Data-Driven Behavior Modeling of an Army Ground-Based Air Defense System (AGBADS)

A.W. Burger, Dr. M.P.D. Schadd, N. M. de Reus
TNO Defense, Safety & Security
The Hague, the Netherlands
{annemarie.burger, maarten.schadd, nico.dereus}@tno.nl

ABSTRACT

This research aims at learning the behavior of operators or role players in military simulators based on observed behavior. The paper describes the principle of so-called imitation learning that we used to learn behavior based on observations of players. We present an overview of the existing examples of the current application of data-driven behavior modeling in the military domain. We apply imitation learning to the army ground-based air defense system (AGBADS) operator behavior, for which live data (from training events) was available. The data that is used in the learning process originates from the 2021 air defense exercise Joint Project Optic Windmill (JPOW). After extensive pre-processing, multiple behavior models were constructed that can be used to provide fire control solutions (when to fire, the salvo size and which launcher to use) for a ground-based air defense system given a tactical situation.

We create explainable models using decision trees, which can help during the after-action review process to give trainees insight in their actions when training as an AGBADS operator. Using less explainable but more precise models, we can create replicas of behavior of an AGBADS, which can be used to simulate the AGBADS in training. Furthermore, the models could be used to aid decision support for AGBADS operators. We achieved quite reasonable performances with regards to the models, especially considering that we had to work with a discrepancy between logged data and the perceived situation as experienced by the operator, making it hard to create an accurate feature representation of the situation.

ABOUT THE AUTHORS

Annemarie Burger is a scientist in the Modeling, Simulation and Gaming department at TNO with over a year experience in the field of data modeling and simulation. Her main research areas include Artificial Intelligence technology for diverse applications, both in the military and the civil domain. She holds degrees in Big Data Management & Analysis and Artificial Intelligence.

Maarten Schadd is a scientist in the Modeling, Simulation and Gaming department at TNO. His main research focus at TNO is on Artificial Intelligence for decision support systems that support the military planning process. He holds a MSc and a PhD degree in Artificial Intelligence and is an expert in Monte-Carlo techniques and optimization algorithms. He has gained experience in developing complex interacting systems in the private sector.

Nico de Reus is a scientist in the Modeling, Simulation and Gaming department at TNO. He holds an M.Sc. degree in Mathematics from Delft University of Technology in the area of optimal control theory and systems optimization. He has over 25 years of experience in simulation. His research areas include C2 to Simulation Interoperability, Concept Development & Experimentation Simulation and Artificial Intelligence for (military) decision support.

Data-Driven Behavior Modeling of an Army Ground-Based Air Defense System (AGBADS)

A.W. Burger, Dr. M.P.D. Schadd, N.M. de Reus
TNO Defense, Safety & Security
The Hague, the Netherlands
{annemarie.burger, maarten.schadd, nico.dereus}@tno.nl

INTRODUCTION

The increasing use of simulation in education, training, analysis and decision support, leads to a higher demand for simulation models. For such models we need to describe the behavior of the entities in the simulator. A high level of realism is essential in order to adequately prepare the training audience for future missions.

Behavioral models date back to over 60 years (Simon, 1955). We define a behavior model as an operational, conceptual, or executable model of the behavior of a human-like, human-controlled, or autonomously operating real-world system. When the system is human-controlled, we consider both systems that are controlled by a single user or multiple users. Examples of real-world systems of which one could create a behavioral model are tanks driven by tank drivers, fighter jets flown by pilots or unmanned aerial vehicles (UAVs), or infantry soldiers. Furthermore, this definition does not restrict the size of the systems, so this can also include for example tank battalions, ship flotillas, fighter jet squadrons or UAV swarms. The question arises how to create behavior models for these systems that display realistic behavior within the simulator.

Creating behavior models is not an easy task. We want to highlight several limitations to behavioral modeling and their integration into simulators. Firstly, a model depends on the abstraction of the simulator that it is running in. For example, when the simulator takes time-steps of one minute, then any behavior model that describes actions for shorter time spans is not necessarily suitable without further aggregation steps. Secondly, a model depends on the available features. When no weather information is provided, any model that takes the weather into account produces additional overhead, or possibly even inaccurate decisions in case a default weather is assumed. A last limitation is that the models are specific to a unit aggregation level. It would be great if models of individual soldiers could be aggregated into a squad model, and several squad models into a platoon model, and so on. However, with such an aggregation, the small errors at the soldier level may be amplified at the platoon level. Furthermore, the running time of such a detailed simulation may be much too large if such a model is to be used for decision support at the brigade level, for example.

When considering these limitations, we regard the probability low that a behavior model that has been created within a certain simulator for a specific goal can be reused, for example in a different simulator for another purpose. This means that in general a new behavior model must be created for each specific context. The creation of behavioral models should therefore be as quick and easy as possible. In this article, we argue that imitation learning, a sub-field of machine learning, can be used to create a behavioral model based on examples of the desired behavior. We test whether a realistic model can be created purely based on actual logged military data, without implementing any manual behavior rules.

As case study we chose to learn the behavior of an Army Ground-Based Air Defense System (AGBADS) operator when defending the surrounding air space against hostile elements such as aircrafts. An AGBADS consists of multiple weapon platforms that need to coordinate in order to decide which platform(s) engage(s) which target(s), at what time, and with how many projectiles. Information such as the current location, speed, altitude, and direction of the target(s) determine the outcome of this decision. The AGBADS operators have to practice so that this decision-making process can be executed efficiently and effectively during real missions. We use data that was generated during the Joint Project Optic Windmill (JPOW) 2021 exercise. This is the leading European Integrated Air & Missile Defense exercise for both the tactical and operational level (Joint Air Power Competence Centre, n.d.). The JPOW exercise is a unique combination of simulated and real forces, entities, and events. Since it practices integrated air and missile defense, the

focus is on eliminating flying targets. Most of these targets are simulated, whereas most firing entities are real troops (using simulated weapons) from a number of different countries. Both real and simulated entities are integrated into a single ground truth dataset, as well as an observed truth dataset. The latter is the information that is displayed to the participating countries. These countries share radar data with one another and coordinate attacks and courses of actions.

The behavioral model of the AGBADS operators developed during this study as a proof of principle has several potential applications that are beneficial to the military. It enables us to explain the current decision heuristics of the operators which can be beneficial to improve after-action reviews during an exercise. In this after-action review one could use our behavioral model to quickly indicate where the trainees deviated from the correct behavior (assuming that the previously-learned behavioral model contains the correct behavior). Our explainable behavioral models can uncover points for improvement as well as set the stage for discussion why the models of different participants differ. Another use of a model that imitates the operator behavior is to use it for simulating an AGBADS in larger scale scenarios, for example in a synthetic wrapping scenario. A last use-case would be to use the behavior models during future conflict situations. The outcome of the simulated scenarios can provide combat decision support to a military commander.

LEARNING BEHAVIOR MODELS BY IMITATION & PREVIOUS WORK IN THE MILATARY DOMAIN

In its bare essence, a behavioral model follows the same steps as humans do when taking a decision. A prominent framework is the OODA-loop (Boyd, 1987). The four phases of this loop are Observe, Orient, Decide, and Act. The observe and orient phase have the single purpose of gaining situational awareness and situational understanding. This military framework has been successfully used in various autonomous agents (Heinze, et al., 2002), and is applied in a large variety of situations (Plehn, 2000) (Tweedale, et al., 2007) (Clough, 2002). A second framework is called BDI, for Beliefs, Desires, and Intentions (Bratman, 1987) (Georgeff & Ingrand, 1989). The basic BDI paradigm is widely used to achieve human-like intelligence, but the artificial agents often lack ideal characteristics such as ‘Coordination and Learning’ (Tweedale, et al., 2007). This has been extended in (Rao & Georgeff, 1995) and is now widely used in practice (Nwana, 1996).

One way to achieve more efficient and effective modeling within the OODA or BDI framework is to use state-of-the-art technology in the research field of artificial intelligence to create behavior models (Roessingh, et al., 2017). A promising research direction, which aims at improving the realism of military behavioral models as well to reduce the modeling time, is to use machine learning based on measured operational data. In machine learning applications, correct and incorrect examples of behavior or decisions are presented to a learning system with the intention that the system is able to generalize the seen examples. This is called supervised learning (Russel & Norvig, 2020). A sub-field of supervised learning is called imitation learning (Schaal, 1999) in which an expert explicitly trains a model to imitate the desired behavior.

When an algorithm has to imitate human behavior but has a different information position, this is defined in the literature as Imitation from Observation (IfO) (Torabi, Warnell, & Stone, 2019). It can be further subdivided into model-based and model-free. In model-based IfO, an algorithm tries to create some form of dynamics model of how the underlying system works, either to predict the next action based on the observed state, or to predict the next state based on current state and chosen action. Within the model-free category, no model of the underlying system is used. We can further distinguish this last category into (1) adversarial methods that use a simulator to collect data and compare the data to the expert demonstrations, and (2) reward engineering (Gupta, Devin, Liu, Abbeel, & Levine, 2017), which learns a reward function for states.

Imitation learning has several applications in the military domain (Nguyen, Garratt, Bui, & Abbass, 2017) (Park & Oh, 2020). For example in (Berthling-Hansen, Morch, Løvlid, & E., 2018), imitation learning is applied to learning decision policies of computer generated forces. The learned behavior can afterwards be used for the training of soldiers in the simulator (Teng, Tan, & Teow, 2013).

AGBADS USE-CASE

This research focuses on learning the behavior of an Army Ground-Based Air Defense System (AGBADS) operator when defending the surrounding air space against hostile elements such as aircrafts. An AGBADS combines multiple weapon platforms; (1) National Advanced Surface to Air Missile System (NASAMS), which fires Advanced Medium Range Air-to-Air Missiles (AMRAAMs) and (2) Fennek Weapon Platform which fires short range Stinger Surface to Air Missiles for targeting rotary aircrafts. Note that the AMRAAM was originally made for Air-to-Air applications but is also used for Surface-to-Air applications. Each of these platforms are manned, and are build up from a Surface-to-Air Missile (SAM), a fire control (SFC) station and a MPQ-64 Sentinel-Radar, controlled by an operator. The total system configuration is visualized in Figure 1.

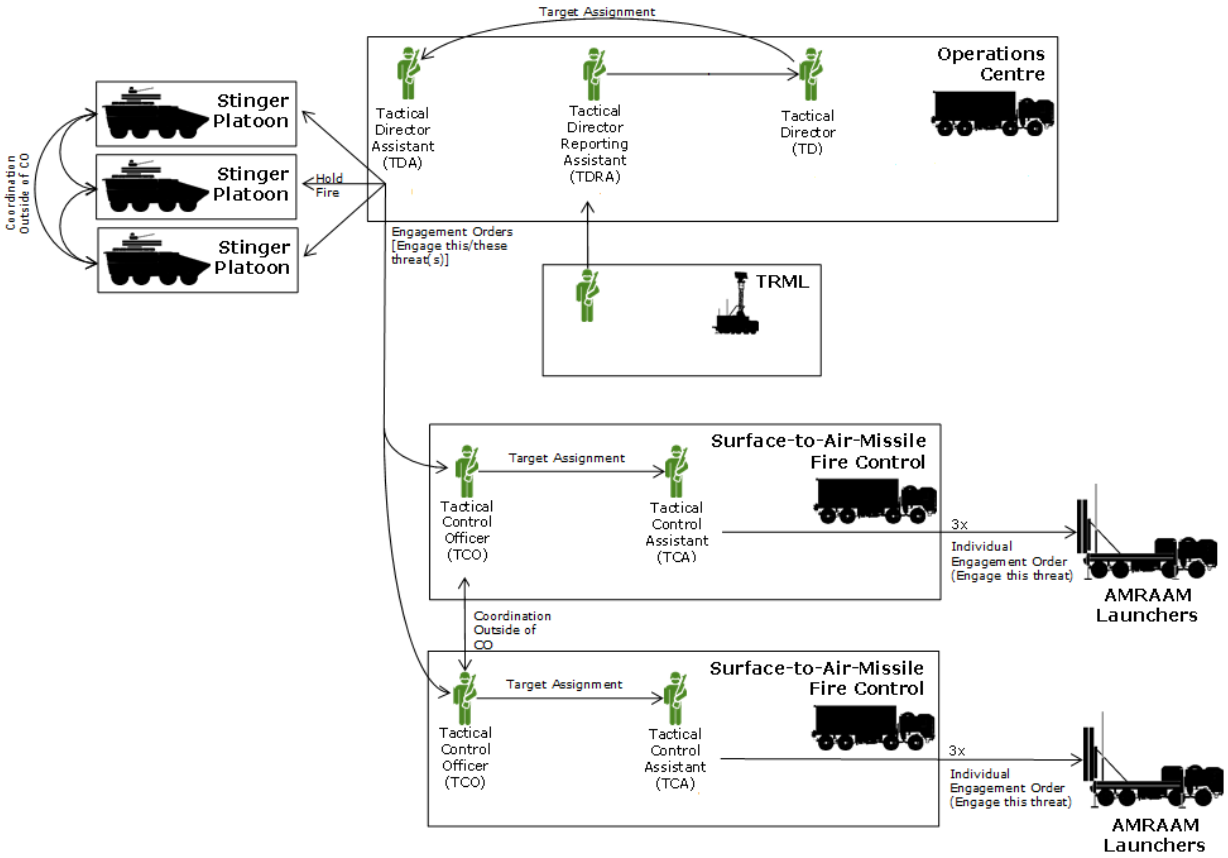

Figure 1: Schematic Representation of an AGBADS

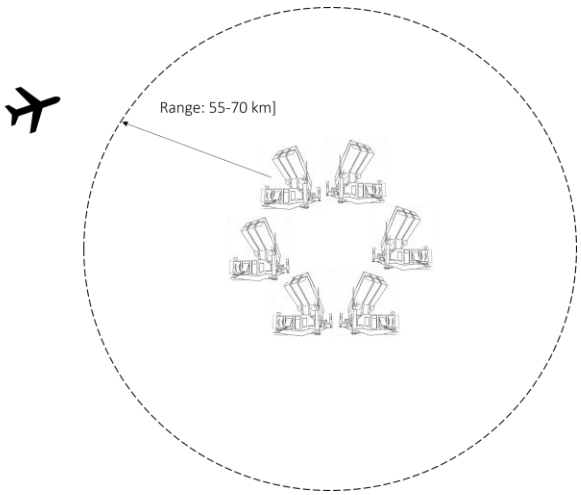

Figure 2: A Typical NASAMS Configuration within an AGBADS

A typical configuration of a NASAMS components cluster with their approximate range according to (Zwarts, 2017) can be found in Figure 2. All are facing a different direction, thereby dividing the airspace. NASAMS are used to fire at helicopters and (unmanned) airplanes, as well as Cruise Missiles (CM) to a lesser degree (Zwarts, 2017).

Decisions of multiple individuals contribute to the behavior of the AGBADS system. To model this behavior, we use a dataset from the Joint Project Optic Windmill (JPOW) 2021 exercise. We must note however, that since we only have data that describes a firing of an AMRAAM, we decided on creating a behavior model that describes the behavior of a NASAMS, which exists of the Surface-to-Air Missile Fire Control components in Figure 1. The approach we present in this article can later be extended to data of Stingers in order to create a behavior model of an entire AGBADS.

As we want to create a model on the behavior of an AGBADS of which we can only observe the NASAMS, we investigated what behavior data of the NASAMSs is available. The first data category is location data. NASAMS ground stations move regularly to new positions during the JPOW exercise. However, when defending the air space, they remain stationary. The location choices are made in the exercise scenario and only local decisions such as if and when to fire missiles are made by the NASAMS crew. Therefore, a movement model does not seem valuable at this time. A second data category is about internal AGBADS communications. Examples include sender, receiver, and type of data. As such information is not logged in the database, it is not possible to create a behavior model on this level. The third data category is behavior data about firing at incoming aircrafts. Decisions such as when firing happens, with what weapon platform and how many missiles are the core responsibility of the NASAMS and interesting to model and learn from. We therefore focus on learning when and how a NASAMS unit fires.

In this research we focus on the following decisions:

1. How long to wait until firing a salvo? Missiles have a certain minimum and maximum effective range, and a distance with maximum effectiveness. As there is a chance of missing the target, there should be sufficient time for a second or even third salvo to be fired at the target.
2. How many missiles should be fired at the target, i.e., what is the salvo size? Firing more missiles means a higher probability of eliminating the target, but may also have consequences for the sustainability of the air defense mission because the system can run out of missiles. Specifically, if the target is not a single threat, but a group of several threats this decision is not trivial.
3. Which NASAMS should be used? Each NASAMS covers a certain part of the air space. Furthermore, reload times, munition and firing doctrine have to be taken into account, as well as obstructions in the terrain.

Since the AGBADS data from the JPOW exercise is classified, we can only provide a high-level summary of the results.

DATA QUALITY

The JPOW dataset contains an abundance of data, including the state of all entities participating in the exercise. This state consists of a timestamp, a location, and a velocity amongst other properties. During the JPOW exercise fire events were logged. These represent the moment an (NASAMS) entity fired at a target and consists of a timestamp, a firing entity, a target entity, and other properties.

As the JPOW exercise is a training, concept development and experimentation exercise that combines real entities with simulated ones, there is one single simulator in charge of simulating all the virtual entities. This is referred to as

'ground truth'. The Distributed Interactive Simulation (DIS) protocol is used to distribute this ground truth among the simulators. A real entity (connected to the simulation environment), standing in the field, uses its sensor to observe either other real entities or simulated ones (which are injected into the sensor). Such live observations are also logged but can only be regarded as 'observed truth' as there may be observation errors and restrictions. An example is that a radar can only observe the position of an aircraft with a certain accuracy. The observed truth data, also called perceived truth and called LINK data, is communicated to, within- and from the AGBADS via the so-called Tactical Data Link. LINK only contains data that was first observed and then communicated between units. Therefore, the location can be inaccurate, and information such as entity IDs is not available.

We removed data points from the dataset that indicated faulty logging. We remove fire events for which: (1) the firing NASAMS launcher was impossibly far away from the target; (2) the firing location did not align with the location of the firing NASAMS launcher. Furthermore, we removed NASAMS state information when these launchers never fired and were located too remotely to be part of the exercise. Recall that state information includes a timestamp, a location, and a velocity amongst other properties.

In total 543 firings of NASAMS launchers are present in the data. When a (group of) target(s) is being engaged, multiple shots can be fired in close succession. This is called a salvo. A salvo may be larger than the group of targets to compensate for potential misses. As some fire events are parts of salvos, these were combined, leaving 317 firing salvos. For each salvo, NASAMS states prior to firing are selected to be part of the dataset, so that predictions of when to fire can be made. These selected states are chosen for every salvo from the time window starting 10 minutes prior to the first fire event in the salvo or starting when the target-entity was in range of the NASAMS components, which is defined as 55-70 km according to Richardson (Stealth Warplanes, Deception, Evasion, and Concealment in the Air, 2001). Together, this information forms a dataset with 5,537 data points.

DATA PRE-PROCESSING

The data was represented in such a way to be easily machine processable. Based on discussions with domain experts, we found that a NASAMS operator uses information that is not directly logged in the data, but which can be derived from the data. This pre-processing step provides a behavior model with similar input information as humans perceive.

The following features are derived from the raw data for this study:

Dependent variables:

- Time to shoot: If a potential target is first observed, a process of deciding whether to engage it, and when, is executed. After this decision, the NASAMS operator may still wait some time to increase the likelihood of a hit. The time to shoot measures the timespan from a current state to actual firing in seconds. This is the value to predict in order to make our first decision, this being 'how long do we wait until firing'.
- Salvo size: Each missile of the same salvo is a separate fire event. We count the amount of fire events by the NASAMSs that occur within 1,000 seconds from each other that is aimed at the same group of targets. The range is chosen so wide since it sometimes occurs that a shoot-look-shoot tactic is used. This means that there can be quite a bit of time between two shots since the operator waits until the first shot hits or misses its targets, and then makes the decision whether to fire another time. This is the value to predict in order to answer our second decision, being 'how many missiles should be fired at the target'.
- Firing NASAMS: A number between 0 and 11 to differentiate between the specific NASAMS, where number 0 represents the NASAMS closest to the target, and 11 the NASAMS furthest away. This is the value to predict in order to answer our third decision, this being 'which weapon system should we use'.

Independent variables:

- Distance: The distance in meters from the target to the NASAMS.
- Altitude: The altitude in meters of the target above mean sea-level, which is referenced by the WGS84 coordinate the target location is defined in.
- Velocity: The velocity of the target. This is derived from the logged x, y and z velocity.

- Group size: The number of aircrafts (either fixed wing or helicopters) that the target group consist of. We assume that all targets within a range of 1,000 meters belong to the same group. The aircraft that was targeted first defines the center of the circle that the targets are in.
- Closing velocity: Since the model needs to be independent of a specific scenario, it needs to be made invariant against rotations of the scenario. Therefore, we calculate the current speed of the target with which it approaches the geographical center of the NASAMSs. In case that the target is flying straight at the NASAMSs, this variable is identical to velocity. In case the target travels at a 90-degree angle, then this variable has value 0, and the value can be negative when the target distances itself from the NASAMSs.
- NASAMS angles: This feature consists of up to 12 sub-features and requires some additional explanation. To be invariant for identifiers of NASAMSs, or specific exercise setups, the NASAMSs are ordered based on their angle towards the incoming target. The data shows that NASAMSs are often placed in a hexagon shape, facing outwards. The NASAMS that is oriented the most towards the target is numbered 1 meaning that their primary target line is most similar to the line on which the target is approaching. The number of active NASAMSs is not constant and varies between 6 and 12. A schematic explanation for this feature can be found in Figure 3. For each NASAMS we report the angle towards the target in each of the sub-features.
- NASAMS distances: This feature consists of up to 12 sub-features in a similar way as the NASAMS angles do. We calculate the distance between each NASAMS and the target and allocate these to the sub-features in ascending order.

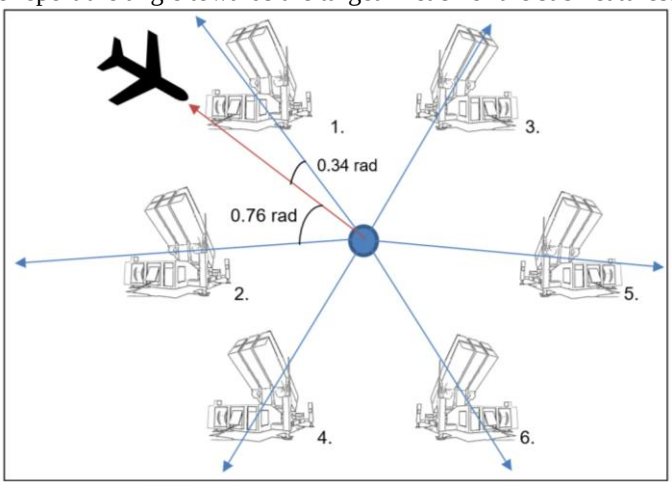

Figure 3: A schematic figure representing the NASAMS angles feature. In the example 6 NAMAMS are present, but it may be up to 12. The NASAMS launcher that has the most similar primary target line as the target approach line with regards to the center of mass of the NASAMS. In this example: NASAMS_angle_1 = 0.34 and NASAMS_angle_2 = 0.76 etc.

We created slightly different datasets for the three decisions we want the models to learn. For all decisions we use the variables for distance, height, velocity, group size, and approach speed. For determining when to fire, we add time to shoot as our dependent variable. For determining the salvo size. we use salvo size as our dependent variable, and we add the NASAMS angles to our independent variables. For choosing the firing NASAMS system, we use the firing NASAMS angles as our dependent variable and add the NASAMS distances to our independent variables.

APPROACH

Explainability, also sometimes referred to as interpretability, of artificial intelligence results is an important topic in military decision making (Gunning & Aha, 2019). Whether explainability is required depends on the specific situation. For instance, when used in decision support tools, the results must be explainable, at least in the development phase of the support, but preferably also in operations, in order to create trust in the correctness of the recommended decision. Also, in exercise situations, an explainable model is important so that it can be discussed in an after-action review. However, for simulating an AGBADS in a different setting, for example when wargaming a course of action, explainability is not needed, only realism is. We therefore created multiple models with these requirements in mind. These are described below and cover both explainable as well as non-explainable concepts.

Decision tree as explainable concept

A decision tree (Russel & Norvig, 2020) works by splitting training examples into distinct groups that differ from each other on a value of a feature (independent variable) that creates the biggest difference in target value (dependent variable). An example of a split could be that all data points with a distance higher than 40.000 meters are in group A and all other data points are in group B. The goal is to find a split in such a way that both groups are more uniform within their own group regarding the target value (e.g., *time to shoot*).

Even though a decision tree is deemed an explainable model, it does however require some base level knowledge. If these decision trees are to be used in an after-action review to debrief NASAMS operators on their training, they first might need a brief explanation on how these work, and the decision tree should be displayed in such a manner that information can be quickly digested.

A fictional example of a decision tree that could be used for an after-action review for NASAMS operators is shown in Figure 4. It tries to predict the time to shoot, based on the distance of the incoming target. Each node contains the elements: (1) *mae* (mean average error), which in this case is the average number of seconds that the prediction differs from the actually observed time to shoot; (2) *samples*, which is the percentage of data points that falls within this node of the tree when following the path from the root to the node, and (3) *value*, which is the current prediction of how many seconds to wait before firing. All the nodes except the lowest level also have a *distance* indicator, which describes the chosen split of the dataset based on the distance of the target position in the observed track. In more complex trees, the split may be made on different attributes on each node of the tree.

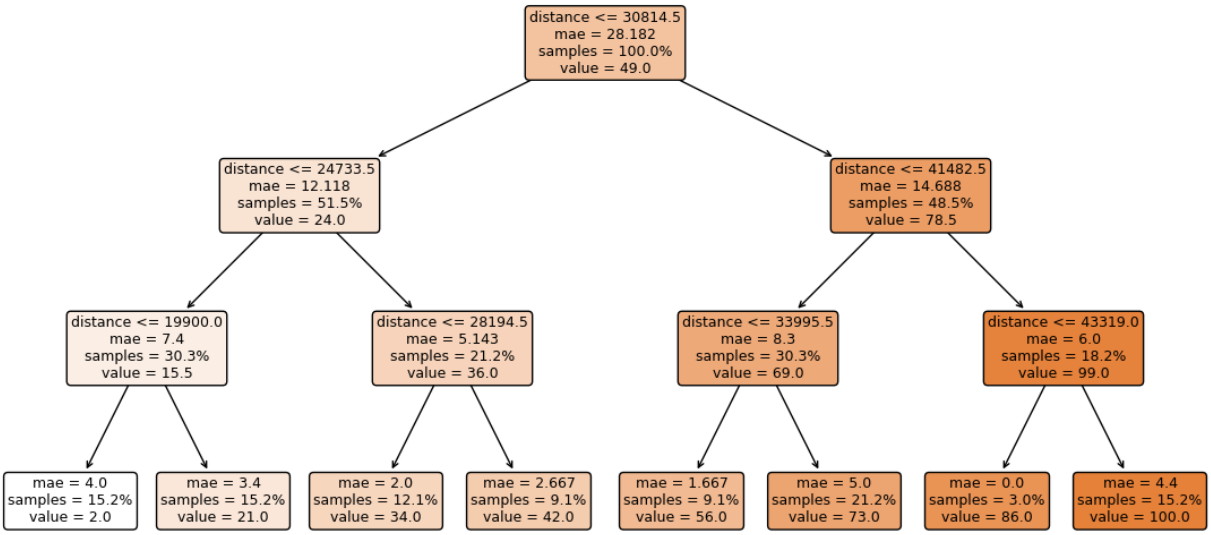

Figure 4: Fictional decision tree of the time-to-shoot predictive model

The tree is read from top (root) to bottom (leaves). The root node contains the entire dataset, and therefore has *samples* = 100%. The best value to predict here for time to shoot is 49 seconds, since it is the median value of the training dataset used to create this (fictional) decision tree.

The algorithm splits the dataset to maximally reduce the mean average error. In the root node one can see that all data points that have a distance less than or equal to 30,814.5 are placed in the left child node, and all other data points in the right child node. We see that in the left node the *mae* has decreased to 12 and in the right node to 14. One can note the difference of the predicted time to shoot between data-points that are close by and far away.

This process is continued until a stopping criterium. This criterium may be a maximum tree-depth, a minimum threshold for *mae* reduction, or various combinations. In our example the tree has a maximum depth of 3. This parameter needs to be carefully tuned to get optimal accuracy without overfitting. This means getting our *mae* as low as possible without making a decision tree that is too specific and only works on the cases that we see in the training data. A highly specific model would lead to low accuracy on predictions using new input data.

Non-explainable concepts

A *random forest model* (Russel & Norvig, 2020) consists of many decision trees. Since data can often be interpreted in many ways, and many splits and distinct groups can make sense on the same data, this model tries to average between those options. It builds many slightly different decision trees and then averages their results to determine a final prediction. This method is less explainable as the decision tree model, as it is based on the average of a large number of decision trees.

XGBoost (Chen & Guestrin, 2016) is a more complex type of model that in its core also makes use of decision trees. It uses gradient boosting to optimize for speed and performance. It is one of the best performing machine learning models, it is widely used and has dominated many machine learning competitions. However, an XGBoost model is not explainable as it combines many decision trees.

RESULTS

For each of the three decisions we trained the three model-types explained in the previous section, so that we can compare how the different model-types perform while catering to various requirements in explainability. The data points were split up in test- and training examples in such a way that data points from the same track are all either in the test or training set. A track is a sequence of data points that belong to the same target. Because the training data is different for different decisions, they all require separate model tuning. An overview of the performances of these nine models can be found in Table 1. For the first two decisions we report the mean absolute error (mae), meaning the average amount of seconds, or average number of missiles that the model prediction differs from the ground truth. For the last decision we report accuracy, meaning the percentage of cases for which the model prediction is exactly equal to the ground truth.

Table 1: Learning performance for various classifiers

	Decision Tree	Random Forest	XGBoost
Time to Shoot **How long to wait until firing?**	70 seconds mae	46 seconds mae	30 seconds mae
Salvo **How many missiles should be fired at the target?**	0.69 mae	0.67 mae	0.75 mae
Firing NASAMS **Which weapon system should be used?**	46% accuracy	54% accuracy	53% accuracy

Decision 1: **How long to wait until firing?**
For the first decision we found that the XGBoost regressor model performed best, with a mean absolute error of 30 seconds. For baseline reference: when always predicting the mean of the 'time to shoot' variable across all training examples, the mean absolute error would be 120 seconds.

When requiring a more explainable model, one can opt for the decision tree regressor. This model had a mean absolute error of 70 seconds, which is a significant decrease in performance.

A visualization of the XGBoost model trained on fictional data for this decision can be found in Figure 5. The location of the AGBADS is represented as a cross and the target as a circle on a map. The shown table contains the input variables as well as the output variable, as discussed in the data pre-processing section. We have added the observed time to shoot as present in the data, which is the ground truth. You can see how the prediction is usually quite similar to the real time to shoot.

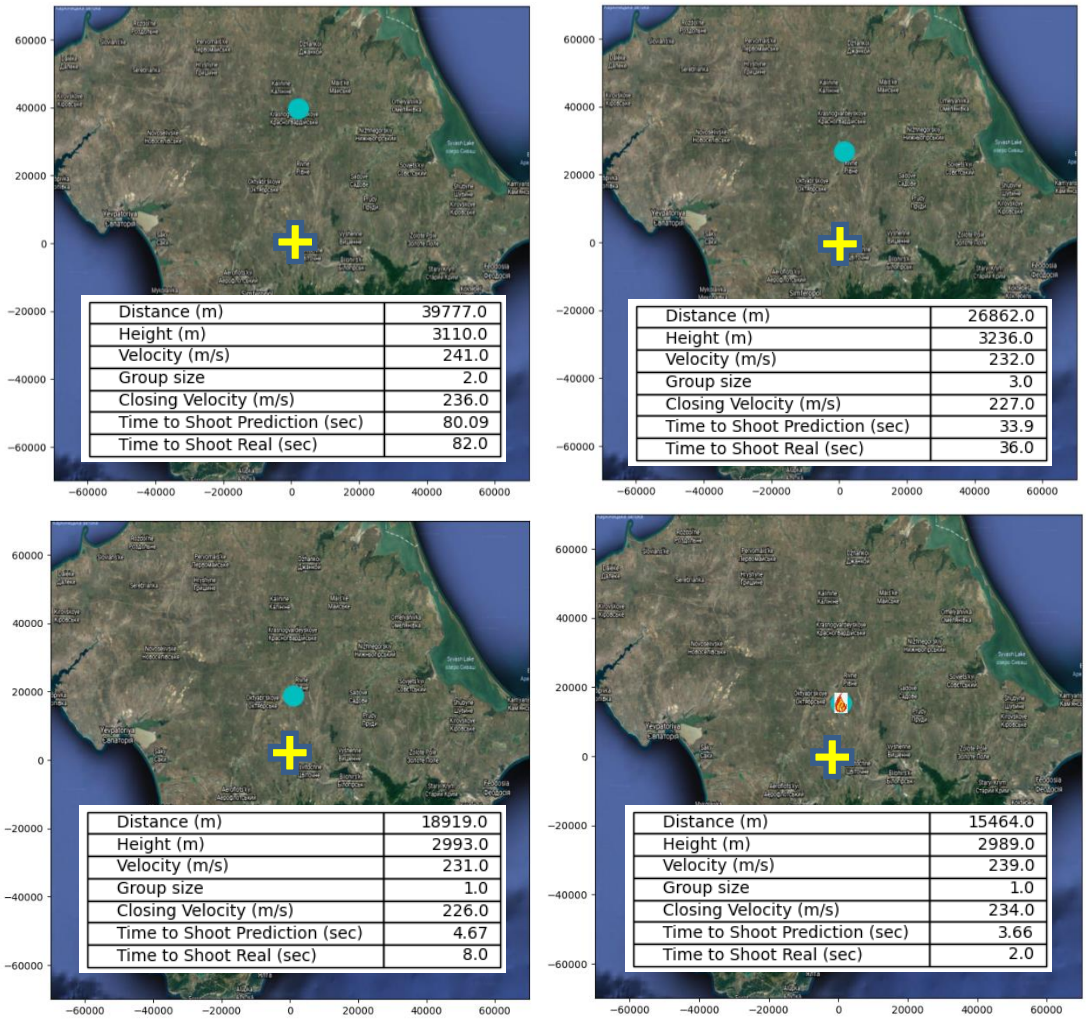

Figure 5: A visualization of the trajectory of an example target, the input variables for the model, and the output variable 'Time to Shoot Prediction' as predicted by the XGBoost model, compared to the 'Time to Shoot Real' as observed in the data.

Decision 2: **How many missiles should be fired at the target?**
For the second decision we found that a Random Forest Regressor model worked best, with a mean absolute error of 0.67. For a baseline reference, when always predicting the mean of the 'salvo' variable, a mean absolute error of 0.86 is produced.

The most important feature for this decision is target group size, which makes sense because one would fire more missiles when targeting a group of hostiles, than when targeting a single hostile. The second most important feature is altitude.

In the more explainable decision tree variant, the mean average error is only 0.02 higher (0.69), making decision trees a valid overall approach.

Decision 3: **Which weapon system should be used?**
For the last decision, a Random Forest multi-class classifier model worked best. This led to an accuracy of 54%. For baseline reference, when always predicting the most commonly used weapon system, which is the one closest to the target, an accuracy of 35% is achieved.

When explainability is important, one can opt for the decision tree model. However, the accuracy reduction of 12% (to 46%) is quite a significant loss of performance.

LIMITATIONS

This research with an AGBADS use-case aims to learn behavior models for three decisions that a NASAMS operator needs to make. To create these models, we collected data, cleaned, and processed this data, and trained several models. All of these steps may have imposed limitations which we will discuss below.

Even though the JPOW dataset logged a large amount of data, we can assume an actual NASAMS operator in the field has access to more and different information than we currently use for our models. For example, the weather can affect decisions, as well as a strategy given by a higher commander. It would be interesting to see how these can be modelled in further research. Furthermore, the data logged has a different perspective than the data the NASAMS operator has access to. For example: the data logged contains the position of the NASAMS and of the target. In the real world, the operator sees an estimation of the distance and relative position based on radar readings. This discrepancy between the perspective of the NASAMS operator and the logged data makes it quite hard to construct a model of the relevant features for the decisions of an NASAMS operator. It is possible that our current feature set over-simplifies the decision-making process of NASAMS operators.

With regards to the results of the models in this research, we achieved quite reasonable performances. This is considering the main targets of the NASAMS are namely helicopters and (unmanned) airplanes. Since these targets do not fly at extremely high speeds, a 30 second error when predicting time-to-shoot is reasonable. One thing to consider however, is that the targets in our dataset were all simulated. NASAMS operators may respond differently when dealing with real-life targets during an actual mission. Also, as mentioned before, we had to work with a discrepancy between logged data and actual operator experience, making it hard to create an accurate feature representation of the situation. For these reasons we believe that additional validation of our models is required in order sufficiently aid in decision support for NASAMS operators. This could be something to research and develop further in future research.

CONCLUSION & WAY AHEAD

This research aimed at learning behavior models of operators in military simulators based on observed behavior. We described the principle of so-called imitation learning that we used to learn behavior based on observations of players and present an overview of the existing examples of the current application of data-driven behavior modeling in the military domain. We apply imitation learning to the Army Ground-Based Air Defense System (AGBADS) operator behavior, for which real-world data (from a training event using simulated targets) was available. The data that is used in the learning process originates from the 2021 air defense exercise Joint Project Optic Windmill (JPOW).

After extensive pre-processing, multiple behavior models were constructed that can be used to provide fire control solutions (when to fire, the salvo size and which launcher to use). We create explainable models with reasonable performance using decision trees, and less explainable but more precise models. An identified challenge is the discrepancy between logged data and the perceived situation as experienced by the operator, making it hard to create an accurate feature representation of the situation.

Overall, we deem the conclusions from this research valuable for three use cases. Firstly, the explainable models can help NASAMS operators during their after-action review. By using such a model, insights into their actions can be gained. For example, an operator can learn that from a data-perspective, the most important feature when deciding when to shoot at an incoming target is distance. The models also can be used to compare a NASAMS behavior with the expected behavior. Any differences between these behaviors can lead to new insights in proper NASAMS

operator behavior. For example, an operator can find that he or she fires way sooner than expected by our models that are trained on historic NASAMS operator data.

The second use-case for using our behavior models is for simulating the behavior of an AGBADS. This allows an AGBADS to be integrated into a simulated training exercise. This is useful when other defense units need to practice a scenario that includes an AGBADS. For this use-case one can use the less explainable, but more precise models.

As third use-case, the created models can be integrated into decision support tools for AGBADS operators in combat situations. For example, could such a system provide advice on when to shoot, how large a salvo to shoot, and which launcher to use based on the current observations in the field. A practical application would require more research and data.

We quickly discovered that NASAMS operators make multiple decisions. We have decided to model three of these separately. However, it is probable that their behavior consists of more decisions than the ones we modelled. Furthermore, it is possible NASAMS operators use more or other input attributes than our system does. These are both interesting topics to explore in further research.

There is a follow-up exercise planned for JPOW in 2023. We are currently working on developing the results of this research into a tool that is scheduled to be used during the 2023 exercise and can aid in after-action review for AGBADS operators.

REFERENCES

Berthling-Hansen, G., Morch, E., Løvlid, R. A., & E., G. O. (2018). Automating Behaviour Tree Generation for Simulating Troop Movements (Poster). In G. L. Rogova, C. Lebiere, O. E. Gundersen, A. Salfinger, & K. Baclawski (Ed.), *2018 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)* (pp. 147-153). Piscataway, NJ, USA: IEEE. doi:10.1109/COGSIMA.2018.8423978

Boyd, J. (1987). *A discourse on winning and losing: Air University Library Document No. M-U 43947.* AL: Maxwell Air Force Base.

Bratman, M. (1987). *Intention, plans, and practical reason.* Cambridge, Mass: Harvard University Press .

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). New York, NY, USA: The Association for Computing Machinery. doi:10.1145/2939672.2939785

Clough, B. (2002). *Metrics, schmetrics! How the heck do you determine a UAV's autonomy anyway?* Wright-Patterson AFB, OH, USA,: Air Force Research Lab.

Georgeff, M., & Ingrand, F. (1989). Decision-making in an embedded reasoning system. In N. S. Sridharan (Ed.), *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)* (pp. 972–978). San Francisco, CA: Morgan Kaufmann Publishers, Incorporated.

Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence (XAI) program. *AI Magazine, 40*(2), 44 - 58. doi:10.1609/aimag.v40i2.2850

Gupta, A., Devin, C., Liu, Y., Abbeel, P., & Levine, S. (2017). Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning. *5th International Conference on Learning Representations, ICLR 2017.* OpenReview.net.

Heinze, C., Goss, S., Josefsson, T., Bennett, K., Waugh, S., Lloyd, I., . . . Oldfield, J. (2002). Interchanging agents and humans in military simulation. *AI Magazine, 23*(2), 37–47. doi:10.1609/aimag.v23i2.1639

Joint Air Power Competence Centre. (n.d.). Joint Project Optic Windmill. *https://www.japcc.org/joint-project-optic-windmill/*.

Nguyen, H., Garratt, M., Bui, L., & Abbass, H. (2017). Supervised deep actor network for imitation learning in a ground-air UAV-UGVs coordination task. *IEEE Symposium Series on Computational Intelligence (IEEE SSCI)* (pp. 1-8). Piscataway, NJ: IEEE. doi:10.1109/SSCI.2017.8285387

Nwana, H. S. (1996). Software Agents: An Overview. *The Knowledge Engineering Review, 11*(3), 205-244. doi:10.1017/S026988890000789X

Park, B., & Oh, H. (2020). Vision-Based Obstacle Avoidance for UAVs via Imitation Learning with Sequential Neural Networks. *International Journal of Aeronautical and Space Sciences, 21*(3), 768–779. doi:10.1007/s42405-020-00254-x

Plehn, M. (2000). *Control warfare: Inside the OODA loop (Master's Thesis).* Maxwell AFB, AL: Maxwell Airforce Base School of Advanced Airpower Studies.

Rao, A. S., & Georgeff, M. P. (1995). BDI Agents: From Theory to Practice. In V. Lesser (Ed.), *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95)* (pp. 312-319). San Francisco, USA: MIT Press.

Richardson, D. (2001). *Stealth Warplanes, Deception, Evasion, and Concealment in the Air.* St Paul, MN, USA: MBI Publishing Company.

Roessingh, J., Toubman, A., van Oijen, J., Poppinga, G., Hou, M., & Luotsinen, L. (2017). Machine Learning Techniques for Autonomous Agents in Military Simulation - Multum in Parvo. *IEEE Interactional Conference on Systems, Man and Cybernetics (SMC).* Piscataway, NJ, USA: IEEE. doi:10.1109/SMC.2017.8123163

Russel, S. J., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach; 4th Edition.* Englewood Cliffs, New Jersey: Prentice Hall.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences, 3*(6), 233-242. doi:10.1016/S1364-6613(99)01327-3

Simon, H. (1955). A behavioral model of rational choice. *The Quarterly Journal of Economics, 69*(1), 99-118. doi:10.2307/1884852

Teng, T.-H., Tan, A.-H., & Teow, L.-N. (2013). Adaptive computer-generated forces for simulator-based training. *Expert Systems with Applications, 40*(18), 7341–7353. doi:10.1016/j.eswa.2013.07.004

Torabi, F., Warnell, G., & Stone, P. (2019). Recent Advances in Imitation Learning from Observation. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (pp. 6325-6331). San Jose, CA: International Joint Conferences on Artificial Intelligence Organization. doi:10.24963/ijcai.2019/882

Tweedale, J., Ichalkaranje, N., Sioutis, C., Jarvis, B., Consoli, A., & Phillips-Wren, G. E. (2007). Innovations in multi-agent systems. *Journal of Network and Computer Applications, 30*(3), 1089 – 1115. doi:10.1016/j.jnca.2006.04.005

Zwarts, F. (2017). Grondgebonden lucht- en raketverdediging anno 2017: veelzijdig en complex. *MILITAIRE SPECTATOR, 186*(6), 269-281.