Providing predictions of adversary movements in a gridworld environment to a human-machine team improves teaming performance

Jeffry Coady, Paul Dysart, Aidan Schumann, Stephan Koehler, Michael Munje, et al.

Jeffry A. Coady, Paul Dysart, Aidan Schumann, Stephan A. Koehler, Michael J. Munje, William D. Casebeer, David M. Huberdeau, "Providing predictions of adversary movements in a gridworld environment to a human-machine team improves teaming performance," Proc. SPIE 12538, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V, 125380N (12 June 2023); doi: 10.1117/12.2663881

SPIE.

Event: SPIE Defense + Commercial Sensing, 2023, Orlando, Florida, United States

Providing Predictions of Adversary Movements in a Gridworld Environment to a Human-Machine Team Improves Teaming Performance

Jeffry A. Coady[a], Paul Dysart[a], Aidan Schumann[a], Stephan A. Koehler[a], Michael J. Munje[a,b], William D. Casebeer[a], & David M. Huberdeau[a]

[a]Artificial Intelligence and Machine Learning Lab, Open Innovation Center, Riverside Research, 70 Westview St., Lexington, MA 02421
[b]School of Computing, Georgia Institute of Technology, 801 Atlantic Dr NW, Atlanta, GA 30332b

ABSTRACT

Knowing the future states of an adversary in an adversary-avoidance game can impart a survival advantage. To assess how predictive modeling helps agents achieve goals and avoid adversaries, we tested the efficacy of three predictive algorithms within a gridworld-based game. For one predictive algorithm, model predictions of adversary moves furnished to an agent helped the agent avoid capture compared to a case without predictions. A human-machine team scenario also benefited from model predictions, while humans alone experienced a ceiling effect. We investigated the efficacy of two additional predictive algorithms and present a maritime vessel pursuit scenario.

Keywords: Human-Machine Teaming; Theory of Mind Modeling; Cognitive Modeling

1. INTRODUCTION

Determining optimal pursuit or avoidance behavior is a well-known problem in optimal control and route-planning, with many solutions under various conditions or assumptions, including proportional navigation [1], deviated pursuit [2], one- or two- sided optimization [3], and many others [e.g., 4, 5, 6]. However, one common principle underlying many optimal pursuit or avoidance strategies is that predicting in advance the future state of an adversary enables an agent to plan a course that is more optimal than in the absence of accurate predictions, whether for adversary avoidance or pursuit. However, obtaining accurate predictions can be challenging, especially in dynamic settings and when there are multiple independent agents that could be pursued or evaded. Here, we implement several methods for predicting the future states of an adversary, including a model inspired by the psychological concept of theory of mind (further defined below), and demonstrate that in a human-machine teaming setting, furnishing these predictions to the team significantly improves teaming performance.

Theory of Mind (ToM) is the ability to attribute mental states to oneself and to others [7]. Individuals with ToM can think about their own and others' emotions, beliefs, desires, and knowledge, and in turn use that information to explain or predict their behaviors. ToM is hypothesized to be integral to human social interactions [8], language development [9], and higher-level cognition, such as decision making [10] and inhibitory control [11]. The hallmark of ToM can be demonstrated through the false belief test [12], which requires an individual to realize that another person has incorrect information, and to anticipate that person will act in a manner consistent with that incorrect information.

Attempts to model ToM began with the work of Baker and colleagues [13, 14]. They created a Bayesian inverse planning model to predict how agents move through artificial maze-worlds modelled as partially observable Markov Decision Processes (POMDP). Agents navigated through their maze-worlds, potentially around obstacles, to one of three target objects. Their Bayesian ToM (BToM) model was given observations of agent trajectories through those environments and tasked with predicting future movements based on the current configuration of the maze-world,

including locations of the navigating agent, target objects, and barriers, along with the agent's previous actions. Their model matched human predictions of agent movements. Rabinowitz and colleagues [15] offered a Machine Theory of Mind (MToM) model to predict how agents navigate a gridword. Their gridworld was an 11 × 11 grid containing an agent, four target objects, and potential barriers, all randomly placed. Their model also learned through observing agents navigate different gridworlds and predicted agent movements based on three modules — (1) a *character net* that parses previous trajectories, (2) a *mental net* that parses the recent trajectory, and (3) a *prediction ne*t that predicts upcoming moves based on character net and mental net embeddings. They reported that their MToM model successfully predicted how different types of agents navigated gridworlds, including agents holding false beliefs. Nguyen and Gonzalez [16, 17] presented a different ToM model based on Instance Based Learning Theory (IBLT, [18]). Their cognitive Theory of Mind (CogToM) model also observed agents navigating gridworlds similar to those used by Rabinowitz and colleagues, and learned to predict agent movements based on previous instances of agent behavior. Like Baker and colleges, Nguyen and Gonzalez reported their model matched human predictions of agent movements, and like Rabinowitz and colleagues, their model passed a false belief task.

Despite different architectures and learning mechanisms, all three groups have successfully demonstrated that computer models can learn to predict how autonomous agents move through simple environments. One next step is to explore how predictions of agent movements might prove useful. To that end, we present experiments meant to extend previous work. The first experiment involves goal-seeking, adversary-avoiding agents in a simple gridworld environment, in three experimental conditions. In the first condition, adversarial agents are introduced into the gridworld simulations. Consistent with previous studies, agents have a goal of capturing a target object. The new adversary, however, seeks to capture the agent before it can capture a target object. Rather than predicting how an agent navigates a gridworld, our ToM model learned to predict how an adversary moves as it attempts to capture the agent, introducing an adversary-avoidance element to the game. If agents are provided predictions of adversary movements, will they adjust their own movements to avoid the adversary, and thus have better performance in capturing their target without being captured themselves? In a second condition, we replace the computer agent with a human agent and ask human subjects to navigate gridworlds while avoiding an adversary in order to capture a target object. Will model predictions of adversary moves help humans capture target objects? Finally, in a third condition, we modify the game to accommodate both a computer and a human agent acting as a human-machine team. Both agents, computer and human, have a goal of capturing one of the target objects while an adversary attempts to capture either agent. Will predictions of adversary moves improve performance for the human-machine team?

In three follow up studies, we tested the effectiveness of different methods for establishing machine theory of mind (i.e. providing predictions of the future actions and states of a pursuing adversary) on the performance of computer agents. In the first, we test the use of a k-level reasoning algorithm [19], where the agent and adversary each recursively predict and account for one-another's action policies in a gridworld environment. In the second, we test a tracking and predictive algorithm, probabilistic data association filter (PDAF), along with particle swarm optimization (PSO) in a multi-adversary scenario. Finally, in the third additional study, we introduced a predictive algorithm that used real-world maritime vessel-tracking data to demonstrate the feasibility of predicting ship movements to introduce machine ToM into a vessel interception setting.

2. EXPERIMENT 1

The typical gridworld experiments include an environment, an agent, and an observer. For this first experiment, the gridworld was modified to include an adversary in all conditions, and a human-as-an-agent in some conditions. In all conditions, we compared success at capturing target objects when computer and/or human agents received no predictions of adversary movements to conditions in which they did receive model predictions.

2.1 Environment

The gridworld environment was based on that used in the MToM [15] and CogToM [16, 17] studies. It was an 11 × 11 grid instantiated as a NumPy matrix. When visualization was required, the NumPy matrix was converted to a text string with equal-width characters representing the different components. Each grid contained between 0 – 4 randomly placed barriers. The environment generation algorithm randomly chose the number of barriers, and for each barrier, the

algorithm chose a random starting point, a random orientation (vertical, horizontal, or diagonal), and a random length (from 1 – 11). Barriers were allowed to overlap. The algorithm then randomly placed four target objects into available squares. The four objects were assigned reward values from a Dirichlet distribution with $\alpha = 0.01$. This resulted in one of the targets having a reward value approaching 1, while the other three had values approaching 0. In this sense, there was a preferred target, randomly determined for each environment. Finally, the agents (computer and/or human) and adversary were randomly placed in open squares. Sample environments are shown in Figure 1.

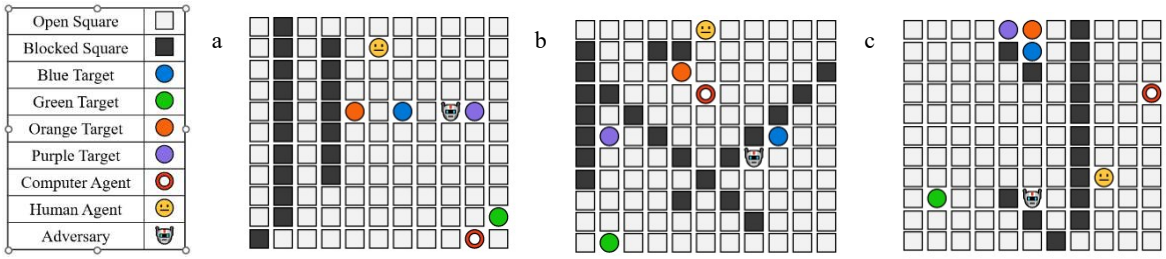

Figure 1. Sample gridworlds. In the leftmost grid (a), both the computer agent and the human agent can reach all targets, and the adversary can reach both agents. In the middle grid (b), the computer and human agents can reach one of the targets, but others are blocked; the adversary cannot reach either agent. In the rightmost grid (c), neither agent can reach any of the targets, and the adversary cannot reach either agent. Most (>95%) instances of the gridworld were of the first type. Rewards were randomly assigned to targets for each instance.

2.2 Computer Agent

Computer agents were provided a policy to navigate their gridworld. Agent policies were calculated using Finite Horizon, a backwards induction algorithm available in the Markov Decision Process (MDP) Toolbox (available at https://pymdptoolbox.readthedocs.io/en/latest/api/mdp.html#mdptoolbox.mdp.FiniteHorizon). The Finite Horizon algorithm requires a transition probability matrix, P, the probability of the next location given a current location and a move command (shape 4 moves, 121 starting states, 121 ending states), and a reward matrix, R, the reward associated with the next location given a current location and a move command (shape 4,121,121). The optimal policy revealed the path resulting in the highest reward, in most cases to the target object with the highest reward. In cases where the target object was blocked, the highest reward typically was achieved by navigating to the target closest to the agent. In cases where the agent was blocked from all objects, it simply ran out the clock.

In the absence of an adversary, a single session proceeds as follows: a random environment is generated, and an agent policy is calculated via Finite Horizon. The agent then moves up, down, left, or right, as determined by the policy for the agent's current location. The barriers and target objects do not move. The agent continues to move until it lands on any one of the four targets, or times out after 31 moves. All moves incur a 0.01 move penalty, and any attempts to move into a barrier or off the grid incur a 0.05 wall penalty, with the agent remaining in its same position. In this base case without an adversary, the agent captures the target with the highest reward in 96.8 percent of trials. It captures one of the non-preferred objects in 2.6 percent of trials and fails to reach any object in 0.6 percent of trials. In cases with two agents, one of the agents captures the highest-reward target in 96.9 percent of trials, one of the lower-reward targets in 3.0 percent of trials, and none of the targets in 0.1 percent of trials.

2.3 Adversary

Like the computer agent, the adversary derives its policy from a Finite Horizon algorithm, except that its target is an agent. In conditions including both a computer and a human agent, both are equally rewarded targets. The adversary incurs the same 0.01 move penalty and 0.05 wall penalty as agents. Further, it is prohibited from capturing any of the target objects, and any attempts to do so incur the 0.05 wall penalty. Sessions are modified so that agent and adversary policies are calculated via Finite Horizon at each step. This allows the adversary to adjust its policy as the agent and/or human move through the grid world, and the agent to adjust its policy in response to adversary movements in cases where model predictions are provided. Sessions continue until (1) the computer or human captures one of the objects, (2) the adversary captures the agent or human, or (3) 31 moves have occurred.

2.4 Observer

We implemented an observer model to predict how an adversary moves through an environment in its goal of capturing an agent, either computer or human. Two different observer models were created — one for simulations with a single agent, either computer or human, and another for simulations with two agents, for the human-machine team conditions. Observer models were convolutional neural networks (CNN) trained to predict how the adversary moved through gridworlds in its goal of capturing the agent. Inputs to the models were matrices containing information about the current state of the grid, potentially with previous states (n_past = 0-10 previous states), and matrices containing the actual moves at each step. Input matrices were layers of 11 × 11 matrices, with one layer for barriers, four layers for the four target objects, one layer for the computer agent's current position (when included), one layer for the human agent's current position (when included), one layer for the adversary's current position, and 0 – 10 layers for the adversary's previous positions (to provide either just the current adversary location of a progressively longer trajectory up to the current adversary position). That is, matrix shapes ranged from (7,11,11) to (19,11,11). Adversary position layers were in reverse chronological order, with the current position listed first, followed by the position at time t-1, then time t-2, etc. Outputs were four posterior probabilities corresponding to the probabilities of the four movements: up, right, down, or left. The highest probability was taken as the predicted move.

A model trained on adversary moves in response to a single computer agent was used for both the computer-agent and human-agent conditions. 350,000 simulations were created, and 40,000 simulations at each trajectory length (0 – 10) were randomly chosen, always based on the adversary's original starting position. The train-test split was 75/25. The CNN observer model used an Adam optimizer and a cross entropy loss function, and ran for 500 epochs. Model predictions were reasonably accurate, with accuracy on unseen test data ranging from 83 percent for the model with no previous states to 93 percent when 10 previous states were included. Results are shown in Figure 2.

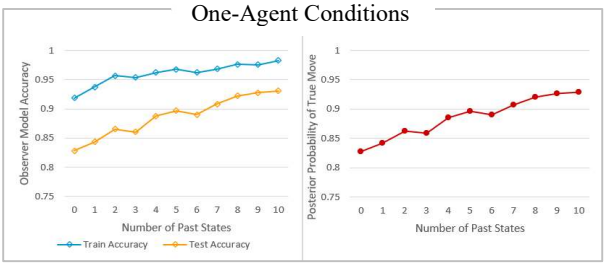

Figure 2. Train and test accuracy (left) and posterior probabilities of true moves (right) for observer models trained to predict adversary moves in one-agent simulations.

For the human-machine team conditions, 1,100,000 simulations were created based on two computer agents, and 40,000 simulations at each trajectory length were randomly chosen. Two-agent simulations resulted in shorter trajectories and so a greater number of simulations were needed to get adequate numbers of the longer trajectories. Results are shown in Figure 3.

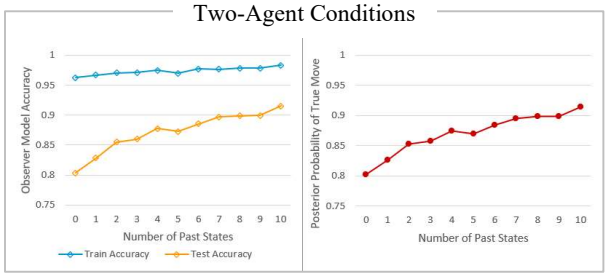

Figure 3. Train and test accuracy (left) and posterior probabilities of true moves (right) for observer models trained to predict adversary moves in two-agent simulations.

For all experimental conditions, predictions of adversary moves were based on the number of previous steps. At the first step, the model prediction was based on the model in which n_past (number of past states included) = 0. At the second step, prediction was based on the n_past = 1 model, etc. After the eleventh step, all predictions were taken from the model based on the most-recent ten past trajectories, n_past = 10.

2.5 Condition 1 – Agent vs. Adversary

In this condition, a computer agent attempted to capture one of the colored objects while the adversary attempted to capture the agent. That is, the agent sought a stationary target while the adversary sought a moving target – the agent. In the baseline case, the agent received full information about the gridworld, including the locations of barriers and colored objects, but no information about the adversary. In the test case, the agent was provided information about the adversary.

Attempting to navigate into a location currently occupied by the adversary incurred a 1.0 penalty, and the reward function was adjusted by adding model posterior probabilities to the typical move penalties for the four possible landing locations of the adversary. Thus, the penalty for moving to the location directly above the adversary was the usual 0.01 penalty plus the model posterior probability of an up move, the penalty for moving to the square on the right of the adversary was 0.01 plus the posterior probability of a right move, etc. We ran 16 trials of 10,000 simulations both with and without model predictions of adversary moves. Results are shown in Figure 4, left bars. Providing agents with predictions of adversary moves resulted in a significantly greater number of instances in which the agent captured one of the target objects [73.5% vs. 89.4%, $t(30) = -310.4$, $p < .001$].

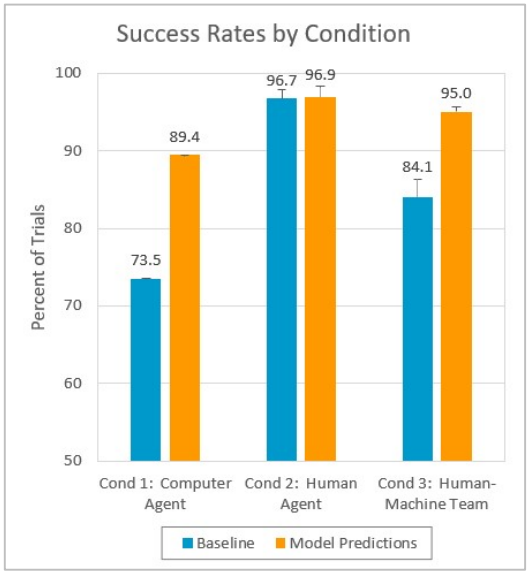

Figure 4. Success rates for computer-agent, human-agent, and human-machine team conditions.

2.6 Condition 2 – Human vs. Adversary

In this condition, the computer agent was replaced with a human agent. The controller program was modified to prompt the human agent to enter a move at each step. In the baseline task, the human agent was given a visual representation of the gridworld, along with a statement about which target object had the greatest reward value. In the experimental task, the human agent was also given the adversary's predicted move. A group of four pilot subjects completed 100 trials of both the baseline and experimental tasks. Results are shown in the middle bars of Figure 4. Model predictions did not help human agents [96.7% vs. 96.9%, $t(3) = 0.24$, n.s.]. It is worth noting, however, that human-agent success rates were near perfect, even without predictions. The lack of an effect here likely reflects a ceiling effect, with human agents capable of predicting an adversary's movements without the predictive model.

2.7 Condition 3 – Human-Machine Team vs. Adversary

This condition included both the computer agent and a human agent. The two agents were not allowed to occupy the same grid location, and in cases where both attempted to move onto the same square, the computer agent was given priority, and the human agent was prompted for a different move. The same four pilot subjects completed 100 trials of baseline and experimental tasks, and the results are shown in the rightmost bars of Figure 4. The human-machine teams achieved more successful trials when given model predictions [84.1% vs. 95.0%, $t(3) = 5.76$, $p < .01$].

2.8 Experiment 1 Conclusions

Results show that model predictions of an adversary's movements improved the performance of computer agents and human-machine teams. Conditions 1 and 3 included computer agents whose performance significantly improved when provided predictions of adversary moves, something we refer to as machine Theory of Mind (ToM). Human agents did not benefit from model predictions, as they were successful even in baseline conditions. These results demonstrate that implementing a functional ToM model into the action-policy of an autonomous agent improves the agent's performance, including in a human-machine teaming setting. Since humans alone demonstrated ceiling-level performance in this task even without the ToM model predictions, this demonstrates that furnishing a computer agent with predictive analytics of an adversary leads to better teaming performance by helping the agent avoid capture by the adversary. This underscores the importance of establishing common models across teammates for better teaming performance.

3. EXPERIMENT 2

The results of Experiment 1 showed that model predictions of adversary moves facilitate agent navigation through gridworlds. However, both the agent and the adversary are limited in that they only respond to the immediate situation. Agents' policies plot the shortest path to the reward object and take no consideration of the adversary unless it blocks

their next step. Similarly, adversaries plot a radiodrome pursuit curve [4], always based on the agent and adversary current positions with adjustments at each timestep. Neither the agent nor the adversary has a mechanism to plot where the other will be within a number of timesteps. To begin to address this, we introduce agents and adversaries that predict each other's trajectories based on k-level reasoning [19].

3.1 k-Level Reasoning

The standard treatment of games makes use of Nash equilibria and unexploitable policies. A Nash equilibrium is a stable state in a multiplayer game where no player can do better by choosing a different action [20]. While every game has at least one (probabilistic) Nash equilibrium there are shortcomings in the application of Nash equilibria to some games. While equilibrium play is non-exploitable, it is not necessarily optimal given a model of your opponent's cognition. For example, in rock-paper-scissors, the equilibrium play is to choose rock, paper, or scissors with probability 1/3. However, if you have information about your opponent's behavior, such as them favoring rock, or that they will most likely play the move that would beat your previous throw, optimal play will look very different from the unexploitable Nash equilibrium because your opponent is employing an exploitable strategy.

The *k*-level framework is an approach to non-equilibrium play that generates a catalogue of strategies for each player that are iteratively generated (i.e. a 1-level strategy is generated from a 0-level strategy, a 2-level strategy is generated from a 1-level strategy and so on) [21, 22, 23]. This more closely mirrors a human-centric method for developing strategies, since calculating Nash equilibria is difficult for most people. A good example of this is the "Guess 2/3 of the Average" game where participants are asked to choose a number between 0 and 100 (inclusive) where the goal is for your guess to be 2/3 of the average of all player's guesses. While there is a Nash equilibrium where all players choose 0 which we would expect to see if you played this game in a department of mathematics, if you play this game with non-mathematicians, playing the equilibrium strategy may be disastrous. See also the Keynesian beauty contest [24, 25].

A game $G = (S, \{A_s\}_{s\in S}, R))$ is a (finite) collection of states S paired with actions $a \in A_{s,p}$ that player p can take given the state $s \in S$ and a reward function R: $S \to \mathbb{R}^2$ that determines the rewards for each player. Critically, we assume that players' actions are taken at the same time. This means that there is necessarily incomplete information, and we need to be able to predict our opponent's moves in order to devise a good strategy.

Given a game, we say that a strategy σ for player p is a mapping from the set of states to a nonempty subset of the action set $A_{s,p}$. During play, a player in state s following a strategy σ will take one of the actions in $\sigma(s)$ (chosen with uniform probability distribution). While it would be more elegant to write σ: $S \to A_{s,p}$, and be able to use a nice single-valued function, this fails to take into account those situations with two or more equally good (or equally bad) options. Thus we need to use the messier domain of $\mathcal{P}(A_{s,p}) \setminus \emptyset$.

Because we are using an inductive approach, we need a base case from which to build our strategies. For the $k = 0$ level, we assume that the agent takes a random walk. In other words, every move is taken based on a uniform probability distribution, or, for every player, $\sigma_{p,0}(s) = A_{s,p}$. Inductively, the k-level strategy for player p, $\sigma_{p,k}$ is optimal given that the other player, p', follows strategy $\sigma_{p',k-1}$. The method through which the strategy $\sigma_{p,k}$ is generated is arbitrary, however in this work, we use a variant of Q learning [23].

3.2 Environment, Setup, and Methods

We take the standard environment of a gridworld with some tiles replaced by obstacles, similar to the method in Experiment 1 above. As with Experiment 1, we have an agent whose goal is to reach a static reward and an adversary whose goal is to intercept the agent. If the agent lands on the reward or if the adversary lands on the same square as the agent, the game stops and one point is added and deducted from the winner and loser, respectively. In this experiment, we examine a 6 × 6 gridworld with 12 tiles replaced by obstacles. During generation, we discard any world where the navigable areas are not

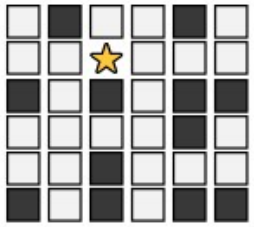

Figure 5. Sample environment

connected which prevents stalemates and yields more meaningful results. Lastly, we place the reward at a random navigable tile. A sample environment is provided in Figure 5.

To develop agents' strategies, we use a modified version of *Q* learning based on the method of relaxation. We use the closed form of the *Q* learning functional equation

$$Q(s,a) = R(s) + \gamma\, \mathbb{E}_{s'}\left(\max_{a' \in A_{s',p}} Q(s',a')\right)$$

for player p, where γ between 0 and 1 is the future-reward discount factor, and the expected value $\mathbb{E}_{s'}$ is taken over the possible states *s*' given action *a* and the opponent's presumed strategy $\sigma_{p',k-1}$. But instead of using it in a reinforcement learning context, we treat this as a differential/functional equation. Because we have complete knowledge of the location and value of the reward, we solve for the *Q* function directly instead of using reinforcement learning and running a trial agent through training runs.

To test the effectiveness of the *k*-level strategies, we train agent and adversary up to a *k*-level of 4 and run every permutation of *k* level through 100 trials in each of 100 maps. The average of the trial results on each map are then averaged over all 100 maps. In each trial, both agents are given a random starting location. The result is a matrix, indexed by the level of agents, with entries equal to the average performance over the course of the 100 trials. However, because this is a 0-sum game, we instead represent these data as an array of values between -1 and 1 (inclusive). Without loss of generality, we choose the scores of the agent (who is trying to reach the static reward but is chased) to be displayed. The scores of the adversary are simply the displayed scores' negative.

3.3 Results

As expected, there was a significant dependence on the k-level of both the agent and adversary on their performance. The average scores for all environments are provided in Table 1. Because the *k* level agent was trained against the *k*-1 level of the adversary, the highest value of every *column* appears directly below the diagonal, highlighted in yellow. These represent conditions in which the agent's k-level was one higher than the adversary's k-level. Similarly, the lowest value in every *row* is to the right of the diagonal, highlighted in blue, corresponding to conditions in which the adversary's k-level was one higher than the agent's.

Agent k-Level \ Adversary k-Level	0	1	2	3	4
0	-0.19	-0.72	-0.40	-0.53	-0.56
1	0.84	0.32	0.30	0.85	0.85
2	0.66	0.61	0.40	-0.07	0.24
3	0.66	0.37	0.59	0.39	0.09
4	0.65	0.02	0.21	0.95	0.37

Table 1. K-Level results. Higher scores indicate the agent prevails, while lower or negative scores indicate the adversary prevails.

Critically, there does not exist one strictly superior *k* value for either player, instead, the relative *k* value between the agent and adversary is what matters. This is because every strategy has a counterstrategy that may be employed by the opponent to thwart the strategy in question. Similarly, simply because one player employs a *k* level that is higher than their opponent does not mean that they will play optimally. It can be just as much a pitfall to overestimate your opponent as it is to underestimate them, which is why scores for which the *k* value difference between agent and adversary is more than one are less optimal than when that difference is one. I.e. the only definitive statement that can be made about the relative performances is when the difference in the *k* levels is one. Thus, the trick, once the *k* level strategies have been developed is to determine at which *k* level your opponent is likely to play and adapt your *k* level to counter them.

3.4 Experiment 2 Conclusions

Experiment 2 provides additional evidence that agents with knowledge of how adversaries will move perform better in a goal-seeking adversary-avoidance game. The results in Table 1 show that agents perform best when their *k* level is one level higher than the adversary's level. That difference in *k* levels suggests that agents are predicting adversary trajectories and planning their own trajectories to avoid adversaries. Like Experiment 1, the environment is abstract, but the agents and adversaries are certainly more sophisticated. This experiment also explored the condition in which both adversary and agent are furnished with predictions of one-another's future actions and outlines a framework for how either the evading agent or the pursuing adversary can get the upper-hand in such games.

4. EXPERIMENT 3

In a third experiment, we draw inspiration from Underwater Autonomous Vehicles (UAV) and simulate an environment that more resembles a realistic operational environment for a UAV. In this case, an agent (the UAV) attempted to traverse an environment, navigating around barriers while avoiding capture by adversaries. Whereas the previous experiments included discrete environments, the current environment is modelled as a continuous space. In addition to including multiple potentially adversarial additional agents, this environment also includes clutter objects. To differentiate clutter from adversaries while also providing the UAV agent with predictions of the paths of other agents in

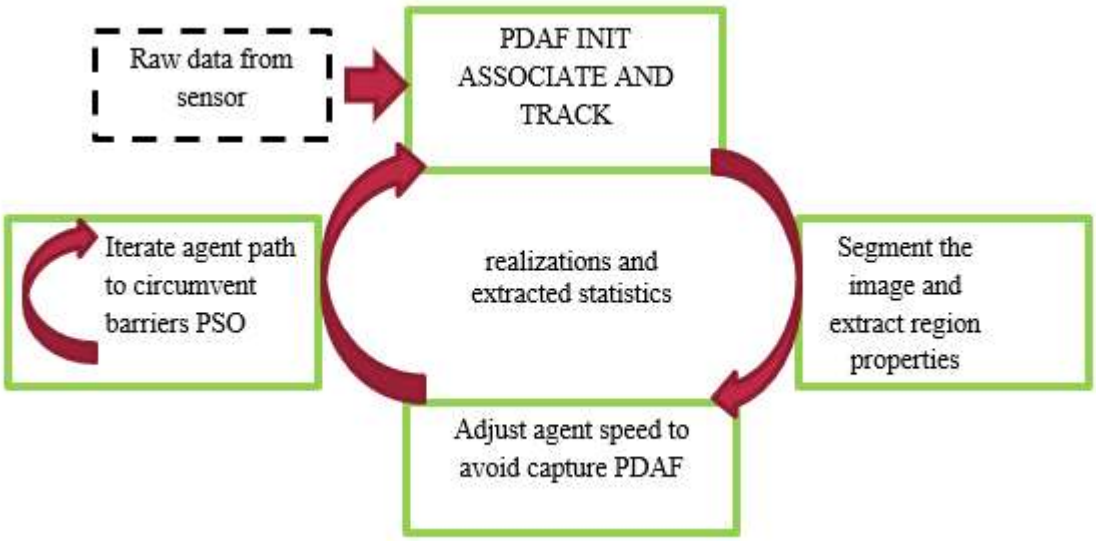

Figure 6. Process flow of the integrated PDAF/PSO system

the environment, we used a probabilistic data association filter (PDAF) algorithm to track the potential adversaries, and employed a particle swarm optimization (PSO) algorithm to compute an optimal traversal path. The process flow shown in Figure 6 illustrates for a single session the complex interaction of agents and adversaries with algorithms configured with a selection of model parameters.

4.1 Environment

The simulation environment is a 2-D physical space bounded in space and time, where the movement of the agent is optimized to achieve a goal of reaching a target location on the opposite side of the spatial boundary for where it starts. The environment is populated by an agent in conflict with adversaries and barriers that jeopardize their success and survival of the agent. The number of agents and adversaries as well as the density of clutter objects representing a noise floor were set as fixed parameters. State variables attached to each object in the simulation environment include position and velocity and are further described by properties such as in-motion, stationary, random, and their specific trajectories.

4.2 Observer Algorithms

In this experiment a Probabilistic Data Association Filter (PDAF [26]) was used to track the relative location and/or movement of the adversary and barriers, while a Particle Swarm Optimization (PSO [27]) algorithm was used to find an optimized path around barriers. The performance of the PSO algorithm depends entirely on the quality of the information provided by the PDAF algorithm. The PDAF algorithm maps all objects in the environment, including adversaries and barriers, and in cases of high clutter/noise, it may provide less than perfect information about the states of adversaries and barriers. In cases where the PSO algorithm receives degraded input from the PDAF algorithm, it may not successfully avoid barriers or adversaries.

4.3 Agent

The agent attempts to navigate through the environment, around barriers, while avoiding capture by adversaries. It receives information about the environment from the PDAF algorithm, which maps and tracks other objects in the environment. If it tracks an adversary on an intercept course, it can take evasive action by adjusting its velocity. The agent initially attempts a linear path between entry and exit points. However, if there are barriers blocking that path, the PSO algorithm can provide nonlinear trajectories to avoid them. Thus, the agent can adjust its velocity via the PDAF algorithm and its trajectory via the PSO algorithm.

4.4 Adversaries

Adversaries are generally intent on capturing agents. In this simplified experiment, we assigned one adversary on an intercept course with the agent, though multi-agent, multi-adversary options are available under this framework. For example, another adversary could be added to patrol the environment along a non-linear trajectory without knowledge of the agent's position. However, in this example, the adversary intersects the agent path along a linear trajectory which was initialized to guarantee intersection by matching its speed along a non-parallel path.

4.5 Game Session

In a single session, the agent's goal is to navigate across the environment around any barriers and exit to a region where it can no longer be pursued by an adversary. As the game begins, the agent attempts to move along a linear trajectory from the starting point to the exit. In the simplest terms, the agent marches across the environment in discrete time steps, applying strategies derived from the observer, essentially the integration of the PDAF and PSO algorithms. Given the dynamic nature of components in the environment, the agent needs to continuously obtain optimized path and trajectory solutions that circumvent barriers and evade capture by adversaries. An example beginning state of a simulation is shown in 7a. In this initial truth state, the adversaries and barriers are labeled, although no tracks have been launched yet. The agent has no knowledge of the adversary or barrier states until they are associated and tracked. In 7b, association and tracking has begun, as indicated by the assignment of a track number and the appearance of an ellipse surrounding the tracked components. At this time, the agent begins to acquire knowledge of state variables. Figure 7b shows that the agent has been assigned a track number of 49 and the adversary is tagged as track 23. The game ends in success or failure when the agent either evades capture and reaches its exit goal, intersects the path of an adversary, or encounters a

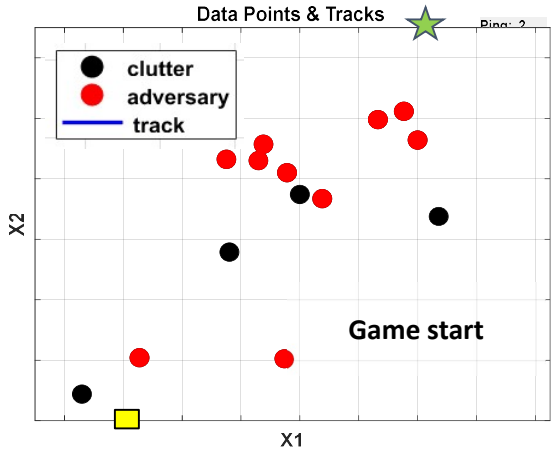

Figure 7a: The initial state of the environment as the game begins. The yellow square at the bottom represents the agent's starting location, while the green star at the top represents the goal state.

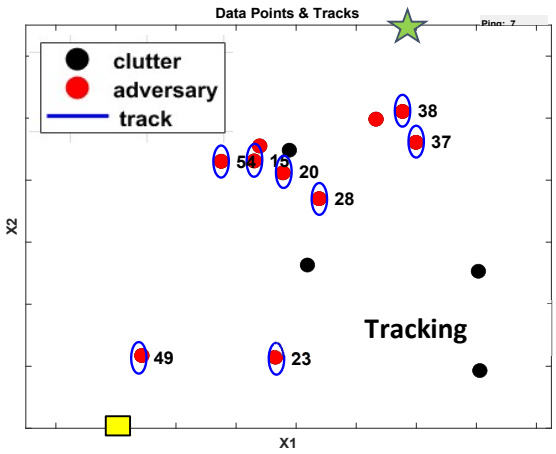

Figure 7b: Tracking of agent, adversaries, and barriers has commenced. Tracked objects are numbered arbitrarily and surrounded by blue ellipses.

barrier by crossing inside the perimeter of that barrier. Agent success also occurs when a time limit based on component velocity and the physical extent of the simulation environment is exceeded.

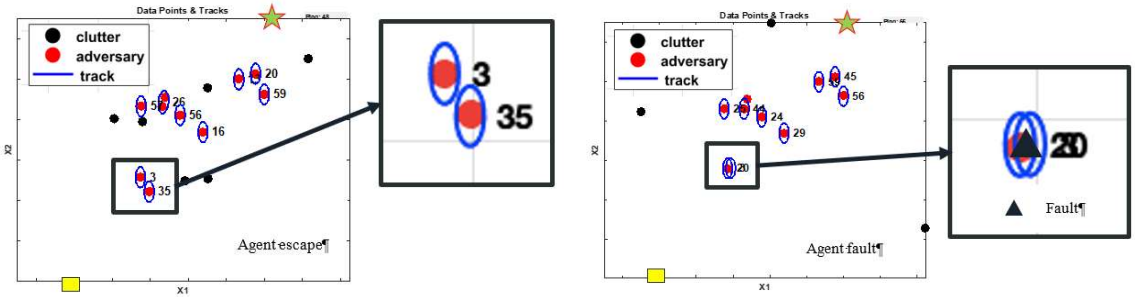

Figure 8. Simulations in which an agent evades capture (left), and when an agent is intercepted (right).

4.6 Efficacy of PDAF and PSO Algorithms

At every point in time the components in the environment are tracked by the PDAF algorithm by state vectors of position, heading, and speed. Objects that persist and move toward the agent are considered threats, in which case the PDAF algorithm changes the agent's velocity to evade capture. The PSO algorithm then takes the mapping from the PDAF algorithm and plots a trajectory around barriers to the goal state. Because evasion is part of the first algorithm, there is no way to remove predictions of adversary moves without also removing the ability to navigate the environment. However, to approximate a baseline condition, we can limit how the agent is allowed to respond to those predictions. For this pseudo-baseline condition, the change in velocity was drawn randomly from a uniform distribution centered around the normal velocity, 6.0 m/s ± 15 percent (5.1 m/s – 6.9 m/s). When the change is greater, the agent should still be able to evade capture, as demonstrated in Figure 8, left. But when the change is minimal, the adversary is more likely to intercept the agent resulting in a fault, as shown in the right side of Figure 8. In 100 simulations, the agent succeeded in reaching the goal state/exit point in 63 cases, while the adversary intercepted the agent in 37 cases. In the test case with both the PDAF and PSO algorithms functioning normally, the agent succeeded in 100 of 100 simulations. Results are shown in Figure 9.

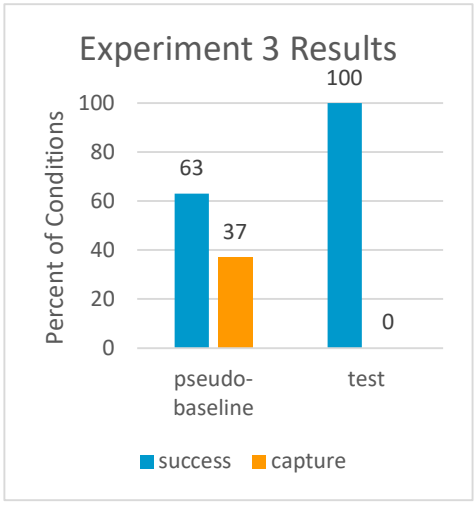

Figure 9. Successes and captures in Experiment 3

4.7 Experiment 3 Conclusions

The results of Experiment 3 again show the usefulness of predictions of adversary moves. Using a more realistic environment and more sophisticated algorithms, we see the same result – agents that can track adversaries and predict how and when they might intercept and capture said agents are better equipped to avoid capture and achieve their goal states.

5 EXPERIMENT 4

This final experiment presents an attempt to extend the goal-seeking, adversary-avoiding approach to another real-world scenario – world-wide marine traffic. From Automatic Identification Systems (AIS) transmissions between January 6, 2023 to January 14, 2023 (provided by SPIRE), we created a model to predict ship trajectories. While we have not yet incorporated an adversary into this scenario, we expect an agent like the U.S. Coast Guard would be able to use predicted trajectories to intercept (rather than avoid) suspicious adversary vessels.

5.1 Vessel Data

The AIS dataset included data from approximately 230,000 ships that transmit approximately 70,000,000 updates per day, with highly variable update intervals ranging from a few seconds to many days. This is not surprising considering that most ships are stationary, either at dock, in storage or moored, and thus their AIS transponders are either simply turned off or transmitting very infrequently. Much of the transmitted data is based upon the captain's manual entries, which generally are not reliable. Thus, we only register a minimal dataset which consists of the ship's Maritime Mobile Service Identity (MMSI), and waypoint information: timestamp, longitude, and latitude. Even this minimal dataset is not completely reliable because in certain cases ship captains will send spoofed waypoints to hide their true location. In some cases this involves illegal activities such as dumping, smuggling, pirating, illegal fishing etc. We have also observed that sometimes the transponder malfunctions or perhaps two ships accidentally share the same MMSI.

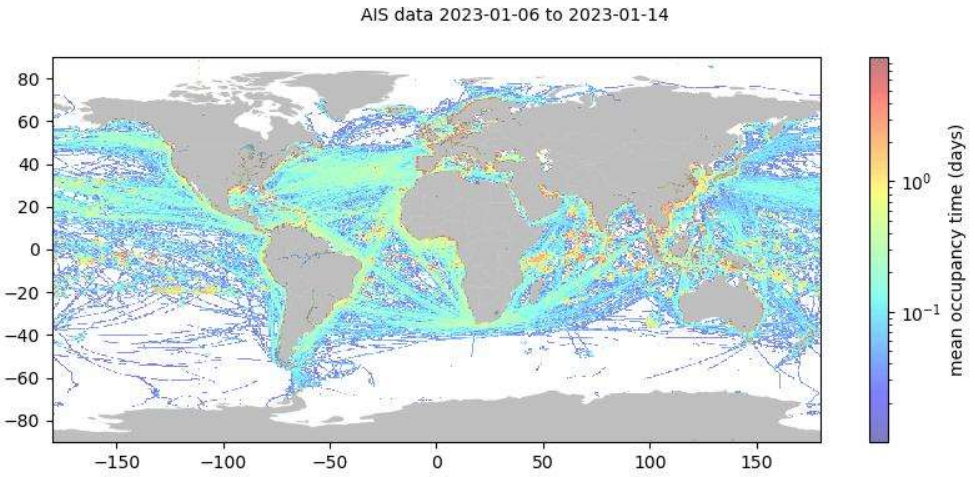

Figure 10. World-wide shipping traffic in terms of average occupancy time per 5 km × 5 km quadrant overlaid on world map

5.2 Trajectory Analysis

The first step in our analysis is parsing the AIS transmissions into grids, which we have chosen to be quadrants 5km × 5km in size. We have accounted for the earth's curvature by using the cosine of latitude to adjust quadrants' width. For each quadrant we register the ship's entry and exit waypoints. Overlaid onto the world map is a bivariate histogram for the overall mean amount of visitation time for each quadrant, shown in Figure 10. This reveals which areas have and have not been visited over that nine-day interval. There is little activity around both poles, and considerable activity along shipping lanes. The colormap is biased towards slowly moving or stationary ships, whereas ships moving quickly through quadrants only contribute little to the mean occupancy time.

We leverage the quadrant-based analysis to identify popular destinations, such as ports or fishing grounds. Rather than a quadrant's average occupancy time, here we consider average ship density and set the lower cut-off to 100 ships/25 km^2 - see Figure 11. This reveals most ports as well as other popular destinations including holding patterns or fishing grounds.

In the spirit of gridworld we study ship trajectories in terms of the sequence of quadrants they visit. We consider similar questions, such as future trajectories or future destinations in terms of the popular destinations shown in Figure 11. However, our analysis is entirely built upon the extensive historic nine-day dataset from SPIRE which consists of one billion waypoints for a quarter of a million ships.

For a given ship we predict its future quadrant-based trajectory from its immediate past and the historic record of all trajectories. In our analysis, we consider the "present" as the last day of the dataset, January 14, 2023, and denote that subset as *P* (about 4,500,000 waypoints). The subset of ships from previous days is the "historic" data (about 37,000,000

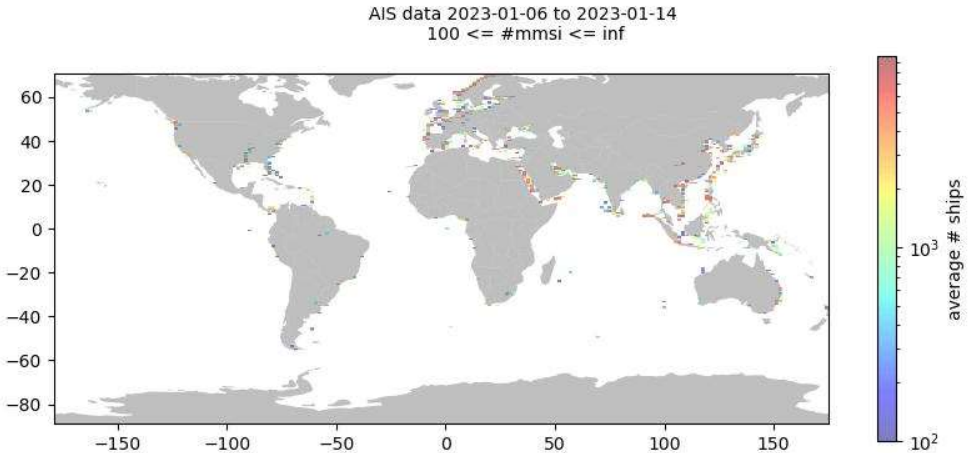

Figure 11. Ship density in terms of the average number of ships per quadarant during the nine-day interval. To reveal high-density regions, an arbitrary cutoff of 100 ships has been imposed.

waypoints). The depth of previously visited quadrants (number of previous steps) from the "present" varies from 1 (only matching a single quadrant), up to matching the last 30 quadrants. For the present subset, P, we define $\underline{S}_d$ as the subset of sister ships in the historic dataset whose trajectories match to depth d. We expect that as the matching requirement increases, this available subset decreases, which confirmed in Figure 12, left panel. Thus, as the trajectory matching depth increases, we observe a reduction in the number of sister ships available for predictions and an increase in prediction accuracy. The right panel of Figure 12 shows high levels of prediction accuracy, with increasing accuracy as a greater number of previous steps are included. With just a single previous step ($d = 1$), prediction accuracy is at 73.3 percent. As d rises, as more previous steps are included, prediction accuracy increases to more than 90 percent accuracy when $d = 30$. As the size of the historic dataset increases (i.e. more than eight days), we expect the number of available sister trajectories will increase without much change in the correct predictions.

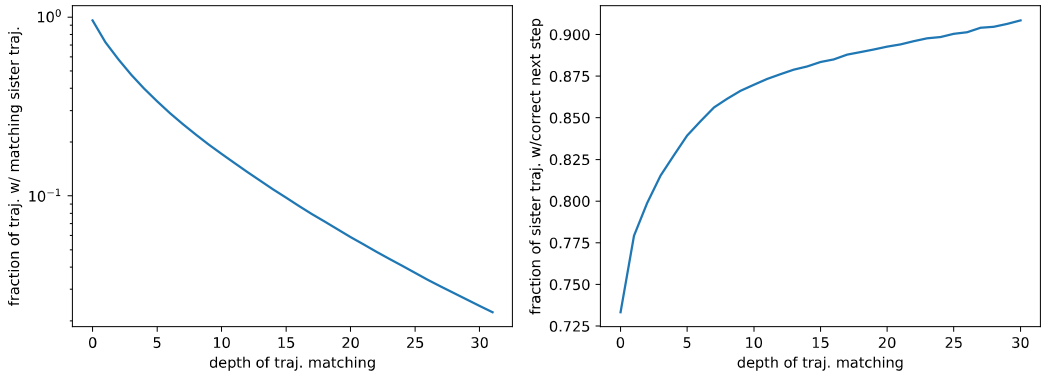

Figure 12. Trajectory matching statistics. left: Dependence of the present trajectories, P, that have matching sister ships in the subset S_d, which decreases with matching depth. Note that the number of ships in P is . right: The fraction of sister trajectories that correctly predict the next step (and possible more steps beyond that) with the matching depth

5.3. Experiment 4 Conclusions

Preliminary results for this experiment reveal that ship trajectories can be predicted from the trajectories of other ships that traverse the same space. Having established a framework for ingesting and analyzing certain aspects of the AIS data, we are well-positioned for additional analysis to improve predictions of the trajectories as well as ports of destination. For example, employing ensemble approaches on the sister trajectories can significantly improve predictions. The ensemble approaches could also use similarity features between the ship of interest and its sister ships to find better

predictors and discard worse ones. In summary, we have demonstrated the utility of the grid-world approach for a large real-world data set.

6 OVERALL DISCUSSION

Prediction is fundamental to many strategies for pursuit or avoidance behavior. Whether as a pursuer or the pursued, knowing the likely future actions and future states of an adversary can facilitate more optimal planning and action selection. Here we used agent-based modeling and a pursuit-avoidance game – implemented in a gridworld – to investigate several different predictive algorithms – a convolutional neural network (CNN)-based predictive algorithm inspired by Theory of Mind modeling [15], a recursive reasoning algorithm known as k-level reasoning [23], and a probabilistic data association filter (PDAF) algorithm [26] in conjunction with particle swarm optimization [27]. As a proof of concept, we also implemented a custom maritime vessel prediction algorithm without an accompanying pursuit-avoidance game to demonstrate a potential real-world use case. In all cases, furnishing predictions from the algorithms enhanced the performance of an autonomous agent being pursued by an adversary. When predictions were furnished to both the agent and the adversary, as in the k-level reasoning experiment, an agent having a k-level one higher than their adversary imparted a benefit, whether the agent was pursuing or being pursued, which is also consistent with predictions providing a behavioral benefit.

Finding optimal pursuit or avoidance behavioral strategies can be especially challenging in settings with multiple agents (including multiple pursuers, multiple agents being pursued, or both). It can be even more difficult to define optimal or near-optimal behavioral strategies for situations in which humans and autonomous agents are operating in the same environment, including as a human-machine team. In such cases, traditional team-based dynamics among human members, such as sharing common goals and robust communication [28], are not necessarily assured because of the inclusion of autonomous agents, nor can the team coordinate purely algorithmically as a team composed only of autonomous agents would be able to do [29]. We hypothesized that one simple strategy for improving the performance of a human-machine team in an avoidance game would be simply to share the predictions of an adversary's movements with both the autonomous agent and the human teammate. We found that furnishing predictions using our CNN-based ToM predictive model improved performance of a human-machine team.

Future work in this area is needed to further exploit these findings. For instance, a systematic comparison of the benefit afforded by different algorithmic predictions could reveal which algorithm is best at enhancing performance in pursuit or avoidance games, and what features of the predictive algorithm are most important to this enhancement, such as greater sensitivity, the specificity, or the number of steps into the future for which predictions are made. While in principle the algorithms tested here could be used by either the pursuer or the agent being pursued, we focused on testing the impact of furnishing predictions to the agent being pursued (apart from the k-level reasoning algorithm, in which both pursuer and the agent being pursued had estimates of one-another). Additional work can test the impact of furnishing predictions of the type tested in Experiments 1 and 3 to the pursuer and to testing the impact of both the pursuer and the evader having predictions of one another. Additional work is also needed to determine whether and to what degree predictive analytics assist teams with multiple humans and/or multiple agents, discover the best ways to communicate predictive analytics to humans, autonomous agents, and human-machine teams, and investigate what additional information helps improve team performance beyond predictions of the future actions of adversaries, such as whether team members having predictive analytics of one-another's future actions, states, and goals improves teaming performance.

REFERENCES

[1] Ho, Y., Bryson, A., Baron, S., 'Differential games and optimal pursuit-evasion strategies', IEEE Transactions on Automatic Control, 10(4), 385-389 (1965).
[2] Shneydor, N.A., Missile Guidance and Pursuit: Kinematics, Dynamics, and Control, Elsevier (1998).
[3] Horie, K., Conway, B.A., 'Optimal fighter pursuit-evasion maneuvers found via two-sided optimization', Journal of Guidance, Control, and Dynamics, 29(1), 105-112 (2006).
[4] Nahin, P.J., Chases and Escapes: The Mathematics of Pursuit and Evasion, Princeton Univ. Pr., Princeton, NJ (2007).

[5] Peterson, A.N., Soto, A.P., McHenry, M.J., 'Pursuit and evasion strategies in the predator-prey interactions of fishes', Integrative and Comparative Biology, 61(2), 668-680 (2021).
[6] Mischiati, M., Lin, H.-T., Herold, P., Imler, E., Olberg, R., Leonardo, A., 'Internal models direct dragonfly interception steering', Nature, 517(7534), 333-338 (2015).
[7] Premack, D, Woodruff, G., 'Does the chimpanzee have a theory of mind?', Behavioral and Brain Sciences, 1(4), 515-526, (1978).
[8] Watson, A.C., Nixon, C.L., Wilson, A., Capage, L., 'Social interaction skills and theory of mind in young children', Developmental Psychology, 35(2), 386-391, (1999).
[9] Milligan, K., Astington, J.W., Dack, L.A., 'Language and theory of mind: Meta-analysis of the relation between language ability and false-belief understanding', Child Development, 78(2), 622-646, (2007).
[10] Frith, C.D., Singer, T., 'The role of social cognition in decision making', Philosophical Transactions of the Royal Society of London B Biological Sciences, 363(1511), 3875-3886, (2008).
[11] Carlson, S.M., Moses, L.J., 'Individual differences in inhibitory control and children's theory of mind', Child Development, 72(4), 1032-1053, (2001).
[12] Wimmer, H., Perner, J., 'Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception', Cognition, 13(1), 103-128, (1983).
[13] Baker, C.L., Saxe, R., Tenenbaum, J.B., 'Action understanding as inverse planning', Cognition, 113(3), 329-349, (2009).
[14] Baker, C.L., Jara-Ettinger, J., Saxe, R., Tenenbaum, J.B., 'Rational quantitative attribution of beliefs, desires and percepts in human mentalizing', Nature Human Behaviour, 1(4), 0064, (2017).
[15] Rabinowitz, N.C., Perbet, F., Song, H.F., Zhang, C., Eslami, S.M.A., Botvinick, M., 'Machine theory of mind', International Conference on Machine Learning, PMLR, 4218-4227, (2018).
[16] Nguyen, T.N., Gonzalez, C., 'Theory of mind from observation in cognitive models and humans', Topics in Cognitive Science, 14(4), 665-686, (2022).
[17] Nguyen, T.N., Gonzalez, C., 'Cognitive machine theory of mind', Carnegie Mellon University, (2020).
[18] Gonzalez, C., Lerch, J.F., Lebiere, C., 'Instance-based learning in dynamic decision making', Cognitive Science, 27(4), 591-635, (2003).
[19] Stahl II, D.O., Wilson, P.W., 'Experimental evidence on players' models of other players', Journal of Economic Behavior and Organization, 25(3), 309-327, (1994).
[20] Nash, J., 'Non-cooperative games', Annals of Mathematics, 54(2), 286-295 (1951).
[21] Crawford, V., Iriberri, N., 'Level-k auctions: Can a non-equilibrium model of strategic thinking explain the winner's curse and overbidding in private-value auctions?', Econometrica, 75(6), 1721-1771 (2007).
[22] Stahl, D.O., Haruvy, E., 'Level-n bounded rationality and dominated strategies in normal-form games', Journal of Economic Behavior & Organization, 66(2), 226-232 (2008).
[23] Fotiadis, F., Vamvoudakis, K.G., 'Recursive reasoning with reduced complexity and intermittency for nonequilibrium learning in stochastic games', IEEE Transactions on Neural Networks and Learning Systems (2022).
[24] Mauersberger, F., Nagel, R., 'Levels of reasoning in Keynesian Beauty Contests: A generative framework', Handbook of Computational Economics, 4, 541-634 (2018)
[25] Camerer, C.F. 'Behavioural game theory', Behavioural and Experimental Economics, 42-50 (2010).
[26] Bar-Shalom, Y., Daum, F., Huang, J., 'The probabilistic data association filter', IEEE Control Systems Magazine, 29(6), 82-100, (2009).
[27] Kennedy, J., Eberhardt, R., 'Particle swarm optimization', Proceedings of the ICNN'95-International Conference on Neural Networks, 4, IEEE, 1942-1948, (1995).
[28] Fiore, S., Salas, E., Cannon-Bowers, J., 'Group dynamics and shared mental model development', How People Evaluate Others in Organizations, 335-362, Psychology Press, (2013).
[29] Doriya, R., Mishra, S., Gupta, S., 'A brief survey and analysis of multi-robot communication and coordination', International Conference on Computing, Communication & Automation, IEEE, 1014-1021 (2015).