A Goal Reasoning Agent for Controlling UAVs in Beyond-Visual-Range Air Combat

Michael W. Floyd[1] and **Justin Karneeb**[1] and **Philip Moore**[1] and **David W. Aha**[2]
[1]Knexus Research Corporation; Springfield, Virginia; USA
[2]Navy Center for Applied Research in AI; Naval Research Laboratory; Washington, DC; USA
{michael.floyd, justin.karneeb}@knexusresearch.com
david.aha@nrl.navy.mil

Abstract

We describe the Tactical Battle Manager (TBM), an intelligent agent that uses several integrated artificial intelligence techniques to control an autonomous unmanned aerial vehicle in simulated beyond-visual-range (BVR) air combat scenarios. The TBM incorporates goal reasoning, automated planning, opponent behavior recognition, state prediction, and discrepancy detection to operate in a real-time, dynamic, uncertain, and adversarial environment. We describe evidence from our empirical study that the TBM significantly outperforms an expert-scripted agent in BVR scenarios. We also report the results of an ablation study which indicates that all components of our agent architecture are needed to maximize mission performance.

1 Introduction

Beyond-visual-range (BVR) air combat is a modern form of air-to-air fighting where aircraft engage each other over large distances, often hundreds of kilometers. Compared to the close-range dogfighting that was common during World Wars I and II, BVR combat tends to be less reactive and involve more long-term planning and strategy. In this paper we describe an agent, called the *Tactical Battle Manager* (TBM), designed to control an unmanned aerial vehicle (UAV) in a BVR combat scenario using numerous integrated AI techniques.

BVR air combat has several key properties that make it an interesting domain to study and necessitates an integrated design rather than any single AI technique. Engagements involve multiple aircraft, both teammates and adversaries, operating in a contested airspace. The environment is continuous, partially observable (i.e., due to limited sensor ranges), and noisy (i.e., due to sensor errors). Additionally, aircraft need to meet tight real-time constraints to evade opponent attacks and avoid dangerous maneuvers (e.g., flying too low, colliding with teammates).

Our integrated architecture uses several parallel components that can each access, create, and modify information in shared data sources. This parallel design allows the components to process information in real time and avoids delays caused by slower components. The TBM uses *goal reasoning* [Aha *et al.*, 2013; Roberts *et al.*, 2016] to dynamically reason about its goals and modify them in response to unexpected events or opportunities (e.g., an opponent attack, an opponent separated from their squadron). *Automated planning* is used to generate plans based on the TBM's goals. Since the plans of opponent aircraft are initially unknown and may change over time, the TBM continuously monitors the actions of opponent aircraft and performs *behavior recognition* to predict their current plans and targets. The TBM uses its own plans, as well as the behaviors of friendly and opponent aircraft, to perform *state prediction* (i.e., anticipate how the environment is likely to change). However, because the TBM may be using incomplete sensory information, have erroneous sensor values, or have incorrect assumptions about opponent behavior, it continuously performs *discrepancy detection* to determine if there are any flaws with its predictions, assumptions, or opponent models.

The remainder of this paper describes our agent architecture and provides justification for the inclusion of each of the integrated components. In Section 2 we describe the BVR air combat domain and formalize the problem being addressed. Section 3 describes our integrated agent design and the role of each component. We empirically evaluate the TBM's design in Section 4, and discuss related work in Section 5. Finally, we discuss aspects of our work in Section 6, and summarize our contributions and mention areas for future work in Section 7.

2 Beyond-Visual-Range Air Combat

BVR air combat involves two opposing teams of aircraft located at large distances from each other (i.e., hundreds of kilometers) and operating in large airspaces (i.e., thousands of square kilometers). The objective of each team is to destroy the opponent aircraft or force them to retreat. Aircraft do not use short-range weapons, as is the case in dogfighting-style air combat, but instead use active radar homing missiles with ranges of approximately 50 kilometers.

We use the Advanced Framework for Simulation, Integration and Modeling (AFSIM) system [Clive *et al.*, 2015], a high-fidelity air combat simulator that allows aircraft to be controlled either programmatically or using physical hardware (e.g., flight sticks, cockpit consoles). AFSIM

allows for the inclusion of a variety of aircraft models, including existing real-world aircraft and custom aircraft. We use models based on the F-16 fighter jet. AFSIM allows for low-level control of an aircraft but also provides higher-level abstractions (e.g., maintaining altitude, moving towards a target, waypoint-based navigation).

The TBM is responsible for controlling a single aircraft in a BVR combat mission. At the start of a mission, the TBM and each of its teammates receives a *mission briefing*. This briefing contains information about the team, which is assumed to be correct, and the opponents, which may be incorrect. Information about the team includes: team leader, capabilities (i.e., each teammate's aircraft type, each teammates missile capabilities), tactics (i.e., initial mission goals, preferred altitudes for engaging and evading enemies, preferred angles of approach and firing angles), speed (i.e., passive speed, approach speed, engagement speed, cornering speed, escape speed), and weapons (i.e., distances at which missiles are expected to hit and miss, distance from an opponent that is considered dangerous, when to take defensive shots). This briefing's information about the opponents contains the number of opponent aircraft, their aircraft type, and weapons capabilities.

At discrete time intervals, each aircraft i receives as sensory input a set s_i^t containing the n_i objects $o_1, \ldots, o_{n_i}$ that are currently visible to it at time t ($s_i^t = \{o_1, o_2, \ldots, o_{n_i}\}$). These objects include positional information (i.e., latitude, longitude, altitude, heading, velocity) about teammate and opponent aircraft, as well as teammate and opponent missiles. Since each aircraft has partial observability, n_i can vary among time intervals and there is no guarantee that the same objects will always be visible (i.e., objects can move into and out of an aircraft's field of vision). However, aircraft on the same team can communicate and share sensory information. If S_{team}^t is the set of all sensory inputs received by an aircraft's team at the current time t, then each of that team's aircraft will have access to a team-level sensory input s_{team}^t that contains all visible objects:

$$s_{team}^t = \bigcup_{s_i^t \in S_{team}^t} s_i^t$$

The aircraft can perform several high-level parameterized actions, including *flying in a specified formation* (i.e., maintaining a speed and direction relative to other teammates), *free flying* (i.e., flying in a straight line in a specified direction at a fixed altitude), *flying relative to a target* (i.e., towards, away from, to its left, to its right), and *firing a missile at a specified target*. The role of the TBM is to use the mission briefing and sensory information to perform actions that intelligently control the aircraft.

3 System Design

In this section we describe the integrated design of the TBM (Figure 1). It uses several integrated components and shared resources that allow the components to interact and provide on-demand services. Reasoning components are shown in orange, discrepancy detectors in yellow, and shared resources in gray. The arrows denote the flow of information between the various components. For example, the Goal Manager modifies the goals stored in the Goals shared resource, which can be accessed and used by the Planner.

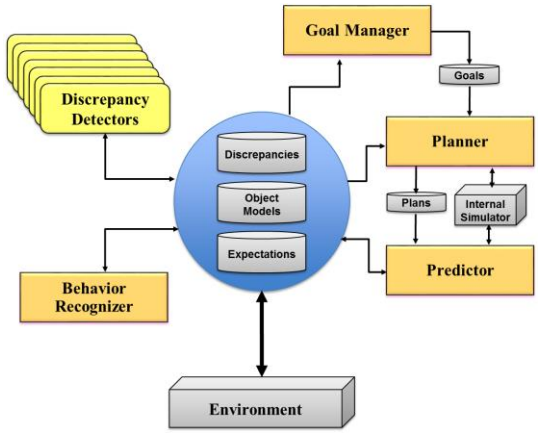

Figure 1: The conceptual diagram of the Tactical Battle Manager (TBM) composed of shared resources (gray), reasoning components (orange), and discrepancy detectors (yellow). Arrows show the movement of information between the various components.

3.1 Shared Resources

The TBM has five data sources that can be accessed and modified by any of its components (although in practice only a subset will actually access each data source): *Goals*, *Plans*, *Discrepancies*, *Expectations*, and *Object Models*. These shared data sources serve as the central communication mechanism through which the components share information. No single component is responsible for modifying all five data sources, so it is only through their integration that sufficient data is available for the system to intelligently reason on and act.

- **Goals**: The Goals data source contains the TBM's currently selected goals. Although the TBM can have multiple simultaneous goals, our current implementation restricts the agent to a single active goal. The representation of goals differs slightly from traditional representations (i.e., a set of grounded literals), and is instead composed of m higher-level *desires*. Each desire has an associated function λ_m that maps the current sensory input (i.e., s_{team}^t) to a value between 0 and 1 ($\lambda_m: S \rightarrow [0,1]$, where S is the set of all sensory inputs), with higher values better satisfying the desire. Each sensory input is represented by a tuple des of desire values $d_1, \ldots, d_m$ ($des = \langle d_1, \ldots, d_m \rangle$). A goal g is represented by a tuple of preferred desire values $pref_1, \ldots, pref_m$ ($g = \langle pref_1, \ldots, pref_m \rangle$), meaning that the TBM will attempt to achieve environment states that match those desire values (i.e., where des and g are similar). Examples of desires include maintaining the TBM's safety, maintaining a teammate's safety, disrupting opponent movements,

avoiding contact with additional opponents, and aggressively targeting opponents.

- **Plans**: The Plans data source contains the grounded plan the TBM is currently executing. A plan π contains a sequence of actions $a_1, \dots, a_l$ to achieve the current goal g ($\pi = \langle a_1, \dots, a_l \rangle$).
- **Expectations**: These are stored at two levels of granularity: *object-level* and *desire-level*. Object-level expectations contain the expected environment state at discrete time intervals (i.e., projections of what s^t_{team} will be at various points in time *t*). If expectations are available for k time points in the future, the object-level expectations E_{ol} is a list of k state expectations ($E_{ol} = \langle s^{t+1}_{team}, s^{t+2}_{team}, \dots, s^{t+k}_{team} \rangle$). The desire-level expectations E_{dl} are a higher-level representation that encodes how well each future state is predicted to satisfy each of the m desires ($E_{dl} = \langle des^{t+1}, des^{t+2}, \dots, des^{t+k} \rangle$). By using two levels of expectations, the TBM can examine whether the current sensory input meets expectations at a high level (e.g., the TBM has the expected level of safety) or at a low level (e.g., an opponent aircraft is at its expected position).
- **Object Models**: These contain detailed information about each observed object in the environment (i.e., aircraft and missiles). If at time t there have been n unique objects observed in the environment (i.e., in the current and all previous team-level sensory inputs), the set $\mathcal{OM}$ contains an object model O_i for each of the n objects ($\mathcal{OM} = \{O_1, O_2, \dots, O_n\}$). Each object model contains the object type ty (i.e., friendly aircraft, hostile aircraft, friendly missile, hostile missile), its plan π, its target tar, and the set OBS of all observations of the object ($O = \langle ty, \pi, tar, OBS \rangle$). For this paper we assume all missiles have the same plan and it does not change (i.e., it flies towards its target).
- **Discrepancies**: These represent unexpected events or environment states that are encountered by the agent. Any discrepancies that are identified by the discrepancy detectors are stored. Each discrepancy d contains the type ty (e.g., the detector that identified the discrepancy), its priority p (i.e., how significant it is), the object O that caused the discrepancy, the object's expected value v_{exp}, and its observed value v_{obs} ($d = \langle ty, p, O, v_{exp}, v_{obs} \rangle$). Discrepancies involving multiple objects are represented as multiple discrepancies, one for each object.

3.2 Goal Manager

The Goal Manager is responsible for selecting new goals for the TBM (i.e., modifying Goals). A goal change can be *externally* or *internally* controlled. An externally controlled goal change occurs when the TBM receives a command from a superior (i.e., a human teammate or lead aircraft). In this situation, the TBM immediately switches to the new goal. An internally controlled goal change is the result of the TBM's own decision making process. An internally controlled goal change may be the result of the TBM determining it has successfully completed its current goal, predicting it will fail to achieve it current goal (e.g., because of unanticipated actions by other entities), or recognizing an opportunistic situation (e.g., an exposed opponent). Goal failure is predicted by periodically reevaluating the current plan to determine if continuing to perform the plan will no longer achieve the TBM's goals (i.e., due to the dynamic and adversarial environment). Similarly, opportunistic situations are a result of the current plan resulting in unanticipated states that are potentially beneficial to the TBM (e.g., an *Opportunistic Target* discrepancy as described in Section 3.6). Being able to dynamically modify its goals gives the TBM the ability to respond to unanticipated events without relying on plans that cover all contingency situations.

The Goal Manager continuously monitors the Discrepancies data source and uses a set of rules to determine if the discrepancy warrants a goal change. For example, consider a situation where a *Model Changed* discrepancy (discussed in Section 3.6) was created because the hostile aircraft O' changed its plan (i.e., as determined by the Behavior Recognizer described in Section 3.3). The following are examples of rules the Goal Manager uses to handle Model Changed discrepancies:

Rule 1:

IF: (O' was not attacking TBM) **and** (TBM was not attacking O') **and** (O' is now attacking TBM) **and** ($distance(O', TBM) < threshold$)
THEN: *Change goal*

Rule 2:

IF: (O' was not attacking TBM) **and** (TBM was not attacking O') **and** (O' is now attacking TBM) **and** ($distance(O', TBM) > threshold$)
THEN: *Ignore discrepancy*

The two rules, while nearly identical, differ in how close the hostile aircraft is to the TBM. In Rule 1, since the hostile is now attacking the TBM and is close, the Goal Manager will change the TBM's goal in response to the immediate threat. In Rule 2, the hostile is now attacking the TBM but is a sufficient distance away (e.g., significantly outside the hostile's attack range) so the Goal Manager will ignore the discrepancy. The implemented version of the Goal Manager uses a collection of rules that are provided by domain experts.

The way in which the Goal Manager selects a new goal is based on goal priorities that are defined as part of an initial mission briefing. For example, the mission briefing may indicate that goals that target a specific high-value opponent are of higher priority than other goals. In this situation, the Goal Manager will select a new goal that targets the high-value opponent unless there are circumstances that prevent it

from doing that (e.g., the high-value target is out of radar range), in which case it will select a lower-priority goal.

3.3 Behavior Recognizer

The Behavior Recognizer examines the history of observations of each entity (i.e., stored in the Object Models), and updates their recognized plans and targets if any changes are detected. Our approach uses an entity's angle relative to each potential opponent aircraft to determine its target. For each potential target, the standard deviation of the angle is calculated, and the opponent with the lowest standard deviation over time is classified as the target. The intuition for this classification approach is that the entity will likely keep its target within its field of vision (i.e., if it is attacking) or at its rear (i.e., if it is evading), whereas the position of non-targets will not influence where it moves (i.e., there will be more variance in the angle relative to those aircraft).

The Behavior Recognizer performs simplified rule-based plan recognition. Each aircraft is labelled as either *Attacking* or *Evading*. The rules use the object's angle relative to its target, speed, and altitude to classify its plan. The following are examples of two rules used by our system:

Rule 1:
IF: (angle is *FACING*) **and** (speed is *FAST*) **and** (altitude is *HIGH*)
THEN: *Attacking*

Rule 2:
IF: (angle is *FACING AWAY*) **and** (speed is *FAST*) **and** (altitude is *LOW*)
THEN: *Evading*

The first rule encodes that aircraft that are facing their target, moving at high speeds, and at a high altitude tend to be performing attacking behaviors. Similarly, aircraft that are facing away from their target, moving at high speeds, and at low altitudes tend to be evading. For each of these classifications, the entity is assumed to be executing a simple plan to achieve the desired behavior. For an *Attacking* entity, the plan involves flying towards and firing missiles at its target, whereas for an *Evading* entity the plan involves flying away from any opponents and missiles. The rules are provided by domain experts.

3.4 Predictor

The Predictor uses the TBM's current plan (i.e., stored in Plans) and the recognized plans of the other entities (i.e., stored in the Object Models) to generate predictions of the future environment states (i.e., to revise Expectations). The TBM uses a lower-fidelity internal simulator to predict the environment changes that occur if each entity continues performing its current plan. More specifically, it uses a lightweight version of AFSIM that uses less sophisticated flight models (i.e., object movement is determined using simple functions rather than full physics models), full observability, and less frequent sensory updates (i.e., sensor values are provided once per second rather than ten times per second), allowing complete simulations to be run quickly (i.e., a fraction of a second).

Before the Predictor runs a new simulation, the previous expectations are cleared (i.e., $E_{ol} = \langle\ \rangle, E_{dl} = \langle\ \rangle$). This is done because the Predictor should have more information than when the previous simulation was performed (i.e., updated positions and plans), so new expectations are assumed to be more accurate than previous ones. At fixed time intervals during the simulation (i.e., $t+1, t+2, \ldots, t+k$), the current environment state is sampled and stored in the expectations (i.e., $s_{team}^{t+1}, s_{team}^{t+2}, \ldots, s_{team}^{t+k}$). We use a sampling rate of once per second. These expectations are coarse, due to the low-fidelity simulation, but provide a general trajectory of future environment states, allowing the TBM to identify if any entities deviate from their current plans or if new objects become visible (e.g., new opponents or missiles). Similarly, although the environment is highly dynamic, longer-term expectations are less important because the TBM will update its expectations frequently.

3.5 Planner

The Planner uses the TBM's current goal (i.e., stored in Goals) and the recognized plans of other entities (i.e., stored in the Object Models) to generate a plan for the TBM to execute (i.e., replace the plan stored in Plans). The Planner is implemented as a plan library planner [Borrajo *et al.*, 2015]; it uses a library of ungrounded plan templates (i.e., each plan is represented as a sequence of actions but the actions do not have their parameters specified). For example, consider the simple plan template:

1. Free fly at a specified *heading*
2. Fly directly at a *target*
3. Fire at the *target*

This specifies that the TBM should *free fly*, *fly*, and *fire*, but leaves the *heading* and *target* ungrounded. The templates in the plan library are provided by BVR domain experts and represent desirable air combat tactics.

During planning, the Planner generates multiple instantiations of any applicable plans (i.e., grounds the actions). The instantiations as exhaustive, but are constrained by any restrictions defined in the mission briefing (e.g., maximum attack speed, attack angle) or resource availability (e.g., number of remaining missiles). To evaluate each candidate plan, their outcome is predicted using the lightweight simulator (i.e., the simulator used by the Predictor). A single simulation is run for each candidate plan. The outcome of each candidate plan's simulation (i.e., represented in terms of how well a plan achieves each desire) is compared to the TBM's goal, and the plan that best achieves the TBM's goal is selected. Achievement of the TBM's goal is measured as the similarity between the TBM's goal and the state reached by performing the plan (i.e., the similarity between the desire values).

3.6 Discrepancy Detectors

The discrepancy detectors identify unexpected events or environment states encountered by the agent. There are seven discrepancy detectors in the TBM [Karneeb *et al.*, 2016]:

Incoming Missile, *Model Changed*, *Flanking Hostile*, *Expectations Violated*, *Out of Ammo*, *Low on Fuel*, and *Opportunistic Target*.

The Incoming Missile detector identifies unexpected hostile missiles. When an unexpected hostile missile appears (e.g., the TBM was not expecting the opponent to fire it), a discrepancy is created. This allows the TBM to dynamically respond to an attack and attempt to evade the missile. The Model Changed discrepancy detector creates a discrepancy whenever a hostile aircraft changes its plan or target. In response, the TBM may need to change its own tactics (i.e., goals or plans) or update its expectations.

The Flanking Hostile detector detects when hostile aircraft deviate from their expected position such that they approach the range at which they can attack the TBM's UAV. An aircraft that approaches attack range becomes a direct threat to this UAV, so creating a discrepancy allows the TBM to respond to the threat and modify the UAV's behavior accordingly. The Expectations Violated detector examines how closely the Expectations deviate from the actual environment states. While the previous three types of detectors look for individual discrepancies (e.g., a single missile or aircraft), this one identifies when the entire environment is deviating too far from expectations. Given the long-term plans and dynamic nature of BVR air combat, such deviations are to be expected, but using this approach allows the TBM to identify when new goals or updated plans may be necessary. The Out of Ammo and Low on Fuel discrepancy detectors are used to identify when the TBM's resources are running low (i.e., it should return to base or a resupply station).

Finally, the Opportunistic Target detector identifies when opponent aircraft that the TBM is not currently targeting become favorable to engage (e.g., can be flanked or attacked from behind). This allows the TBM to quickly respond to opportunistic targets and capitalize on an opponent's tactical errors.

4 Evaluation

Our empirical evaluation assesses the TBM's ability to operate effectively in a BVR air combat domain. Our experiments concern the following hypotheses:

H1: The TBM will outperform a baseline scripted agent in BVR scenarios

H2: Each component of the TBM contributes positively to overall mission performance

4.1 Experimental Conditions

The evaluation scenarios involve two teams of aircraft with four aircraft per team. A prototypical scenario was created where each team is aligned in a column with aircraft spaced 10 nautical miles from each other and opposing teams spaced 45 nautical miles from each other. The prototypical scenario was used to create 100 different random scenarios where each aircraft's position is modified by between -4 and 4 nautical miles (according to a uniform random distribution) in both the North/South and East/West directions. Figure 2 shows a graphical representation of one such random scenario. All evaluations use the same random scenarios. Since the initial positions may provide an advantage to one team, each random scenario is run twice with the teams switching positions between runs. This results in 200 total runs per evaluation. A run ends when one team is completely destroyed or 20 minutes elapse.

We used eight different teams of aircraft. One is controlled by the TBM, one is controlled by a scripted agent, and the remaining six are ablations of the TBM that have one component replaced with a simplified (but still functional) version. The teams are:

- $\boldsymbol{TBM_{full}}$: Aircraft are controlled by TBMs using the full agent architecture described in this paper.
- $\boldsymbol{RIPR}$: Aircraft are controlled by scripted agents that are implemented using the Reactive Integrated Planning aRchitecture (RIPR) [Clive *et al.*, 2015]. RIPR agents were designed with the assistance of subject-matter experts (i.e., ex-pilots) and are modelled using behavior trees. The agents perform competent BVR combat behavior across all aspects of an encounter (e.g., target pursuit, attacking, escaping danger).
- $\boldsymbol{TBM_{GM}}$: Offensive goals use less-sophisticated target selection that considers only the nearest opponent.
- $\boldsymbol{TBM_{plan}}$: The instantiated plans are restricted to consider only direct motion routes (i.e., flying directly to a target rather than a path that flanks it).
- $\boldsymbol{TBM_{pred}}$: Desire-level expectations are generated randomly according to a uniform random distribution.
- $\boldsymbol{TBM_{BR}}$: Performs behavior recognition for only the first 15 seconds of the mission (i.e., does not perform additional recognition afterward).
- $\boldsymbol{TBM_{DD}}$: All detectors are turned off except the Incoming Missile detector.
- $\boldsymbol{TBM_{none}}$: The simplified versions of all five components are used.

Each agent is given an initial goal to destroy all opponent aircraft while minimizing team casualties. We measure how well that goal is achieved using the following metrics:

- **Kills**: The number of opponent aircraft destroyed
- **Wins**: The number of scenarios that end due to all opponents being destroyed

4.2 Results

Each experiment involves the TBM_{full} team competing against one of the other seven teams. The results, shown in Table 1, measure the number of kills and wins for both the TBM_{full} team (labeled as TBM_{full}) and their opponents (labeled as *Opponent*). Also, the percent increase in the performance of the TBM_{full} team versus the opponent team is displayed (labeled as *Increase*), with positive values indicating TBM_{full} outperformed its opponent. For *Wins*, the

values do not sum up to 200 because some scenarios ended in draws (i.e., the time limit was reached with aircraft on both teams remaining). In all experiments, using the full integrated TBM architecture resulted in statistically significant improvements to BVR air combat performance (using a single-tailed *t*-test with $p < 0.01$).

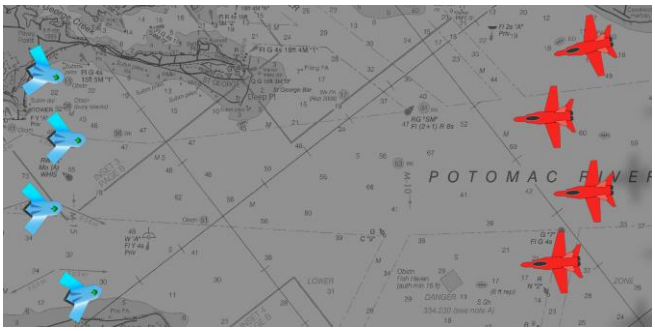

Figure 2: Graphical representation of the starting conditions in a constrained random 4 vs 4 scenario (aircraft size not to scale)

Table 1: Results of 200 scenarios comparing a team of four TBM_{full} agents to a team of four opponents

Opponent	Kills			Wins		
	TBM_{full}	Opponent	Increase	TBM_{full}	Opponent	Increase
$RIPR$	756	224	237.5%	165	5	3200.0%
TBM_{GM}	700	549	27.5%	118	54	118.5%
TBM_{plan}	663	580	14.3%	106	67	58.2%
TBM_{pred}	697	577	20.8%	112	50	124.0%
TBM_{BR}	679	571	18.9%	99	63	57.1%
TBM_{DD}	722	566	27.6%	124	53	134.0%
TBM_{none}	718	545	31.7%	127	50	154.0%

4.3 Discussion

Although TBM_{full} had sizable victories over all other teams, the most significant improvement was TBM_{full} over $RIPR$, with our goal reasoning agent beating the scripted agent the vast majority of the time. This provides strong evidence in support of **H1**. We conjecture that the highly dynamic and uncertain nature of the BVR environment made it difficult for a scripted agent to react to the full range of discrepancies and unexpected events. Although RIPR performed competent BVR behavior (i.e., could intelligently perform the full range of BVR combat behavior) and was designed by domain experts, it was infeasible for its behavior tree to cover all unexpected events in all situations. The TBM did not suffer a similar limitation because it can dynamically modify its goals or replan in response to unexpected events.

The full TBM agent architecture outperformed each of the ablations that used component simplifications. This provides support for **H2** and demonstrates that our integrated architecture relies on each of the TBM's reasoning components to achieve full performance. Our motivation for using simplified components rather than removing the components entirely was to ensure that the resulting agents could still perform full BVR behavior without being excessively handicapped. We conducted an additional test that compared the performance of the TBM that uses all five simplified components (TBM_{none}) versus the scripted agent. Although TBM_{none} is not as competent as TBM_{full}, it still significantly outperformed the scripted agent (133 wins, 23 losses, an increase of 478.3%). This provides further support for **H1** by demonstrating that the TBM, even with simplified versions of each component, outperformed a competent scripted agent that was authored by domain experts.

5 Related Work

To the best of our knowledge, there have been relatively few applications of AI agents in high-fidelity BVR scenarios, with the exception of the RIPR agent we discussed in our evaluation [Clive *et al.*, 2015]. In low-fidelity simulators (i.e., simple 2D environments without sophisticated flight and aircraft models), genetic algorithms have been used to optimally assign targets to each aircraft [Luo *et al.*, 2005] and select initial team formations [Mulgund *et al.*, 1998]. These approaches represent a subset of the complete agent behavior performed by the TBM. Also, they are performed only before the start of a scenario; they do not respond to changes in the environment or unexpected opponent behavior.

Our previous work in the BVR domain has primarily focused on individual reasoning components in isolation and has not evaluated the effectiveness of the integrated agent architecture. This includes measuring the effectiveness of various Behavior Recognition algorithms [Alford *et al.*, 2015; Borck *et al.*, 2015] and a smaller set of discrepancy detectors [Karneeb *et al.*, 2016]. In this paper we instead presented the complete TBM architecture, empirically evaluated the influence of each reasoning component, and for the first time directly compared it to an expert-authored BVR agent.

Goal Reasoning has been successfully used to control agents in several autonomous systems domains. Wilson et al. [2016] control an autonomous underwater vehicle as it performs a surveying task. The agent monitors for unexpected surface vehicles and modifies its goals depending on whether the vehicle is hostile. Their work differs from ours in that their agent uses only a single discrepancy detector (i.e., unexpected surface vehicle), performs a simpler form of behavior recognition (i.e., labeling a vehicle as hostile or not), and does not actively engage opponents (i.e., it stops surveying until the adversary leaves). GRIM [Johnson *et al.*, 2016] allocates unmanned aerial vehicles to search specific regions during disaster relief scenarios. GRIM responds to events including changing resource levels, adding or removing vehicles, unexpected environment factors, and changing mission priorities. This differs from the TBM in

that GRIM is focused on collaborative team goals rather than the real-time control of individual vehicles. This is possible in GRIM because each vehicle performs a predefined search pattern once assigned and does not need to respond to attacks from hostile enemies.

The Autonomous Squad Member (ASM) [Gillespie *et al.*, 2015] is another example of an integrated goal reasoning agent. The ASM agent controls a simulated unmanned ground vehicle and operates as a member of a human-robot squad. Based on observation of the environment, the agent hypothesizes about the actions of other actors and the occurrence of external events, recognizes the goals and plans of its teammates, and modifies its own goals in response. The primary difference between the ASM and the TBM is that the ASM agent does not perform the full behavior of a human teammate (i.e., it could not replace a human member of the team). Instead, it performs a limited set of behaviors meant to support human teammates.

While some of the previously described agents operate in the presence of hostile agents in the environment, none directly engage with them. In the real-time strategy game StarCraft, Goal Reasoning has been used to control an agent that attempts to defeat an opponent in a military-style battle [Weber *et al.*, 2012]. This is similar to the TBM in that it requires recognizing the opponent's intent and responding to unexpected opponent actions. However, the primary differences are that their system contends with only a single opponent in each scenario (although the opponent controls many individual units, as does the Goal Reasoning agent GDA-C [Jaidee *et al.*, 2013]), whereas the TBM engages a team of opponent agents, and their discrepancy detector looks only for changes in the number of visible objects the opponent controls (e.g., buildings and units), whereas the TBM uses a richer set of discrepancy detectors.

Dynamically changing an agent's goals and performing real-time replanning in response to a changing environment has been studied in a robotic search and rescue domain [Talamadupula *et al.*, 2011]. This differs from our work in that the agent does not detect unexpected events or select its own goals (i.e., this information is provided by the robot's operator). MADbot [Coddington *et al.*, 2005] uses a set of internal motivators to evaluate goals and trigger goal changes. Unlike the TBM, MADbot's goal changes are in response to internal factors and do not take into account external events or the actions of other agents.

6 Discussion

Our integrated agent was designed for high performance in a specific application domain, beyond-visual-range air combat, so it uses a considerable amount of domain-specific information. This is especially true of the rule-based components that rely heavily on rules provided by domain experts. Given the level of complexity of BVR combat and the well-established tactics used by pilots, it was necessary to use domain knowledge to ensure that the TBM achieved a high level of performance while behaving is such a way that a human teammate would consider its behavior reasonable and predictable. However, although our agent incorporates a significant amount of domain-specific information, we feel that the overall design could be used in other domains. Goal management, behavior recognition, planning, prediction, and discrepancy detection are all components that would be valuable to include in agents in other domains. Additionally, a number of the domain-specific aspects of our agent could be transferred into similar domains. For example, the Incoming Missile discrepancy detector could be modified to work in other domains where hostile agents fire projectiles or the Behavior Recognizer's rules for target detection could be modified for other combat environments.

One of the primary lessons learned from the development of our agent was that using some state-of-the-art algorithms for the components resulted in decreased overall performance in our domain. While state-of-the-art algorithms often improved performance on a specific subtask, the improvements were often not significant enough to offset their increased computational costs. Instead, we found it was more beneficial to use simpler low-cost techniques. The overall performance of our system was a result of numerous integrated techniques working together, so allowing real-time execution of all components was more important than small component-level performance improvements. However, our approach uses a modular architecture so we can replace existing components with state-of-the-art algorithms if they meet real-time constraints or the UAV adds additional computational resources.

7 Conclusions

In this paper, we presented an integrated agent architecture, the Tactical Battle Manager, for controlling an unmanned aerial vehicle in simulated beyond-visual-range air combat scenarios. The primary novelty of the TBM is that it combines techniques from goal reasoning, automated planning, opponent behavior recognition, state prediction, and discrepancy detection. Our empirical evaluation demonstrated that the TBM significantly outperforms an expert-authored BVR agent in a set of combat scenarios. Additionally, our ablation study demonstrated that each individual reasoning component of the TBM positively influenced mission performance; maximum mission performance was only achieved when the fully integrated architecture was used.

One area of future work we plan to address is to integrate learning into the discrepancy detection process. For example, this could include allowing the TBM to learn models of opponent aircraft and missiles, and using those models to detect different hardware configurations (e.g., new types of aircraft or more advanced missiles). We also plan to add capabilities for the TBM to identify opportunistic targets and communicate with other UAVs to perform small-team tactics (e.g., surround the opponent, create a diversion).

Acknowledgments

Thanks to OSD ASD (R&E) for supporting this research, and our subject matter experts for their many contributions.

References

[Aha *et al.*, 2013] David W. Aha, Michael T. Cox, Héctor Muñoz-Avila (Eds.). Goal Reasoning: Papers from the ACS Workshop (Technical Report CS-TR-5029). *University of Maryland, Department of Computer Science*, 2013.

[Alford *et al.*, 2015] Ron Alford, Hayley Borck, Justin Karneeb, and David W. Aha. Active behavior recognition in beyond visual range air combat. In *Proceedings of the Third Conference on Advances in Cognitive Systems*, 2015.

[Borck *et al*., 2015] Hayley Borck, Justin Karneeb, Michael W. Floyd, Ron Alford, and David W. Aha. Case-based policy and goal recognition. In *Proceedings of the 23rd International Conference on Case-Based Reasoning, Frankfurt am Main, Germany*, pages 30-43, 2015.

[Borrajo *et al.*, 2015] Daniel Borrajo, Anna Roubíčková, and Ivan Serina. Progress in case-based planning. *ACM Computing Surveys*, 47(2): 35:1–35:39, 2015.

[Clive *et al.*, 2015] Peter D. Clive, Jeffrey A. Johnson, Michael J. Moss, James M. Zeh, Brian M. Birkmire, and Douglas D. Hodson. Advanced Framework for Simulation, Integration and Modeling (AFSIM). In *Proceedings of the 13th International Conference on Scientific Computing*, pages 73-77, 2015.

[Coddington *et al.*, 2005] Alexandra M. Coddington, Maria Fox, Jonathan Gough, Derek Long, and Ivan Serina. MADbot: A motivated and goal directed robot. In *Proceedings of the 20th AAAI National Conference on Artificial Intelligence*, pages 1680-1681, 2005.

[Gillespie *et al.*, 2015] Kellen Gillespie, Matthew Molineaux, Michael W. Floyd, Swaroop S. Vattam, and David W. Aha. Goal reasoning for an autonomous squad member. In *Goal Reasoning: Papers from the ACS Workshop*, pages 52-67, 2015.

[Jaidee *et al.*, 2013] Ulit Jaidee, Héctor Muñoz-Avila, and David W. Aha. Case-based goal-driven coordination of multiple learning agents. In *Proceedings of the 21st International Conference on Case-Based Reasoning*, pages 164-178, 2013.

[Johnson *et al.*, 2016] Benjamin Johnson, Mark Roberts, Thomas Apker, and David W. Aha. Goal reasoning with information measures. In *Proceedings of the Fourth Conference on Advances in Cognitive Systems*, 2016.

[Karneeb *et al.*, 2016] Justin Karneeb, Michael W. Floyd, Philip Moore, and David W. Aha. Distributed discrepancy detection for BVR air combat. In *Goal Reasoning: Papers from the IJCAI Workshop*, 2016.

[Luo *et al.*, 2005] De-Lin Luo, Chun-Lin Shen, Biao Wang, and Wen-Hai Wu. Air combat decision-making for cooperative multiple target attack using heuristic adaptive genetic algorithm. In *Proceedings of the 4th International Conference on Machine Learning and Cybernetics*, pages 473-478, 2005.

[Mulgund *et al.*, 1998] Sandeep Mulgund, Karen Harper, Kalmanje Krishnakumar, and Greg Zacharias. Air combat tactics optimization using stochastic genetic algorithms. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 3136-3141, 1998.

[Roberts *et al.*, 2016] Mark Roberts, Daniel Borrajo, Michael T. Cox, Neil Yorke-Smith (Eds.). Goal Reasoning: Papers from the IJCAI Workshop, 2016.

[Talamadupula *et al.*, 2011] Kartik Talamadupula, Paul Schermerhorn, J. Benton, Subbarao Kambhampati, and Matthias Scheutz. Planning for agents with changing goals. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling: System Demonstrations*, pages 71-74, 2011.

[Weber *et al.*, 2012] Ben G. Weber, Michael Mateas, and Arnav Jhala. Learning from demonstration for goal-driven autonomy. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1176-1182, 2012.

[Wilson *et al.*, 2016] Mark A. Wilson, James McMahon, Artur Wolek, David W. Aha, and Brian H. Houston. Toward goal reasoning for autonomous underwater vehicles: Responding to unexpected agents. In *Goal Reasoning: Papers from the IJCAI Workshop*, 2016.