

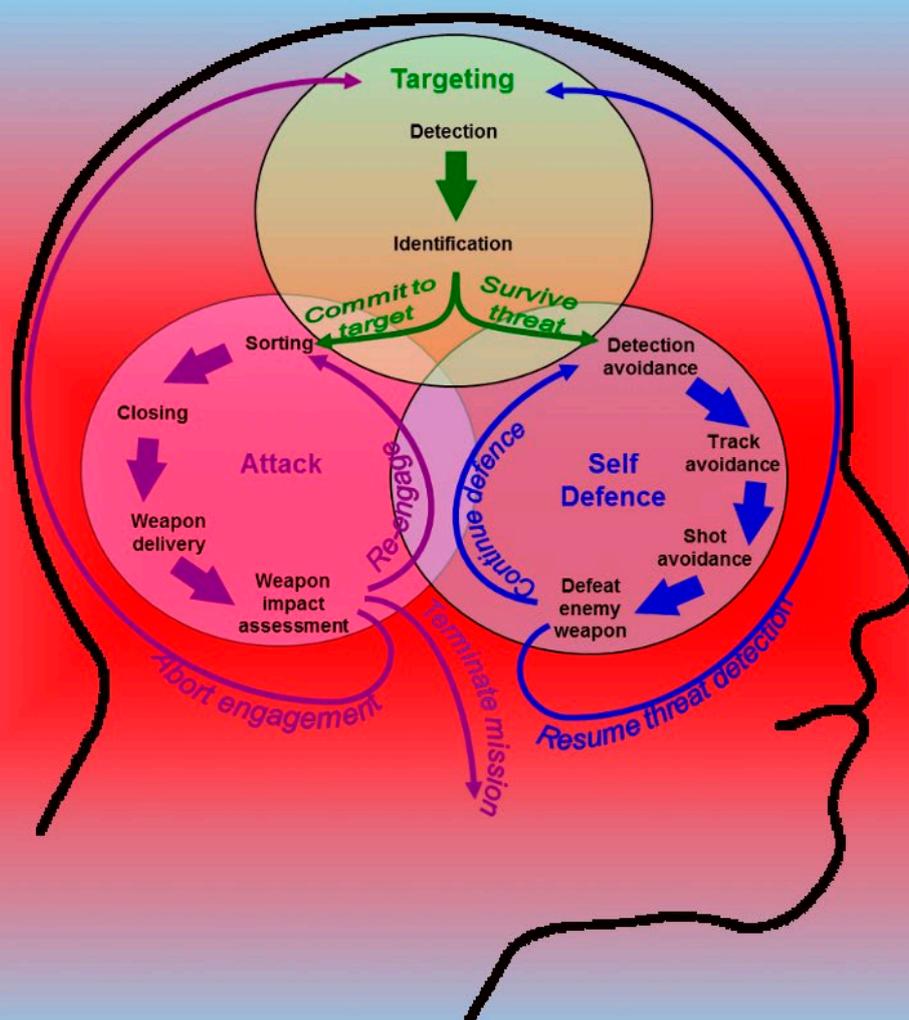
Modeling CGF Behavior with Machine Learning Techniques

Requirements and Future Directions

Customer

National Aerospace Laboratory NLR

NLR-TP-2015-426 - February 2016



National Aerospace Laboratory NLR

Anthony Fokkerweg 2

1059 CM Amsterdam

The Netherlands

Tel +31 (0)88 511 3113

www.nlr.nl

EXECUTIVE SUMMARY

Modeling CGF Behavior with Machine Learning Techniques

Requirements and Future Directions



Problem area

Computer Generated Forces (CGF) packages are widely used in modelling and simulation for training purposes. Conventional CGF packages often include artificial intelligence (AI) interfaces, with which the end-user (scenario developer, modeller or sometimes the instructor) defines CGF behaviours.

Machine Learning (ML) techniques can be beneficial to the behaviour modelling process, yet such techniques seem to be underused and perhaps underappreciated.

This paper aims at bridging the gap between academia and the military when it comes to ML and AI. Military user requirements and how they can be addressed by ML techniques are highlighted with the focus on the added ML value to CGF packages.

Report no.

NLR-TP-2015-426

Author(s)

A. Toubman
J.J.M. Roessingh
G. Poppinga
M. Hou
L. Luotsinen
R.A. Løvlid
C. Meyer
R.J. Rijken
M. Turcaník

Report classification

UNCLASSIFIED

Date

February 2016

Knowledge area(s)

Training, Mission Simulation and
Operator Performance

Descriptor(s)

Computer Generated Forces
Machine Learning
Artificial Intelligence
Simulation
Training

Description of work

AI and ML specialists from the Netherlands, Sweden, France, Canada, Norway, and Slovakia work together in NATO Research Task Group (RTG) IST-121 RTG-060 'Machine Learning Techniques for Autonomous Computer Generated Entities' from 2014 to 2016. For NLR and the Royal Netherlands Air Force, the international cooperation was sought as part of the 'Smart Bandits' project. The latter project aimed to develop intelligent and adaptable behavioral models to be used in tactical training simulations for air-to-air combat training. The NATO RTG that was proposed by the Netherlands under the IST-panel has broader objectives: ML applicable to all kind of simulation applications. The group started in 2013 with a review of-existing and relevant Machine Learning techniques and their (potential) application to autonomous behaviour of CGFs. This paper is based on that work.

Results and conclusions

ML techniques can be beneficial for modelling CGF behaviours, and guidance needs to be developed for end users. It is recommended that the developers of COTS/MOTS CGF packages start incorporating ML techniques, thereby providing the capability to create CGFs that possess richer behaviours in complex environments and are better tailored to the knowledge and skills of the trainee. This paper is essentially a call for further development of CGFs that are capable of learning, using the proposed architecture in this paper.

Applicability

Military end-users of ML are mainly scenario developers, modellers and instructors. For the modeller, who develops algorithms for CGFs, to generate realistic human behaviour, a set of algorithmic requirements have been drafted in this paper. These algorithmic requirements, which are on the one hand computational requirements, and on the other hand functional requirements, support the modeller in his/her choice of ML technique. These requirements can also be used by developers of CGF packages to incorporate ML techniques in their packages or in the AI plug-ins that are used in conjunctions with these packages. A modular, scalable architecture that is conceptualized in this paper can be applied for the purpose of integrating ML techniques in CGF packages/ AI plug-ins. Finally, four basic requirements are provided for a User Interface (UI) that allows users to employ CGFs with ML: Different end-users (scenario developer, modeller or instructor) should be able to use such UI for its own purposes.

National Aerospace Laboratory NLR

Anthony Fokkerweg 2, 1059 CM Amsterdam,
P.O. Box 90502, 1006 BM Amsterdam, The Netherlands
Telephone +31 (0)88 511 31 13, Fax +31 (0)88 511 32 10, www.nlr.nl



Modeling CGF Behavior with Machine Learning Techniques

Requirements and Future Directions

A. Toubman, J.J.M. Roessingh, G. Poppinga, M. Hou¹,
L. Luotsinen², R.A. Løvlid³, C. Meyer⁴, R.J. Rijken⁵ and
M. Turcaník⁶

¹ Defense Research and Development Canada

² Swedish Defence Research Agency (FOI)

³ Norwegian Defence Research Establishment (FFI)

⁴ Thales

⁵ Min. van Defensie

⁶ Armed Forces Academy, Slovakia

Customer

National Aerospace Laboratory NLR

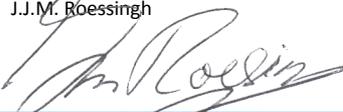
February 2016

This report is based on a presentation held at the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2015, Orlando, FL, USA, November 30 – December 3, 2015.

*The contents of this report may be cited on condition that full credit is given to NLR and the authors.
This publication has been refereed by the Advisory Committee AEROSPACE OPERATIONS.*

Customer National Aerospace Laboratory NLR
Contract number - - -
Owner National Aerospace Laboratory NLR
Division NLR Air Transport
Distribution Unlimited
Classification of title Unclassified
Date February 2016

Approved by:

Author J.J.M. Boessingh 	Reviewer J. van Oijen 	Managing department H.G.M. Bohnen 
Date: 27-OCT-2015	Date: 01-NOV-2015	Date: 08-NOV-2015

Summary¹

Commercial/Military-Off-The-Shelf (COTS/MOTS) Computer Generated Forces (CGF) packages are widely used in modelling and simulation for training purposes. Conventional CGF packages often include artificial intelligence (AI) interfaces, with which the end user defines CGF behaviors. We believe Machine Learning (ML) techniques can be beneficial to the behavior modelling process, yet such techniques seem to be underused and perhaps underappreciated. This paper aims at bridging the gap between users in academia and the military/industry at a high level when it comes to ML and AI. Also, specific user requirements and how they can be addressed by ML techniques are highlighted with the focus on the added ML value to CGF packages. The paper is based on the work of the NATO Research Task Group IST-121 RTG-060 ‘Machine Learning Techniques for Autonomous Computer Generated Entities’.

¹ The language standard for I/ITSEC papers is US-English. These US-English spelling rules are maintained throughout the remainder of this Technical Publication (TP). The executive summary and the summary at the beginning of this TP is not part of the original paper and uses UK-English spelling.



Content

Abbreviations	6
1 Introduction	7
2 State-of-the-art of AI in CGF packag	9
3 The added value of machine learning	13
4 Requirements and desired future capabilities	14
4.1 Identifying end-users	14
4.2 Algorithmic Requirements	15
5 Realization	17
5.1 ML Techniques	17
5.2 Architecture	18
5.3 User Interface	19
6 Conclusion	21
7 References	23

Abbreviations

Acronym	Description
AI	Artificial Intelligence
C2	Command & Control
CGF	Computer Generated Forces
COTS	Commercial Off The Shelf
DNW	German-Dutch Wind tunnels
IST	Information Systems Technology
ML	Machine Learning
MOTS	Military Off The Shelf
NATO	North Atlantic Treaty Organization
NLR	National Aerospace Laboratory NLR
PEO-STRI	US Army Program Executive Office for Simulation, Training and Instrumentation
R&D	Research and Development
RTG	Research Task Group
SAF	Semi Automated Forces
SF	Synthetic Forces
SME	Subject Matter Expert
UI	User Interface
VBS	Virtual Battle Space

1 INTRODUCTION

A trainee often has to interact with other agents in a training simulation. Virtual agents or so called Computer Generated Forces (CGFs) inhabit training simulations to make the simulations more realistic. These CGFs usually carry out the role of ally (e.g. squad mate), adversary (e.g. enemy fighter jet), or some neutral role (e.g. civilian traffic).

The term CGF is commonly substituted with terms such as Synthetic Forces (SFs) or Semi-Autonomous Forces (SAFs) and other terms, depending on the specific application, whereas CGF is the more general term. CGFs can be presented as individual elements (for example, individual soldiers or aircraft) or as 'aggregates', that are groups (for example a platoon) of individual elements that behave as a cohesive element in the simulated environment. The creation of CGFs has generally two design aspects. The first is their physical presence, as apparent from their size, shape, maneuverability, weapons, sensors, etc. The second design aspect is to make CGFs actually act out a role and concerns the representation of their behavior, typically based on mimicking the doctrines, strategies, tactics, rules-of-engagement, techniques, procedures and other abstract features of their real-world counterparts.

However, modeling behavior that is credible and sufficiently representative for the role remains a great challenge. Behavior definitions and implementation require time and expert knowledge that is not always available. Furthermore, once a behavior model is implemented, the behavior is usually set in stone and has to be manually altered to provide variations and different levels of sophistication or challenge.

Machine learning (ML) techniques may provide a solution, through the automatic generation of behavior models. While the use of artificial intelligence (AI) techniques in CGF packages has already been documented (see e.g. Abdellaoui, Taylor & Parkinson 2009), previous work did not thoroughly discuss the use of ML techniques to generate behavior of CGFs. In light of the current revival of ML, illustrated by examples such as IBM's Watson and Google DeepMind, we think that it is the time to focus on the use of ML for CGF behavior.

Since the field of ML is obviously very large, with many techniques, implementations, and tools, we will not attempt to provide a complete overview. Rather, we restrict ourselves to requirement definitions of these techniques, from the viewpoint of ML techniques integration into CGF packages. In these requirements, we also take into account the end users that might have to be able to operate these ML techniques. Furthermore, we discuss how a ML component could be integrated into the larger architecture of a CGF package.

Modeling CGF Behavior with Machine Learning Techniques

This paper is based on the current state of the work of the NATO Research Task Group IST-121 RTG-060 'Machine Learning Techniques for Autonomous Computer Generated Entities', which runs until the end of 2016.

2 STATE-OF-THE-ART OF AI IN CGF PACKAGES

Most Commercial/Military-Off-The-Shelf (COTS/MOTS) packages provide at least a rudimentary AI interface, through which the scenario developer has to define the behavior for their CGFs. On the other side of the spectrum, a few companies provide custom AI solutions that can be integrated with existing CGF packages. Mainly, AI introduction consists of adding some degree of decision autonomy to simulated entities. None of the mainstream CGF packages that are currently used operationally introduce ML capabilities.

In 2009, Abdellaoui et.al. (2009) analyzed and compared several well-known modeling and simulation packages with respect to CGF and AI capabilities. The packages were evaluated based on architecture, autonomous operation, learning, organization and realism. Results from the study showed that none of the evaluated packages provided tools, processes, user interfaces, etc. that would enable behavior modeling through observational learning ('supervised learning', see, for example, Mohri, Rostamizadeh, & Talwalkar, 2012) nor did these packages provide experiential learning ('reinforcement learning', see, for example, Sutton & Barto, 1998).

These packages have been re-evaluated to investigate if they have improved their learning capabilities or not since the evaluation by Abdellaoui et al (2009). Table 1 shows the review result with additional modeling and simulation packages commonly used by military organizations for training and decision support purposes². Most of these packages support Distributed Interactive Simulation (DIS) protocols, High Level Architecture (HLA) and are compatible with other standards of the Simulation Interoperability Standards Organization (see, for example, SISO, 2003).

Our review shows that these packages, to the best of our knowledge, still do not provide adequate capabilities to model behaviors through learning processes. Some packages, however, incorporate modern behavior modeling techniques/representations borrowed from the gaming/entertainment industry to address reusability, scalability and modeling complexity issues.

² However, many more packages that are also on the market could not be considered.

Table 1. Modeling and simulation packages: AI modules, representations and learning capabilities

Framework	AI Module	Representation	Learning
OneSAF, ModSAF	Various (specific for agencies)	No information found	No
VR-Forces	DI-Guy AI	Hierarchical finite state machines	No
STAGE	AI.Implant	Binary decision trees	No
VBS3	Discovery Machine Behavior Modeling Suite	Tree structures	No
MASA SWORD	MASA Life	Behavior trees	No
FLAMES	Cognitive model	Scripting and functions	No

Below, we give some examples of commonly used simulation environments with state-of-the-art AI interfaces.

OneSAF, which replaced ModSAF, is a modular package to construct CGFs. A single operator can create and control large numbers of entities that are sufficiently realistic that trainees are not aware that most of the maneuvering is done by computers, rather than humans. The US Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) offers a version to U.S. government users and an international version.

The SAF/CGF VR-Forces (VT MÄK, 2015) developed by VT MÄK supports both aggregate level simulations as well as simulation of single vehicles and soldiers. VR-forces includes basic tasks like “move along route”, “move to location” and “set engagement rule”, and new tasks can be created with Lua-scripts. The simulated entities or aggregates can be assigned independent tasks or plans consisting of multiple tasks.

STAGE (Presagis, 2015) is a scenario generation and CGF software suite. As a scenario generation tool, STAGE provided a level of fidelity and abstraction which is well suited to devise intelligent and autonomous CGFs in air-to-air combat simulations that were developed for the Netherlands Air Force (Roessingh, Merk, Huibers, Meiland, & Rijken, 2012). Where a higher level of fidelity in platform dynamics, sensor or weapon models is required, the basic CGF functionality provided by STAGE can be extended. CGFs in STAGE can also be enhanced with core AI capabilities through Presagis’ AI.Implant tool. The tool models movement and behavior of humans within a simulation, particularly for generating crowds in urban environments.

Virtual Battlespace 3 (Bohemia Interactive, 2015) allows CGFs to be scripted, and comes with a sample of high level behavior. Training instructors are given the capability to edit a scenario as it is running. This also includes triggering pre-scripted events.

SWORD (MASA group, 2015) is used to train brigade and division command post staff in large-scale conflicts and operations, focusing on improving decision-making capabilities. SWORD simulates a diverse range of situations in which trainees may lead thousands of autonomous subordinate units on the virtual field.

FLAMES (Ternion, 2015) is a family of software products that provide the framework and basic functionality needed for constructive simulations as well as the ability to interface with other simulations and live entities.

Scenario development for simulation training is about exploiting the capabilities of the CGFs to generate training scenarios that facilitate training goals. The scenario developer should therefore be a Subject Matter Expert (SME) with extensive expertise about the domain and the training goals. Ideally, the scenario developer should not have to be a programmer or software developer. However, CGFs capable of sophisticated behavior are complex software models. It may therefore be very hard for an SME to define the different high-level behaviors of CGFs. Whether relevant or not, some models may be so complex that it becomes almost impossible for regular end users of CGFs (training instructor, animators, scenario developers, etc.) to configure the AI so as to fulfill the scenario requirements or behavior expectations. Particularly, the translation of military expertise into classical AI models is very difficult. Even worse is the translation of real “observed” behaviors (as the “observed” behavior of a real entity embeds interpretation of orders/instructions and human factors). Therefore, most scenario development with complex AI is currently done by programmers rather than SMEs. In essence, this is a user interface issue, as the SME also has to be an expert on translating their knowledge to computer commands.

Furthermore, managing the level of detail and realism is extremely difficult. For example, modeling behaviors using doctrines results in behavior models that are too good (optimized) to be realistic. Likewise, employing SMEs can result in subjective behavior models that sometimes do not adequately reflect the true behavior of the CGF's real-world counterpart.

Finally, integrating CGFs in a Command and Control (C2) structure is challenging. Ideally, commanding CGFs should behave the same as commanding live forces. This would require the CGF system to receive and interpret orders directly from the C2 system. In current command and staff training, human operators are needed to decompose the higher level tasks from the C2 system and manually enter more detailed sets of instructions into the CGF system. More autonomous CGFs would require fewer personnel to carry out an exercise, and lead to new

usages like support in operational planning and decision making (Hyndoy, Mevassvik, & Brathen, 2014).

Using more autonomous CGFs in C2 systems requires careful consideration of the many concerns involved: behaviors at various levels need to be modeled correctly, consistently and need to be adaptive at the same time. Although various efforts are undertaken to create more autonomous CGF, these seem to be driven by R&D initiatives that do not result in AI implementations in current COTS/MOTS packages.

The Norwegian Defense Research Establishment (FFI) developed a prototype of a more autonomous CGF that receive battalion level orders directly from a C2 system, and used a multi-agent system to interpret the high level task in the order and decompose them into lower level tasks that most CGF systems can understand (Alstad, Løvlid, Bruvoll, & Nielsen, 2013; Løvlid, et al., 2013). Further examples of usage of CGFs in C2 systems are SCIPIO (Thalesgroup, 2015a) , which is an army command post simulator for the army in which MASA-SWORD is used, and SETHI (Thalesgroup, 2015b), which introduces modern AI approaches: based on motivational free-flow hierarchies and classifier systems, they provide CGFs with perceptible adaptive behaviors. However, none of these examples use ML techniques.

In conclusion, concerns with current CGF packages are undesirably high programming complexity for end users, lack of realism and detail in behavior and the lacking ability of CGFs to understand high level commands from C2 systems. Furthermore, the number of staff needed to build a scenario and run a simulation is a major cost driver. Solutions for the aforementioned concerns with behavioral modeling of CGFs seem not to have evolved beyond ad-hoc partial solutions that require a specific and detailed knowledge. The development of CGFs that possess richer behavior in complex environments and are better tailored to the knowledge and skills of the trainee is therefore hampered. Moreover, development of CGFs often remains a painstaking development of a set of rules (for example 'if-then rules') that need to be derived for each specific problem or situation to be resolved, based on the manual elicitation of operational expertise.

3 THE ADDED VALUE OF MACHINE LEARNING

The use of ML techniques does not seem very well exploited in the current approaches. Our hypothesis is that using ML to generate behavior might be faster and lead to more desirable behavior than the traditional approach where the behaviors of the CGF are manually generated using hand-crafted rules, scripts and probabilities, etc.

ML can be used to generate behavior from scratch, but may also be based on existing sources of data, such as recordings of demonstrated behavior. These methods might discover causalities that SMEs are not conscious about, while enabling the CGFs to generalize to new situations that did not crop up in the training process. This process will take considerably less time from SMEs than the current practice of building scenarios. Currently, many examples exist in which training scenarios with duration of, say, half-an-hour, may take weeks or months to prepare.

The ability of ML methods to objectively extract behavior rules from data and generalize to new situations might lead to more desirable behavior: more autonomous CGFs, with fewer personnel needed to build a scenario, to run a simulation and to carry out an exercise. Also, this capability of ML enables using CGFs in new applications such as a what-if-analysis during the operational planning and during the operation itself.

Faster development of new behavior also opens up the possibility for behaviors that are tailored for specific training objectives. For example, for training purposes one often wants enemy behavior that provokes specific training elements. Also, when training in an environment with CGFs as a part of own forces, one might want the CGFs to be at approximately the same performance level as the trainees, such that different variations of the same basic behavior can be trained in a team context. For planning purposes one probably wants friendly ('blue') and enemy ('red') CGFs acting according to their own doctrines. These types of behaviors can be developed fast with ML.

4 REQUIREMENTS AND DESIRED FUTURE CAPABILITIES

4.1 Identifying end-users

ML techniques and tools, and the CGFs created with these, need to be usable by all people involved in a training simulation. Different end-users may be distinguished, depending on characteristics of training and scale of the organization. In this paper, we consider the following persons as possible end-users of CGFs:

- **the training instructor**, who identifies individual, team, and collective training objectives and translates these objectives (including performance criteria) into representative scenario events to provide opportunities for trainees to demonstrate competencies related to the training objectives. They observe and provide feedback on team processes and outcomes;
- **the modeler**, who develops algorithms for CGFs to generate realistic human behavior, i.e., cognitive modeling and behavior emulation models. Also physical events and combat interactions on the battlefield need to be modeled;
- **the scenario developer**, who, on the basis of a software architecture that supports multiple entities, takes CGF models and algorithms developed by modelers and build plans, tasks, event triggers, behavior sets, and user interfaces into custom applications of the simulation. They create tools (e.g., real-time and post simulation analysis) to link performance data to a historical performance database, allow individual and/or team strengths and weaknesses to be diagnosed, and serve to focus future training events;
- **the CGF operator** interacts with the scenario while the simulation is running and instructs CGFs through a user interface which allows scenario generation by positioning forces, creating routes and waypoints, assigning tasks and plans, and triggering events to achieve simulation goals;
- **the trainees** benefit in their learning process from the learning events in which the CGFs act out their role. They demonstrate competencies related to the training objectives, which may be formulated for the individual trainee, the team, or the collective.

In practical applications it was generally observed during the survey of the Research Task Group (RTG-060) that end users (with a focus on the modelers) select different ML techniques and mix them together in a hybrid fashion. We believe that it should be possible to create truly intelligent (autonomous and adaptive) CGFs through using packages that are available on the market. In this section, we define requirements for ML techniques in behavior modeling software. Also, we

discuss the future capabilities of behavior modeling that we believe are instrumental for an integrated approach.

4.2 Algorithmic Requirements

The field of training simulations is closely related with that of video games. Both training simulations and games take one or more participants into a virtual world, although for different goals (training versus entertainment). Both also require these virtual worlds to be populated with virtual entities that display behavior that is believable for some particular setting.

Video games have been and continue to be fertile testing grounds for behavior generation through ML for both academia and industry. One of the main focus points has been adaptive AI that responds to the manner in which a player plays a game. Spronck et al. have compiled a list of computational and functional requirements that ML techniques should adhere to if they are to be used for adaptive game AI (Spronck, Ponsen, Sprinkhuizen-Kuyper, & Postma, 2006). These requirements also seem to be a good fit for training simulations, as they share many qualities. The requirements are as follows.

The computational requirements are:

- **Speed.** Behavior generation should be fast, as it is (possibly) done live.
- **Effectiveness.** Generated behavior should be effective, even while the system is still learning.
- **Robustness.** Generated behavior should be able to cope with randomness and unexpected events.
- **Efficiency.** Generated behavior should quickly be optimized based on few interaction moments with the human participant.

The functional requirements are:

- **Clarity.** Generated behavior should be easily interpretable by human operators.
- **Variety.** A variety of behaviors should be generated, as repeated behavior can be uninteresting or even suspicious.
- **Consistency.** The number of interaction moments needed to generate or adapt behavior should have low variance and should be independent from the behavior of the human participant.
- **Scalability.** Generated behavior should be scalable to the skills of the human participant.

For training simulations, we propose the addition of a new requirement. A key feature in CGF packages is the ability of the instructor or CGF operator to take over control of one or more

entities. Ideally, a CGF operator can select a level of autonomy at which each entity operates, ranging from fully automatic to fully manual control (Parasuraman, Sheridan, & Wickens, 1997). ML techniques, the behavior models they generate, and the tools with which they are controlled should facilitate such takeovers, and the behavior of the CGFs should adapt gracefully. Hence a specific functional requirement for CGFs in training simulations is:

- **Transfer of control.** An end-user (instructor, CGF operator) should be able to take control over the CGF without interruption in behavior.

5 REALIZATION

We introduce some ML techniques, and propose a hypothetical architecture and user interface with which these can be employed.

5.1 ML Techniques

ML techniques can be applied both online and offline. Essentially, using ML techniques, CGFs learn to map observed situations to particular actions. Offline learning means this mapping is learned from a set of known examples, before the CGFs start operating in their environment. In turn, online learning means that observation-action mapping is learned without any prior data, and that the CGFs learn from data as it comes in during exploitation of the built-in models.

Both online and offline learning have their advantages and disadvantages. Online learning methods have time constraints, as it is done during the operation of some system. However, they are also capable of learning from new, unseen situations as they arise. Offline learning methods have no time constraints, as they do the learning before operation. This also allows testing the properties of the learned behavior models beforehand. However, retraining the models requires adding new training instances and adjusting parameters. Some methods are flexible and can be used both offline and online, providing the best of both worlds. However, which method is best depends on the application.

A second division of ML techniques is that between supervised, unsupervised and reinforcement learning methods. Supervised methods are fed pre-labelled data, e.g. situations labelled with 'correct' actions. The ML algorithm then tries to discover the correct mapping between the situations and the labels, so that unseen situations can be acted upon. On the other hand, unsupervised methods are fed unlabeled data, and it is left to the algorithm to discover 'categories' of situations, and then map them to actions.

Finally, certain ML techniques are reinforcement learning methods. Reinforcement learning methods require an evaluation function which assigns a score to displayed behavior. This would allow CGFs to learn behavior in certain 'offline' environments, before being put in an environment together with human actors. However, depending on the evaluation function, the CGFs may continue to evaluate their own behavior in the new 'live' environment, and keep adjusting their behavior online.

Modeling CGF Behavior with Machine Learning Techniques

Various types of ML techniques are available, for both online and offline ML applications. As a suggestion, the following non-exhaustive list of ML techniques covers various aspects of CGF behavior modeling:

- Decision Tree Learning (Mitchell T. M., 1997) is a technique for automatically learning hierarchical decision structures, which allows incorporating expert knowledge such that this is easily understandable and verifiable by human users.
- Artificial Neural Networks (Haykin, 1998) are algorithms based on biological neurons, usually used to model complex relationships and find patterns in data.
- Bayesian Learning (Jie Cheng, 2001) allows for a modular representation of uncertain knowledge, providing an intuitive representation of domain knowledge.
- Genetic Algorithms (Mitchell M. , 1998) allow for finding solutions to optimizations and search problems, through generation of candidate solutions in a biologically inspired process.
- Hidden Markov Models (Rabiner, 1989) are statistical models aimed to determine the hidden parameters of an underlying model based on visible output, and can be used to recognize temporal patterns, for example.
- Reinforcement Learning (Sutton & Barto, 1998) in general is learning what to do by trial and error, driven by discovery of the most rewarding action.
- Dynamic Scripting (Spronck, Ponsen, Sprinkhuizen-Kuyper, & Postma, 2006) is a technique that finds optimal combinations of behavior rules, which are taken from a pre-populated rule base.

It is speculated that for most practical applications, a combination of various ML techniques is required for modeling realistic and adaptive CGF behavior, taking intent, intrinsic restrictions and guidelines into account.

5.2 Architecture

Usually practitioners select different ML techniques and mix them together as a hybrid approach. For example, practitioners mix reinforcement learning with evolutionary computing and neural networks. A solution could therefore be based on the following two principles:

- Decoupling learning CGF models from the simulation application or from the scenario management application,
- Enabling the distribution of such models at different “client” CGFs.

Such a solution enables mixing different ML techniques in a user friendly way. We therefore propose a modular architecture that provides ML capabilities to a CGF package, based on Roessingh, Merk, Huibers, Meiland, & Rijken (2012).

Figure 1 shows the proposed architecture. The software package runs a simulation, which lets several human actors and CGFs play out a certain scenario. The CGFs are driven by a ML component. This component should have access to some repository containing behavior models and associated ML techniques. The ML component should also be able to take some high-level behavior specification as input. The component should then be able to produce valid behavior, by modifying the behavior models to suit the behavior specification.

The ML component can be made independent of the simulation package by using a Mediator component similar to the one described by Roessingh et al. (2012) to translate instructions originating in a behavior model to package-specific instructions.

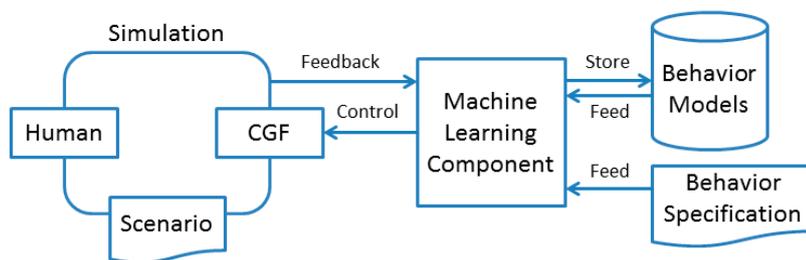


Figure 1. Proposed architecture for integrating ML in a CGF package

5.3 User Interface

A user interface (UI) is needed for end users to employ ML techniques for defining, developing, and controlling CGF behaviors. A ML technique-based UI should provide the following basic requirements for user to employ CGFs in military training simulations.

- Automatic generation of behavior models (whether offline or online) should be possible with minimum effort from the end user. For the ease of use, technical details should be hidden as much as possible, yet should still be accessible to experts. Earlier in this paper we have identified five different possible end users, each of which should be able to use the UI for their purposes.

Modeling CGF Behavior with Machine Learning Techniques

- Behavior models should be reusable to save development time. This includes identifying which types of CGF a specific behavior model is usable for. Ideally, a behavior is transferrable between different types of CGFs with minimum efforts.
- Behavior models should be easily testable, to demonstrate and verify learned behavior. The UI should contain a component in which a minimal scenario can quickly be built for one or more CGFs using a specific behavior model. These scenarios should have testable, user-definable conditions with which the test case can be labeled a success or failure. Such scenarios can then be grouped into test suites for automated testing.
- The end user should be able to add constraints and goals to generated behavior, e.g. to indicate training events that a CGF should facilitate. This should be possible in a way that is intuitive for the end user.

None of these matters are trivial, and they are most likely partially the reason why ML for CGF behavior has seen little commercial interest. However, if no efforts are made in this area at all, no knowledge will be gained. To unlock the full power of ML techniques, it is important to start making the first steps in offering them to end users.

6 CONCLUSION

This paper takes the position that machine learning (ML) techniques are lacking in CGF packages that are currently on the market. The abundant possibilities of ML are currently not exploited within military simulation and serious games. The development of CGFs remains a painstaking effort with ad-hoc solutions that are based on the manual elicitation of operational expertise. We believe that it should become possible to create truly intelligent (autonomous and adaptive) CGFs with the packages available on the market.

Modeling of realistic CGF behavior requires ML techniques and tools that can deal with many aspects, such as intent, interaction, and intrinsic responsibilities, tasks, restrictions and guidelines. Task Group RTG-60, under the NATO RTO Information and System Technology (IST) panel, therefore takes the challenges to bring these ML techniques to the end-users: training instructors, modelers, scenario developers, CGF operators and trainees. However, to apply learning CGFs in simulation and games, specific end-user requirements and functional requirements need to be taken into account first.

Various requirements for modeling CGF behavior have been identified. Many requirements are similar to the requirements for ML applications in video games. From the computational perspective, speed, effectiveness, robustness and efficiency are a prerequisite. Furthermore, ML techniques should fulfill the requirements of clarity, variety, consistency, and scalability for military training simulation. Finally, we have identified the need for generated behavior models to be able to cope with transferring control to human operators.

An initial list of ML techniques to be supported is provided. An architectural solution is suggested, based on the principles of decoupling learning CGF models from the specific application and on enabling distribution of such models at different client-CGFs. This enables the freedom of pursuing hybrid techniques in a user friendly way.

In summary, we believe that applying ML techniques can be beneficial for modeling CGF behaviors, and guidance needs to be developed for end users. We would recommend that the developers of the COTS/MOTS CGF packages start incorporating ML techniques, thereby providing the capability to create CGFs that possess richer behaviors in complex environments and are better tailored to the knowledge and skills of the trainee. This paper is essentially a call for further development of CGFs that are capable of learning, using the architecture we proposed.

ACKNOWLEDGEMENTS

The authors thank Christopher Roos (NLR) for describing his experience with the VBS package.

7 REFERENCES

- Abdellaoui, N., Taylor, A., & Parkinson, G. (2009). Comparative Analysis of Computer Generated Forces' Artificial Intelligence. *NATO Modelling and Simulation Group (NMSG) Symposium (MSG-069): Use of M&S in Support to Operations, Irregular Warfare, Defence against Terrorism, and Coalition Tactical Force Integration*. Brussels, Belgium
- Alstad, A., Løvliid, R. A., Bruvoll, S., & Nielsen, M. N. (2013). *Autonomous battalion simulation for training and planning integrated with a command and control information system*. FFI Tech Report no.: 2013/01547. Retrieved from <http://rapporter.ffi.no/rapporter/2013/01547.pdf>
- Bohemia Interactive. (2015). *Virtual Battlespace 3 | BISim*. Retrieved from <https://bisimulations.com/virtual-battlespace-3>
- Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR
- Hyndoy, J., Mevassvik, O., & Brathen, K. (2014). Simulation in Support of Course of Action Development in Operations. *Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*. Orlando, Florida
- Jie Cheng, R. G. (2001). *Learning Bayesian belief network classifiers: Algorithms and system*.
- Løvliid, R. A., Alstad, A., Skogsrud, G., Bruvoll, S., Mevassvik, O. M., & Bråthen, K. (2013). *Modelling battle command with context-based reasoning*. FFI Tech Report no.: 2013/00861. Retrieved from <http://rapporter.ffi.no/rapporter/2013/00861.pdf>
- MASA group. (2015). *SWORD*. Retrieved from <http://masa-group.biz/products/sword/>
- Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press.
- Mitchell, T. M. (1997). *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc.
- Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of Machine Learning*. The MIT Press
- Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (1997). A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 30(3), 286-297
- Presagis. (2015). *STAGE | Scenario Generation Software | Presagis | Presagis*. Retrieved from http://www.presagis.com/products_services/products/modeling-simulation/simulation/stage/
- Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *IEEE*, (pp. 257-286)
- Roessingh, J., Merk, R.-J., Huibers, P., Meiland, R., & Rijken, R. (2012). Smart Bandits in air-to-air combat training: Combining different behavioural models in a common architecture. *21st Annual Conference on Behavior Representation in Modeling and Simulation*
- SISO. (2003). *Realtime-Platform Reference Federation Object Model (RPR FOM 2.0d17)*
- Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., & Postma, E. (2006). Adaptive game AI with dynamic scripting. *Machine Learning* 63.3, 217-248
- Sutton, R. S., & Barto, A. G. (1998). *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press
- Ternion. (2015). *FLAMES*. Retrieved from <http://www.ternion.com/>
- Thalesgroup. (2015a). *Scipio Army Command Post Training Centre*. Retrieved from <https://www.thalesgroup.com/en/canada/defence/scipio-army-command-post-training-centre>
- Thalesgroup. (2015b). *Army Training and Simulation (PDF)*. Retrieved from https://www.thalesgroup.com/sites/default/files/asset/document/thales_army_simulationcanada.pdf
- VT MÄK. (2015). *VR-Forces Online*. Retrieved from <http://www.mak.com/products/simulate/computer-generated-forces.html>

This page is intentionally left blank.

WHAT IS NLR?

The NLR is a Dutch organisation that identifies, develops and applies high-tech knowledge in the aerospace sector. The NLR's activities are socially relevant, market-orientated, and conducted not-for-profit. In this, the NLR serves to bolster the government's innovative capabilities, while also promoting the innovative and competitive capacities of its partner companies.

The NLR, renowned for its leading expertise, professional approach and independent consultancy, is staffed by client-orientated personnel who are not only highly skilled and educated, but also continuously strive to develop and improve their competencies. The NLR moreover possesses an impressive array of high quality research facilities.



NLR – *Dedicated to innovation in aerospace*

www.nlr.nl