Learning Objective Agent Behavior using a Data-driven Modeling Approach

Farzad Kamrani and Linus J. Luotsinen
FOI - Swedish Defence Research Agency
SE-164 90 Stockholm, Sweden
Email: {farkam, linluo}@foi.se

Rikke Amilde Løvlid
FFI - Norwegian Defence Research Establishment
NO-2007 Kjeller, Norway
Email: rikke-amilde.lovlid@ffi.no

***Abstract*—This paper presents a data-driven approach towards the modeling of agent behaviors in a full-fledged, commercial off-the-shelf simulation milieu for tactical military training. The modeling approach employs machine learning to identify behavioral rules and patterns in data. Potential advantages of this approach are that it may improve modeling efficiency and, perhaps more importantly, increase the realism of the training simulator.**

In this work, we present an architecture outlining the main components of the data-driven behavior modeling approach. Using a prototype that implements the approach, we conduct and present results from an experiment targeting the learning of cooperative military movement tactics. It is shown that the prototype is capable of identifying the rules of the tactics. Moreover, it is shown that the agents are able to generalize such that the learned behavior can be applied in a new setting different from the one observed in the training data.

I. Introduction

Autonomous agents, which in the context of military simulations are often called Computer Generated Forces (CGFs), are frequently used to populate simulation models. Development of CGFs is influenced, to a great extent, by computer games programming, and as such, adopts a wide range of methods from *ad hoc* solutions and heuristics to modern Artificial Intelligence (AI) and Machine Learning (ML) techniques (e.g. see [1], [2]).

We are especially interested in applying ML to generate rules "automatically" for CGFs using observations and recorded data, which henceforth we refer to as Data-Driven Behavior Modeling (DDBM). The main goals of this approach are to improve the efficiency of the modeling process and to increase the creditability of the agents' behavior. The ambition is to replace (the main part of) the manual work of extracting behavioral rules from doctrines with a method based on acquiring, editing and labeling datasets that are processed by machine learning algorithms. This may not only improve the efficiency of the modeling process, but may also enhance the realism of the agent by creating objective and more human-like agent behaviors.

In earlier work, we have used toy problems to show the potential of the approach (e.g. see [3]). In the current work, we apply DDBM in real-world military applications to further investigate and gain insights into the following research questions:

- Is DDBM more efficient, with respect to cost and time, compared to the traditional modeling approach? That is, what can be gained by shifting the modeling work from manually hand-crafting behavioral rules to acquiring, creating, editing, and labeling or pre-processing datasets?
- Can DDBM be used to create more objective behavior models that imitate the behavior of their real-world counterparts? Can DDBM create behavioral models for agents that are more credible than those that are created using traditional methods?

To shed light on the stated research questions, we have developed a platform that integrates the main components of the DDBM approach. The platform is intended as an intuitive behavior modeling tool that can interact with a simulator and use recorded data and ML methods to create models for computer agents. For this study, we have chosen *Virtual Battle Space 3 (VBS3)*[1], which is a widespread military simulation application. The solution and platform, however, are not restricted to any specific simulator and VBS3 can be substituted by other simulation tools without any major modifications of the platform.

This paper is organized as follows. In Section II, we present background and related works. In Section III, we introduce DDBM and its main components. In Section IV, we briefly describe the implementation of our DDBM platform. In Section V, we present experimental results using the DDBM-prototype. Finally, conclusions and future works are presented in Section VI.

II. Background and Related Works

The promise of DDBM is that by using machine learning algorithms, a computer agent (*target agent*) can learn from the gathered data to act in a manner similar to a human or simulated agent (*original agent*) from which the data were collected. In this context, the learning strategies can be divided into three different categories; experiential, observational, and a hybrid approach that combines the first two methods. In experiential learning, the desired target agent learns and optimizes its behavior using a trial-and-error approach (similar to

[1] https://bisimulations.com/

human learning through practice). The target agent, usually in a simulator, executes a sequence of actions in different settings and learns from the outcome of taken actions. The learning is an iterative process, starting from a random solution. In each iteration, the performance of the agent is measured according to how well the task is performed. Some reinforcement mechanism ensures that those behaviors that show higher performance are strengthened, and finally, the one with the highest performance is presented as the agent's behavior [4].

It is important to bear in mind that in experiential learning, the performance rating cannot determine the correctness of the action or its similarity with human behavior at any given time, and only the result of the overall behavior of the agent for the entire activity is graded. Consequently, the agent may learn optimal behavior that is inappropriate. That is, the performance criteria are met, but the agent may act in a different way than what is expected or considered "natural". Some authors relate this phenomenon with *computational creativity* [5] and suggest that experiential learning might find creative solutions that are not found by humans [6].

Examples of published research outputs that discuss experiential learning can be found in [7], [8], [9], [10], however, not all of these references use the term experiential learning explicitly.

Observational learning takes another approach. Here, the idea is to extract the behavior model of the target agent by observing the behavior of another (possibly human) agent (so called original agent). The data collected from the original agent performing an activity, either in a simulation or in the real world, are used to train the target agent to act similarly when attempting to perform the same activity [11]. The terms *learning from demonstration* and *learning by imitation* are also used in literature, largely synonymous to observational learning.

In learning from demonstration, a human purposely demonstrates how to perform a given task in order to make the agent perform the same task, whereas in observational learning in general, the original agent does not have to be a willing or knowing participant [12]. The term learning from demonstration is often used in robotics [13], [14]. Learning from imitation (mimicking) can also be viewed as a special case of observational learning where the purpose of learning is to reproduce exactly the same actions.

In the machine learning community, the term learning by observation has a broader meaning and refers to the fact that in supervised learning, the training data is a set of collected observations. However, although in supervised learning the observed data are used to learn (e.g. in handwritten character recognition), in general, the data do not demonstrate how to perform a task, or teach any behavioral skills [15]. In this paper, observational learning is used in a more specific context, which refers to learning the behavior of an observed agent performing some activity.

In observational learning, the data used for training the target agent are collected by registering the state of a human (or another agent) performing an activity over time, and it is not necessarily clear where the activity starts and ends. Furthermore, the labels are not explicitly known as in traditional supervised learning, where the input data, their features and labels are explicitly defined [11].

Observational learning has been used in different domains using a variety of techniques (e.g. see [15], [16], [12], [17]). Learning team behavior from observation is discussed in [16], [17], where the presented method is semi-autonomous, and the collected data are manually processed to identify domain specific contexts representing different states of the observed behavior. This is necessary to suppress the amount of the training data to only those significant within the context.

In order to prove the concept and verify the applicability of the methods, in previous works, we have used DDBM approach to solve different small-scale problems. In [3], observational learning approach was used to imitate the behavior of multiple collaborative agents performing tasks of increasing complexity. In this work, the goal was to learn the behavior of a team of players, which exercised passing a puck in a hockey-player simulator environment. In the first experiment, four players located at the corners of a square, passed the puck in a clockwise manner with no opponent player present. In the second exercise, with the same setting, one of the players randomly passed the puck diagonally approximately 50% of the time. In the third exercise, five players kept the puck within the team by avoiding passing to team members who were covered or intercepted by an opponent player. For each exercise, synthetic datasets that were consistent with the specified behavior were created. The datasets were applied to our DDBM prototype to learn the positioning and passing behavior of each agent using the back-propagation algorithm [18] for standard neural networks and the ID3 algorithm [19] for decision trees, respectively. Presented results showed that it was possible to learn the desired behaviors for all exercises using relatively small training datasets and in short time interval (less than 10 minutes). The behavior of each agent was verified by visualizing the rules embedded in the generated decision trees. Collaborative behavior of all agents was verified using the hockey-simulator capable of visualizing the movement of the players and the puck given the learned behavior model.

In [6], we used a predator-prey simulation to test the ideas presented in the experiential learning strategy. In this simulation, the predator was represented by a wolf, which was hunting a herd of sheep controlled by a modified version [20] of a flocking algorithm [21]. The goal was to optimize the behavior of the predator agent so that it annihilated the herd of sheep as efficiently as possible. Unlike observational learning, in experiential learning, the simulator was integrated within the learning or evaluation phase. In this experiment, the DDBM prototype used Genetic Programming (GP) [22] to generate the wolf behavior. The GP was configured to use a population size of 1000 individuals (programs) and terminated its search after 10 generations. The result of the GP was compared with a human-programmed code, representing an intuitive behavior where the wolf continuously turned and struck towards the center of the herd. While the human-programmed solution killed all the sheep in 1700 game ticks, the DDBM generated code required only 1000 cycles for killing all sheep, thus outperforming the human-generated code [6].

Although the findings of the aforementioned literature show that the DDBM approach can be used for modeling the behavior of agents, most of the available results involve only toy problems, and to the best of our knowledge, there is no previous work applying the DDBM approach in real-world military settings. The main contribution of this paper is that we have applied the DDBM method to generate behavior models for Computer Generated Forces for practical use in a full-fledged training simulator such as VBS3.

III. Data-driven Behavior Modeling

In a broad categorization, the learning component of DDBM can be divided into the following three types:

- *Observational learning*, in which a target agent learns from data collected over how an original agent behaves in a similar situation (in live or simulated scenarios).
- *Experiential learning*, in which the target agent learns without having any role model to learn from and learns in a trial-and-error manner, starting usually from a random behavior.
- *Hybrid approach*, which is a combination of the first two approaches, i.e. the target agent first learns from observing an original agent and then improves the learned model by experiential learning.

Since the focus of this paper is observational learning, we will briefly explain different components of this method.

A. Observational Learning

In observational learning, the goal is to develop the behavior model of an agent by observing the behavior of another agent whose behavior should be learned while it performs the same activity under similar condition [11], [3]. However, we interpret the term “observation” in a less literal sense and use it to describe all relevant data that are collected and saved in a persistent database, while the original agent performs the desired task. The data can be collected from real-world exercises or while executing the scenario in a simulator, where agents are controlled by human players (or by scripts).

Fig. 1, illustrates the entire process of observational learning and how it is employed. The left part of the figure illustrates the learning steps of the observational learning DDBM, which starts with gathering data in a persistent database. The data include changes of various states of all agents involved in the scenario including their time stamps. Moreover, environment parameters that might change during the scenario execution are recorded. A feature extraction module then extracts the set of relevant features and corresponding labels for each agent. An appropriate machine learning algorithm deduces a model (e.g. a decision tree) using each feature and the corresponding label. The collection of these models constitutes the behavior model of each CGF and is the input to the target simulator. Theoretically, the learning simulator and the target simulator do not need to be the same, but using the same simulator on both sides, considerably facilitates the process.

The right part of Fig. 1 depicts how the behavior models are employed in a simulator. As the simulator starts, the states of the CGFs over a predefined time span (so called sliding window) are collected in an in-memory database. It is required that the data over a sufficient duration of time (sliding window) are available, since some states may depend not only on the current state of the agent and other agents, but also on the history of the events.

A feature extraction module, which corresponds exactly to the feature extraction module used during the learning phase, extracts the relevant features for each CGF. In each tick of the simulator, or at predefined decision points, a decision maker module uses the behavior models to deduce the desired state for each feature of the CGFs. If required, a command is issued to the simulator to establish the desired state.

IV. Implementation

We have developed a software platform that includes all required elements to develop behavior models using the DDBM approach, necessary modules to apply the models to CGFs inside a target simulator (e.g. VBS3), and interfaces to interact with the target simulator. The platform is a highly modularized, loosely coupled and plug-in based application that integrates different components for data-acquisition, visualization, feature extraction, machine learning algorithms, and interfaces to communicate with and control the target simulator. It is intended to be an intuitive and easy to use tool, with the ultimate goal of functioning and creating behavior models for CGFs without involvement of any software engineers, so that it can readily be operated by subject matter experts.

The platform is a further development and refinement of the software presented in [3]. Added capabilities in the new version include, but are not limited to, interfaces to the target simulator and extending the feature extraction to a large number of new features.

V. Experiments

We have used our platform to apply the DDBM approach for modeling the behavior of CGFs in VBS3, which is a simulation tool widely used within defense sector across many countries.

A. Virtual Battlespace 3

VBS3 (developed by *Bohemia Interactive Simulations*)is a computer game-based software system, which is used for soldier training by a majority of NATO partners. It is a tactical first-person military training simulation program and provides a three-dimensional visually rich environment with flexible scenario and terrain options. VBS3 supports more than 100 combined arms training tasks, from the individual soldier to company level. Examples of tasks in VBS3 are entering and clearing a building, conducting an attack, conducting a defense, conducting an artillery raid, and conducting route reconnaissance.

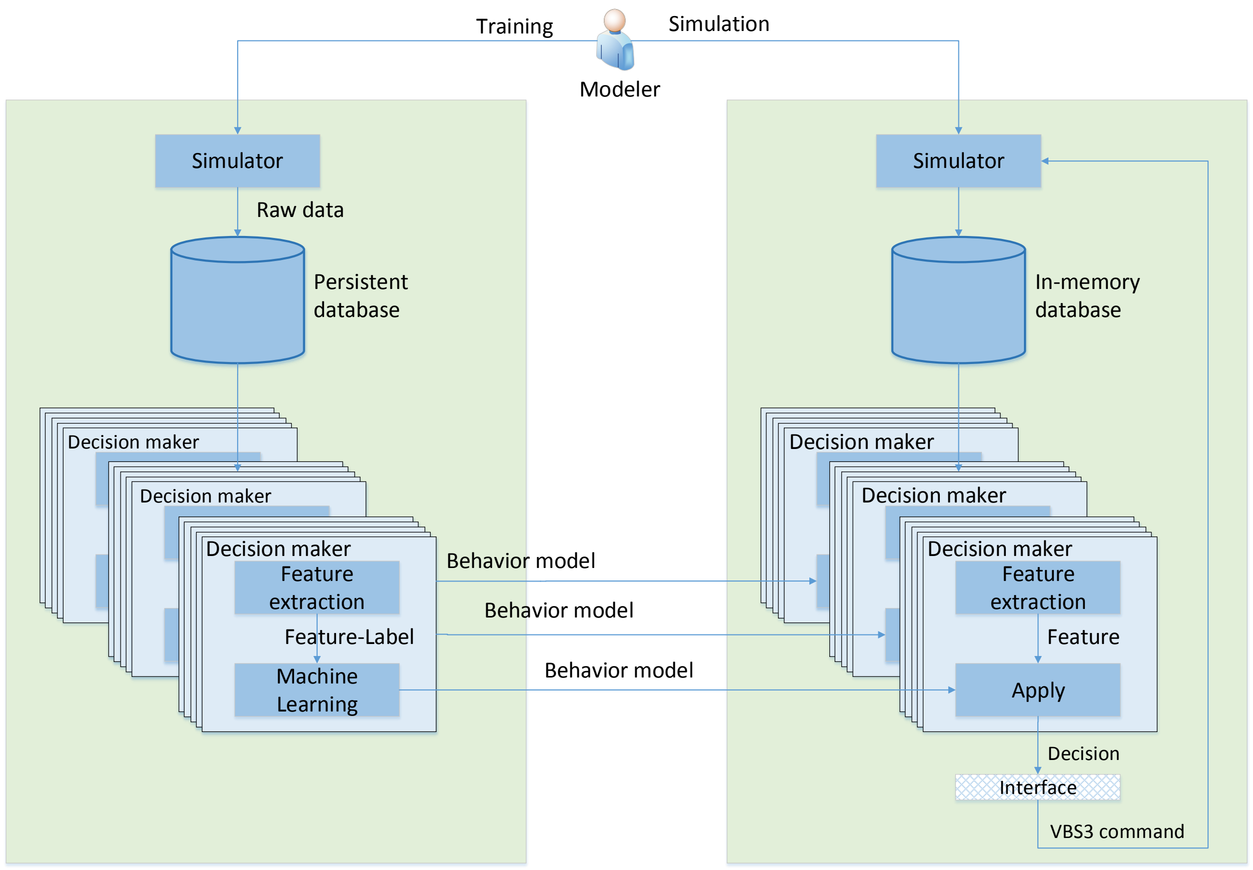

Fig. 1. An overview of the main components of the observational learning DDBM approach and its interaction with the target simulator. In the left part of the figure, three CGFs each having four decision maker modules are trained. For each CGF, four different models are derived (from which only one is shown to avoid cluttering). In the right hand part of the figure, the CGFs use the derived models in the target simulator.

Apart from the player and playable entities (i.e. entities controlled by possibly other players connected over a network), other units and entities are CGFs that act autonomously according to some predefined rules. The CGFs give the possibility to configure and train larger scenarios that would otherwise be impossible if all units had to be controlled by human players. VBS3 provides several means and tools, such as scripting tools, Real-time editor (RTE) to influence the scenario during training, and VBS TACTICS that enables users to configure doctrine-based orders of battle and planning mission.

In the following, we outline a case scenario, *bounding overwatch*, for which we have applied the DDBM approach to learn collaborative CGF behaviors from observations using the VBS3 simulator.

B. Bounding Overwatch

Bounding overwatch is a military movement tactic of coordinated units used when enemy contact is likely. The characteristic of this tactic is prioritizing security over speed by giving an overwatch unit the possibility of suppressing enemy fire if required. In bounding overwatch, two units have different roles; while the bounding unit takes the lead towards the target point and moves forward, it is protected by the overwatch unit (see Fig. 2). There are two types of bounding overwatch: (i) *alternating bounding*, in which the bounding unit is protected by the halted overwatch unit, and the two swap their roles when the bounding unit has moved forward to the overwatch unit's protection range, and (ii) *successive bounds*, in which the roles of bounding unit and overwatch unit remain the same throughout the movement [23].

In order to be credible, CGFs that perform the bounding overwatch tactic should consider a wide range of factors, including distance to the enemy, distance to own units, direction and intensity of the enemy fire, own weapon type, and the amount of ammunition. Writing a program that considers all these features in an appropriate way is a time-consuming and non-trivial endeavor.

Fig. 3 illustrates a model of the bounding overwatch tactic represented graphically by a *behavior tree (BT)*. A behavior tree is a graphical representation for execution of actions used in computer game industry for modeling the behavior of non-player characters (NPCs) [24], [25]. In recent years, BTs have also been used in robotics and control systems for multi-robot systems [26], [27]. The strengths of BTs are in their modularity and ability to compose complex behaviors by using simple

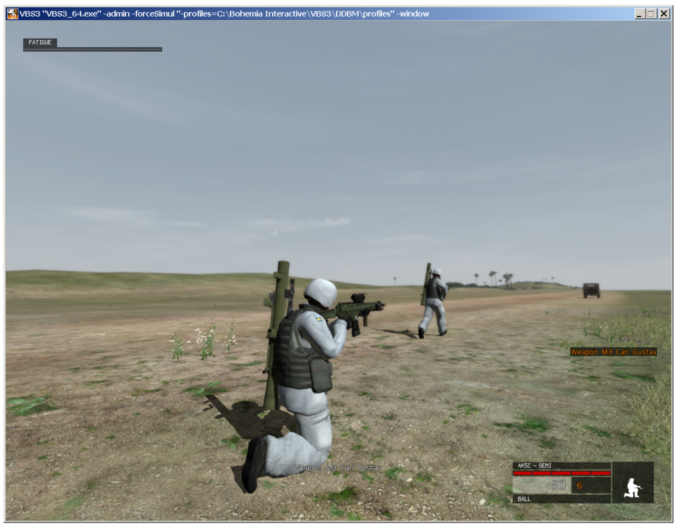

Fig. 2. Two entities in VBS3 performing bounding overwatch. While the entity on the left is overwatching, the entity on the right is moving forward towards the target.

tasks.

BT is a directed tree consisting of a single root, *control flow* nodes and *execution* leaves. Control flow nodes are either *sequences* or *selectors* and execution leaves are either *conditions* or *tasks*. In Fig. 3, the root is marked by "ϕ", sequence nodes are marked by "$\rightarrow$", selector nodes are marked by "?", condition leaves are the ovals and tasks are rectangles.

The execution of the BT starts from the root, which sends *ticks* at fixed intervals to its only child. Ticks are discrete signals that traverse through the tree in a depth-first manner (from left to right), changing the state of a node. Each node is in one of the states *not executed*, *running*, *success* or *failure*. A node's type determines how its children are traversed and how its state is changed. A sequence node executes its children sequentially from left to right by sending ticks to them, until all its children succeed. If one of the children returns with failure or running state, the sequence node will return immediately with the same state, otherwise after all children have returned with success state, the sequence node will also return with success state.

A selector attempts to execute its children sequentially from left to right by sending ticks to them. If all children return with failure state, then the selector node will also return with a failure state, otherwise if any of the children returns with success or running state, the selector node will return immediately with the same state.

A condition node returns with success if the predication is true, otherwise it returns with failure.

A task node is the interface of the BT with the environment and triggers actions such as movements and reading sensors. Execution of a task node means that its state sets to running and the corresponding action is scheduled to be performed asynchronously. The node stays in the running state until the activity is finished. The state changes to success or failure depending on the result of the activity.

There are some additional node types, which are extension or modification of the four types of nodes described here. However, they are considered to be out of the scope of this paper, and are not discussed.

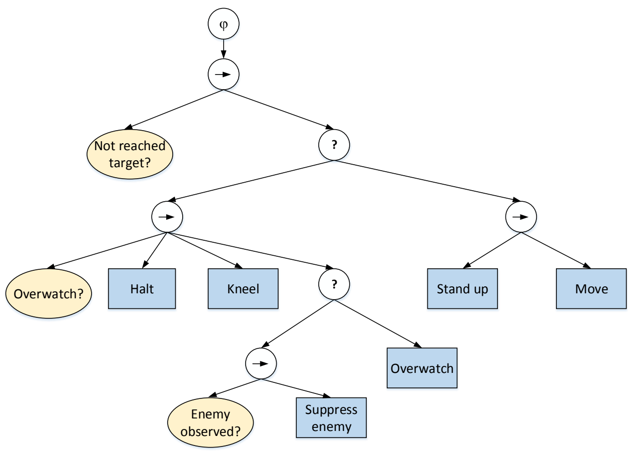

Fig. 3. Example of a behavior tree model for a unit conducting the bounding overwatch task.

C. Experiment Model and Settings

We make some assumptions and create a simplified model of the bounding overwatch problem. First, we assume that the group consists of only two CGFs; one bounding and one overwatch entity. Moreover, to keep the model simple, we ignore the *suppress enemy* behavior (represented by the sub-tree in the lower center in Fig. 3) in this paper, and leave it to future work. It should also be clarified that in this work, the DDBM is used to derive the models of the low-level behaviors defined as the leaves of the behavior tree, and the structure of the tree is hand-crafted.

In the data gathering phase, we use VBS3 and a script that controls the group according to one of the two types of bounding overwatch (either alternating bounding or successive bounds). As a part of the input to this script, we provide a set of waypoints, which define the path of the entities. Among others, data over the following states (including timestamps) for each agent are collected and saved persistently in a file, periodically.

- The position of the agent (a continuous value).
- The orientation of the agent (a continuous value).
- The velocity of the agent (a continuous value).
- The agent's stance, which is one of the three values *prone*, *kneeling*, or *standing* position.

Only this part of the saved data is used by the machine-learning algorithms, and other fields of data are just collected to be used when new features and capabilities are added to the model.

The raw data as it is saved cannot directly be used by the machine-learning algorithms, and we need to refine and process the data to extract generalized features of the state of the agents. This is necessary in order to help the machine-learning algorithms to learn from even a restricted amount of data. Using the allegory of learning by observation, the feature extraction process resembles "teaching" the learner instead of letting her "discover" a situation by observing it frequently. For instance, a huge amount of raw data is required for an agent to discover that it should avoid all positions in which it is in the gunshot range of an adverse agent. However, by extracting the distance between the two agents, we can "teach" the agent the concept of distance, which is a prerequisite for learning how to avoid hazardous situations.

For the current scenario, the following features for each agent have been extracted:

- The distance feature, that is, the relative distance of the agent to the other agent, measured in the number of waypoints.
- The stance feature, that is, the stance of the agent.
- Speed feature, that is, the velocity of the agent.
- Orientation feature, that is, the angle between direction of the overwatch agent's weapon and the line passing through the agent and the next waypoint.
- Waypoint location feature, that is, which waypoint the agent is located at, if any.
- Waypoint destination feature, that is, towards which waypoint the agent is heading.
- Moving feature, that is, whether the agent is moving or not.
- Turning feature, that is, how much the overwatch agent should turn to hold an appropriate direction.

Based on the features, for each of the two involved agents, four models are created that are later used to make decision about the behavior of the agent. It needs to be clarified that since the gathered data for the two agents are not the same, the derived models are not identical either (although they are of the same type). The developed models are as follows:

- The distance and moving features are used to derive a decision tree model, which is used to make decision about whether the agent should move or not.
- The distance and stance features are used to derive a decision tree model, which is used to determine the stance of the agent.
- The speed, orientation and turning feature are used to derive an Artificial Neural Network (ANN) model, which is used to make decision about how much the overwatch agent should turn to correct the direction of its weapon.
- The speed, location and destination features are used to derive a decision tree model, which is used to make decision about the waypoint the agent should move towards.

Unfortunately, there is no single silver bullet method for choosing features or type of the models and it is a matter of trial-and-error experimentation to select adequate features and find models that work appropriately.

We use the open-source RapidMiner [28] library, which is incorporated in our platform, to create the decision tree and ANN models.

In the application phase, the models are used to answer queries about how each agent should behave according to the created behavior models; that is whether the agent should move or not, which stance should it take, towards which waypoint should it move, and how much should it turn to correct the direction of its weapon. A series of waypoints (different from the waypoints used in the training phase) are provided to the DDBM platform, which define a new path for the agents. The DDBM platform initiates the agents and in a similar manner to the data collection phase, gathers data (states of the agents in the VBS3 simulator) over a predefined time span (sliding window). Whenever one of the agents arrives at a waypoint (a so-called decision point), the same feature extraction functions are used to extract the known features and make query about the unknown features (moving, stance, destination and turning).

D. Results

The test scenarios for both alternating bounding and successive bounds show that the method works for both tactics, that is the model learns the behavior from the training data generated by a single run and is capable to repeat the same behavior without any errors.

It should be emphasized that the learned behavior is independent of the path for which the agents have learned the tactic. Once the agents learn a tactic, they can generalize the behavior to different paths defined by completely different waypoints, even in other directions.

Although in this experiment the studied tactics are rather simple and elementary, the result is very promising and show that the approach is applicable and can be implemented on more complex tactics and behavior, which are part of the future work of this project.

E. Discussion

The training tactics, which our system is tested against, are rather simple and consist of a set of states that are clearly distinct, observable and not overlapping (stance, moving, direction and destination). This is not by accident, but a deliberate decision to start testing the approach for more simple tactics on a sophisticated simulator, and incrementally add features that are more complex. The reason for this decision has been twofold: first, to prove the concept and show that the DDBM approach is working even when using real-world simulation tools, second, to master the difficulties in interactions with a sophisticated and proprietary software with no source code available.

VI. Conclusions and Future Work

In this work, we have presented a DDBM platform for modeling agent behavior in a commercial off-the-shelf simulation-based military training software, VBS3. The platform is customized to interact with VBS3 and is used both to generate training data and to develop behavior models. The DDBM concept is tested for two military movement tactics, alternating bounding and successive bounds for a group consisting of two entities. First, data is generated for these two tactics using a script interacting with VBS3. The generated data is used as training data. VBS3 is used to determine if a group with the same configuration as the original scenario can learn from the observed data and mimic the same behavior. The conducted experiments clearly demonstrate that the entities show the same behavior, using the data generated by a single run of the script.

Although the concept of DDBM has been previously tested on toy problems and simplistic models, to the best of our knowledge, this is the first time the concept is successfully applied on a real-world and sophisticated simulation tool as VBS3, something that we consider as the main contribution of this paper.

The DDBM approach appears promising, however, as previously pointed out, the trained tactics used in this work are relatively uncomplicated, and there are still much more interesting and involved problems to be tested and evaluated.

Moreover, in order to use this approach in practice, it is required to address several challenges, from which we list the following:

- Complex behavior models, including those with overlapping states.
- Data problems such as insufficient, incomplete and noisy data.
- Real-time simulation problems caused by advanced feature extraction functions (e.g. terrain analysis, route planning).
- Verification and validation problems related to black-box representations such as neural networks that are difficult to visualize and analyze by humans experts.
- The need for intuitive and easy-to-use DDBM authoring tools capable of visualizing, editing and processing datasets acquired synthetically or from live exercises.

Acknowledgment

The work presented in this paper is the result of a collaborative effort by the Swedish Defence Research Agency (FOI) and the Norwegian Defence Research Establishment (FFI). The work was supported by the FOI research project "Synthetic Actors", which is funded by the R&D programme of the Swedish Armed Forces, and by the FFI research project "LVC simulation for training, exercises and experiments".

References

[1] I. Millington and J. Funge, *Artificial Intelligence for Games*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

[2] B. Geisler, "Integrated machine learning for behavior modeling in video games," in *Challenges in game artificial intelligence: papers from the 2004 AAAI workshop*, D. Fu, S. Henke, and J. Orkin, Eds. Menlo Park, CA, USA: AAAI Press, 2004, pp. 54–62.

[3] L. J. Luotsinen and R. A. Løvlid, "Data-driven behavior modeling for computer generated forces," in *NATO Modelling and Simulation Group Symp. M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence (MSG-133)*, 2015, pp. 1–13.

[4] G. Stein and A. J. Gonzalez, "Building high-performing human-like tactical agents through observation and experience," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2011, pp. 792–804.

[5] J. McCormack and M. d'Inverno, *Computers and creativity*. Springer, 2012.

[6] L. J. Luotsinen, F. Kamrani, P. Hammar, M. Jändel, and R. A. Løvlid, "Evolved creative intelligence for computer generated forces," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2016.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 1998.

[8] D. Aihe and A. Gonzalez, "Context-driven reinforcement learning," in *Proceedings of the Second Swedish-American Workshop on Modeling and Simulation*, Cocoa Beach, FL, 2004.

[9] K. Merrick and M. L. Maher, "Motivated reinforcement learning for non-player characters in persistent computer game worlds," in *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ser. ACE '06. New York, NY, USA: ACM, 2006.

[10] T.-H. Teng, A.-H. Tan, and L.-N. Teow, "Adaptive computer-generated forces for simulator-based training," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7341–7353, 2013.

[11] G. Stein, A. J. Gonzalez, and C. Barham, "Combining NEAT and PSO for learning tactical human behavior," *Neural Computing and Applications*, pp. 1–18, 2014.

[12] S. Ontanon, J. L. Montana, and A. Gonzalez, "Towards a unified framework for learning from observation," in *IJCAI Workshop on Agent Learning Interactively from Human Teachers*, 2011.

[13] C. G. Atkeson and S. Schaal, "Learning tasks from a single demonstration," in *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, 1997, pp. 1706–1712.

[14] D. C. Bentivegna and C. G. Atkeson, "Learning from observation using primitives," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2001, pp. 1988–1993.

[15] H. K. G. Fernlund, "Evolving models from observed human performance," Ph.D. dissertation, University of Central Florida, 2004.

[16] C. L. Johnson and A. J. Gonzalez, "Learning collaborative behavior by observation," in *ICMLA*, S. Draghici, T. M. Khoshgoftaar, V. Palade, W. Pedrycz, M. A. Wani, and X. Zhu, Eds. IEEE Computer Society, 2010, pp. 99–104.

[17] ——, "Learning collaborative team behavior from observation," *Expert Systems with Applications*, vol. 41, no. 5, pp. 2316–2328, 2014.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, pp. 696–699.

[19] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[20] M. Barksten and D. Rydberg, "Extending Reynolds' flocking model to a simulation of sheep in the presence of a predator," Bachelor's thesis, School of computer science and communication, KTH - Royal Institute of Technology, Stockholm, Sweden, 2013.

[21] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[22] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[23] L. J. Luotsinen, "Recognizing teamwork activity in observations of embodied agents," Ph.D. dissertation, University of Central Florida, Orlando, FL, USA, 2007.

[24] D. Isla, "Handling complexity in the Halo 2 AI," in *Proceedings of the Game Developers Conference*, 2005.

[25] C. Lim, R. Baumgarten, and S. Colton, "Evolving behaviour trees for the commercial game DEFCON," in *Applications of Evolutionary Computation*. Springer, 2010, pp. 100–110.

[26] P. Ögren, "Increasing modularity of UAV control systems using computer game behavior trees," in *AIAA Guidance, Navigation and Control Conference*, 2012.

[27] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ögren, "Towards a unified behavior trees framework for robot control," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Robotics and Automation Society, 2014, pp. 5420–5427.

[28] M. Hofmann and R. Klinkenberg, *RapidMiner: Data Mining Use Cases and Business Analytics Applications*, ser. Chapman & Hall/CRC Data Mining and Knowledge Discovery. CRC Press, 2013.