Extended Abstract

Motivation The prevalence of sea surface drones from companies like Saildrone necessitates an increased understanding and training of policies for these drones' autonomous use. These policies are then relevant across multiple types of operations, from defense, to scientific data collection and surveying, and even rescue missions. Compared to aerial drones, the underwater environment is more complex, and less research has been done in this field. This is due in part to limited sensor ranges, more sensor noise, and more deployment challenges. Moreover, targets can move in a three dimensional space, changing their depth and acoustic signatures for evasion.

Method We created a 20×20 km ocean area with semi-realistic sonar physics and a uniform depth of 500m. On the surface, there are eight autonomous drones with sonars (260 dB source level, 3.5 kHz) that need to search, find, and track a submarine operating at a depth between 200 and 500m that follows non deterministic movement patterns. For each drone, the detection probability follows the general sonar equations with added simulation of transmission loss, spherical spreading, absorption / attenuation, Doppler shift, and ambient noise. To avoid environment size issues, we make the world wrap around. In this environment we test a Kalman filter baseline, a deep Q learning (DQN) model with Hindsight Experience Replay, and a decentralized multi agent proximal policy optimization (PPO) model with continuous control.

Implementation For the Kalman filter baseline, the drones are randomly initialized, and then do an initial randomized search. Upon detection of the submarine, they switch to tracking mode, with the detecting drone following the submarine directly, and other drones go on predicted intercept paths instead. The DQN model uses a flattened vector representing a binned and discretized state space and a five layer fully connected network. It outputs eight discrete, binned headings for each drone. For increased training given the large state space, we implement hindsight experience replay and decaying epsilon learning. Finally, the multi agent PPO implementation is trained for each drone, with input observations containing the drone's state and detection information from three nearest neighboring drones within a 1km communication range. For the later two models, the reward function highly incentivizes positive detections, and strongly disincentivizes losing the submarine after already having detected it. This help the model learn to track the submarine after finding it.

Results We note that the Kalman fitler baseline got around 0.4 detections per agent per step, taking just under 50% of its iterations before first detection, and then reliably tracking the target. The DQN model did not learn very well, capping at 8-10 detection events per episode with no improvement in search-to-track transitions. Namely, it only learned how to optimally spread out the drones in the environment to maximize pings, but did not learn to track the submarine. Finally, the PPO model got two to four times higher detection rates than DQN, even matching the Kalman baseline's performance despite using more realistic communication constraints. However, it showed high variance during training, and periodic collapse. Even though it managed 80% target lock after detection, this performance was relatively unstable through the 1,000,000 training timesteps.

Discussion Interestingly, our baseline ended up being stronger than anticipated, given that the PPO model only matches the Kalman filter performance. The challenges of credit assignment and reward structures in the multi agent setting make training these RL models quite difficult. Moreover, these models can't distinguish between lack of signal and loss of signal, making it difficult to learn both a searching and tracking phase in the same policy.

Conclusion In this paper, we show that traditional reinforcement learning algorithms like DQN under perform when learning multi agent policies because of the large state space, getting worse results than the simpler Kalman filter baseline. However, dedicated multi agent solutions overcome this issue through decentralized control, matching the quality of the Kalman baseline, even with less information available to each agent. However, there is still work to be done in understanding the training characteristics of PPO and how to avoid policy collapse. Overall, we train a model that successfully finds a submarine and then manages to maintain an 80% lock rate on the submarine.

Reinforcement Learning for Sea-surface UAV Tracking of Underwater Submarines

Maxime de Belloy
Department of Computer Science
Stanford University
belloy@stanford.edu

Varun Agarwal
Department of Computer Science
Stanford University
varun4@stanford.edu

Abstract

This paper looks at multi agent, autonomous submarine tracking through active sonar in an ocean environment. We present an approach for multiple surface drones to cooperatively search and track an actively maneuvering submarine at depth. We implement Kalman filter, deep Q learning (DQN), and proximal policy optimization (PPO) approaches to this problem. We show that both the PPO and Kalman filter implementations are able to search and track the target, while the DQN model struggles from the size of its state space and only learns a searching policy. Overall, PPO demonstrates consistent detection and an over 80% lock on target after detection.

1 Introduction

The prevalence of sea surface drones from companies like Saildrone necessitates an increased understanding and training of policies for these drones' autonomous use. These policies are then relevant across multiple types of operations, from defense, to scientific data collection and surveying, and even rescue missions. Unlike aerial drones, the underwater environment is more complex, given the more complex propagation physics and more noisy sensors. In this paper, we look at implementing techniques from multi agent RL to the maritime environment. In particular, we look at the task of searching for and tracking an underwater submarine. We first set up this environment with a series of drones with active sonar that can communicate with each other and share information. After simulating a submarine with different action states and properly modeling its underwater sonar response, we train a Kalman Filter, a deep Q learning model, and proximal policy optimization to act on or learn policies to track the moving, erratic underwater target.

2 Related Work

Prior literature largely discusses the application of RL to UAV swarming and algorithm development for active sonar tracking, but not at the intersection of the two. Arranz et al. (2023) proposes an AI system to search for and track ground targets for security purposes. The chief technical goal of the paper is to avoid obstacles (static and dynamic) and prevent collisions between agents. In pursuit of this, they use a centralized swarm controller to assign high-level search and tracking tasks to UAVs; cooperative sub-agents act as individually trainable models responsible for low-level actions. The authors applied Proximal Policy Algorithm to learn to search an area and effectively "lock on" to a target even when the tracking has to be spread across multiple drones due to obstacles in sightlines. They demonstrate adaptability to varying numbers of targets and drones, but exhibit difficulties with computational cost and convergence. Their work holds relevance to the Methods discussed later; as opposed to computer vision, our agents perceive noisy sonar data instead, creating a different distribution to learn.

One of the issues with submarine tracking is that they are purpose-built for evasion. Work from Reddy et al. (2024) looks at how to distinguish underwater mines from rocks, achieving an accuracy of approximately 84% with a suite of non-deep learning classifiers - Support Vector Machines, Decision Trees, Random Forests, KNNs, XGBoosts, and Ensembles. While their work uses supervised learning methods, their setup will be helpful in finding submarines when they are not moving and may appear like rocks.

To evaluate the results of our system, we can compare against benchmarks found by Wang et al. (2010), which looks at performance results for different active sonar tracking algorithms - including a few versions of the Kalman filter and other probabilistic methods. They find that both the Extended Kalman Filter and the Converted Measurements Kalman Filter achieve equivalent performance in both single and multi-target tracking scenarios (though, CMKF produced marginally lower RMS errors and track fragmentation). Their work will also be useful to validate how improvement scales when the data collection for these algorithms is spread across multiple sensors in the drone fleet.

Finally, Wakefield et al. (2024) have shown promising results in applying deep RL techniques for smart tracking of an underwater sound emitting object with passive sonar. However, their work focuses on one sonar that has to be repositioned to triangulate a moving object. Moreover, with a lack of immediate range information, detection and localization become challenging problems. Since our work will focus on a swarm of drones, the triangulation can happen from different drones instead of having to move one sensor around. This work will be useful as a starting point for our passive sonar approach.

Our work appears at a gap in the literature between sonar tracking and reinforcement learning applications for controlling drone swarms.

3 Method and Environment

We built our own simulation environment, based off a 20×20 km area in the ocean, with a constant depth of $D = 500$ m. To describe the sound propagation speed through the environment, we defined uniform water characteristics with temperature T = 15°C and salinity S = 35 ppt. Applying Mackenzie (1981)'s equation for sound speed in oceans, we get:

$$c = 1448.96 + 4.591(15) - 5.304 \times 10^{-2}(15)^2 + 2.374 \times 10^{-4}(15)^3 \tag{1}$$

$$+ 1.340(35 - 35) + 1.630 \times 10^{-2}(250) + 1.675 \times 10^{-7}(250)^2 \tag{2}$$

$$- 1.025 \times 10^{-2}(15)(35 - 35) - 7.139 \times 10^{-13}(15)(250)^3 \tag{3}$$

$$\approx 1500 \text{ m/s} \tag{4}$$

Finally, we set a global ambient noise level of 40 dB re $1\mu Pa$.

3.1 Submarine

We have a submarine that operates within this environment at depths of 200-800 m following basic dynamics and constraints:

- **Kinematics:** Position update as $\mathbf{p}(t + \Delta t) = \mathbf{p}(t) + v\hat{\mathbf{n}}(\theta)\Delta t$ where $v \in [3, 10]$ m/s, θ is heading, $\Delta t = 5$ s
- **Depth dynamics:** $z(t + \Delta t) = z(t) + \dot{z}\Delta t$, where $|\dot{z}| \leq 1$ m/s
- **Turn dynamics:** $\theta(t + \Delta t) = \theta(t) + \omega\Delta t$, where $|\omega| \leq \pi/60$ rad/s

The submarine changes between multiple operation states following a continuous time Markov chain. The submarine can either go forward normally, zigzag, burst forward, run in silent mode (slower, but less detectable), or dive / climb to different depths.

The submarine then has the following acoustic signature:

$$S(f) = N_b \cdot H_e(f) + N_b \cdot H_r(f) + C(f) + W(f) + \xi(f) \tag{5}$$

where N_b is the base noise level, $H_e(f)$ is the engine noise transfer function, $H_r(f)$ the hull resonance responses, $C(f)$ the the cavitation noise, $W(f)$ the wake induced distortion, and $\xi(f)$ the stochastic noise with variance proportional to the base noise level. Specific definitions and values for each of these variables, as well as the Markov transition matrix for the submarine can be found in Appendix A.

3.2 Drones

The environment also contains 8 autonomous surface vehicles (ASV or drones), that are trying to track the submarine. The drone's have an onboard active sonar system with source Level: $SL = 260dB$ re $1\mu Pa$ @ $1m$, operating frequency: $f_0 = 3.5$ kHz, and ping frequency of $3.33Hz$. The drones search at 15 m/s, track at 8m/s, and can accelerate and turn at $3m/s^2$ and $\pi/4\ rad/s$ respectively.

To calculate the detection probabilities, we first need to define the signal to noise ratio or SNR, which captures how noisy a sonar reading is. It follows:

$$\text{SNR} = \text{SL} - 2\text{TL} + \text{TS} - \text{NL} \tag{6}$$

with TL the transmission loss, TS the target strength, and NL the noise level. The transmission loss is modeled as:

$$\text{TL} = 20\log_{10}(R) + \alpha R + \beta D \tag{7}$$

The first term represents noise spreading loss in dB, $\alpha = 0.03$ dB/km represents signal absorption in seawater, and $\beta = 0.03$ dB/100m models attenuation from signal depth.

The target strength is:

$$\text{TS} = \text{TS}_0 + 15\log_{10}(\sigma_s) + 5B(\theta) \tag{8}$$

with base level $\text{TS}_0 = 20$ dB, a noise term proportional to the acoustic signature level σ_s of the submarine, and a beam localization factor $B(\theta) = \cos(\pi r_{rel}/2)$ where $r_{rel} = d_h/r_{cone}$ represents the submarine's relative position within the sonar cone, as seen in Figure 1.

Finally, the noise level is

$$\text{NL} = 10\log_{10}(10^{N_a/10} + 10^{N_s/10}) \tag{9}$$

with ambient ocean noise of $N_a = 40$ dB and each drone producing $N_s = 5$ dB of operation noise.

Finally, for each drone, the detection probability is calculated a sigmoid function:

$$P_d(\text{SNR}) = \frac{1}{1 + \exp(-(\text{SNR} - \text{SNR}_t)/k)} \tag{10}$$

with detection threshold $\text{SNR}_t = -10$ dB and slope $k = 2$.

When the drones detect the submarine, they get noisy, Doppler-shifted information about the submarine's range ($\hat{R}$), bearing ($\hat{\theta}$), and confidence of detection (C):
$\begin{cases} \hat{R} = R + \epsilon_R, \text{ where } \epsilon_R \sim \mathcal{N}(0, 0.01R) \\ \epsilon_\theta \sim \mathcal{N}(0, 0.03) \text{ rad} \\ C = \min(0.98, \text{SNR}/20 \cdot m_D \cdot m_R \cdot m_\theta) \\ m_D = 1 + |\Delta f|/1000,\ m_R = 1 - |\epsilon_R|/(0.1R_{max}),\ m_\theta = 1 - |\epsilon_\theta|/(0.5\theta_{BW}) \end{cases}$

The environment is visualized in a 3D plotted world as shown in Figure 1

4 Kalman Filter Baseline

We set up a Kalman filter based baseline to compare against our RL agent. In this baseline, the drones are randomly deployed in the environment on the surface. They then move around randomly until they first detect the submarine.

When they get a positive ping (SNR > -10 dB and confidence > 0.2), the drones start tracking the submarine by predicting the submarine's (depth invariant) position.

The initial detection state x is defined as a function of the detecting drone's current position and noisy range and bearing estimates. Similarly, the covariance matrix P_0 uses position uncertainty $\sigma_p = 50$ m:

$$\begin{cases} x = [x, y, v_x, v_y]^T = [x_v + \hat{R}\cos(\hat{\theta}), y_v + \hat{R}\sin(\hat{\theta}), v_{sub}\cos(\hat{\theta}), v_{sub}\sin(\hat{\theta})]^T \\ P_0 = \text{diag}[\sigma_p^2, \sigma_p^2, v_{sub}^2, v_{sub}^2] \end{cases} \tag{11}$$

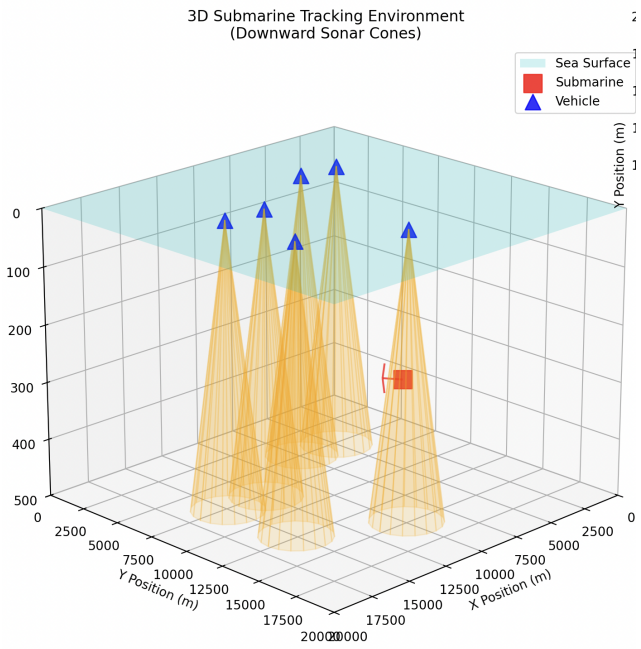

Figure 1: Submarine Tracking environment

Then, state transitions follow simple velocity updates with state transition matrix F, process noise covariance Q, and covariance prediction:

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q\,, \quad q = 0.1m/s^2 \tag{12}$$

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; Q = q^2 \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix} \tag{13}$$

When the drones get new detections, the Kalman filter is updated as:

$$\begin{cases} \mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R})^{-1} \\ \mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{x}_{k|k-1}) \\ \mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1} \end{cases} \tag{14}$$

with measurement model, observation matrix, and detection confidence covariance as:

$$\mathbf{z}_k = \begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x_v + \hat{R}\cos(\hat{\theta}) \\ y_v + \hat{R}\sin(\hat{\theta}) \end{bmatrix}, \; \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \; \mathbf{R} = \frac{\sigma_m^2}{\max(0.5, C)}\mathbf{I}_2 \tag{15}$$

With this model of the submarine's position and bearing, the current detecting drones follows the predicted submarine position, while the remaining drones are assigned to spaced intercept positions in the direction the submarine is currently moving. These intercepts are recalculated every couple of steps (8 seconds).

5 Deep Q-Learning Baseline

We also decide to implement a single policy Deep Q learning model to track the submarines. The learned policy acts on all 8 drones simultaneously. To decrease the large state space, it is discretized to a $32 \times 32 \times 3$ flattened vector describing the surface position of the drone clipped to a grid and the

detection confidence for each drone. To further simplify, each drone can only operate on 8 discrete headings with angles $\theta_a = 2\pi a/8$ for $a \in \{0, 1, ..., 7\}$.

The underlying network has 5 intermediate fully connected layers with sizes 256, 128, 128, 64, and 64. The resulting output produces $n * 8$ actions $= 64$ Q values. These give the value for each action for each drone.

We use a simple reward function that incentivizes finding and following the submarine:

$$r(s,a) = \begin{cases} 10.0 & \text{if submarine detected} \\ -0.5 & \text{if previously tracking but lost} \\ -0.1 & \text{otherwise (searching)} \end{cases} \tag{16}$$

The training also uses Hindsight Experience Replay (Andrychowicz et al. (2018) - HER), to help get signal from rollouts where the submarine was not found. In rollouts where the submarine was not found, HER substitutes one of the achieved drone position as the goal position. With HER ratio $\rho = 0.3$ and $k = 2$ virtual goals per episode, the augmented reward is:

$$r_{\text{HER}} = -||\mathbf{p}_{\text{achieved}} - \mathbf{p}_{\text{goal}}||_2 \tag{17}$$

The model is trained using Adam (Kingma and Ba (2017)) for 10000 epochs using ϵ greedy exploration that decays from $\epsilon = 1.0$ to $\epsilon_{\text{min}} = 0.05$. Episodes run for 1000 steps, with 8 drones initialized randomly at 3km from the center of the grid.

5.1 DQN-Learned Policy

The average policy and drone direction after training are plotted in Figures 2 and 3 (note that the environment is wrap-around):

We see from those figures that the policy learns only to maximally spread out the drones across the space. This way, it covers more of the area, and detects the submarine more frequently. However, likely due to the still too large state space, the policy doesn't learn to track the submarine after first contact.

This incomplete training may also be due to the global policy, instead of letting each agent learn its own. While the single policy has more information to work with, the resulting state and action spaces are eight times as large.

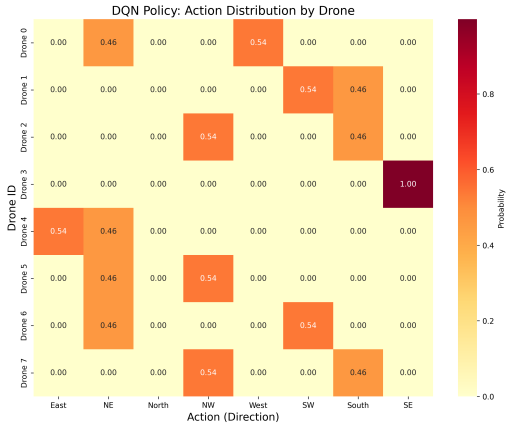

Figure 2: DQN Policy per Drone (1000 sims)

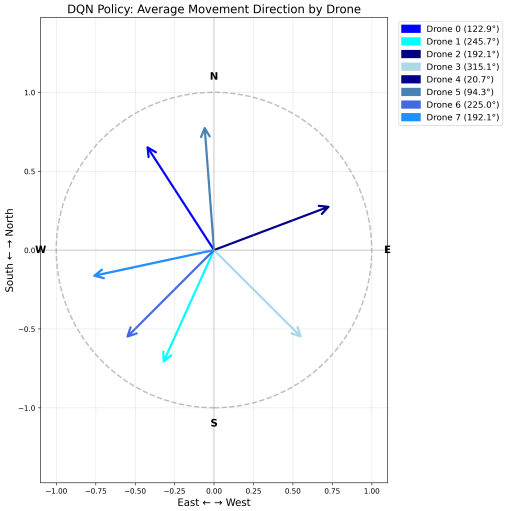

Figure 3: DQN Direction per Drone (1000 sims)

6 multi agent PPO

In this scenario, the cooperative task is modeled as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \Omega, O, N, \gamma \rangle$ ($N = 8$ is the number of ASV agents). The state space $\mathcal{S}$ encompasses the continuous 3D positions, headings, and internal states of the submarine and all ASVs. Each agent $i \in \{1, ..., N\}$ selects a continuous action $a_i \in \mathcal{A}_i$.

Critically, the state $\mathbf{s}$ is not fully observable. Each agent i receives a local observation $o_i \in \Omega_i$ from a shared observation function $O(\mathbf{s}, i)$, which is conditioned by stochastic sensing modalities (sonar). Agents aim to learn a decentralized policy $\pi_i(a_i|o_i)$ that collectively maximizes the expected discounted future return.

Each agent controls its heading (turn rate / direction). The policy network outputs a continuous, normalized action $\hat{a}_i \in [-1, 1]$. This is then scaled to the vehicle's maximum turn rate $\dot{\theta}_{\text{max}}$, yielding the final action $a_i = \hat{a}_i \cdot \dot{\theta}_{\text{max}}$. In building a system to coordinate homogeneous agents, focusing on the heading offers a powerful simplification that abstracts away specific vehicle dynamics while retaining autonomy in movement. Agent speed is governed by a simple heuristic: a higher speed for tracking is triggered when an agent has a high-confidence detection; else a default "search" speed is used. Narrowing the policy to maneuver decisions enabled more efficient rollouts and ease of training, while retaining adaptive behavior.

The design of the observation vector o_i for each agent is critical for learning effective cooperative strategies - the information shared led to significant differentials in detection rates (all other hyperparameters kept constant). We construct a fixed-size observation vector composed of two parts:

1. Self-Agent Information: normalized range, absolute bearing, relative bearing (target bearing vs agent heading), and confidence of sonar detection. Positional and relative information is necessary to provide clear, immediate error signals for maneuvering and disambiguate heading from the target's frame of reference.

2. Team Information: to enable cooperation, agents share information with their nearest neighbors. Here, we pass data for the $K = 3$ closest neighbors within a communication radius R_c (1000m). For each neighbor j, the following information is included relative to agent i to ensure the policy is invariant to translation and rotation: relative position $(\Delta x, \Delta y)$ (normalized by R_c), relative heading $(\theta_j - \theta_i)$, neighbor's last-known sonar detection confidence. In the case that the agent has fewer than K present neighbors, we pad the remaining space in the vector with 0s, to simplify the fixed-size input for the network (and batch training).

The most complex part of this design was engineering a reward that surpasses the conventional naive, sparse reward for detection and leverages multi agent behaviors. Subsequently, we created a denser reward function that promotes a strategy with search and track phases. When the team of agents are not tracking the submarine, they are engaged in search. Here, we apply a 'coverage penalty', in which we penalize agents which come into proximity of each other (beyond a radius R_{cov}). The goal is to directly incentivize the agents to fan out, maximize coverage, and accelerate detection of the submarine. Formally, we can write this (for agent i) as: $R_{\text{cov}} = \sum_{j \neq i} w_{\text{cov}} \cdot \max(0, 1 - d_{ij}/R_{\text{cov}})$. Tracking mode is initiated by an agent producing a detection confidence > 0.5 (a tunable hyperparameter); it only disengages if there is no detection after a period of $t = 10$ seconds past the previous detection (temporarily pausing the penalties while they attempt to reacquire contact). The goal in this mode is structured to include: a team reward (distributed to all agents) proportional to the maximum confidence achieved by an agent ($R_{\text{team}} = w_{\text{team}} \cdot \max(c_1, ..., c_N)$), an individual reward (distributed to the agents currently generating detections), and a convergence bonus (distributed to non-detecting agents) - in which the reward inversely correlates with the distance to the actively-detecting agent(s). In all timesteps (regardless of mode), there is a minute time penalty applied to instantiate urgency. The structure of this reward creates richer, localized reward densities which allow PPO to outperform our DQN baseline.

Our proposed model involves centralized training via PPO. A single shared policy network π_ϕ and value function V_ψ (composed of shared parameters), are trained on the collective experience of all N agents for (a) computational efficiency and (b) leveraging the homogeneity of agents in this setup. The policy is trained on-policy through rollouts of 16384 agent-steps (2048 steps / agent). Furthermore, we use Generalized Advantage Estimation (GAE) to compute stable advantage targets: $\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$, $\delta_t = r_t + \gamma V_\psi(o_{t+1}) - V_\psi(o_t)$. The policy is updated by optimizing

the PPO-clip objective function over $K = 6$ epochs on the collected batch. The objective for the actor is: $L^{\text{CLIP}}(\phi) = \hat{\mathbb{E}}_t \left[\min \left(\frac{\pi_\phi(a_t|o_t)}{\pi_{\phi_{\text{old}}}(a_t|o_t)} \hat{A}_t, \text{clip} \left(\frac{\pi_\phi(a_t|o_t)}{\pi_{\phi_{\text{old}}}(a_t|o_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right]$. Here, $\epsilon = 0.1$ is the clipping hyperparameter. The critic is updated by minimizing MSE and entropy is applied to the loss to promote exploration. The architecture follows a classic actor-critic approach, with a shared feature extraction backbone composed of an MLP with two fully-connected hidden layers (256, 128) with ReLU activations. The actor produces a mean of a Gaussian probability distribution for the action space; standard deviation is separately trainable to permit exploration variance tuning. Increasing the width or depth of the network (our largest network had dimensions 512, 256, 256, 128) led to no discernable improvement on the task, suggesting that the bottleneck was in the observation vector and reward function.

7 Results

7.1 Quantitative Evaluation

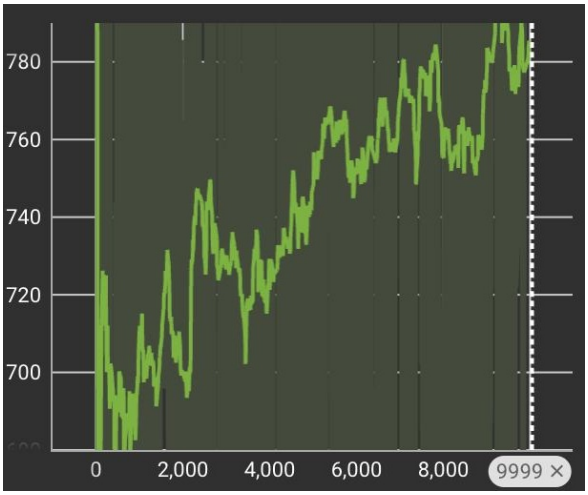

Figure 4: DQN Baseline

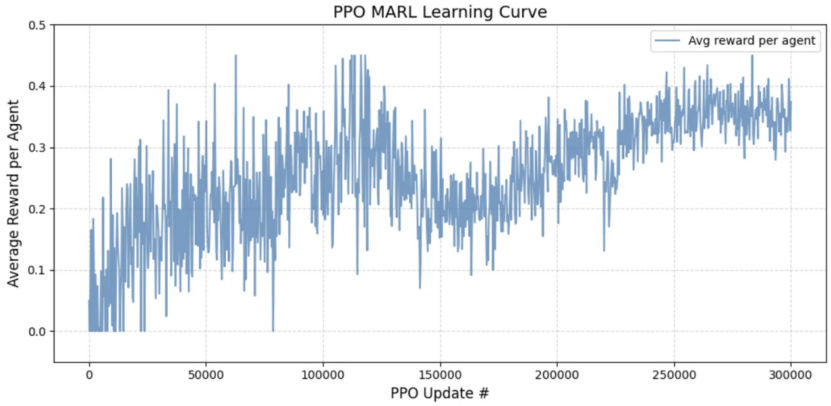

Figure 5: Successful Training Run of PPO-MARL

The Kalman filter baseline (KF) exhibits a detections per agent per step of approximately 0.4. Across simulations, KF typically consumes 45-55 percent of the time in the search phase, as agents remain stationary or engage in random walks. However, when an initial stream of detections is made, Kalman converges quickly on and consistently follows the target, contributing to a high detection rate per episode. However, the caveat is that all KF agents communicate when tracking occurs (whereas we

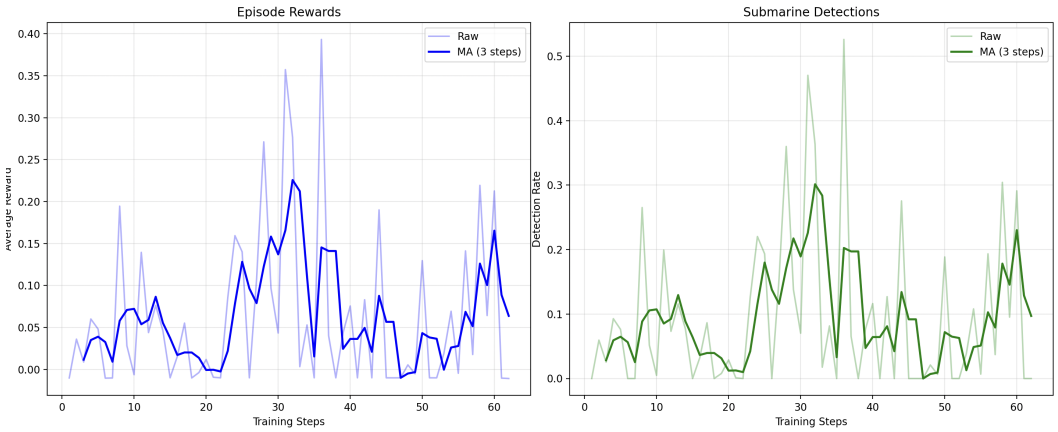

Figure 6: Policy Collapse of PPO-MARL

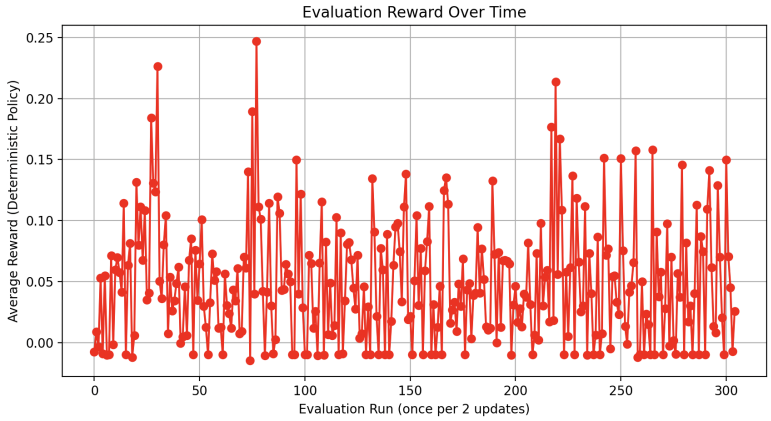

Figure 7: High Variance Evaluation of PPO-MARL

introduced realistic communication limits in our PPO-MARL scenario). The DQN baseline exhibits minimal effective learning - Figure 4 demonstrates how the agents remained at approximately 8-10 detection events per episode (per agent), with marginal improvements to either search minimization and/or tracking. PPO-MARL demonstrates the capacity to navigate the continuous, sparse-reward space - as seen in Figure 5, where it shows consistent detection; considering travel time to the submarine and reaching around 80 percent lock once detected, PPO MARL is similar to an optimal outcome. PPO-MARL outperforms the DQN's detection rate between 2-4x (when normalizing detection / agent / step), and matches the KF baseline (which has perfect information). However, on average, it has inconsistent and unstable behavior. Figure 6 illustrates how the model routinely undergoes policy collapse - oscillating from high-reward outcomes to rollouts containing 0 detections periodically. Moreover, Figure 7 depicts the high variance and uninterpretable performance of the model as we trained the system across 1000000 timesteps.

7.2 Qualitative Analysis

A core limitation of DQNs is its inability to map onto continuous action spaces; despite the discretization of the space into cardinal directions in this instance, the model failed to learn beyond the scope of expanding coverage during tracking. Many of the model's shortcomings are systemic issues shared by PPO-MARL. The PPO-MARL system continued to struggle with the non-stationary dynamics of the targets and collaborative agents; given that the Markov assumption is violated and the rules are continuously evolving (considering the 20km x 20km x 1km space of positions, trajectories, and noisy pings), it is not unexpected that the model failed to establish a consistent policy over the course

of 500,000 timestep training runs. Moreover, performing credit assignment remains a tricky task - decomposing individual contributes to the reward function meant that the signal the model used to learn was heavily diluted - indicating the necessity for separability in or of the policy used to train the agents. Additionally, there is the issue of intractability of constantly shifting, partially-observable spaces obscured by noise (both spectral and in confidence approximations shared between limited sets of neighbors).

8 Discussion

Despite the scale of model size, training, and observation / reward setup, the deep learning models presented were unable to measurably surpass the KF baseline. They proved - in the scope of training conducted - unable to learn a search-and-track strategy that reliably coordinates and converges on a submarine. Given the feedforward nature of models and lack of observation-stacking, they had to operate purely reactively to incoming observations - their inability to distinguish between no signal during search and loss of signal during tracking (despite the implementation of a grace period in PPO-MARL to re-establish contact) likely contributed to the rapid decay of rewards we see in the training trends. In comparison, KF baseline is not a memoryless approach - the use of recursive estimation and probability-data-associations enables it to better map plausible trajectories of the target and maintain robust belief (aided by the perfect information setup in the baseline). Summatively, our findings convey the necessity of using memory-based methods to adapt to high-dimensional, non-stationary environments - regardless of the deployment of RL.

9 Conclusion - Future Directions

Looking forward, we see a number of directions in which we can improve the simulation realism and model performance. We had built environments containing procedurally generated obstacles of varying size and position, but they suffered from significant slowdown (as collision avoidance introduces expensive computations). In the near future, trialing in environments with obstacles optimized to mitigate rollout efficiency loss is a key goal. Moreover, exploring Graph Attention Network-based message passing between agents (create embeddings to share, and use attention coefficients to model confidence or uncertainty of reports) and meta-learning (adding a controller/meta-level may allow for formation of subgroups of UAV activity, joint optimization, and more robust task solutions), are core goals to pursue for model improvement. In terms of the task structure, there are four major areas of improvement: (1) adversarial training for submarine evasion - enabling the submarine to learn dynamic, fine-grained maneuvers beyond hard-coded patterns encoded in a transition-state matrix, (2) systems with agent specialization / heterogeneity to evaluate how novel optimal strategies shift and parameter or policy-sharing evolves, (3) systems with multiple target submarines, and (4) transitioning to a full-scale simulator using a ROS/Gazebo acoustic plugin with generated bathymetry.

10 Team Contributions

- **Group Member 1 (Maxime):** Worked on setting up the environment and the Kalman filter baseline. Also, wrote and trained the DQN model.
- **Group Member 2 (Varun):** Helped fix the sonar characteristics and introduce noise and obstacles in the environment. Wrote and trained the multi agent PPO model.

Both members worked on the poster and final report equally.

Changes from Proposal In the proposal, we wanted to simulate passive sonar techniques. However, it was not feasible to accurately simulate the ambient sound spectrograph of the ocean floor and the sound signal and acoustic signature of the submarine.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2018. Hindsight Experience Replay. arXiv:1707.01495 [cs.LG] https://arxiv.org/abs/1707.01495

Raúl Arranz, David Carramiñana, Gonzalo de Miguel, Juan A. Besada, and Ana M. Bernardos. 2023. Application of Deep Reinforcement Learning to UAV Swarming for Ground Surveillance. *Sensors* 23, 21 (Oct. 2023), 8766. https://doi.org/10.3390/s23218766

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

K. V. Mackenzie. 1981. Nine-Term Equation for Sound Velocity in the Oceans. *The Journal of the Acoustical Society of America* 70, 3 (1981), 807–812. https://doi.org/10.1121/1.386920

B. Nikhil Krishna Reddy, Rashmi M. R, Chockalingam Aravind Vaithilingam, and S. Kamalakkannan. 2024. SONAR Based Under Water Mine Detection Using Machine Learning Algorithms. In *2024 4th International Conference on Innovative Practices in Technology and Management (ICIPTM)*. 1–4. https://doi.org/10.1109/ICIPTM59628.2024.10563616

Joshua J. Wakefield, Adam Neal, Stewart Haslinger, and Jason F. Ralph. 2024. Sonar Path Planning Using Reinforcement Learning. In *2024 27th International Conference on Information Fusion (FUSION)*. 1–8. https://doi.org/10.23919/FUSION59988.2024.10706484

J. Wang, A. von Trojan, and S. Lourey. 2010. Active sonar target tracking for anti-submarine warfare applications. In *OCEANS'10 IEEE SYDNEY*. 1–7. https://doi.org/10.1109/OCEANSSYD.2010.5603790

A Submarine Characteristics

A.1 State changes Markov Chain:

The transition probability matrix for the submarine is:

$$P = \begin{bmatrix} 0.80 & 0.10 & 0.05 & 0.03 & 0.02 \\ 0.70 & 0.20 & 0.05 & 0.03 & 0.02 \\ 0.60 & 0.15 & 0.10 & 0.08 & 0.07 \\ 0.70 & 0.10 & 0.08 & 0.07 & 0.05 \\ 0.70 & 0.10 & 0.08 & 0.07 & 0.05 \end{bmatrix}$$

for the ordered state space: {normal, zigzag, burst, silent, depth}

Depending on the position of the closest drone, the submarine starts with state drawn from a Poisson process with rate $\lambda = 0.2\ \mathrm{s}^{-1}$ (baseline) or $\lambda = 0.5\ \mathrm{s}^{-1}$ (under threat, distance < 500 m). The length of time spent in a state is exponentially distributed.

The submarine's acoustic signature is calculated as

$$S(f) = N_b \cdot H_e(f) + N_b \cdot H_r(f) + C(f) + W(f) + \xi(f) \tag{18}$$

with:

- $N_b = 1.2(1 + 0.5v/v_{max} + 0.3|\omega|/\omega_{max} + 0.2|\dot{z}|/\dot{z}_{max})$ the base noise level
- $H_e(f) = 0.8$ for $|f - 50| < 20$ Hz (engine noise at 50 Hz)
- $H_r(f) = 0.6$ for $|f - f_r| < 10$ Hz, $f_r \in \{120, 240, 360\}$ Hz (hull resonances)
- $C(f) = N_b \cdot 0.2 \cdot \max(0, 2(v/v_{cav} - 1))$ for $f > 200$ Hz (cavitation noise, $v_{cav} = 0.6v_{max}$)
- $W(f) = (1 + 1.5w_s)$ for $f < 100$ Hz (wake induced distortion, $w_s = 0.8v/v_{max}$)
- $\xi(f) \sim \mathcal{N}(0, 0.1N_b)$ (gaussian noise)

When the submarine is in stealth mode, N_b is reduced by a factor of 0.8.