to three separate strategies. The first strategy involves restricting the trained agent policy to be deterministic by taking the mean of the distribution as an action instead of a random sample from the distribution. This strategy serves to provide a comparison that elucidates the benefit provided by the Gaussian tails in the trained policy. The second strategy was to train each agent

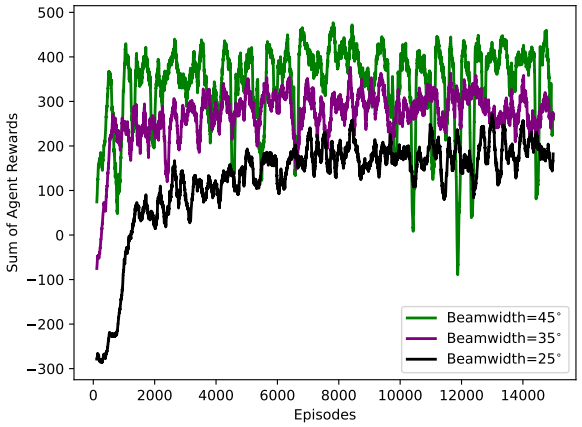

Fig. 4. Training reward curves for three different values of agent beamwidth. Note the variation in the reward curves due to the stochastic environment.

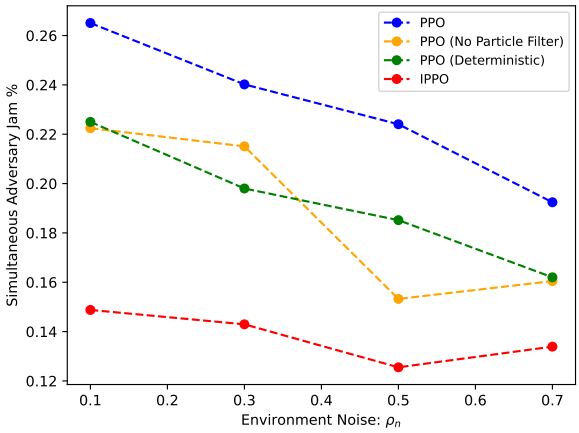

Fig. 5. Percentage of intervals in which both adversaries are simultaneously jammed for different learning models as a function of environmental noise level. Results are averaged over 4 random seeds.

independently (IPPO). Instead of a CTDE strategy, each agent has its own actor and critic, attempting to maximize its rewards. The third strategy consisted of passing each observation $o_w^{(t)}$ into the networks instead of using particle filters for belief state estimation. This served to understand the distinct benefit gleaned from the particle filter implementation. In each strategy, all environment and PPO parameters were held identical to Table II (including the actor and critic learning rates) to provide a useful comparison.

The performance of our MAPPO strategy is compared with that of each strategy for various values of the environmental noise variance, ρ_n, in Fig. 5 and Fig. 6. All results reported are averaged over four separate seeds. For each environmental noise value ρ_n, our stochastic PPO outperforms every other strategy in terms of both average reward and simultaneous jamming percentage.

When comparing our PPO solution to deterministic PPO (using only the mean of the distribution), we see in Table III that the deterministic policy tends to focus its jamming on the leader adversary significantly more than the follower, while the stochastic strategy has a more even distribution of jamming between the adversaries, demonstrating the benefit of our stochastic policy. Also, the stochastic PPO is able to more often jam both adversaries, even without knowing the position of the other agent's beam and outperforms the deterministic strategy in reward (see Fig. 5 and Fig. 6).

When the results are compared to the IPPO strategy, the benefits of the CTDE framework for cooperation become apparent. For each trained policy, our stochastic PPO with CTDE outperforms IPPO in terms of how often both adversaries are jammed (see Fig. 5). This drop in performance by the IPPO strategy is due to the lack of shared estimates in the critic during

TABLE III
RESULTS COMPARISON WITH STRATEGY 1 ($\rho_n = 0.1$)

Metric	Stochastic	Deterministic
Leader Jammed %	70	74
Follower Jammed %	38	27
Both Jammed %	25	20

training. Although both techniques are decentralized in deployment, the exploitation of a centralized critic in training and shared policy in deployment aids in the development of cooperative strategies. We also note that the results for IPPO have significant instability across different ρ_n. This instability is a recorded phenomenon in independent MARL (see [11], [12]), impacting performance in stochastic settings. Our method with CTDE effectively serves to stabilize the results while also outperforming IPPO at each ρ_n.

The final strategy comparison completed was against simulations that do not have the particle filters for state estimation. In this situation, the observation $o_w^{(t)}$ was directly passed into the networks, bypassing the particle filters. Much like the comparisons with IPPO, these simulations underperformed our PPO strategy, and presented significant instability caused by the lack of belief state tracking. From Fig. 5 and Fig. 6, one can note the instability of the non-particle filter simulations while our PPO strategy remains consistent, outperforming the non-particle filter simulations at all values of ρ_n.

To demonstrate scalable performance, we show performance results for two agents and either three or four adversaries in Fig. 7. Because of the high ratio of adversaries to agents, rewards overall are increased, as there are more targets for the agents to jam. We note that the trend of decreasing rewards with increasing ρ_n applies for higher numbers of adversaries.

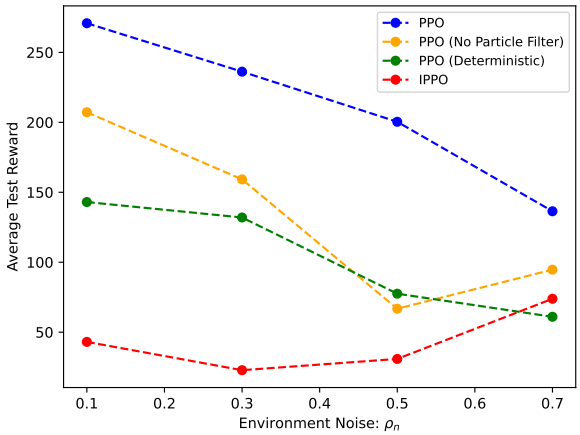

Fig. 6. Average reward for effective jamming for different learning models, as a function of environmental noise level. Results are averaged over 4 random seeds.

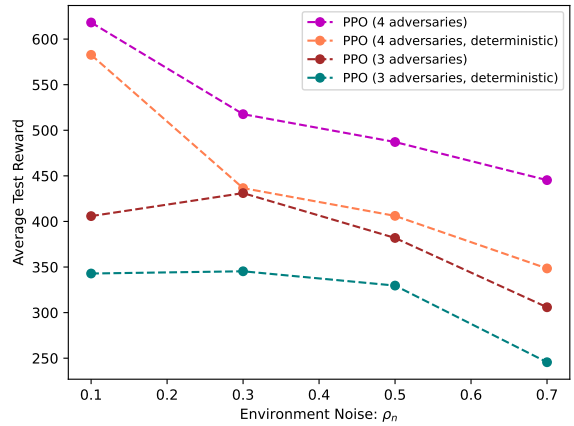

Fig. 7. Average reward for effective jamming with PPO for 3 and 4 adversaries, as a function of environmental noise level.

V. Conclusion

We considered the control of distributed jammers to protect an area from intrusion by mobile adversaries and formulate the problem as a POMDP. Because of the partial observability with a continuous state space and observations, we applied a particle filter for belief-state estimation. We evaluated the performance in a jamming scenario in which the adversaries are traversing a protected rectangular region in a leader-follower architecture. The performance of multi-agent PPO with the derived particle filters was compared with deterministic PPO, PPO without belief state estimation, and independent PPO approaches. The results demonstrate that the use of stochastic policies achieves an emergent form of coordination among the agents for the CTDE-trained PPO, resulting in significantly better performance than all compared strategies.

References

[1] H. Pirayesh and H. Zeng, "Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey," *IEEE Commun. Surveys & Tutorials*, vol. 24, no. 2, pp. 767–809, 2022.

[2] S. Bhattacharya, A. Gupta, and T. Başar, "Jamming in mobile networks: A game-theoretic approach," *Numerical Algebra, Control and Optimization*, vol. 3, no. 1, pp. 1–30, 2012.

[3] S. Bhattacharya and T. Başar, "Differential game-theoretic approach to a spatial jamming problem," *Advances in Dynamic Games: Theory, Applications, and Numerical Methods for Differential and Stochastic Games*, pp. 245–268, 2013.

[4] P. Valianti, S. Papaioannou, P. Kolios, and G. Ellinas, "Multi-agent coordinated close-in jamming for disabling a rogue drone," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3700–3717, 2022.

[5] P. Valianti, K. Malialis, P. Kolios, and G. Ellinas, "Multi-agent reinforcement learning for multiple rogue drone interception," in *Proc. Int. Conf. on Unmanned Aircraft Syst. (ICUAS)*, 2023, pp. 1037–1044.

[6] ——, "Cooperative search and track of rogue drones using multiagent reinforcement learning," in *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2024, pp. 2091–2096.

[7] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proc. Conf. on Theoret. Aspects of Rationality and Knowledge*, San Francisco, CA, USA, 1996, pp. 195—-210.

[8] C. D. Hsu, H. Jeong, G. J. Pappas, and P. Chaudhari, "Scalable reinforcement learning policies for multi-agent control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, 2021, pp. 4785–4791.

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

[10] W. Ran, R. Luo, F. Zhang, R. Luo, and Y. Xu, "Research on efficient multiagent reinforcement learning for multiple UAVs' distributed jamming strategy," *Electronics*, vol. 12, no. 18, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/18/3874

[11] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. Int. Conf. on Machine Learning*, 1993, pp. 330–337.

[12] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, "Learning with opponent-learning awareness," 2018. [Online]. Available: https://arxiv.org/abs/1709.04326

[13] C. Zhang, C. Tao, Y. Xu, W. Feng, J. Rasol, T. Hui, and L. Dong, "Autonomous defense of unmanned aerial vehicles against missile attacks using a GRU-based PPO algorithm," *Int. J. Aeronaut. and Space Sci.*, vol. 25, no. 3, pp. 1034–1049, 2024.