USING NEURAL-NETWORKS TO REDUCE ENTITY STATE UPDATES IN DISTRIBUTED INTERACTIVE APPLICATIONS

Aaron McCoy, Tomas Ward, Seamus McLoone and Declan Delaney

Department of Electronic Engineering,
National University of Ireland Maynooth,
Maynooth, Co. Kildare, Republic of Ireland.
E-mail: {amccoy,tomas.ward,seamus.mcloone}@eeng.nuim.ie

ABSTRACT

Dead reckoning is the most commonly used predictive contract mechanism for the reduction of network traffic in Distributed Interactive Applications (DIAs). However, this technique often ignores available contextual information that may be influential to the state of an entity, sacrificing remote predictive accuracy in favour of low computational complexity. In this paper, we present a novel extension of dead reckoning by employing neural-networks to take into account expected future entity behaviour during the transmission of entity state updates (ESUs) for remote entity modeling in DIAs. This proposed method succeeds in reducing network traffic through a decrease in the frequency of ESU transmission required to maintain consistency. Validation is achieved through simulation in a highly interactive DIA, and results indicate significant potential for improved scalability when compared to the use of the IEEE DIS Standard dead reckoning technique. The new method exhibits relatively low computational overhead and seamless integration with current dead reckoning schemes.

1. INTRODUCTION

Distributed Interactive Applications (DIAs) are virtual reality systems through which participants can share information via individual and collaborative interaction with each other and their environment [1]. They offer the realization of simulated virtual worlds that embody a modern extension of communication, encompassing the concepts of *shared time*, *space* and *presence* [2]. The definition of DIAs includes a wide range of applications that have seen rapid advances in technology and global popularity due to the widespread availability and ease-of-use of the Internet [3, 4].

The two key factors that limit large-scale deployment of DIAs are *network latency* and *bandwidth*. Bandwidth refers to the rate at which data can be communicated per unit time between two end-points, while latency refers to the delay in communication. High latency and low network bandwidth capacity represent the largest contributors to the difficulties faced by DIAs in supporting dynamic shared state consistency, potential scalability and real-time interactivity. A wide variety of techniques exist that aim to reduce the amount of network traffic transmitted during the execution of a DIA [2-4]. One of the most commonly used techniques is the predictive contract mechanism known as *dead reckoning*.

Formally defined within the IEEE Standard for Distributed Interactive Simulation (DIS) [5], dead reckoning reduces network traffic by preventing update packets from being sent if the local participant's state has not varied from a low-fidelity model of that state by a pre-determined error threshold. It often relies exclusively on the replication of instantaneous derivative information between controlling hosts for remote entity modeling. However, it ignores available contextual and a priori information that may be influential to the state of an entity, and sacrifices remote predictive accuracy in lieu of low computational complexity and resource usage [2].

This paper presents a novel extension of dead reckoning, termed *neuro-reckoning*, that uses neural-networks to allow for expected future entity behaviour when transmitting entity state updates (ESUs). Each controlling host employs a set of time-delayed neural-networks trained to predict future changes in entity velocity over a series of progressively increasing prediction horizons. Future changes in entity location are determined using a process of forward simulation through time, producing an estimate for the total expected change in entity location over a temporal interval defined by the maximum prediction horizon. On exceeding the error threshold, the controlling host issues an ESU containing the predicted neuro-reckoning velocity vector instead of the standard dead reckoning velocity vector. Hence, by distributing ESUs that implicitly encode future entity behaviour, neuro-reckoning succeeds in reducing the spatial error associated with remote entity modeling. This, in turn, achieves a reduction in network traffic through a decrease in the frequency of ESU transmission due to error threshold violation.

The remainder of this paper is structured as follows. A mathematical representation of the application domain is given in the next section. This is followed by a detailed explanation of the novel neuro-reckoning approach in Section 3. A short description of the experimentation is given in Section 4, while Section 5 presents the analysis and results, comparing the proposed method with the IEEE DIS Standard dead reckoning. Finally, the paper ends with some conclusions and directions for future research in Section 6.

2. THE APPLICATION DOMAIN

To demonstrate and validate our proposed neuro-reckoning framework within an application domain exhibiting a realistic scope, experiments are conducted under the guise of a simple First-Person Shooter (FPS) style game scenario, designed to produce patterns of repeatable human-user behaviour (suitable for training time-delayed neural-networks). The data from this genre of games can be considered a fair representation of the type of data one would expect to see in commercial networked multiplayer computer games [6].

In general, we assume the current state and the current action of an entity at time t to be described by real-valued feature vectors $\mathbf{s}_t \in \Re^n$ and $\mathbf{a}_t \in \Re^m$ respectively. Further, we assume the environmental (i.e. external) influences affecting the state of an entity at time t to be described by a real-valued feature vector $\mathbf{e}_t \in \Re^p$. Given these real-valued feature spaces, reactive modeling of user behaviour in the discrete-time domain can subsequently be viewed as a problem of function approximation. This is represented by the following non-linear input-output model describing the dependence of the *future change in entity state* i steps ahead on both the current and k previous states, actions and environmental influences:

$$\Delta \mathbf{s}_t^{t+i} = f(\mathbf{s}_{[t-k,t]}, \mathbf{a}_{[t-k,t]}, \mathbf{e}_{[t-k,t]}) \tag{1}$$

Data-based learning problems become exponentially more difficult with the increasing size of the state-space [7, 8]. To reduce the complexity of our particular state-space, we employ domain specific knowledge that is partly realized through a series of consciously introduced application constraints, allowing us to make assumptions regarding the information relevance of state-space features for adequately approximating the functional mapping presented in Eq. (1). The following assumptions are made:

1) *Action State-Space*: We assume the consequences of actions performed by entities are both implicitly and sufficiently encoded within the changes in entity state that occur during multiple consecutive time-steps;

2) *Environmental State-Space*: We assume that any environmental influences are plausibly negligible over a sufficiently fine temporal-granularity;

3) *Degrees of Freedom Constraints*: We assume that vertical-component space can be ignored by exclusively constraining the manipulation of entity state to two degrees of freedom along the horizontal plane;

4) *Translation and Rotation Invariant*: We assume that changes in entity state are translation and rotation invariant, meaning that entity reactions are independent of absolute location (this ties in with Assumption (2) above).

Assumptions (1) and (2) remove the dependence on the action and environmental influence state-spaces **a** and **e** respectively, reducing Eq. (1) to a problem of *multi-variate time-series prediction*, represented as:

$$\Delta \mathbf{s}_t^{t+i} = f(\mathbf{s}_t, \mathbf{s}_{t-1}, \mathbf{s}_{t-2}, \ldots, \mathbf{s}_{t-k}) \tag{2}$$

From our FPS scenario, we assume the state of an entity at time t to be accurately represented by a series of location $\mathbf{L}_t$, velocity $\mathbf{V}_t$, and orientation $\mathbf{O}_t$ vectors, jointly comprising a 9-dimensional real-valued state-space:

$$\mathbf{s}_t = (\mathbf{L}_t, \mathbf{V}_t, \mathbf{O}_t) : \mathbf{L}_t, \mathbf{V}_t, \mathbf{O}_t \in \Re^3 \tag{3}$$

Assumption (3) removes the dependence on vertical-component space (thus reducing from $\Re^3$ to $\Re^2$), while Assumption (4) removes the dependence on the location vector space **L**, reducing Eq. (3) to a more compact form:

$$\mathbf{s}_t = (\mathbf{V}_t, \mathbf{O}_t) : \mathbf{V}_t, \mathbf{O}_t \in \Re^2 \tag{4}$$

By combining Eqs. (2) and (4), we arrive at the final compact state-space representation, where k represents the *current time-delay* and i represents the *current prediction horizon*:

$$\begin{aligned}(\Delta \mathbf{V}_t^{t+i}, \Delta \mathbf{O}_t^{t+i}) &= f(\mathbf{V}_{[t-k,t]}, \mathbf{O}_{[t-k,t]}) \\ &= f(\mathbf{V}_t, \ldots, \mathbf{V}_{t-k}, \mathbf{O}_t, \ldots, \mathbf{O}_{t-k})\end{aligned} \tag{5}$$

For the remainder of this paper (and without a loss of generality), we shall restrict our attention exclusively to the prediction of changes in entity velocity only. Hence, as a final step, we remote the change in entity orientation from the output of the dependency relationship in Eq. (5):

$$\Delta \mathbf{V}_t^{t+i} = f(\mathbf{V}_t, \ldots, \mathbf{V}_{t-k}, \mathbf{O}_t, \ldots, \mathbf{O}_{t-k}) \tag{6}$$

3. THE NEURO-RECKONING APPROACH

Eq. (6) can be used to define a set of predictors **P** for the reactive modeling of user behaviour, represented by:

$$\mathbf{P} = \{ P_i \}, \forall\, i \in \{1, \ldots, q\}$$
$$P_i \xrightarrow{predicts} \Delta\mathbf{V}_t^{t+i} \qquad (7)$$
$$\mathbf{V}_{t+i} = \mathbf{V}_t + \Delta\mathbf{V}_t^{t+i}$$

In order to realize **P**, we employ a collection of time-delayed neural-networks (TDNNs) for predicting *future changes in entity velocity* over a series of progressively increasing prediction horizons (up to a total of q steps ahead), as given by Eq. (7) [9]. In this respect, a separate TDNN is trained for each prediction horizon, resulting in a total of q individual networks. Each TDNN consists of an input layer, a hidden layer, and an output layer arranged in a feed-forward architecture comprising an additional tap-delay line designed to delay the input signal by a total of k time-steps (see Figure 1), producing a total of $4(k + 1)$ input neurons, m hidden neurons, and 2 output neurons. Non-linear tan-sigmoid transfer functions are used by neurons contained within the hidden layer, while linear transfer functions are used by neurons contained within the output layer. Each TDNN utilizes the well-known mean squared error (MSE) performance criterion [7] and is batch trained using the popular Levenberg-Marquardt (LM) update rules for propagating the scaled output-error to individual network weights and biases [10]. As per Eq. (4), state vectors reside in a 2-dimensional vector space. As a result, neurons pictured as residing in either the input or the output layer represent two components of entity state, physically implemented as two separate neurons in the final model.

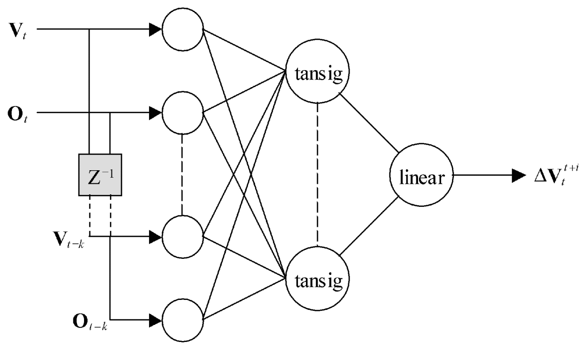


Figure 1: Time-delayed neural-network (TDNN) topology for a single prediction horizon i. Given a maximum prediction horizon q, a total of q such TDNNs (one for each prediction horizon) are required to predict entity state (see Eq. (7) above). Each TDNN contains $4(k + 1)$ neurons in the input layer, m neurons in the hidden layer, and 2 neurons in the output layer.

The set of predictors **P** provides controlling hosts with the capability to generate a representation for the expected evolution of *future entity velocity* up to a total of q steps ahead (for each entity under the host's control). In order to generate a *spatial* representation of the expected evolution of *future entity location*, controlling hosts can combine the velocity estimations by a process of forward simulation through time (a process we refer to as locally '*unfolding*' an entity trajectory). Future changes in entity location are estimated by simulating the effect of each predicted change in entity velocity using a standard first-order, one-step extrapolation equation over a constant simulation time-step Δt (equal to the sampling rate of the set of training exemplars used to train the predictors) (see Table 1). The net effect of this entire process is an estimate for the *total expected change in future entity location* over the temporal interval τ defined by the maximum prediction horizon q and the constant simulation time-step Δt, where $\tau = (q + 1)\Delta t$. This procedure of unfolding is illustrated in Table 1, presented as an iterative process that is performed over the progressively increasing prediction horizon i (this process is performed by a controlling host for a local entity every time an error threshold violation is detected).

Time	**TDNN Predictor**	**Entity Velocity**	**Entity Location**
t	——	$\mathbf{V}_t$	$\mathbf{L}_t$
$t+1$	$P_1 \xrightarrow{predicts} \Delta\mathbf{V}_t^{t+1}$	$\mathbf{V}_{t+1} = \mathbf{V}_t + \Delta\mathbf{V}_t^{t+1}$	$\mathbf{L}_{t+1} = \mathbf{L}_t + \mathbf{V}_t\Delta t$
………	………	………	………
$t+i$	$P_i \xrightarrow{predicts} \Delta\mathbf{V}_t^{t+i}$	$\mathbf{V}_{t+i} = \mathbf{V}_t + \Delta\mathbf{V}_t^{t+i}$	$\mathbf{L}_{t+i} = \mathbf{L}_{t+i-1} + \mathbf{V}_{t+i-1}\Delta t$
………	………	………	………
$t+q$	$P_q \xrightarrow{predicts} \Delta\mathbf{V}_t^{t+q}$	$\mathbf{V}_{t+q} = \mathbf{V}_t + \Delta\mathbf{V}_t^{t+q}$	$\mathbf{L}_{t+q} = \mathbf{L}_{t+q-1} + \mathbf{V}_{t+q-1}\Delta t$
$t+q+1$	——	——	$\mathbf{L}_{t+q+1} = \mathbf{L}_{t+q} + \mathbf{V}_{t+q}\Delta t$

Table 1: The iterative process of unfolding an entity trajectory.

To exploit the process of unfolding, controlling hosts must compress the resulting predictive state information for replication to remote hosts in a compact form that: (1) implicitly encodes the expected evolution of entity state in both temporal and spatial form, and (2) retains suitability for efficient remote extrapolation of entity state.

Both prerequisites can be satisfied by employing a suitable '*aggregation policy*' designed to summarize the predictive state information contained within the unfolded entity trajectory. In theory, an aggregation policy can assume any functional means intended to reduce the unfolded trajectory to a compact form for replication to remote hosts. On the other hand, the format of the expected output of an aggregation policy will be pre-defined should maintaining compatibility with existing predictive contract mechanisms (such as dead reckoning) be considered a priority – in our case, we wish to maintain compatibility with current standard first-order, one-step extrapolation mechanisms employed within many DIAs. As such, we require an aggregation policy that produces a *single predictive velocity vector*, implicitly describing the expected evolution of entity behaviour over time, while also maintaining suitability for replication in the form of a standard entity state update (ESU).

Eq. (8) defines the aggregation policy employed throughout this paper for use within our proposed neuro-reckoning framework by controlling hosts for each unfolded entity trajectory. The estimated total expected change in entity location ($\mathbf{L}_{t+q+1} - \mathbf{L}_t$) is normalized to a unit vector and scaled by the magnitude of the current instantaneous entity velocity at time t (i.e. current speed):

$$\mathbf{V}_t^{\mathrm{NR}} = \frac{\mathbf{L}_{t+q+1} - \mathbf{L}_t}{\left| \mathbf{L}_{t+q+1} - \mathbf{L}_t \right|} \left| \mathbf{V}_t \right| \tag{8}$$

When an entity is deemed to have violated the dead reckoning error threshold, the controlling host issues an ESU consisting of current entity location $\mathbf{L}_t$, current entity orientation $\mathbf{O}_t$, and the *predictive velocity vector* $\mathbf{V}_t^{\mathrm{NR}}$. Remote hosts subsequently extrapolate entity location using a first-order, one-step extrapolation equation defined in Eq. (9). In this way, the reliance of controlling hosts on the use of neuro-reckoning for replicating predictive state information is entirely opaque to remote hosts, who only ever view the transmitted ESU packets:

$$\mathbf{L}_{t+1} = \mathbf{L}_t + \mathbf{V}_t^{\mathrm{NR}} \Delta t \tag{9}$$

4. EXPERIMENTATION

In order to collect the type of data that one would expect to observe in a real-world DIA (and to illustrate the potential of neuro-reckoning for network traffic reduction), we utilize the commercially available Torque Game Engine [11] as a customisable research platform for performing experiments in a controlled manner [6]. We present results for three human-user test subjects (each possessing a varying degree of expertise with respect to networked multiplayer computer games – Test Subject 1 being the most experienced, Test Subject 3 the least), who were each required to individually compete in a series of 12 consecutive experiments (consisting of a pre-specified 'hit-point' score limit of 10) against a scripted computer-controlled opponent operating under the influence of a simple dynamic shortest-path behavioural model [12].

Data is collected at a rate of 20 samples per second (i.e. a constant simulation time-step of Δt = 50ms), and subsequently normalized in order to improve training efficiency and provide good generalization capability. Of the 12 experimental datasets collected for each user, the first 2 sets were discarded due to the probable appearance of transient behaviour related to initial learning strategies adopted by each human-user test subject. In addition, the final 2 sets were kept isolated from the data pertaining to training the neural-networks for the purposes of providing an unbiased means of performance evaluation. The remaining 8 experimental datasets were combined, after which time a series of exemplars were extracted and then divided randomly into suitable training (70%), validation (15%), and testing (15%) sets. Based on early evaluations performed using various network topologies, the total time-delay k was set at 3 (longer time-delays were seen to produce negligible improvement in training performance relative to the additional network complexity), and the maximum prediction horizon q was set at 10.

5. RESULTS AND ANALYSIS

Table 2 presents the mean squared error (MSE) training performance over the training, validation, and test sets for each test subject with respect to the deviation between predicted changes in entity velocity and the correct changes in entity velocity over each set of TDNN predictors. Each individual TDNN learns to predict over a specific prediction horizon, where the training procedure continues until the validation set determines the 'early-stopping' point for maximum generalization [7] (or alternatively, the training period reaches 100 epochs). As can be observed, each of the datasets exhibits similar results over all of the prediction horizons, indicating good generalization ability across the entire set of predictors. Evident from inspection of Table 2 is the fact that as the prediction horizon increases, we can observe an approximately linear trend in the decreasing accuracy of each neural-network. Hence, the predictive performance of each neural-network is directly proportional to the prediction horizon.

Prediction Horizon	Test Subject 1 (9618 Samples)			Test Subject 2 (10114 Samples)			Test Subject 3 (9835 Samples)		
	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
1	0.0004	0.0005	0.0004	0.0006	0.0006	0.0006	0.0003	0.0003	0.0004
2	0.0015	0.0015	0.0015	0.0019	0.0020	0.0020	0.0010	0.0010	0.0011
3	0.0032	0.0033	0.0031	0.0039	0.0041	0.0041	0.0022	0.0024	0.0026
4	0.0055	0.0059	0.0054	0.0069	0.0072	0.0072	0.0037	0.0039	0.0044
5	0.0079	0.0086	0.0078	0.0099	0.0104	0.0104	0.0057	0.0059	0.0063
6	0.0100	0.0111	0.0100	0.0131	0.0135	0.0138	0.0080	0.0087	0.0091
7	0.0121	0.0133	0.0117	0.0163	0.0171	0.0176	0.0108	0.0113	0.0117
8	0.0140	0.0154	0.0136	0.0195	0.0207	0.0209	0.0135	0.0140	0.0145
9	0.0160	0.0174	0.0151	0.0229	0.0239	0.0249	0.0157	0.0166	0.0172
10	0.0174	0.0193	0.0165	0.0248	0.0261	0.0270	0.0187	0.0198	0.0202

Table 2: MSE results over training, validation and test sets for each individual human-user test subject.

Table 3 presents simulation results that were conducted over a series of increasing error thresholds for Test Subject 1, Test Subject 2, and Test Subject 3. All presented error threshold values are measured in terms of Torque world units. To put the error thresholds into perspective, the height of an entity within our test environment is approximately 2.3 Torque world units. Ideal network conditions were assumed to ensure unbiased evaluation of the performance of the first-order, one-step

Threshold	Test Subject 1 (21595 Samples)			Test Subject 2 (27749 Samples)			Test Subject 3 (32477 Samples)		
	DR	NR	% Red	DR	NR	% Red	DR	NR	% Red
0.5	2694	2392	**-11.2**	4103	3717	**-9.4**	4309	3926	**-8.9**
1	1823	1656	**-9.2**	2778	2436	**-12.3**	3005	2426	**-19.3**
1.5	1423	1297	**-8.9**	2211	1956	**-11.5**	2397	1961	**-18.2**
2	1217	1111	**-8.7**	1869	1668	**-10.8**	2025	1665	**-17.8**
2.5	1064	994	**-6.6**	1639	1470	**-10.3**	1791	1513	**-15.5**
3	974	893	**-8.3**	1481	1318	**-11.0**	1618	1365	**-15.6**
3.5	872	809	**-7.2**	1352	1234	**-8.7**	1481	1267	**-14.4**
4	786	742	**-5.6**	1232	1132	**-8.1**	1364	1172	**-14.1**
4.5	733	699	**-4.6**	1155	1077	**-6.8**	1284	1104	**-14.0**
5	679	650	**-4.3**	1092	1013	**-7.2**	1186	1051	**-11.4**
10	422	400	**-5.2**	683	661	**-3.2**	753	694	**-7.8**
25	221	218	**-1.4**	325	312	**-4.0**	368	350	**-4.9**

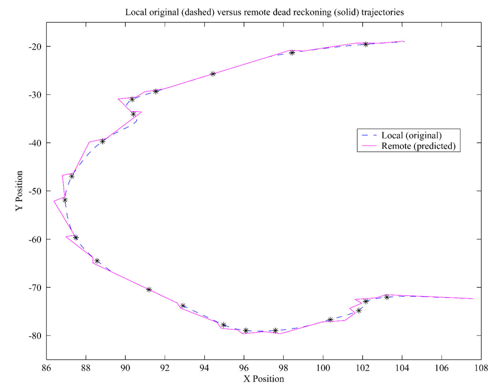
Table 3: Entity state updates generated and totalled over all 12 recorded trajectories for each individual test subject (see Section 4).

dead reckoning ('DR') model and the proposed first-order, one-step neuro-reckoning ('NR') technique.

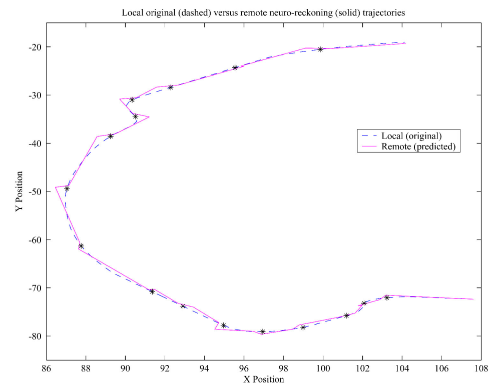
Throughout Table 3, packet counts for both scenarios are listed under the headings 'DR' and 'NR' respectively. The term '% Red' refers to the percentage reduction (or increase) in the number of ESU packets generated due to error threshold violation, where negative values are indicative of superior performance by the proposed neuro-reckoning framework in comparison with the standard dead reckoning model. Packet counts are generated and totalled over all of the 12 recorded trajectories for each individual test subject (see Section 4), and all simulations were performed at a constant rate of 20Hz (i.e. the original sampling rate of the data) using a maximum prediction horizon of q = 10 (i.e. 10 TDNNs).

From inspection of the results, it is noted that in every situation, neuro-reckoning offers a reduction in the number of ESU packets generated that ranges from small bandwidth savings (in the region of 2% or lower packet reduction) to very large bandwidth savings (in the region of just under 20% packet reduction). Furthermore, percentage reductions in the number of ESU packets generated are observed over the entire series of simulated error thresholds. Of particular interest are the large packet reductions typically observed at lower error thresholds, implying excellent potential for use within DIAs requiring a high degree of tightly coupled synchronization between locally and remotely modeled entity state. In general, there appears to be an approximately linear trend between the (decreasing) reported percentage reductions and the (increasing) error threshold. This implies a relatively stable predictive accuracy with respect to the gross performance gain (i.e. expected overall packet reduction) resulting from the use of our proposed neuro-reckoning framework in contrast to the DIS dead reckoning model. It should be noted that although the number of users limits our ability to generalize the results, it is sufficient for demonstrating the efficacy of the neuro-reckoning technique for cases where good neural-network prediction is available for a user.

Figure 2 (a)-(b) presents the results generated using both ESU mechanisms for Test Subject 1 over an error threshold of 0.5 Torque World Units (TWUs), visually illustrating the potential of neuro-reckoning for further network traffic reduction over a small sample section from one of the trajectories recorded for that user.



(a) First-order DIS dead reckoning (DR) (20 ESUs generated).



(b) First-order neuro-reckoning (NR) (16 ESUs generated).

Figure 2 (a)-(b): ESU packet generation results for Test Subject 1 over an error threshold of 0.5 Torque World Units (TWUs) for a small sample section extracted from a recorded user trajectory.

In both cases, the remotely modelled entity trajectory is represented as a solid line that is overlaid on top of the original trajectory represented as a dashed line, where the number of solid black stars represents the number of ESUs generated due to error threshold violation.

From inspection of the spatial plots, we can observe how first-order neuro-reckoning accurately compensates for expected changes in future entity behaviour, reducing the number of ESUs required to maintain an equivalent spatial consistency to that provided by the first-order DIS dead reckoning mechanism from 20 to 16.

6. CONCLUSIONS AND FUTURE WORK

The proposed neuro-reckoning model provides a statistical foundation for the modelling and prediction of repeatable patterns of human-user behaviour [13, 14]. By forecasting expected future entity behaviour in advance, and distributing this information to remote hosts in a compact form, neuro-reckoning succeeds in reducing the spatial error associated with remote modelling of entity state when compared with the traditional use of first-order instantaneous derivative information. Consequently, neuro-reckoning achieves a reduction in the number of ESUs generated due to error threshold violation.

Presented simulation results validate the potential of our proposed framework for improving the accuracy of remote entity modelling over the use of a pure dead reckoning approach throughout our experimental set-up, exhibiting a reduction in network bandwidth requirements over a wide range of error thresholds and indicating viable potential for satisfying many spectrums of consistency requirements throughout a broad range of applications.

Future work includes the investigation of confidence-interval estimation methods to quantify the reliability of the predictions made by each set of neural-networks (with a view to dynamic model switching) [15], in addition to a complexity analysis for the use of the neuro-reckoning algorithm by controlling hosts during real-time execution. Further validation of the technique on more test subjects, comprising a wider array of scenarios involving multiple human-users, complex real-world environments, and realistic network conditions, will also be undertaken.

7. ACKNOWLEDGEMENTS

This material is based upon work supported by the Science Foundation of Ireland and Enterprise Ireland under grant no. IRCSET/SC/04/CS0289, and also by Enterprise Ireland under grant no. SC/2002/129.

8. REFERENCES

[1] E. F. Churchill, D. N. Snowdon and A. J. Munro, *Collaborative Virtual Environments: Digital Places and Spaces for Interaction*. London, UK: Springer-Verlag, 2001.

[2] S. K. Singhal and M. J. Zyda, *Networked Virtual Environments: Design and Implementation*. New York, New York: Addison-Wesley, ACM Press SIGGRAPH Series, 1999.

[3] D. Delaney, T. Ward and S. McLoone, "On Consistency and Network Latency in Distributed Interactive Applications: A Survey – Part I," *Presence: Teleoperators and Virtual Environments*, vol. 15, no. 2, pp. 218-234, April 2006.

[4] A. McCoy, D. Delaney and T. Ward, "Game-State Fidelity Across Distributed Interactive Games," *ACM Crossroads*, vol. 9.4 (Networking Issue), pp. 4-9, Summer 2003.

[5] *IEEE Standard for Distributed Interactive Simulation – Application Protocols*, IEEE Standard 1278.1-1995, 1995.

[6] A. McCoy, D. Delaney, S. McLoone and T. Ward, "Investigating Behavioural State Data-Partitioning for User-Modelling in Distributed Interactive Applications," in *Proc. 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '04)*, Budapest, Hungary, October 2004, pp. 74-82.

[7] J. C. Principe, N. R. Euliano and W. C. Lefebvre, *Neural and Adaptive Systems: Fundamentals through Simulation*. New York, NY, USA: John Wiley and Sons, Inc., 2000.

[8] G. I. Webb, M. J. Pazzani and D. Billsus, "Machine Learning for User Modeling," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1-2, pp. 19-29, March 2001.

[9] R. Bone and M. Crucianu, "Multi-step-ahead Prediction with Neural Networks: A Review," *Approches Connexionnistes en Sciences Economiques et en Gestion*, vol. 2, pp. 97-106, November 2002.

[10] M. T. Hagan and M. B. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989-993, November 1994.

[11] J. Lloyd, "The Torque Game Engine", *Game Developer Magazine*, vol. 11, no. 8, pp. 8-9, September 2004.

[12] A. McCoy, D. Delaney, S. McLoone and T. Ward, "Dynamic Hybrid Strategy Models for Networked Multiplayer Games," in *Proc. 19th European Conference on Modelling and Simulation (ECMS 2005)*, Riga, Latvia, June 2005, pp. 727-732.

[13] W. S. Sarle, "Neural Networks and Statistical Models," in *Proc. 19th Annual SAS Users Group International Conference*, SAS Institute Inc., Cary, NC, USA, April 1994, pp. 1538-1550.

[14] I. Zukerman and D. W. Albrecht, "Predictive Statistical Models for User Modeling," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1-2, pp. 5-18, March 2001.

[15] G. Papadopoulos, P. J. Edwards and A. F. Murray, "Confidence Estimation Methods for Neural Networks: A Practical Comparison," IEEE Transactions on Neural Networks, vol. 12, no. 6, pp. 1278-1287, November 2001.