REAL-TIME SYNCHRONIZATION AND MODIFICATION OF A BEHAVIORAL VEHICLE MODEL FOR DISTRIBUTED SIMULATION

William J. Gerber and Avelino J. Gonzalez
University of Central Florida
Orlando, FL 32816-2450

Distributed simulations have become valuable tools for individual and group training. A combination of live, virtual and constructive distributed simulations that is highly promising for greater realism in training at reduced costs, called embedded simulation, is being explored by the U. S. Army's Simulation, Training and Instrumentation Command (STRICOM) Inter-Vehicle Embedded Simulation Technology (INVEST) Science and Technology Objective (STO) program for use in combat vehicles. Among the many technical challenges to be overcome is that of providing a simulation environment in which live vehicles, manned vehicle simulators, and computer generated forces can interact with each other as well as with the battlefield environment in real-time over a geographically diverse, distributed network. The main problem is the high communications requirements imposed by the need to convey large amounts of data among the various players. The Vehicle Model Generation and Optimization for Embedded Simulation (VMGOES) project at the University of Central Florida is focusing on this aspect of the INVEST STO program. The approach is to use a behavioral vehicle model that is context-based to match the actions of the human-controlled entity on the battlefield. By observing the surrounding environment of the vehicle model's location in the simulation at each update time step, the model will determine what context it should be in and perform the actions that are appropriate for that context. This will allow the vehicle model to match the human-controlled entity's behavior for a longer period of time than is possible with only dead-reckoning updates, thus reducing the communications bandwidth required. However, discrepancies between the vehicle model and the human controlled entity will inevitably occur and these must be detected and resolved to allow the vehicle model to function efficiently. The portion of our model that addresses this need, the Difference Analysis Engine (DAE), will be resident on the human-controlled entity. It will be able to observe the actual vehicle's actions as well as the simulation environment and the vehicle model itself. It then must evaluate whether significant discrepancies exist. If they do, it will immediately take the action needed to synchronize the vehicle model with the actual entity. These corrections can involve a simple State Realignment to update the vehicle model's location, direction and speed; a forced vehicle model Context Shift to match the context of the human-controlled entity; a Model Correction to change the way the model itself responds; and, as a last resort, a Model Suspension to revert to standard dead-reckoning until the DAE can recognize what context the human-controlled vehicle actually is in. This paper will focus on those DAE functions and on how techniques, such as temporal template based reasoning, neural networks and genetic algorithms, are being used to accomplish those DAE functions.

ABOUT THE AUTHORS

William Gerber received his B.S.E.S. in both Astronautics and Engineering Sciences from the United States Air Force Academy. He has an M.S.E. in Nuclear Engineering from the University of California at Los Angeles and an M.S.Cp.E. in Knowledge Based Systems from the University of Central Florida. He is currently a Ph.D. candidate in computer engineering at the University of Central Florida, a Research Assistant on the VMGOES project and a Research Fellow at U.S. Army STRICOM.

Avelino Gonzalez received his bachelor’s and master’s degrees in Electrical Engineering from the University of Miami, in 1973 and 1974, respectively. He obtained his Ph.D. degree from the University of Pittsburgh in 1979, also in Electrical Engineering. He is currently a professor in the Electrical and Computer Engineering Department at UCF, specializing in human behavior representation.

REAL-TIME SYNCHRONIZATION AND MODIFICATION OF A BEHAVIORAL VEHICLE MODEL FOR DISTRIBUTED SIMULATION

William J. Gerber and Avelino J. Gonzalez
University of Central Florida
Orlando, FL 32816-2450

INTRODUCTION

Simulations, whether constructive, virtual or live, have become valuable tools for individual and group training. They generally allow acceptably realistic training at a lower cost than using the actual equipment in a fully operational capacity. Additionally, the use of distributed simulations has allowed for team and large scale unit training in a constructive or virtual environment without the expense of massive movements of equipment to a central training area and the scheduling restrictions of that limited training area resource. Furthermore, the introduction of Computer Generated Forces (CGF's) has added greater realism into the simulation by allowing representations of the enemy forces as well as the inclusion of greater numbers of simulated friendly forces than can be reasonably assembled for any given training exercise.

The most realistic training, however, occurs when the individuals and crews can train on their own operational equipment in the actual environment. This is done now in live simulations using instrumented large-scale exercises at locations such as the U. S. Army's National Training Center. There, the live fire of weapons is replaced for increased safety and reduced expense by the use of instrumented vehicles where the outcome of individual engagements is determined by computer processing of the data in real time. The initial data and the results are transported over a communications network, the Range Data Measurement System (RDMS), which allows only 2400 bits per second peak bandwidth per vehicle for the information exchanges between vehicles. [Bahr and DeMara, 1996] The time lags and bandwidth of the communications network, though, still restrict the realism.

A combination of live, virtual and constructive distributed simulations that is highly promising for greater realism in training at reduced costs, called embedded simulation, is being explored for use in combat vehicles by a program of the U. S. Army's Simulation, Training and Instrumentation Command (STRICOM). That program is called the Inter-Vehicle Embedded Simulation Technology (INVEST) Science and Technology Objective (STO) program. Among the many technical challenges the program must overcome is that of providing a simulation environment in which live vehicles, manned vehicle simulators, and computer generated forces can interact with each other as well as with the battlefield environment in real-time over a geographically diverse, distributed network.

The main problem is the high communications requirements imposed by the need to convey large amounts of data among the various players. The Vehicle Model Generation and Optimization for Embedded Simulation (VMGOES) project at the University of Central Florida is focusing on this aspect of the INVEST STO program. The approach is to use a behavioral vehicle model (VM) that is context-based to match the actions of the human-controlled entity on the battlefield. By observing the surrounding environment of the vehicle model's location in the simulation at each update time step, the model will determine what context it should be in and perform the actions that are appropriate for that context. This will allow the vehicle model to match the human-controlled entity's behavior for a longer period of time than is possible with only dead-reckoning updates, thus reducing the communications bandwidth required.

However, discrepancies between the vehicle model and the human controlled entity will inevitably occur and these must be detected and resolved to allow the vehicle model to function efficiently. The portion of the model that addresses this need, the Difference Analysis Engine (DAE), will be resident on the human-controlled entity. It will be able to observe the actual vehicle's actions as well as the simulation environment and the vehicle model itself. It then must evaluate whether significant discrepancies exist. If they do, it will immediately take the action needed to synchronize the vehicle model with the actual entity. These corrections can involve a simple State Realignment to update the vehicle

model's location, direction and speed; a forced vehicle model Context Shift to match the context of the human-controlled entity; a Model Correction to change the way the model itself responds; and, as a last resort, a Model Suspension to revert to standard dead-reckoning until the DAE can recognize what context the human-controlled vehicle actually is in. This paper will focus on those DAE functions and on how techniques, such as temporal template based reasoning, neural networks and genetic algorithms, are being used to accomplish those DAE functions

EMBEDDED SIMULATION

Embedded simulation, the simulation refinement for providing the most realistic training, combines live, virtual and constructive simulations. [Bahr, 1997] For this type of training, the individuals and crews use their own operational equipment, but use them in a simulation mode. The controls still function but the inputs are additionally provided to a simulation environment, which in turn provides feedback to the individuals and crew in the form of visual, auditory and other stimuli. This has the advantages of the live simulation in that the equipment is as authentic as possible. Furthermore, it can be used in either the deployed operational environment or in a benign, non-deployed state at the home location.

The refinement that comes from the addition of the virtual and constructive simulations allows a training scenario to be filled out with virtual, manned simulators and CGF vehicles for more realistic force-on-force training. The viewports of the live vehicles can have displays of the virtual forces and CGF's overlaid and registered with representations of the live forces so that a seamless presentation is made to the crewmembers of the entire set of forces involved in the scenario. This training can be very flexible and can be based on either fully mobile forces within an actual training area (most similar to the live simulation), stationary forces using CGF's (most similar to a virtual simulation with manned simulators), or a combination of the two. Likewise, the number of participants can vary from the single soldier practicing individual tasks using simulations of the other crew members/other crews to large-scale exercises with numerous live individuals/crews mixed in with manned simulators/CGF's. Again, the realism is restricted by the time lags and the bandwidth of the communications network available for each participant. As the number of participants increases, the available bandwidth can become so limited that updates to the representations of the forces are delayed, resulting in a decrease in the realism of the simulation.

In addition to being able to have a set of canned training scenarios with specific training objectives for the crews, embedded training is also operationally useful for mission rehearsal training. Forward-deployed crews can practice an operational scenario before actual execution to familiarize themselves with the mission before trying it the first time while under the strain of combat. As an additional benefit, previously undetected flaws in a plan can be found before they become too costly.

In the past, this use of embedded simulation for training has not been pursued because it was viewed as too difficult and expensive to implement. [Bahr, 1997] Currently, models of vehicles transmitted to other simulations do not address behavior, even in a generic way, beyond using a dead-reckoning model that assumes that the current actions taking place will continue without change until updated by a Protocol Data Unit (PDU) set of communication packets. There is thus no means for vehicle changes in reaction to the simulation environment to be reflected in the distributed model except by PDU's using the scarce bandwidth. Therefore, for embedded simulation to be used in large exercises, reactive models must be developed for each vehicle to cover numerous situations, a significant undertaking to develop manually. Furthermore, each deployed model on other vehicles must be continuously correlated with the actual behavior of its vehicle throughout the entire training period. Since the manually developed models, for cost reasons, must be generic, they cannot reflect the subtle differences between individuals and crews of different skill levels and operational styles using the equipment being modeled. As a result, these differences must be accounted for by frequent updates to correlate the model with the vehicle's actions, a further drain on the available bandwidth.

BEHAVIORAL VEHICLE MODEL

At the U.S. Army's STRICOM, the INVEST STO program was initiated in 1997 to address these issues of embedded simulation. [Bahr, 1997] Included in the INVEST STO program is the Vehicle Model Generation and Optimization for Embedded Simulation (VMGOES) research.

It is investigating the use of techniques to reduce the effort required for developing the models and to cut down on the bandwidth required for supporting the virtual/live vehicle representations in the embedded simulation. [Gonzalez, DeMara, and Georgiopoulos, March 1998; Gonzalez, DeMara. and Georgiopoulos, May 1998; Henninger, Gerber, DeMara, Georgiopoulos and Gonzalez, 1998]

Architecture

The VMGOES concept uses a Vehicle Model (VM) and a Difference Analysis Engine (DAE) (see Figure 1). Both are linked to each other using context-based reasoning and are located on each host vehicle. [Gonzalez and Ahlers, 1995] The VM portion includes the VM control, a neural network engine and neural network libraries while the DAE portion includes the DAE control, a template engine, and template libraries. Both portions can observe the simulation environment, the state data store for the host vehicle model, and state data stores for vehicle models of other vehicles of interest to the host. (The data stores for the other vehicles are also referred to as Clone data stores, where Clone X VM Data Store would be the data store for Vehicle X that is of interest to the host.) The DAE can also observe the true state of its host vehicle and send corrections for the state data store of its host vehicle model via a PDU, when needed. The PDU is sent through its host vehicle to the VM on the host vehicle and to the VM on any of the other vehicles observing its host vehicle. The PDU is used by the VM, at each location where a model of that vehicle is being maintained, to update the appropriate vehicle state data store.

The VM portion will address the massive manual development effort by allowing automation of the process. The VM will be trained by observation of the host vehicle's surroundings in the virtual environment to determine its local behavioral context and, consequently, which actions to take while in that context. (Actions are also considered sub-contexts.) The VM will also be trained by observation of the specific actions within each context to model those actions. This training will be stored in the neural network libraries, which are called by the neural network engine to produce the predictions needed, such as for context or for speed and heading. The VM control uses the predictions to update the vehicle state at each time step. When properly trained to reliably predict the actions that the vehicle will take, the VM should substantially reduce the number of PDU updates required and thus the bandwidth needed for those updates.

The DAE portion also addresses the bandwidth reduction for PDU's needed to keep the VM synchronized with the actual vehicle by real-time observation and evaluation of the actual vehicle's context and actions as well as the VM's predicted context and actions. The DAE, like the VM, will be trained by observation, but by observation of the host vehicle as well as its surroundings in the virtual environment. It will be trained to recognize the actual vehicle context and sub-contexts, which will be constrained to match the ones that the VM has been trained to recognize and utilize. The training for the DAE will be stored in template libraries. The template engine sorts through the templates to find the one that most closely matches the observations, i. e., achieves the highest score for similarity of conditions, and selects that one to predict the host vehicle's actual context. The DAE control considers the magnitude of observed positional errors, a history of those errors, and the actual vehicle context in determining whether or not to send out a PDU and, if so, what kind. Thus, the DAE will determine not only positional mismatches, as is done in current dead-reckoning, but also context mismatches when they occur and model error biases between the VM and the real vehicle. It will then provide appropriate positional, context or bias corrections to the VM in real time to both synchronize it with the real vehicle state and to modify the vehicle model actions to improve its performance.

Operation

In operation in an embedded simulation, the dynamic data for the host vehicle's VM state would exist in its modeled host vehicle's data store and would be replicated identically in the data store for other vehicles whenever they receive a PDU. The trained neural network data for contexts and actions for each modeled vehicle in the exercise is pre-stored identically in each vehicle and, since every environment presumably is the same for what each VM would observe, each VM would react identically. Thus, the data store for the host vehicle, located on the host vehicle itself, is a good reference for the DAE of what the VM data stores on other vehicles are using to represent its host vehicle. The additional processing required for the VM and DAE computations onboard the operational vehicles configured for embedded training is considered a tradeoff for reducing the

bandwidth requirements and the associated infrastructure. Since the VM would be able to react to the environment as the vehicle on which it

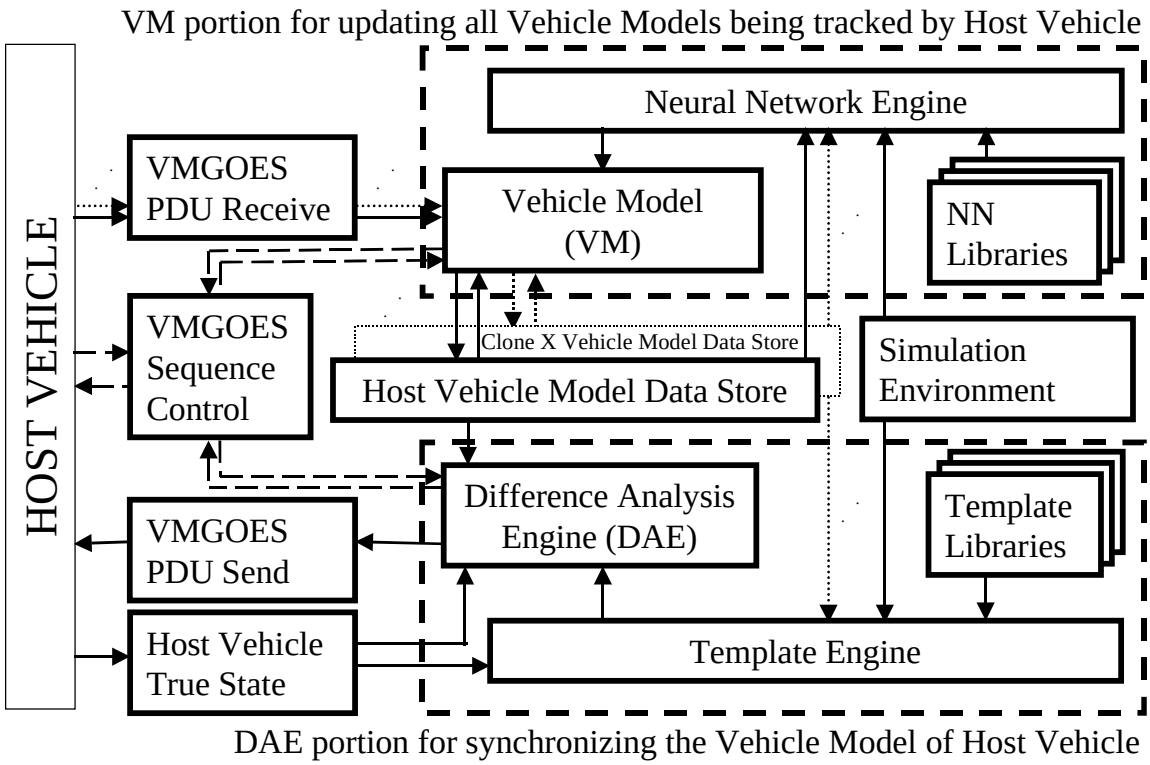

Figure 1. VMGOES Architecture with the interactions for the Vehicle Model and Difference Analysis Engine portions of the Host Vehicle shown.

was trained would react, it should be able to operate for a longer time before needing updates than the current dead-reckoning method would allow.

Of course, the VM is still only a model. Therefore, differences over time between the VM predicted actions and the actual modeled vehicle's actions will be inevitable. Since each Clone VM data store for its current state will be maintained in a vehicle other than the vehicle it is a model for, it will not have visibility to the modeled vehicle's actual actions. Thus it cannot predict actions based on what the actual vehicle is currently doing. For example, a real crew may miss the fact that an enemy is present or, conversely, may misidentify a friendly force as an enemy. That crew's unforseen actions would then diverge from what the VM would logically predict. The DAE will observe those differences, whether from minor errors in predicted actions over time, errors in the model context for the observed host vehicle, or the vehicle taking unusual actions. It will then take the necessary corrective actions to correlate the VM with the actual vehicle and, if appropriate, modify the VM either in context or in the actions it takes.

RESEARCH

Research on the vehicle model portion of the VMGOES is ongoing, as is that on the difference analysis engine portion. However, this paper is addressing only the research into the synchronization and modification of the behavioral vehicle model. Thus, it assumes that the vehicle model research will be successful and describes only what will be done for research for the DAE portion.

Overview

The purpose of this research is to find a means of determining, in real-time, the high-level behavioral intentions of an individual or crew controlled vehicle through learning by observation the

actions taken by the vehicle over a recent time period as well as observing the environment in which the vehicle is operating. Additionally, the learning by observation must be automated to make it feasible. Further, the use of adaptive, dynamic learning of the differences between the model's prediction of low-level actions and the actual entity's actions will aid the synchronization between the model and the entity by modifying the model in real-time. This research has application over a wide range of distributed simulations and intelligent tutoring systems. For distributed simulations, it will allow for more efficient vehicle models that will result in an increase in simulation realism and an increase in the number of participants able to train in the same simulation exercise. [Bahr, 1997] For intelligent tutoring systems, the intentions of a student can be captured in real-time for immediate training feedback on mistakes, as well as for use in an after-action review. [Drewes and Gonzalez, 1995] Additionally, the capability to observe the vehicle's actions, compare them with the model and then, when necessary, resynchronize the model provides a possible validation tool for behavioral vehicle models. By keeping the model synchronized with the actual vehicle throughout the validation test, the actions of the vehicle model can be compared with the actions of the human-controlled vehicle in a meaningful way even if the model makes different, though possibly correct, decisions during the test. [Gonzalez and Murillo, 1999]

Approach

The specific research being addressed here, however, is the real-time synchronization of an embedded simulation behavioral model of a human-controlled vehicle operating in a simulation environment. To accomplish this research, the following four questions must be answered. First, to keep the vehicle model efficiently correlated with the actual vehicle, how can one modify the embedded simulation model to correct observed model positional errors, model speed and heading prediction errors and model context errors? Second, to make the correct modifications, how can one determine the reason for discrepancies between the embedded simulation model and the actual vehicle? Third, to support the discrepancy reasoning, how can one determine the vehicle's actual context by observing its actions and the simulation environment? And fourth, how can one implement the solution into an actual embedded simulation environment architecture?

Template-Based Reasoning At the center of the model synchronization research is the challenge of determining the actual entity's intentions/context. It is planned that temporal template-based reasoning will be used, where each defined intention/context that the vehicle can assume is associated with a template. [Drewes, 1997] Each template will have selected attributes that represent portions of the vehicle's state and simulation environment (see Figure 2). The set of attributes selected will include only those that are germane to the determination of that particular template's validity. At each update cycle, each template's attributes will be evaluated and multiplied by a weighting value to determine an overall score for the template. If a minimum threshold value is not reached, the template is not considered further as a candidate for the vehicle's intention/context. The template chosen as the one representing the actual vehicle's intention/context will be the one with the highest score.

Automating Template Training by Observation Closely associated with the use of the templates is the challenge of how to automate the setting of the attribute weights. In earlier research with temporal templates, the values for the weights were manually determined by the researcher adjusting the values until the results were satisfactory. [Drewes, 1997] That was possible for the limited domain that was involved. However, for a much larger domain, such as for the ground vehicle in a battlespace with many possible courses of action, a practical method of automating the setting of the weights by observation is essential. This not only will allow the determination of the weights, but will also allow the templates to be readily created to match various vehicle entities whose actions may differ in the same circumstances. For example, generalized templates could be tailored for individuals/crews whose training levels are expert, average or novice and specialized templates could be created to represent a specific individual/crew for most efficient representation. It is anticipated that neural networks or genetic algorithms could be used for determining these weights.

Behavioral Vehicle Model Corrections On the list of model synchronization challenges is the process of determining what corrections to make to the vehicle model and how to make those corrections. The corrections can be categorized into one of the following four types, where the correction to apply is shown in parentheses:

(1) State Realignment Update, where the position and orientation state differences between the vehicle model and the actual vehicle exceed an allowable threshold while the vehicle model is operating in the same context as the actual vehicle. (The vehicle model's state is updated.)

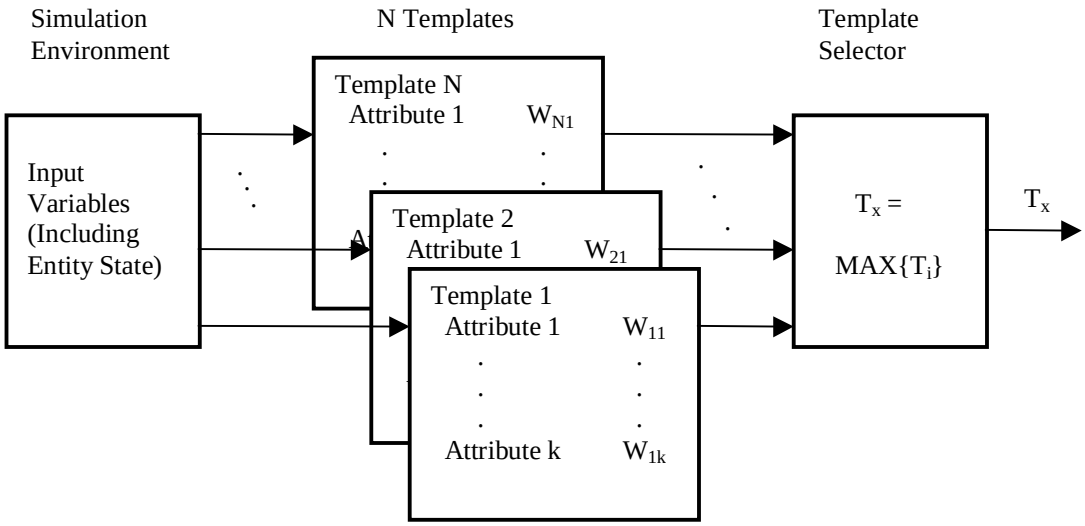

Figure 2. Temporal Template Components. The score for each template is the sum of the weighted values for the attributes or zero, if not above a critical value. The selected template has the largest score.

(2) Model Bias Correction, where the state differences between the vehicle model and the actual vehicle exceed an allowable threshold and a significant pattern of error in the output of the vehicle model is observed. (The vehicle model is modified to remove the observed bias and the vehicle model's state is updated.)

(3) Context Shift, where the state differences between the vehicle model and the actual vehicle exceed an allowable threshold and the context of the vehicle model is different than the observed context of the actual vehicle. (The context of the vehicle model is modified and the vehicle model's state is updated.)

(4) Model Suspension, where the state differences between the vehicle model and the actual vehicle exceed an allowable threshold but the context of the vehicle model cannot be determined. (The vehicle model's behavior is suspended and simple dead-reckoning is used.)

The type of correction to make is determined by comparison of the state differences and context differences between the vehicle model and the actual vehicle. This can be viewed as a Finite State Machine (FSM) implementation with the four types of corrections along with the case of no update required as the fifth state.

System Architecture Finally, the entire system has to be integrated into a workable architecture. It is anticipated that the DAE evaluation will occur immediately following each completed VM controlled update cycle of the vehicle model state data stores. The vehicle model will update the present state of each represented vehicle based on extrapolation using its last vehicle model state (position, velocity, orientation), predicted actions to take (based on the determination of its context at the last update), and elapsed time between the previous and current update cycle. For each PDU received since the previous update cycle, the VM would modify its vehicle's state information to reflect the PDU update.

After the vehicle model cycle is completed and the VM has updated its host vehicle model's current state and context, the difference analysis engine will compare the actual host vehicle's state and its determination of the host vehicle context against the vehicle model's currently determined state and context. If updates are required, the DAE will send out an update PDU to the vehicle model on the same host vehicle as the DAE and to the

identical vehicle models for the actual entity located on the other vehicles.

Temporal Template Selection and Training

This section discusses the temporal templates, the basis for their development, and how they will be trained using learning by observation.

Temporal Templates The use of temporal templates for template-based reasoning is similar to case-based reasoning except that the comparison of attributes will be time-dependent as well as potentially sequenced. That is, it can allow for the requirement that an event or sequence of events must precede another event, or have occurred within a certain time period before it, to increase the template weighting in support of the template.

Context/Sub-context Selection This VMGOES research is being conducted using a two tracked approach with both a ModSAF tank entity and a Manned Module of the M1A2 serving as the host vehicle being modeled. The purpose of this approach is to first develop the system and computer code with the ModSAF simulator system before involving the more expensive to operate Manned Module simulator with trained, human crewmembers. However, for the contexts and sub-contexts identification, a critical link between the VM and DAE since they form the basis for the actions that the VM is attempting to model, the result is two different sets of templates. This occurs because the ModSAF operator has to choose from a preset selection of contexts for the actions and they do not necessarily match the doctrinal contexts that would be used internally by an M1A2 tank crew. Additionally, the VMGOES research is restricted in scope to a Road March with the only transitions allowed being those due to the presence of obstacles or opposing forces.

Matching Templates with VM Contexts/Sub-contexts In the initial assessments of contexts for ModSAF, 29 were identified as applicable for a complete system with only four required for the VMGOES scenarios. These were for Road March, Contact Drill, Assault, and Withdraw. For use with the M1A2 Manned Simulator, the doctrinally based Combat Instruction Sets (CIS's) developed for the U. S. Army's Close Combat Tactical Trainer simulator system were evaluated to determine an initial set of contexts. Of the 47 that were identified as being needed for a complete system, eight were found to be required. They were for Conduct Tactical Road March, Execute Contact Drill, Execute Action Drill (Front/Right/Left/Rear), Take Actions at an Obstacle, and Perform Assembly Area Activities.

Each of these major contexts have a set of attributes that distinguish between them and a set of associated sub-contexts that determine what actions within the major context are being performed. The number of attributes were four or five for the VMGOES ModSAF major context templates and ranged from three to seven for the VMGOES CIS-based major context templates. For example, the CIS-based Conduct Tactical Road March template had only four attributes (see Table 1). Except for possibly the first attribute, an evaluation would have to be performed to determine a value for the attribute. Other templates, called Supporting Templates, which will be discussed later, would perform these evaluations.

Table 1. Template Attributes for Conduct Tactical Road March Template (CIS-based)

(1) Moving
(2) Meeting fixed times for waypoints
(3) Road nets available nearby
(4) Enemy contact unlikely

The number of sub-context templates for the VMGOES major context templates ranged from two to seven for ModSAF templates and from three to 12 for the CIS-based templates. To use again the same CIS-based Conduct Tactical Road March template as an example, 11 sub-context templates were identified (see Table 2). Note that the action associated with the eleventh sub-context template would control the turret. That action would occur simultaneously with the action associated with one of the other 10 sub-context templates, each of which would affect the action of the tank hull.

Table 2. Sub-Context Templates for Conduct Tactical Road March (CIS-based)

(1) Movement to start point
(2) Follow road segment-starting
(3) Follow road segment at 75% of march speed
(4) Follow road segment-increase to march speed
(5) Follow road segment at march speed
(6) Follow road segment-increase to catch up

speed
(7) Follow road segment at catch up speed
(8) Follow road segment-decrease to march speed
(9) Follow road segment-decrease to 75% march speed
(10) Follow road segment-halting
(11) Turret scan relative to platoon position

Each sub-context template also has a set of attributes to distinguish between them. For example, the fourth sub-context from Table 2, "Follow road segment-increase to march speed", has three attributes identified (see Table 3). Others may be added later, if needed.

Table 3. Template Attributes for Follow Road Segment-Increase To March Speed Sub-Context Template (CIS-based)

(1) After start point time
(2) After critical point time and past critical point
(3) Accelerating between 75% and 100% march speed

Supporting Templates Various supporting templates will be defined to abstract the environment into useable attributes for other temporal templates. They provide an evaluation, using their own attributes, to determine a value for the attribute of interest. Note that supporting templates may also use other supporting templates for more than one layer of abstraction and to reduce the redundant computation of attribute values when the same attribute is used by more than one template. For example, the fourth attribute from Table 1, "Enemy contact unlikely", would be used by more than one template. As a supporting template, it would have to be determined only once in a vehicle model's computation cycle, regardless of how many templates used it as an attribute.

Automated Learning by Observation Once the contexts and sub-contexts have been established, the task become one of determining how to capture the knowledge about recognizing the various contexts and sub-contexts and discriminating between them. The inputs that are of significance to the crew in making their decisions about which "action" to take would logically be the inputs to consider. A Subject Matter Expert (SME) can be consulted to provide an initial cut of which factors are important. Additional research can also be conducted to determine if other attributes may be of use. The ultimate limit on what can be used is the physical constraint on what variables are, in fact, available within the simulation to be observed.

However, although a large number of observations can be made and collected, the question still remains about how to reduce that data into a form that can be used. One can make the analogy that each template is similar to a single layer neural network, i. e., the set of inputs (attributes) are each individually multiplied by their own weights and the result is summed to produce an output (see Figure 3). That being so, a neural network training procedure, such as back-propagation, could be used to set the value of the weights. In this case, the template score, the sum of each input attribute multiplied by its associated weight, corresponds to the output of the neural network with a linear activation function.

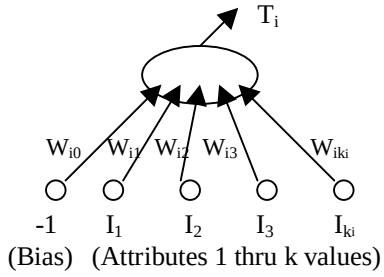

Figure 3. Attribute Learning by Observation Using Neural Network Framework

As an alternative training method, genetic algorithms may also be considered since the information to use for training them would be very similar. The same set of training examples and outputs could be used for either training method. One could have values for genes that would be used to make up the template weights. They could be recombined or allowed to mutate in the training process and only the best weights would be kept for each succeeding generation. Yet another approach to training could use a combination of methods, such as initial training with genetic algorithms and refinement with neural network training.

To address the template cut-off score, an arbitrary value can be set, such as 0.5, below which each template representing a doctrinal context will not be considered. This has utility in that the uses of

some contexts are anticipated which will be simple actions not representing doctrinal decisions. For example, the DAE might be able to recognize that the actual vehicle is heading in a particular general direction but that the action doesn't have relationship to any known context. Passing that behavior which is not matched with any doctrinal context to the VM may still allow the VM to adequately model the action. Thus, some lower level behaviors may be added as contexts that would have a cut-off of, perhaps 0.25, but a maximum value for the template restricted to less than 0.5. This would allow the lower level behaviors to be considered only when no doctrinal context can be found.

Development of Training Examples To train using back-propagation, each example set of inputs will have to have an output for each template to train against, since the relative values of the templates must be considered as well as the minimum cut-off value. Also, since each of the templates will be using their own, probably different, sets of inputs, all the possible inputs may have to be collected for each example.

Next, the development of training examples is complicated by the fact that a vehicle in one context with an observable set of inputs might have the same or very similar action in response to those inputs were it to be in a different context or contexts. Thus, the training examples could give conflicting information to other templates if the only output considered as valid is the context that the vehicle was actually in that generated that particular example. Therefore, either every example may have to be evaluated as to how well it would match with other template actions to produce an output for training the other templates, or else it might have to be used for training only the template that generated it. One could argue that if the outputs were so similar, it would make no difference if the wrong context were selected since the context changes are only sent out when a PDU is required. The research will investigate that possibility, since it could result in a much simpler generation of training examples where each example is used only for a positive output for the context that generated it.

This leads to the question of what schema to use for the output value for each training example. Not considering the arguments given in the preceding paragraph, several possibilities exist and the following three will be investigated.

First, a simple binary value could be used, where the template that generated the example is given a value of one and all other templates are assigned a value of zero. This could be readily used with the ModSAF templates, since the context is definitely known and could be recorded along with the inputs at the time that the example was generated. For the Manned Module, either the crew and/or a trained observer will have to determine and record the actual context being used.

For a second schema, continuous values could be chosen between zero and one where values of 0.5 to one are used for correct templates and zero to 0.5 are used for incorrect templates. A value of one represents complete certainty that the match is correct, zero represents complete certainty that the match is incorrect, and values in between those represent a gradation of certainty. The values would have to be determined by a SME or trained observer. In this schema, the same examples do not need to be used with every template and the templates are not given values based on the values given to other templates. However, for proper training, examples that do not represent the context would have to be used to ensure that the templates would score below the cutoff value for inputs that do not represent the context of the template. This schema would allow expansion of the system with additional templates being added without having to necessarily retrain the existing templates.

For the third schema, consideration of other templates is introduced into the output values, but evaluation by a SME or trained observer is still required. The following numerical values with verbal descriptions could be used as guidelines in determining what value to assign to a template for a given training example:

(1) 1.0, if absolutely confident of correct match and no other contender exists,

(2) 0.8, if strong confidence of correct match and no other contender is more likely,

(3) 0.6, if possible match, but other contenders are more likely,

(4) 0.4, if unlikely match, but cannot be ruled out entirely, or

(5) 0.0, if absolutely confident of no match.

This schema would be used with the output for all the possible contexts considered for each training example. It would be the most comprehensive set of training examples and would address the problem of a set of inputs being valid for multiple contexts. Unfortunately, it would also be the most labor intensive for development of the training examples and would require an almost complete retraining of the templates whenever a new context/template were added.

SUMMARY

While an embedded behavioral vehicle model within a distributed simulation may be able to reduce the communication needs below what would be required using current dead-reckoning procedures, it still must be synchronized with the actual vehicle and its actions to be effective. This research is exploring the methods that can be used to synchronize the behavioral vehicle model with both the position and orientation of the actual vehicle as well as the actions that it is performing. The action synchronization includes modifications of both the context of the vehicle model and the predictions that it makes, where the context of the actual vehicle is determined using temporal template matching and the prediction modification needed is determined by observing biases in the errors.

ACKNOWLEDGEMENTS

This work was sponsored by the U.S. Army Simulation, Training, and Instrumentation Command as part of the Inter-Vehicle Embedded Simulation and Technology (INVEST) Science and Technology Objective (STO), contract N61339-98-K-0001. That support is gratefully acknowledged.

CITED REFERENCES

Bahr, H., (1997). Embedded simulation for ground vehicles. Proceedings of the Spring 1998 Simulation Interoperability Workshop. Orlando, FL.

Bahr, H. and DeMara, R. F., (1996). A concurrent model approach to scaleable distributed interactive simulation. Proceedings of the 15th Annual Workshop on DIS, Institute for Simulation and Training. Orlando, FL.

Drewes, P. J. (1997). Automated student performance monitoring in training simulation. Unpublished doctoral dissertation, University of Central Florida, Orlando.

Drewes, P. and Gonzalez, A., (1995). Instructor assistance using template based reasoning. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Volume 2, 1918-1923. Piscataway, NJ.

Gonzalez, A. J. and Ahlers, R. H., (May 1995). Context-based representation of intelligent behavior in simulated opponents. Proceedings of the 5th Computer Generated Forces and Behavioral Representation Conference. Orlando, FL.

Gonzalez, A. J., DeMara, R. F. and Georgiopoulos, M., (March 1998). Vehicle model generation and optimization for embedded simulation. Proceedings of the Spring 1998 Simulation Interoperability Workshop. Orlando, FL.

Gonzalez, A. J., DeMara, R. F. and Georgiopoulos, M., (May 1998). Automating the CGF model development and refinement process by observing expert behavior in a simulation. Proceedings of the 7th Computer Generated Forces and Behavioral Representation Conference. Orlando, FL.

Gonzalez, A. J. and Murillo, M., (1998). Validation of human behavioral models. Proceedings of the Spring 1999 Simulation Interoperability Workshop. Orlando, FL.

Henninger, A., Gerber, W., DeMara, R., Georgiopoulos, M., and Gonzalez, A., (1998). Behavior modeling framework for embedded simulation. Proceedings of the 20th Interservice/Industry Training, Simulation and Education Conference. Orlando, FL.