Modeling Semi-Automated Forces with Neural Networks: Performance Improvement through a Modular Approach

Amy E. Henninger
Avelino J. Gonzalez
Michael Georgipoulos
Ronald F. DeMara
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2450
amy@isl.engr.ucf.edu

Keywords:
Neural networks,
Human behavior representation

ABSTRACT:. *A recent report by the National Research Council (NRC) declares neural networks "hold the most promise for providing powerful learning models". While some researchers have experimented with using neural networks to model battlefield behavior for Computer Generated Forces (CGF) systems used in distributed simulations, the NRC report indicates that further research is needed to develop a hybrid system that will integrate the newer neural network technology into the current rule-based paradigms. This paper supports this solicitation by examining the use of a context structure to modularly organize the application of neural networks to a low-level Semi-Automated Forces (SAF) reactive task. Specifically, it reports on the development of a neural network movement model and illustrates how its performance is improved through the use of the modular context paradigm. Further, this paper introduces the theory behind the neural networks' architecture and training algorithms as well as the specifics of how the networks were developed for this investigation. Lastly, it illustrates how the networks were integrated with SAF software, defines the networks' performance measures, presents the results of the scenarios considered in this investigation, and offers directions for future work.*

1. Introduction

The combination of computer simulation and networking technologies has provided military forces with an effective means of training through the use of Distributed Interactive Simulation (DIS). DIS is an architecture for building large-scale simulation models from a set of independent simulator nodes that represent entities in the simulation [1]. These simulator nodes individually simulate the activities of one or more entities in the simulation and report their attributes and actions of interest to other simulator nodes via the network. DIS nodes simulating combat vehicles, such as M1 Abrams tanks, are crewed by soldiers being trained. The trainees operate the controls of the simulators as they would in the actual vehicles, and the simulators implement actions in the simulated battlefield. Since, in a synthetic battlefield, the trainees need opposing forces against which to train, a type of DIS node known as a Computer Generated Force (CGF) system was developed.

CGFs are computer-controlled behavioral models of combatants used to serve as opponents against whom trainees can fight or as friendly forces with which the trainees can fight. At a minimum, the behavior generated should be feasible and doctrinally correct. For example, behaviors should be able to emulate the use of formations in orders, identify and occupy a variety of tactical positions (e.g., fighting positions, hull down positions, turret down positions, etc), and plan reasonable routes.

Researchers in [2], [3], and [4] have experimented with using neural networks to model battlefield behavior for CGF systems used in military simulations. This technology has been identified as one that "holds the most promise for providing powerful learning models" in a recent National Research Council Report [5]. Also asserted in this report, however, is the need for further research to develop hybrid systems that will integrate the newer neural network technology into the current rule-based paradigms. This investigation considers one such approach by using a framework based on modular decomposition to develop and apply the neural networks generating SAF behavior. Specifically, this research examines the performance improvements made to a

neural network based near-term movement model by adopting a modular approach.

2. Modular Decomposition

The use of a modular approach to a modeling task can be beneficial in a variety of ways. For example, it can be used for the purposes of improving performance. In other words, although the task could be solved with a monolithic set, better performance is achieved when it is broken down into a number of expert modules. Once the task is decomposed it is possible to switch to the most appropriate module, depending on the current circumstances or context. Switching has been discussed in the control literature [6][7], as well as the literature on behavior-based robotics [8].

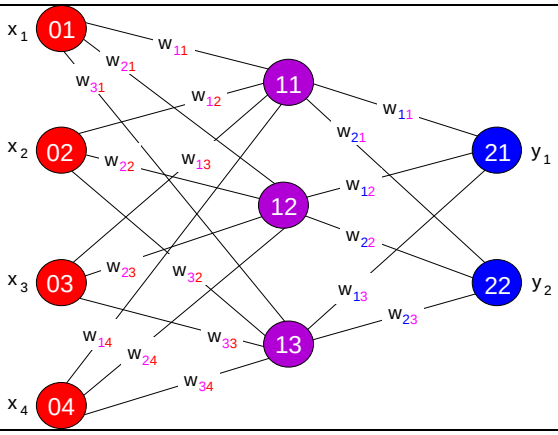
In addition to performance improvement, other motivations for adopting a modular approach to a problem include a reduction in model complexity and construction of the overall system such that it is easier to understand, modify, and extend. Thus the "divide and conquer" principle is used to reduce the complexity of a single net system. This enables the use of different neural net architectures or algorithms to be applied to individual sub-problems, making it possible to exploit specialist capabilities. Moreover, where appropriate, some of these components could make use of non-neural computing techniques. This justification has been noted [9][10] and is common to engineering design in general. Another motivation for adopting a modular approach is the reduction of network training times [11]. Finally, in well-defined domains, the use of a priori knowledge can be used to suggest an appropriate decomposition of a task. This approach complements the knowledge acquisition efforts and knowledge representation paradigms used in current SAF systems [12] and can be easily extended to the acquisition of knowledge and tactics for SAF systems [13].

The decomposition of a problem into modular components may be accomplished automatically or explicitly. When the decomposition of the task into modules is determined explicitly, this usually relies on a strong understanding of the problem. The division into sub-tasks is known prior to training [14], and improved learning and performance can result. An alternative approach is one in which the task is automatically decomposed according to the blind application of a data partitioning technique. Automatic decomposition is typically applied with the intent of performance improvement, whereas explicit decomposition could have the aim of either improving performance or accomplishing tasks that might not be accomplished as easily or as naturally with a monolithic net.

3. Neural Networks

A variety of researchers have worked in modeling human driving skills such as acceleration, steering, and vehicle following with neural networks [15], [16], [17], and [18]. A neural network is a collection of simple processors or nodes interconnected with each other that learn from examples and store the acquired knowledge in their interconnections, referred to as weights. Neural networks can solve a variety of problems related to non-linear regression and discriminant analysis, data reduction, and non-linear dynamic systems. One of the practical characteristics of neural networks is that they lend themselves to parallel-distributed processing using simple processing units rather than a complex CPU. This makes their execution very fast.

The multi-layer feed-forward network is one of the more typical network designs used in neural network applications. The example in Figure 1 is a 3-layer feed-



Where

- x1, x2, x3 and x4 are inputs (k = 1 to 4)
- y1 and y2 are outputs (i = 1 to 2)
- there are three hidden nodes (j = 1 to 3)
- there are three layers (L = 0,1,2)
- node "12" is node 2 in layer 1
- weight "12" connects node 1 from layer L+1 to node 2 of layer L

Figure 1. 4-3-2 Feed-Forward Architecture

forward network with four nodes in the first layer representing each dimension of the input vector, two nodes in the last layer representing each dimension of the output vector, and a hidden layer consisting of three nodes. This network attempts to develop a matching function between the input and output vectors by using some training algorithm. One of the more popular training algorithms is a method known as back-propagation. This method is based on finding the outputs

at the last layer of the network, calculating the errors between the actual and the predicted outputs, and then adjusting the network weights to minimize the error. Weight changes are implemented in a backward fashion starting from the weights converging to the output layer and proceeding backwards to the weights that converge to the hidden layer closest to the output layer. These computations are repeated such that the error is propagated back until the weights converging to the hidden layer closest to the input layer are reached.

In short, back-propagation involves a two step process. The first step, the forward pass, propagates the effects of the inputs forward through the network to reach the output layer. This step is governed by three forms of equations. First, in equation 1.1, the total weighted input to the j^{th} node for pattern p is given by:

$$net_j^{(1)}(p) = \sum_{k=1}^{K=4} x_k^{(0)}(p) w_{jk}^{(1)} \quad \text{for j = 1 to 3} \tag{1.1}$$

Next, the output of the jth node, $y_j^{(1)}(p)$, is given by:

$$y_j^{(1)}(p) = g(net_j^{(1)}(p)) = \frac{1}{1+e^{-net_j(p)}} \tag{2.1}$$

where $g()$ is the frequently used sigmoid function.

The net input to the i^{th} node for pattern p is similar to equation (1.1) and is given by:

$$net_i^{(2)} = \sum_{j=1}^{J=3} y_j^{(1)}(p) w_{ij}^{(2)} \quad \text{for i = 1 to 2} \tag{1.2}$$

and the output of the i^{th} node for pattern p, $y_I(p)$, is given by:

$$y_i^{(2)}(p) = g(net_i^{(2)}(p)) = \frac{1}{1+e^{-net_i(p)}} \tag{2.2}$$

where $g()$ is shown as the sigmoid function.

Lastly, the error function associated with the p^{th} input/desired output ($d_I(p)$) pair, is given by:

$$E^p(w) = \frac{1}{2}\sum_{i=1}^{I=2}[d_i^{(2)}(p) - y_i^{(2)}(p)]^2 \tag{3}$$

Once the error is computed, the weights are adjusted such that E^p is minimized. This occurs in the second pass, the backward pass, by computing the negative gradient of the error function and taking the partial derivatives of this function with respect to the weights (equations 5.1 and 5.2). This allows errors at the output layer to be propagated backward toward the input layer in proportion to the change in activity at the previous layer.

$$\Delta w_{ij}^{(2)} = -h\,(\frac{\partial E}{\partial w_{ij}^{(2)}}) \tag{4.1}$$

$$\Delta w_{jk}^{(.)} = -h\,(\frac{\partial}{\partial w_{jk}^{(.)}}) \tag{4.2}$$

h is a user defined parameter.

By applying the chain rule of derivation (see [19] for complete derivation), equations 5.1 and 5.2 reduce to:

$$\Delta w_{ij}^{(2)} = h d_i^{(2)}(p) y_j^{(1)}(p) \quad i = 1,2 \quad j = 1,2,3 \tag{5.1}$$

$$\text{where } d_i^{(2)}(p) = g'(net_i^{(2)}(p))[d_i^{(2)}(p) - y_i^{(2)}(p)]$$

$$\Delta w_{jk}^{(1)} = h d_j^{(1)}(p) x_k(p) \quad j = 1,2,3 \quad k = 1,2,3,4 \tag{5.2}$$

$$\text{where} \quad d_j^{(1)}(p) = g'(net_j^{(1)}(p))\sum_{i=1}^{I=2} w_{ij}^{(2)} d_i^{(2)}(p)$$

For the example in Figure 2, these equations reduce to:

$$\Delta w_{ij}^{(2)} = h\,(y_i(p))(1-y_i(p))(d_i(p)-y_i(p))\cdot (y_j^{(1)}(p)) \tag{6.1}$$

$$\Delta w_{jk}^{(1)} = h\,(y_j(p))(1-y_j(p))\sum_{i=1}^{I=2}[(y_i(p))\cdot (1-y_i(p))(d_i(p)-y_i(p))w_{ij}^{(2)}]\cdot x_k(p) \tag{6.2}$$

Each of these weight adjustments directs the network towards a solution to the input/output mapping. That is, these weights are training the network to produce a certain output given a set of inputs. This is one of the fundamental benefits of the neural network approach. With the proper training and representation, the network will self-organize to arrive at a mapping of how the responses are formed and there is no need to acquire and represent an expert's knowledge in terms of rule sets.

4. Methodology

This study used ModSAF, a CGF system for training and research, to focus on the near-term movement behavior of a SAF. The near-term movement behavior was selected because it is computationally challenging and highly observable [20], [21], [5]. It also provides a direct correspondence to ESPDUs resulting from errors in entity position. This proves useful in finding a way to measure the performance of this study as explained in the next section (section 5). However, since moving in the battlefield is a highly complex behavior depending on many factors, the problem was scoped to specifically consider how a single entity (i.e., a ModSAF M1A2) performs a "Road March". This is accomplished by estimating the changes in an M1A2 entity's speed and orientation given its previous state and the previous state of the simulated world

For this application, a feed-forward architecture with back-propagation training was used in each of experimental systems. In the first system, Model 1, two

networks were considered. One of these networks predicts the change in an entity's speed and the second network predicts the change in the entity's heading. In the second system, Model 2, four networks were considered. Of these four neural networks, two predict the change in the entity's speed and the other two predict the change in the entity's orientation. Each of these pairs of networks can be further distinguished by whether the M1A2 entity is traveling a straight segment of the road or is entering a turn. The rule used to distinguish between these segment types was acquired from the code used to generate the SAF's behavior and is based on the straight-line distance to the current waypoint. This decomposition category is referred to as a "context" and the change from one context to another context is referred to as a "context transition" [22]. In this problem, for example, there are two contexts: straight and turning. Also, there is a rule defining when to shift between contexts:

if distance to next waypoint is <= 25 m,
context = turning
else,
context = straight.

This rule represents an explicit decomposition of the task and the application of this or other similar rules results in the generation of four networks identified as straight-speed (SS), straight-heading (SH), turn-speed (TS), and turn-heading (TH).

Each network used a sigmoid function at the hidden nodes and a linear transformation at the output nodes. The configuration of the networks in each of the models may be seen in Table 1 and the inputs were normalized according to equations 7.1 – 7.18 below. Fundamentally, the inputs for each of the networks were a function of the M1A2 entity's state at the last simulation clock and how this state related to the road characteristics and March Order parameters.

$$S_t = S_{t-1} + \Delta S_t$$
$$\text{where } \Delta S_t = f(Ra_{t-1}, Rb_{t-1}, Rc_{t-1}, Rp_{t-1}, Rs_{t-1}, HRab_{t-1}, HRbc_{t-1}, Hz_{t-1}) \quad (7.1)$$

$$q_t = q_{t-1} + \Delta q_t$$
$$\text{where } \Delta q_t = f(Ra_{t-1}, Rb_{t-1}, Rc_{t-1}, Rp_{t-1}, Rs_{t-1}, HRab_{t-1}, HRbc_{t-1}) \quad (7.2)$$

where

$$Ra_t = S_t / (Da_t + M) \quad (7.3)$$
$$Rb_t = S_t / (Db_t + M) \quad (7.4)$$
$$Rc_t = S_t / (Dc_t + M) \quad (7.5)$$
$$Rp_t = S_t P_t / M \quad (7.6)$$
$$Rs_t = S_t / M \quad (7.7)$$
$$HRab_t = Hab_r \times Hxy_t \quad (7.8)$$
$$HRbc_t = Hbc_r \times Hxy_t \quad (7.9)$$
$$S_t = \text{entity speed at } t \quad (7.10)$$
$$Da_t = \text{distance to previous waypoint} \quad (7.11)$$
$$Db_t = \text{distance to current waypoint} \quad (7.12)$$
$$Dc_t = \text{distance to next waypoint} \quad (7.13)$$
$$M = \text{march order speed} \quad (7.14)$$
$$P_t = \text{perpindicular distance to road} \quad (7.15)$$
$$Hab_t = \text{direction of road segment ab} \quad (7.16)$$
$$Hbc_t = \text{direction of road segment bc} \quad (7.17)$$
$$Hxy_t = \text{entity orientation} \quad (7.18)$$

		Context	Architecture	Predictors	Response
Speed Networks	**Model 1**	No context Switching	8-20-5-1	$Ra_{t-1}, Rb_{t-1}, Rc_{t-1}, Rp_{t-1}, Rs_{t-1}, HRab_{t-1}, HRbc_{t-1}, Hz_{t-1}$	ΔS_t
	Model 2	Straight (SS)			
		Turning (TS)			
Heading Networks	**Model 1**	No context switching	7-20-5-1	$Ra_{t-1}, Rb_{t-1}, Rc_{t-1}, Rp_{t-1}, Rs_{t-1}, HRab_{t-1}, HRbc_{t-1}$	Δq_t
	Model 2	Straight (SH)			
		Turning (TH)			

Table 1. Architecture by Model Type

Data for training and validation were gathered over the 45, segment route shown in Figure 2. This route is

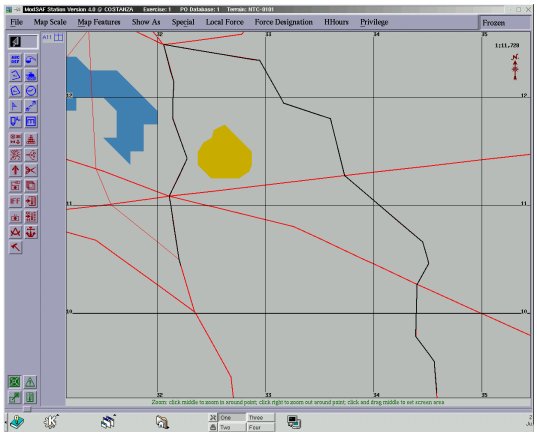


Figure 2. Route Used for Neural Network Training

approximately 7 kilometers long and took the tank about 15 minutes of simulation time to travel at a March Order speed of 8 m/s. From this period, a total of 13760 examples were generated at a rate of 15 HZ.

Of these, Model 1 used 860 examples for training the speed network, 860 examples for training the heading network, and 859 examples for validating the training of both of these networks. Alternatively, Model 2 used 568 examples for training the straight-heading (SH) network, 292 examples for training the turn-heading (TH) network, 100 examples for training the turn-speed (TS) network, and 760 examples for training the straight-speed (SS) network. An approximately equal number of examples were used for validating the training results of each network. The training rate was selected as 0.01 and the initial momentum parameter was .9. The momentum parameter was periodically adjusted to speed the rate of descent along the error surface. The training and validation results for each of the four networks may be seen in Table 2.

5. Experimental Results

Essential to the task of determining whether one model out-performed another is a metric to make such a comparison. Validating SAF models has typically been performed subjectively by SMEs and the DIS community has no known quantitative performance measure to evaluate the performance of a SAF near-term movement model. However, given the level of resolution of SAF maps, it is impractical to assume that a SME could detect a noticeable difference in models due to the addition of a context shift. In other words, even if a human observer could visibly discriminate between two different types of movement models, it is unlikely that he could visibly detect the difference in the same movement model represented by a monolithic neural network versus represented by a module of networks. Because the SME validation and comparison of the models used in this research was susceptible to error, the investigators made use of the DIS entity state synchronization concept to evaluate model performance. This was accomplished by implementing each of the models as dead-reckoning models (as opposed to SAF behavioral models) and then comparing the numbers of PDUs generated by each of models 1 and 2. Implicit in this measure of performance is the assumption that the model with the lower PDU count is the model that best fits the source data used to develop the model and from which the PDU count is derived.

The comparison of the entity's true position and the position according to the dead-reckoning model occurs in the ModSAF *libentity* library. As such, the neural network models used in this investigation replaced the dead-reckoning code in the *libentity* library. The implementation of this functionality in ModSAF is illustrated in Figure 3.

	Delta Speed ΔS **Error(m/s)**	**Delta Heading** Δq **Error(rads)**	**Context**	**Model Type**
Training	0.259977±2.045589	0.004578±0.007818	**N/A**	**No Context Shifting Model (1)**
Validation	0.206374±0.825324	0.014221±0.067663		
Training	0.179351±1.672473	0.000150±0.0022	**straight**	**Context Model (2)**
Validation	0.083991±0.299115	0.000153±0.0024		
Training	0.097374±0.268849	0.002010±0.0174	**turning**	
Validation	1.417666±0.748291	0.002704±0.0175		

Table 2. Training and Validation Mean Square Error and Standard Deviation for Models 1 and 2

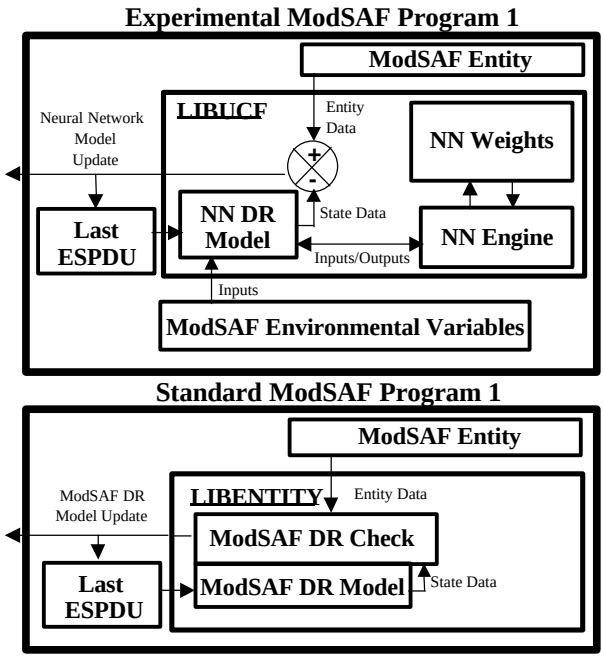


Figure 3. Functional Relationship of Neural Networks to ModSAF Dead-Reckoning Code

The PDUs, evidenced as errors in location or orientation generated by the monolithic neural network, Model 1, was compared with the number of PDUs generated by the context-based neural network, Model 2. As evidenced in Figure 4, there was an 11% reduction in Model 2's total PDU count over Model 1's PDU count. This suggests that the performance of neural networks applied to a low-level SAF reactive-task can be improved by the use of the context-based task decomposition scheme.

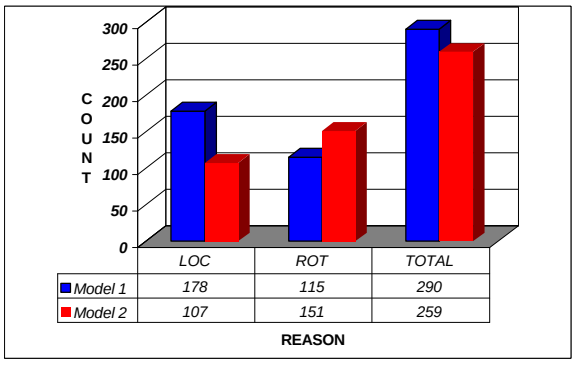


Figure 4: PDU Results from Model 1 and Model 2

6. Future work

The use of the context-based decomposition scheme (Model 2) resulted in a PDU reduction of 11% over the monolithic neural network model (Model 1). This suggests that the performance of neural networks applied to a low-level SAF reactive-task is improved by the use of the context-based task decomposition scheme. Future work in this aspect of the study includes investigating methods of automating the learning of the task decomposition and hence, the context-shifting rules. Also, the improvement of the neural networks' performance continues to be explored. This includes considering alternative types of architectures, inputs, normalization schemes, and sampling strategies. Since the ModSAF infrastructure to collect data and evaluate models is now in place, more work can be done to improve these preliminary results.

7. 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.

8. References

[1] Smith, S., and Petty, M. (1992). Controlling Autonomous Behavior in Real-Time Simulation. In Proceedings of the Second Conference in Computer Generated Forces and Behavior Representation. Orlando, FL., May, 1992.

[2] Crowe, M. (1990). The Application of Artificial Neural Systems to the Training of Air Combat Decision-Making Skills. Proceedings of the 12th Interservice/Industry Training Systems Conference, Orlando, FL. Nov., 1990.

[3] Jaszlics, S.L. (1993). Artificial Intelligence in Tactical Command and Control Applications: Architecture and Tools. In Proceedings of the Third Conference on Computer Generated Forces and Behavior Representation. Orlando, FL., March, 1993.

[4] Morrison, J.D. (1996). Real-time Learning of Doctrine and Tactics Using Neural Networks and Combat Simulations. Military Operations Research, vol. 2, no. 3, pages 45-60.

[5] Pew, R.W., and Mavor, A.S., eds. (1998). Modeling Human and Organizational Behavior: Application to Military Simulations. Washington, DC: National Academy Press.

[6] Murray-Smith, R., and Johansen, T.A. (1997). Multiple Model Approaches to Modelling and Control. Taylor and Francis, UK.

[7] Narendra, K. S., Balakrishnan, J., and Ciliz, K. (1995). Adaptation and Learning Using Multiple Models, Switching and Tuning. IEEE Control Systems Magazine, June, 1995, pp. 37-51.

[8] Brooks, R.A. (1986). A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, RA-2:14-23.

[9] Gallinari, P. (1995). Modular Neural Net Systems: Training Of. In M.A., Arbib, editor, The Handbook of Brain Theory and Neural Networks, pp. 582-585. Bradford Books: MIT Press.

[10] Hrycej, T. (1992). Modular Learning in Neural Networks. John Wiley, Chichester.

[11] Pratt, L.Y., Mostow, J., and Kamm, C.A. (1991). Direct Transfer of Learned Information Among Neural Networks. In Proceedings of the Ninth National Conference on Artificial Intelligence (AAI-91), pp. 584-589. Anaheim, CA, 1991.

[12] Ourston, D., Blanchard, D., Chandler, E., and Loh, E., (1995). From CIS to Software, In Proceedings of the Fifth Conference on Computer Generated Forces and Behavior Representation. Orlando, FL., May, 1995, pp. 275-285.

[13] Henninger, A., and Gonzalez, A. (1997). Automated Acquisition Tool for Tactical Knowledge, In Proceedings of the 10th Annual International Florida Artificial Intelligence Research Symposium, Melbourne FL., May, 1997, pp. 307-311.

[14] Hampshire, J.B., and Waibel, A.H. (1992). The Meta-P,I Network: Building Distributed Representations for Robust Multiource Pattern Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(7): 751-769, 1992.

[15] Fix, E., and Armstrong, H.G., (1990). Modeling Human Performance with Neural Networks. Proceedings of the International Joint Conference on Neural Networks, vol. 1, pages 247-252.

[16] Nechyba, M.C. and Xu, Y. (1997). Learning and Transfer of Human Real-Time Control Strategies. Journal of Advanced Computational Intelligence, vol.1, pp. 137-154.

[17] Pomerlau, D., Thorpe, C., Longer, D., Rosenblatt, J.K., and Sukthankar, R., (1994). AVCS Research at Carnegie Mellon University. Proceedings Of Intelligent Vehicle Highway Systems America 1994 Annual Meeting, p. 257-262.

[18] Pomerlau, D.A.., Gowdy, J., and Thorpe, C.E. (1991). Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance. Journal of Engineering Applications of Artificial Intelligence 4(4): 279-285.

[19] Rummelhart, D., and McClelland, J. (1986). Parallel Distributed Processing. MIT Press, Cambridge, MA.

[20] Schricker, S.A., Franceschini, R.W., and Mukherjee, A. (1998). Feasibility of Hardware-Based Computer Generated Forces for Embedded Training, Proceedings of the 1998 Interservice/Industry Training, Simulation and Education Conference, Orlando, FL, November 30-December 3, 1998.

[21] Smith, J. (1994). Near-term Movement Control in ModSAF. In Proceedings of the Fourth Conference in Computer Generated Forces and Behavior Representation. Orlando, FL, May 1994.

[22] Gonzalez, A., and Ahlers, R. (1995). Context-based Representation of Intelligent Behavior in Simulated Opponents. In Proceedings of the Fifth Conference on Computer Generated Forces and Behavior Representation, Orlando, FL., May, 1995.

Author Biographies

AMY HENNINGER is a doctoral candidate in computer engineering at the University of Central Florida and a former Research Fellow for the Army Research Institute at U.S. Army STRICOM. She has earned B.S. degrees in Psychology, Industrial Engineering, and Mathematics from Southern Illinois University, an M.S. in Engineering Management from Florida Institute of Technology, and an M.S. in Computer Engineering from UCF.

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.

MICHAEL GEORGIOPOULOS is an Associate Professor at the Department of Electrical and Computer Engineering of the UCF. His research interests lie in the areas of neural networks, fuzzy logic and genetic

algorithms and the applications of these technologies in cognitive modeling, signal processing and electromagnetics. He has published over a hundred papers in scientific journals and conferences.

RONALD DEMARA is a full-time faculty member in the Electrical and Computer Engineering Department at the University of Central Florida. Dr. DeMara received the B.S.E.E.. degree from Lehigh University in 1987, the M.S.E.E. degree from the University of Maryland, College Park in 1989, and the Ph.D. degree in Computer Engineering from the University of Southern California, Los Angeles in 1992.