Active Behavior Recognition in Beyond Visual Range Air Combat

Ron Alford RONALD.ALFORD.CTR@NRL.NAVY.MIL
ASEE Postdoctoral Fellow;
Naval Research Laboratory (Code 5514); Washington, DC; USA

Hayley Borck HAYLEY.BORCK@KNEXUSRESEARCH.COM
Justin Karneeb JUSTIN.KARNEEB@KNEXUSRESEARCH.COM
Knexus Research Corporation; Springfield, VA; USA

David W. Aha DAVID.AHA@NRL.NAVY.MIL
Navy Center for Applied Research in Artificial Intelligence;
Naval Research Laboratory (Code 5514); Washington, DC; USA

Abstract

Accurately modeling uncontrolled agents (or recognizing their behavior and intentions) is critical to planning and acting in a multi-agent environment. However, behavior recognition systems are only as good as their observations. Here we argue that acting, even acting at random, can be a critical part of gathering those observations. Furthermore, we claim that acting intelligently via automated planning can significantly reduce the time it takes to confidently classify agent behaviors. We present a formalism and algorithm for integrated planning and recognition, as well as its implementation in a beyond visual range air combat simulator. We found that it yields better behavior recognition than non-integrated approaches. This provides evidence that behavior recognition is not just a necessary component of an intelligent agent, but that good behavior recognition requires intelligent acting.

1. Introduction

Behavior recognition is often referred to by (or confused with) other names, including activity, plan, and goal recognition, but for multi-agent systems, the basic premise is the same: given a series of observations, classify the actors against a set of prescriptive models. This is fine, and perhaps ideal, when either the environment provides plenty of quality observations, or when the system performing recognition has no agency in the environment. However, agency allows one to affect the actions of others in the environment, and this may be critical to determining the intent and strategy of others.

Thrillers in television and film, after all, often feature the incredibly obvious tail (TV Tropes, 2015), wherein a protagonist is followed by car or on foot. The pursuers are made obvious by their interaction with their target - they repeatedly make the same turns while maintaining a constant distance. If these scenes were transplanted to a highway in Kansas, there would be nothing to distinguish the pursuers from innocent bystanders traveling in the same direction. More mundane examples include job interviews, or simply directly asking a question when someone is pacing an area ("What are you looking for?"). Interaction is key to disambiguating agents when the environment fails to provide enough clues.

Acting at random may be enough to distinguish the inept tail, but this strategy may have foreseeable downsides when applied to interviewing job candidates. Fortunately, there are a wide range of automated planning techniques that can help an agent to navigate the known consequences of its actions (Ghallab, Nau, & Traverso, 2004).

We review related work that combines acting and behavior recognition in Section 2. In Section 3, formalize acting with behavior recognition as a special case of partially-observable Markov decision processes (POMDPs), where the hidden variable is the specific model that controls the environmental agents. We give an approximation of this POMDP as a fully-observable Markov decision process (MDP). We adapt a case-based reasoning (CBR) behavior recognition system to our formalism, and apply it to an unmanned aircraft (UAV) in an air combat simulation in order to classify the strategies (and predict the actions) of opposing aircraft (Section 4). In our experiments (Section 5), we show that the UAV acting at random is a surprisingly good at classifying the behavior of the opponents. However, when the behavior recognition system is paired with a Monte-Carlo based MDP planner, the UAV is able to confidently classify the opponents in significantly less time. Finally, we discuss future work in Section 6 and then conclude.

2. Related Work

The focus of our work in this paper is on the relation between acting and behavior recognition, and how explicitly planning our interactions with other agents can reveal information about them. Behavior recognition and acting have been separately studied in extensive detail. In games, where behavior recognition is often referred to as *opponent modeling*, we find an application where recognition meets acting. More specifically, we mean in (1) two-player repeated games, where opponent modeling has long held an important role, and (2) real-time and imperfect-information games, where actions have long-ranging consequences, making planning more important.

Poker is a perennial favorite in the opponent modeling literature (Billings et al., 1998; Southey et al., 2005). However, opponent modeling plays a vital role even in simple games that have known “optimal” strategies (e.g., rock-paper-scissors) (Tesauro, 2003). This suggests behavior recognition is important even as we approach near-optimal play against rational opponents, as in smaller games of poker (Bowling et al., 2015). Agents for games like poker and rock-paper are evaluated by pairing them against another agent, and repeatedly playing the game against that agent for an arbitrary number of rounds (unknown to the player). Other than the shared history of interactions, each iteration of these games is independent. When this causes the value of exploration to be small, agents tend to play the known optimal strategy without deliberately exploring (Carr et al., 2009). While these works demonstrate the importance of behavior recognition, they describe techniques that perform behavior recognition over multiple independent interactions. In this work, we are concerned with behavior recognition during the course of a single, (roughly) continuous interaction, where each state (and our available actions) depends on the previous choice of actions.

Opponent modeling is also used in longer-form games, especially those that violate the standard minimax assumptions. Implicitly, a form of opponent modeling is used in many Monte-Carlo tree-search implementations, where the random roll-outs are biased towards moves seen in previous games (Ciancarini & Favini, 2010; Browne et al., 2012). Opponent modeling has been used explicitly in real-time strategy games (Schadd, Bakkes, & Spronck, 2007) and in a football simulator (Laviers et al., 2009). In each of these, though, the effect of an agent's action on its knowledge of the opponent strategy is not considered.

In two-player games, PrOM (Donkers, Uiterwijk, & van den Herik, 2001) uses a probability distribution over the opponent's choice of heuristic functions to perform a mini-max-like search. However, PrOM is only given the initial probability distribution as input, a fixed value throughout the search. Given that some choices are more likely for one heuristic than another, PrOM will miss opportunities to exploit disambiguating actions.

The closest work to our is that of Hidden Goal MDPs (HGMDPs), which models an assistant alternates acting with an agent, where the assistant must act in a way to maximize the agent's undisclosed utility function (Fern et al., 2014). In the next section, we will describe a formalism, MARA, which could be viewed as a specialization of HGMDPs in which the assistant's utility function is independent of the other agent. Where as HGMDP approach assumes an explicit representation of the state and agent action probabilities, we will use a black-box simulator. This leads us to use a sampling-based UCT algorithm, as opposed to the more traditional MDP solvers used for HGMDPs.

3. A Formalism for Active Behavior Recognition

When planning in a multi-agent environment, agents may make decisions not only on the current world state, but also on their previous interactions with other agents. Here, we will assume that the planner has access to a set of candidate models, each of which predicts the behavior and reactions of all other agents in the environment. Although unintuitive, using a single model to predict the behavior of all agents allows the planner to abstract away any communication and coordination between agents. Under this assumption, then given a set of m independent models for individual agents and k agents in the environment, we can construct m^k candidate models for the planner to use.

Finding the model that matches best against the observed behaviors is the task of *behavior recognition*. We assume the planner also has access to a behavior recognizer that takes as input the current history of observations and returns a probability distribution over the candidate models, representing the likelihood that a given model produced that history.

We define a multi-agent recognition and acting (MARA) problem to be the tuple

$$\mathcal{X} = \left(M, H, A, P_{(\cdot,\cdot)}\left(\cdot,\cdot\right), C_{\cdot}\left(\cdot\right), R, h_0\right)$$

where:

- M is a finite set of agent models, where any $m \in M$ represents a policy for controlling *all* non-self agents in the environment.
- H is the (presumably infinite) set of all possible histories.

- A is a finite set of actions.
- $P_{(m,a)}(h_1, h_2)$ is the probability that we transition from history h_1 to h_2 given an agent model $m \in M$ and an action $a \in A$. This allows for non-deterministic agent models.
- $C_m(h)$ is the agent modeler (or classifier), giving the probability that model m could produce history h.
- $R : H \times A \times H \to \mathcal{R}$ is the reward function over histories and actions.
- h_0 is the initial history

A MARA policy is a partial mapping of histories to actions. The goal of a MARA planner in our formalism will be to take as input a MARA problem $\mathcal{X}$ and a finite look-ahead N, and produce a policy that maximizes the expected sum of rewards from R.

A MARA problem is essentially a partially-observable Markov decision process (POMDP) with one partially-observable variable (the agent model) that never changes. However, planning with POMDPs is computationally expensive. Therefore, we approximate the MARA problem with a fully-observable MDP, where the agent model can change after every action, but only according to the probability given by the model for the current history. This forms a probability function $P_a^C(h_1, h_2) = \sum_{m \in M} P_{(m,a)}(h_1, h_2) \cdot C_m(h_1)$ for any action and pair of histories.

4. Active Behavior Recognition in Air Combat Simulators

Our motivation for creating an active behavior recognition system stems from a larger problem that we are addressing: that of creating an effective agent that can function in an imperfect information multi-agent air combat scenario. To properly respond to hostile agents, it is prudent to first understand how they will react. Prior work has used passive behavior recognition to solve this problem. However, its effectiveness was tied to the actions taken by the observed aircraft. Specific actions are effective in reducing the ambiguity among possible hostile behaviors. Thus, by merging the behavior recognizer with the UAV's planner, it can actively choose actions to reduce this ambiguity.

4.1 Tactical Battle Manager

The Tactical Battle Manager (TBM) is a high level architecture designed to control an unmanned aircraft (UAV) flying as a wingman for a human-piloted allied aircraft. The TBM is designed to function with imperfect information about the state of the world and the behaviors of any hostiles it may encounter during a scenario. The TBM will use Goal Reasoning, Planning, Behavior Recognition, Prediction and other components to complete this task. To date, work on the TBM has focused primarily on its Behavior Recognizer, Plan Execution Predictor, and Planner components, which (as we will describe) collaborate to perform Active Behavior Recognition through planning.

Figure 1 displays an overview of the TBM. It receives raw imperfect state information from an air combat simulator through an abstract interface. This data is then consumed by various "Knowledge Updaters" that attempt to parse relevant state information and fill in possible gaps in the system's knowledge. These include:

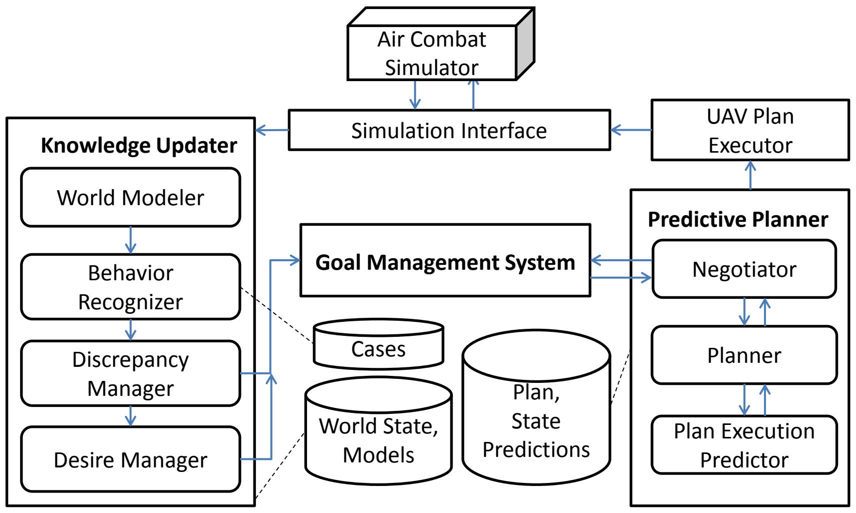

Figure 1. High level architecture of the TBM.

- World Modeler: Maintains a high level overview of the world state
- Behavior Recognizer: Determines behaviors of hostiles based on observations
- Discrepancy Manager: Notifies the system of contradictions between internal models and observations
- Desire Manager: Continuously re-evaluates to what degree the desires of the UAV are being met.

Once the raw state data has been consumed, the Goal Management System uses goal reasoning techniques to select a current goal for the UAV. This information is handed off to the planner, which uses a prediction system to generate and score plans created for a given goal. For the purposes of this paper, the goal reasoning subsystem and its corresponding knowledge updater components (Desire Manager, Discrepancy Manager) have been disabled. The knowledge acquisition goal is passed directly to the planner in order to showcase and test active behavior recognition. The Behavior Recognizer and Planner components will be detailed below.

4.2 Simulation

The TBM was designed to receive information updates about the world state from an external entity through a data bridge. This data can arrive from any source that properly implements the message protocol used by the TBM. Currently, we are using two air combat simulators: the Next Generation Threat System (NGTS) (NAVAIR, 2013) and the Analytical Framework for Simulation Integration and Modeling (AFSIM) (Zeh et al., 2014).

4.2.1 NGTS

NGTS is a government-owned high fidelity simulator that models aircraft, ground platforms, weapons and agent coordination. Its models are customizable, allowing parametric definitions to be set per entity type. It is designed to be highly parallelized, efficiently taking advantage of multiple CPUs/cores. The actions of an entity are governed by a "Tacticianer" which by default is represented by a behavior graph. NGTS behavior graphs function similarly to finite state machines and allow for complex but static behaviors. NGTS simulations run at a multiple of real time, depending on processing power. At every time step, each entity is given a chance to update itself.

The TBM can be run in the NGTS simulation using plugins that allow for more direct aircraft control. The scenarios and aircraft used in these simulations were created by domain experts to be as realistic as possible using unclassified information. We describe these scenarios in Section 5. For this paper, we used NGTS as the primary simulation engine for testing and evaluating the active behavior recognition components.

4.2.2 AFSIM

AFSIM is a lower fidelity simulator than NGTS, and has fewer restrictions. It uses a custom scripting language to configure scenarios and scenario entities. These scripts grant detailed control over every aspect of the scenario.

AFSIM simulations can be run in modified real-time update cycles (like NGTS) or in an event-driven mode. In the latter, the various scripted components register as listeners for events (e.g., radar tracks, weapon impacts). This allows AFSIM to run lengthy simulations quickly so long as the number of computationally intensive events are kept minimal.

The TBM's Plan Execution Predictor (Jensen et al., 2014) uses AFSIM as a means of prediction. It can configure a scenario using information from its world model and predict a duration of time into the future. These predictions allow the TBM to score and rank possible actions against their outcomes. However, these predictions are only accurate if the correct behavior model is chosen for the other agents. Our modification of the PROST planner (Section 4.4) uses AFSIM to predict how any given combination of action and behavioral model will play out.

4.3 Case-Based Behavior Recognizer (CBBR)

Our Behavior Recognizer uses observations and CBR techniques to classify hostiles against previously observed behaviors. We define a *behavior* as a tendency or policy of an agent over time. Currently, CBBR populates its case base through observational training with perfect information.

We represent cases as ⟨problem,solution⟩ pairs. The *solution* is the ground truth behavior being run by the agent while the *problem* is a discretization of observed states and actions into features. There are two types of features, time step features (TSF) and global features (GF). Time step features capture information over a small duration of time. An example TSF is FACINGOPPOSINGTEAM, which is a floating point value that denotes to what degree the hostile is facing an opposing force. This can be telling especially when combined with other features such as ISINWEAPONRANGE to help determine possible behaviors. In contrast, global features capture information about a hostile over the course of an entire simulation. An example GF is HASINTERESTINOPPOSINGTEAM, which tracks whether the hostile appears to react to the UAV or its allies. These features were tested and were shown to be an effective means of behavior recognition (Borck et al., 2015).

Several changes have been made to CBBR for this paper. This includes a new case pruning algorithm, our method for case retrieval, and solution reporting. Previously, the pruner removed all ambiguous cases (i.e., those with identical solutions and whose problems have above-threshold similarity) from the case base except for a single representative. This led to prominent cases being treated equally with atypical cases. The new pruning algorithm still only stores one case in ambiguous situations, but tracks the number of ambiguously pruned cases for a given problem, and is thus able to represent the difference between prominent and atypical cases. Specifically, when adding a new case c to the case base L its similarity is computed against all other cases in L. If the most similar case $c_s \in L$ has a similarity above a threshold, the weight of c_s is incremented. Otherwise c is added to L with an initial weight of 1.

Previously, the CBBR case retrieval algorithm retrieved the most similar case to the currently observed situation in L and returned its solution. This led to issues where, if the highly similar cases were dominated by one behavior but the most similar case had a different behavior, only one of the two were reported. Now, CBBR returns a probability distribution over the known behaviors denoting the likelihood that the given hostile is the respective behavior. In this paper we use four behaviors: *All Out Aggression (AA), Safety Aggressive (SA), Passive (PA) and Oblivious (OB).* These behaviors range from always flying a pure pursuit and attacking whenever possible to completely ignoring the UAV entirely, a not unrealistic scenario given that 80–90% of combat aircraft are shot down unawares (Shaw, 1985). To estimate behavior probabilities, CBBR retrieves the set of cases K whose similarity to the observed case is above a threshold. For each behavior b its probability is estimated as the normalized percentage of cases in K whose solution is b (Equation 1). These changes permit more detailed analysis of the predicted behavior by other system components, as they can consider the relation of the highest-probable behavior with others.

$$p(b) = \frac{\sum_{C_b \in K}(C_b.weight)}{\sum_{C \in K}(C.weight)} \tag{1}$$

4.4 Active Planning using PROST, CBBR and Simulation

To perform active behavior recognition using CBBR in beyond visual range (BVR) air combat, we modified PROST (Keller & Eyerich, 2012), which is a domain-independent probabilistic planner based on UCT (Kocsis & Szepesvári, 2006). PROST uses a standard UCT tree comprised of decision and chance nodes. Decision nodes have exactly one successor node per action and are used to denote possible choices. Chance nodes can have many successor nodes, each of which are labeled with their probabilistic likelihood of occurring. Together they can be used to create a tree that contains information both on active decisions and their probabilistic outcomes. PROST is an anytime algorithm that returns a non-random decision whenever queried. Through the use of Monte-Carlo rollouts the tree is constructed from selected actions and their outcome probabilities where each rollout traverses the search tree from the root node to a leaf. At each node, PROST selects the next successor. For decision nodes it uses the UCT score of the node and for chance nodes it selects using a random distribution. They show that their probabilistic version of UCT is viable in domain-independent planning problems. PROST is particularly applicable for active behavior recognition because chance nodes allow for an intuitive way to branch the search tree based on recognition ambiguity.

We use a variant of PROST where decision nodes represent each potential action a that can be taken by the UAV and chance nodes represent possible future states by combining a and the model m used for all other agents. The resulting tree alternates between these decision and chance nodes at every depth of the tree. We define recognition confidence as the expected reward of a rollout, which is the largest probability returned by CBBR for the observed time period. Our PROST variant actively chooses actions that disambiguate CBBR's recognitions. At each chance node, an AFSIM simulation is run using the proposed action a and the agent behavior model m drawn from the probability distribution calculated by CBBR. The recognition confidences are then re-evaluated by running CBBR over the predictions generated by AFSIM. Rollouts are performed in parallel to increase the speed at which the tree is searched. Currently, there are four possible actions the UAV can take at any moment with respect to a hostile:

- Fly 0: Fly directly at given hostile
- Fly 45: Fly at a 45 degree offset to given hostile
- Fly 90: Fly at a 90 degree offset to a given hostile
- Fly 180: Fly at a 180 degree offset to a given hostile

These actions discretize basic BVR air combat maneuvers while keeping the search space relatively small. Figure 2 shows an example partial search tree at the start of a simulation. The tree starts with the currently executing action, Fly: 0, and a hostile with an unknown behavior. The figure shows the subsection of the tree where *SafetyAggression* was randomly chosen to represent the hostile and how the recognition confidence changed by choosing the Fly: 45 action.

In its current implementation, we run our PROST variant planner over 30 second time intervals with a maximum tree depth of 10. Each interval starts a new tree with a root action equal to the solution of the previous run. When time is up, the action with the highest average expected reward is returned. Since this planner is always run for the same duration of the executing action, plans are only a single action in length. In the future we will improve this process by returning full plans and using plan refinement techniques rather than performing a full replan at every step.

5. Experiments

In Section 5.1 we describe the experiments we ran to validate two hypotheses:

H1: Acting as an agent yields better recognition results than not acting.

H2: Acting intelligently through planning yields better recognition results in a shorter time period.

In Section 5.2 we then discuss the results of our experiments and how they support both hypotheses.

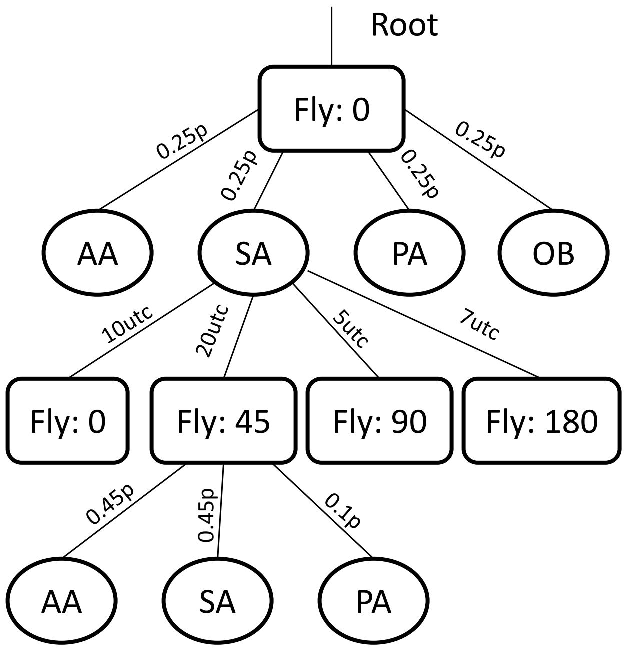

Figure 2. Example Monte-Carlo Tree with root action Fly: 0 against one hostile

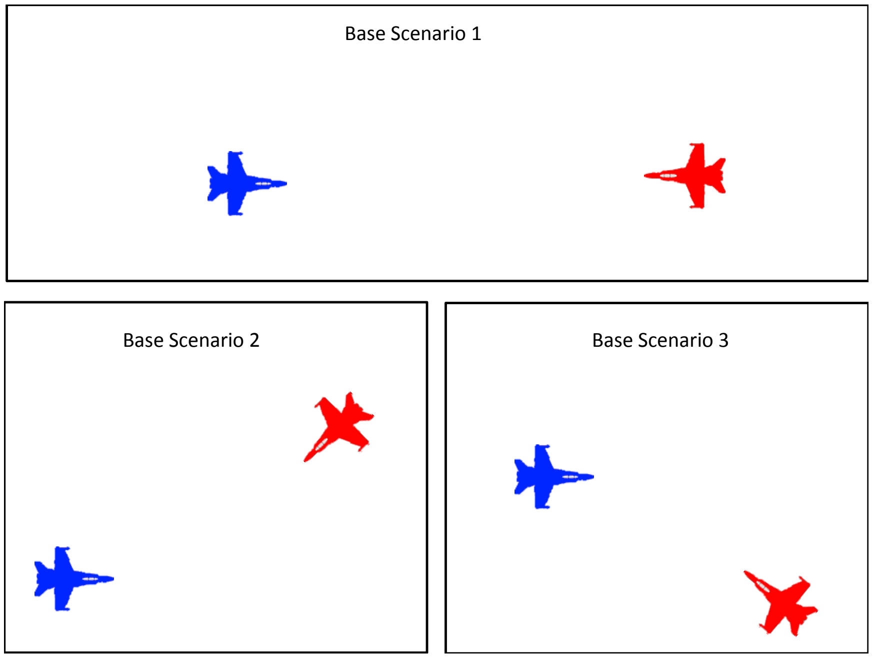

Figure 3. Scenario templates used for testing and training

5.1 Experimental Setup

We divided our experiment into training phases, to gather the case base for the CBBR component, and testing phases. Trials were run using three scenario templates, which we created with the help of input from subject matter experts. Figure 3 displays the base scenarios. The first consists of a UAV (blue) and hostile (red) agent flying directly at each other. The second and third scenario templates consist of a UAV flying straight initially and a hostile flanking from the top or bottom of the scene. In these templates the UAV and hostiles have the same radar and weapon ranges. We create 1,000 specific scenarios for each of the scenario templates by randomizing agent headings and positions. The headings and positions were randomly selected within bounds to generate "valid" scenarios (i.e., in which the agents are eventually within radar range of each other). For each random scenario we created a trial with a hostile agent using one of the four behaviors we currently model. This yielded 12,000 trials for training. We created an additional 840 random trials for testing.

We recorded the performance of the following four methods for controlling the behavior of the UAV using the baseline scenarios described above. The opponent's (hostile's) behavior was always randomly generated from among the four behaviors described in Section 4.3:

1. **Random Behavior**: UAV executes a random behavior (among the four described in Section 4.3)
2. **Safety Aggressive**: UAV executes a Safety Aggressive behavior
3. **Passive**: UAV executes a Passive behavior
4. **Active Planner**: UAV executes a behavior selected by the Active Planner (among the four described in Section 4.3)

The first three of these methods serve as baselines, while the fourth uses the active planner described in Section 4.4. We ran each of these methods to generate a training case base.

To assess our results, we measured the probability given by the CBBR system for the ground-truth behavior (i.e., the behavior controlling the hostile aircraft). A value of "1" would be absolute confidence in the correct behavior, whereas a value of "0.25" would be the expected value for guessing at random among the four behaviors. We report the results in two ways: the average probability given for the ground-truth behavior for all trials (i.e., offline), and the average probability over time (i.e., online). The behavior recognizer outputs a probability every 20 seconds, after the first 30 seconds of the scenario (allowing it to gather enough information to make a first recommendation), which yields 23 data points per 8 minute trial.

5.2 Results

The results support our hypothesis **H1** (i.e., acting as an agent yields more accurate recognition probabilities). We also found evidence for **H2** (i.e., intentionally acting yields even better results than acting randomly).

Figure 4 displays behavior probability over time averaged for all trials. During the beginning of the scenarios all but the Passive method performed similarly. Passive instructs the UAV to immediately fly at a 60°offset to the hostile when the hostile enters its radar range. This helps the passive

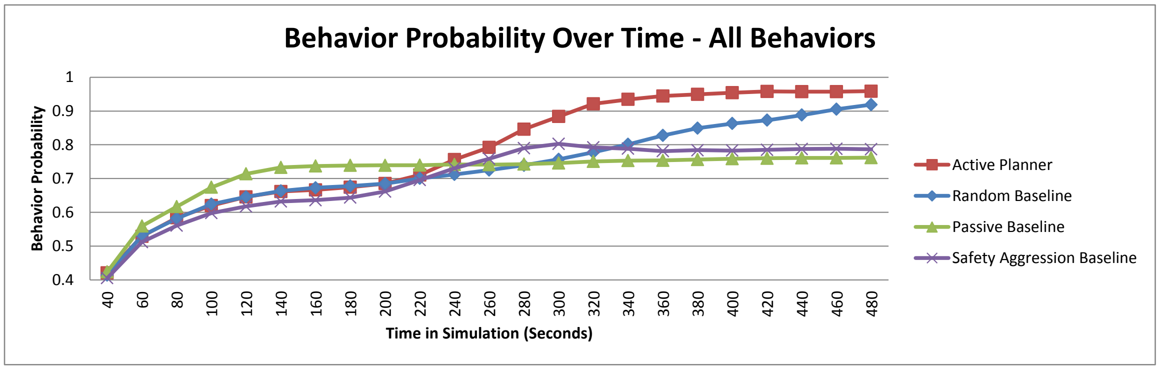

Figure 4. Behavior probability over time for all behaviors averaged over all trials.

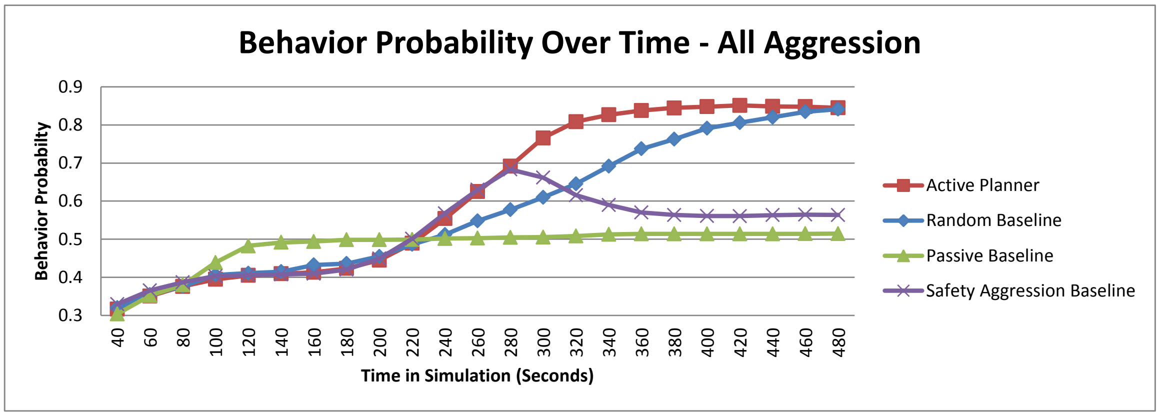

Figure 5. Behavior Probability over time for *All Aggression* behavior averaged over all trials

UAV to quickly increase its behavior probability, but the relative inaction in the later stages of the behavior yields stagnant probability scores. In contrast the random action baseline displays a steady increase in behavior probability. The Active Planner's recognition accuracy increases more quickly and achieves higher overall scores than any of the baselines.

The Active Planner's benefits are even more apparent when the All Out Aggression behavior is broken out, as in Figure 5. In this graph we again see the increase in behavior probability with the Passive baseline because it quickly executes a telling action. However, it again stagnates at a recognition probability score of 0.5. The Active Planner performs well, along with the Safety Aggressive baseline until the 280 second mark. In contrast to Safety Aggressive, the Active Planner attains a high behavior recognition probability of approximately 0.85 and does not decrease. This indicates the actions the Active Planner selects allow the CBBR component to be highly confident in its recognition attempt from the 320 second mark. Around the 220 second mark the Active Planner starts to take actions that are clearly better for recognition, in contrast to Random and Passive. A paired t-test ($p > .001$) confirms the Active Planner significantly outperforms Random for All Aggression from 240 until 440 seconds.

We conjecture that the small size of the action set is what causes the Random baseline to obtain the best results possible from the CBBR behavior recognizer. We also expect the speed and recognition performance of the Active Planner will increase by exploring more of the tree. The planner performed approximately 250 rollouts per re-plan which, while not all rollouts would necessarily be good to traverse, is still a very small portion of the tree. We will explore each of these conjectures in our future work.

6. Future Work

During experimentation several issues arose with our implementation of PROST that we will address in future work. When performing active behavior recognition through planning, recognition speed is important. We calculated expected reward by averaging the behavior accuracy at each stage of the rollout. This led to situations where the planner could spend considerable time in a subtree that contained a highly favorable action far down in a rollout. Alternative methods for calculating expected reward may reduce the required time to find solutions. We used a depth limit of 10 for our Active Planner; this coincides with experiment length. However, in more realistic scenarios tree depth may need to be dynamic. The original PROST implementation varied the search depth through reward locks, which may fix this limitation.

We also plan to expand to use larger scenarios involving multiple allies and hostiles. CBBR has already been tested in situations with multiple adversaries. However, the increase in tree size would need to be addressed. Research into the actions themselves and determining when they are useful could help provide insight into intelligent Q-value initialization and better guide the search algorithm.

7. Conclusions

In this paper, we argued by analogy and experimental evidence that acting and behavior recognition are inextricably linked. We formalized integrated acting and recognition as a special case of POMDP planning and provided an MDP relaxation of the problem. We implemented this relaxation in the context of a BVR air combat simulator using a case-based behavior recognizer and (active) Monte-Carlo planning techniques. The results from our empirical study revealed that an agent's ability to recognize the behavior of other agents depends on its choice of actions. Moreover, by actively planning its actions, an agent can increase its behavior recognition accuracy more than by acting randomly or by a non-intentional behavior.

Our results in this paper are limited to claims about behavior recognition in fully-observable environment. In practice, we expect agents to use behavior recognition in the pursuit of some larger set of goals, and to act in a partially-observable environment. Our work is an initial foray, showing that behavior recognition cannot be fully divorced from acting, and that intelligent agents may be able to exploit this dependency.

References

Billings, D., Papp, D., Schaeffer, J., & Szafron, D. (1998). Opponent modeling in poker. *Proceedings of the Fifteenth Annual Conference on Artificial Intelligence* (pp. 493–499). Madison, WI.

Borck, H., Karneeb, J., Alford, R., & Aha, D. W. (2015). Case-based behavior recognition in beyond visual range air combat. *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference*. Hollywood, FL.

Bowling, M., Burch, N., Johanson, M., & Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science*, *347*, 145–149.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., & Colton, S. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*, 1–43.

Carr, R., Raboin, E., Parker, A., & Nau, D. (2009). Near-optimal play in a social learning game. *Proceedings of the Third International Conference on Computational Cultural Dynamics*. College Park, MD: AAAI Press.

Ciancarini, P., & Favini, G. P. (2010). Monte Carlo tree search in kriegspiel. *Artificial Intelligence*, *174*, 670–684.

Donkers, H., Uiterwijk, J. W. H. M., & van den Herik, H. J. (2001). Probabilistic opponent-model search. *Information Sciences*, *135*, 123–149.

Fern, A., Natarajan, S., Judah, K., & Tadepalli, P. (2014). A decision-theoretic model of assistance. *Journal of Artificial Intelligence Research*, *50*, 71–104.

Ghallab, M., Nau, D. S., & Traverso, P. (2004). *Automated planning: theory & practice*. Morgan Kaufmann.

Jensen, B., Karneeb, J., Borck, H., & Aha, D. (2014). *Integrating afsim as an internal predictor* (Technical Report AIC-14-172). Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, DC.

Keller, T., & Eyerich, P. (2012). PROST: Probabilistic planning based on UCT. *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling* (pp. 119–127). Sao Paulo, Brazil.

Kocsis, L., & Szepesvári, C. (2006). Bandit based monte-carlo planning. *Proceedings of the Seventeenth European Conference onMachine Learning* (pp. 282–293). Berlin, Germany: Springer.

Laviers, K., Sukthankar, G., Molineaux, M., & Aha, D. W. (2009). Improving offensive performance through opponent modeling. *Proceedings of the Fifth Artificial Intelligence for Interactive Digital Entertainment Conference* (pp. 58–63). Stanford, CA.

NAVAIR (2013). Next Generation Threat System. `http://www.navair.navy.mil/nawctsd/Programs/Files/NGTS-2013.pdf`.

Schadd, F., Bakkes, S., & Spronck, P. (2007). Opponent modeling in real-time strategy games. *Proceedings of the Eigtht International Conference on Intelligent Games and Simulation* (pp. 61–68). Bologna, Italy.

Shaw, R. L. (1985). *Fighter combat: Tactics and maneuvering*. Naval Institute Press.

Southey, F., Bowling, M. P., Larson, B., Piccione, C., Burch, N., Billings, D., & Rayner, C. (2005). Bayes' bluff: Opponent modelling in poker. *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence* (pp. 550–558). Edinburgh, Scotland: AUAI Press.

Tesauro, G. (2003). Extending q-learning to general adaptive multi-agent systems. *Advances in neural information processing systems 16* (pp. 871–878). Vancouver and Whistler, British Columbia, Canada.

TV Tropes (2015). Incredibly obvious tail. `http://tvtropes.org/pmwiki/pmwiki.php/Main/IncrediblyObviousTail`. Accessed 2015-02-21.

Zeh, J., Birkmire, B., Clive, P. D., Johnson, J., Krisby, A. W., Marjamaa, J. E., Miklos, L. B., Moss, M. J., & Yallaly, S. P. (2014). *Advanced framework for simulation, integration, and modeling (AFSIM) version 1.8 overview* (Technical Report 88ABW-2014-5041). Air Force Research Laboratory, Aerospace Systems, Wright-Patterson Air Force Base, Ohio.