Comparison of Predictive Contract Mechanisms from an Information Theory Perspective

XIN ZHANG, TOMÁS WARD, and SÉAMUS MCLOONE, National University of Ireland Maynooth

Inconsistency arises across a Distributed Virtual Environment due to network latency induced by state changes communications. Predictive Contract Mechanisms (PCMs) combat this problem through reducing the amount of messages transmitted in return for perceptually tolerable inconsistency. To date there are no methods to quantify the efficiency of PCMs in communicating this reduced state information. This article presents an approach derived from concepts in information theory for a deeper understanding of PCMs. Through a comparison of representative PCMs, the worked analysis illustrates interesting aspects of PCMs operation and demonstrates how they can be interpreted as a form of lossy information compression.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Distributed*; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*; H.1.1 [**Models and Principles**]: Systems and Information Theory—*Information theory*

General Terms: Measurements, Performance, Theory

Additional Key Words and Phrases: Consistency, collaborative virtual environments, dead reckoning, distributed interactive applications, distributed interactive simulation, distributed virtual environments, networked multi-player computer games, networked virtual environments, predictive contract mechanisms

ACM Reference Format:

Zhang, X., Ward, T., and McLoone, S. 2012. Comparison of predictive contract mechanisms from an information theory perspective. ACM Trans. Multimedia Comput. Commun. Appl. 8, 2, Article 18 (May 2012), 18 pages.
DOI = 10.1145/2168996.2168998 http://doi.acm.org/10.1145/2168996.2168998

1. INTRODUCTION

Distributed Virtual Environments (DVEs) is a class of software systems that enable geographically distant users to collaboratively interact with each other. Typically this takes the form of a virtual environment or virtual world that offers shared time, shared space, and shared presence [Singhal and Zyda 1999]. Examples of significant DVE deployments range from distributed military simulations (e.g., SIMNET [Calvin et al. 1993; Miller and Thorpe 1995], DIS [IEEE 1998], HLA [IEEE 2000]) and academic virtual networked community (e.g., DIVE [Frécon and Stenius 1998], NPSNET [Capps

This work is supported by the Irish Research Council for Science, Engineering, and Technology (IRCSET): funded by the National Development Plan.
Authors' addresses: X. Zhang, T. Ward, and S. McLoone, Department of Electronic Engineering, National University of Ireland Maynooth, Maynooth, Co. Kildare, Ireland; email: {xzhang, tomas.ward, seamus.mcloone}@eeng.nuim.ie.

DOI 10.1145/2168996.2168998 http://doi.acm.org/10.1145/2168996.2168998

et al. 2000], PECOLE [Saddik et al. 2008]) to large-scale online entertainment systems (e.g., Ultima-Online [Origin Systems 1997], Quake [Kushner 2002], DIP [Zimmermann et al. 2008]).

Conceptually, a real-time DVE can be seen as a set of virtual entities (or objects) interacting within a simulated and synchronized virtual environment [Delaney et al. 2006]. The interactive entities can be represented by a number of state variables, and manipulated by dispersed human-users through host machines that maintain local instances of the virtual environment. The local controlling host notifies the changing state of the corresponding entity by sending synchronization messages (or update packets) across the underlying communication network. Remote host machines receive the messages and implement the changes to their own virtual environment instances. This "updating" enables the DVE to provide to the human users the illusion of a single, seamless virtual environment and real-time interaction.

Despite the diversity of this application class, the fundamental functionality of any DVE is to maintain a consistent view of the virtual world for all the participating users and to hide the distributed nature of the application. Maintaining sufficient consistency has been one of the most persistent problems in large-scale DVEs because of the two fundamental challenges of network data transmission, that is, finite network bandwidth and nonzero network latency [Capps and Stotts 1997; Delaney et al. 2006]. In the face of the limited network resources, increasing the update frequency for a finer remote state replication may lead to higher throughput and latency if the traffic exceeds the available bandwidth, and ironically compromises consistency. This *Consistency-Throughput Trade-off* states that it is impossible to achieve a both highly consistent and dynamic world at the same time [Singhal and Zyda 1999].

To deal with the Consistency-Throughput Trade-off, one important group of consistency maintenance mechanisms, collectively known as Predictive Contract Mechanisms (PCMs) [Mellon and West 1995; Delaney et al. 2006], employs predictive approaches to reduce the amount of data transmission for optimal bandwidth usage and minimum latency. These techniques have been widely utilized in military training simulations and networked games [Calvin et al. 1993; Pantel and Wolf 2002; Yu et al. 2007]. In PCMs, a local controlling host runs a high-fidelity motion that represents the true entity state and a low-fidelity local model estimated from contextual dynamics by some prediction scheme. The local host compares the low-fidelity model to the true entity state and send an update packet only when the difference between the two violates a given threshold. The same prediction scheme is also applied at every remote host to produce a remote model of the entity state from the most recently received packet. In doing so, PCMs maintain a controlled inconsistency for reduced traffic.

The most commonly deployed PCM is known as Dead Reckoning (DR), which makes use of the current state derivatives and polynomial equations to extrapolate future entity states [Miller and Thorpe 1995; IEEE 1998; Pantel and Wolf 2002]. There has been significant research focused on improving prediction accuracy either by employing advanced modeling techniques to generate better derivative information used by those predictors [Pantel and Wolf 2002; Hanawa and Yonekura 2006; McCoy et al. 2007] or dynamically switching among prediction models according to certain accuracy criterion [Lee et al. 2000; Delaney et al. 2003; McCoy et al. 2005]. One problem with these alternative predictors is that they only produce better accuracy when the user motion complies with the motion model implied by the predictor under use. Another line of research carefully selects suitable error thresholds in order to balance the trade-off between the number of update packets and consistency [Lee et al. 2000; Yu and Choy 2001; Chen and Chen 2005; Roberts et al. 2008; Kenny et al. 2009]. These techniques have all proven effective through simulation. However, in so far as the authors are aware, there exists no analytical measure of the contribution of such mechanisms in reducing data transmission.

The work presented in this article takes a new approach to examine PCMs from the perspective of information theory. Concepts such as Entropy and Mutual Information [Cover and Thomas 2006] are

employed to measure the knowledge related to future entity states in the state derivatives used by the predictors. The advantage of information-based measurement is that it does not imply any presumed model or pattern, and provides a general metric of the user motion predictability of any possible form. Also, different prediction schemes can be assessed and compared under a single framework. Such an information perspective has been proposed to measure PCM operation, but is either restrained on the local host [Zhang et al. 2008] or only with the simplest standard dead reckoning [Zhang et al. 2009]. In this article, the idea is considerably extended by a complete "life cycle" analysis of the synchronization messages that shows how information is included in and extrapolated from the messages to build the remote model. The inconsistency caused by discarding prediction errors within the threshold limit and network latency in data transmission is measured as information loss. Such a perspective facilitates an analytical study of the trade-off between consistency and throughput. PCMs are seen consequently as an information generation, compression and reconstruction process. An analysis of our model is presented through an experimental study. Finally, a comparison of two standard DR algorithms and a newly proposed Neuro-Reckoning technique [McCoy et al. 2007] using the information model is presented. The results show that our information approach applies to both standard and novel PCMs.

The remainder of this article is organized as follows. In the next section, principles of general PCMs are outlined. A brief overview of two standard polynomial predictors and Neuro-Reckoning (NR) is also presented. The information model, along with basic concepts in information theory, is discussed in detail in Section 3. In Section 4, the information model and information analysis is demonstrated using a user motion dataset from a representative DVE scenario. The three PCMs are then examined and compared using thiis model. Finally, Section 5 gives some conclusions and discusses future directions.

2. PREDICTIVE CONTRACT MECHANISMS (PCMs)

The operation of predictive contract mechanisms is commonly divided into two main components: prediction and convergence [Singhal and Zyda 1999]. Instead of updating entity state all the time, PCMs make use of prediction models to generate estimation of future entity state and therefore reduce packet transmission. The local and remote host apply the same prediction model to generate local and remote models of the entity dynamic. On receiving the packets, the remote host uses convergence algorithms to correct prediction errors in a way that more natural motion is rendered to the human-users.

The prediction algorithms estimate future entity state from state derivatives included in the latest message. The local host responsible for the entity keeps this estimation as the local model, and compares it with the true motion. Whenever the prediction error exceeds a predefined threshold, a message containing the current state derivatives is generated, and the local model is corrected to the current entity state immediately. The local model acts as a reference of how the predictor is performing on the remote host such that the local host can determine when to send a new message to reduce inconsistency. On the remote host, the remote model is produced by extrapolating the most recently received message using the same predictor.

The convergence aspect of these methods defines how the diverged remote model is corrected when the remote host receives a new message, so that the rendered motion looks perceptually plausible. Generally, instead of abrupt correction or "snap," error is gradually corrected over several steps along a modeled path. Currently, polynomial equations are the most commonly used convergence algorithms [Singhal and Zyda 1999; McCoy et al. 2007]. The convergence process does not affect update packet generation or bandwidth consumption. Therefore, we do not consider convergence algorithms in the scope of this work.

Figure 1 illustrates the concept of a typical predictive contract mechanism. The time it takes to transmit the message makes the remote model suffer from additional latency-induced error beyond the local error threshold. Consider the time of simulation step $k = 0$ at which the local model error

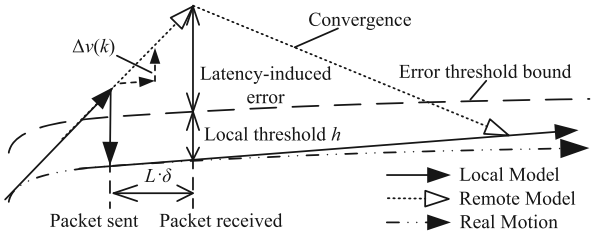

Fig. 1. Visual illustration of a typical PCM, with linear extrapolation and first-order convergence.

exceeds the local threshold h and a new update is generated. The local model is corrected immediately but the remote model keeps diverging over the period of the network latency of L steps until the new message arrives. The accumulation of the remote error R_e is described as [Zhang et al. 2009]:

$$R_e = h + \sum_{k=0}^{L-1} \Delta v(k)\delta = h + L\delta\overline{\Delta v}, \tag{1}$$

where δ is the simulation interval, $\Delta v(k)$ is the difference in velocity between the real dynamic and the remote extrapolation at the simulation tick k , and $\overline{\Delta v}$ is the average difference in velocity over the time of the message transmission. From a spatial perspective, Equation (1) indicates that the prediction algorithm is the core of the whole mechanism, in that aside from the deterministic local threshold and the uncontrollable network latency, the difference in velocity depends on how the prediction model fits the entity dynamic.

2.1 Polynomial Predictors

The polynomial predictors used in standard DR can be generally described in the form of a truncated Taylor expansion. For a time-dependent, real-value continuous function $f(t)$, the n^{th}-order polynomial equation that predicts the value of $x(k) = f(t_k)$ at the time step $t_k = t_0 + k\delta$ using state derivatives at the time t_0 is as follows [McCoy et al. 2007]:

$$x(k) \approx \sum_{i=0}^{n} \frac{x^{(i)}(0)}{i!}(k\delta)^i, \tag{2}$$

where $x^{(i)}(0)$ is the i^{th} derivative of $f(t)$ at time t_0, and δ is the simulation step size. Naively, one might assume that higher-order derivatives improve prediction accuracy. However, higher-order derivatives introduce higher sensitivity to rapid changes of the entity motion and may result in highly jerky approximations. In addition, due to the discrete nature of DVE simulation, estimation error related to numerical approximation of high-order derivatives [Pantel and Wolf 2002] makes them unreliable. Consequently, high-order predictors may lead to larger numbers of updates [Hanawa and Yonekura 2005]. Therefore, 1^{st} and 2^{nd}-order DR are the most commonly deployed PCMs [Singhal and Zyda 1999], and are further investigated in the following sections. For clarity, the 1^{st} and 2^{nd}-order DR predictors are explicitly expressed in (3) and (4):

$$1^{st}\text{-order DR} : x(k) = x(0) + \dot{x}(0) \cdot k\delta \tag{3}$$

$$2^{nd}\text{-order DR} : x(k) = x(0) + \dot{x}(0) \cdot k\delta + \frac{\ddot{x}(0)}{2} \cdot (k\delta)^2. \tag{4}$$

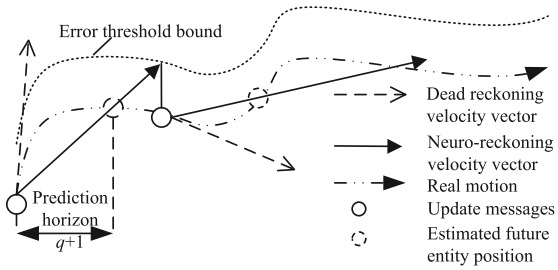

Fig. 2. Visual illustration of neuro-reckoning.

2.2 Neuro-Reckoning

Neuro-reckoning extends 1^{st}-order DR in two aspects [McCoy et al. 2007]. Firstly, instead of just referring to the current velocity of the entity when generating an update packet, it produces a "long-term" velocity that compensates for expected changes of entity velocity over multiple steps ahead. Secondly, NR employs Artificial Neural Networks to capture behavioral patterns in past entity motion to better estimate future velocity changes. The basic philosophy is illustrated in Figure 2. At the time of an error threshold violation, NR first employs a bank of neural network predictors to estimate a probable position of the entity on a relatively long time scale. It then replaces the current velocity in 1^{st}-order DR (DR velocity) with a "NR velocity" that extrapolates from the current entity state through the estimated future position. In doing so, the local and remote model are oriented towards more likely, and thus more stable, directions, and packet transmissions can be further reduced [McCoy et al. 2007].

The NR prediction module uses a bank of neural network predictors to estimate the entity state over a prediction horizon of $q+1$ simulation steps (Figure 2) from $d+1$ most recent entity state vectors (e.g., multidimensional position $\boldsymbol{x}(k)$, velocity $\boldsymbol{v}(k)$, and orientation $\boldsymbol{\theta}(k)$). The neural network predictors are trained over a representative dateset [McCoy et al. 2007]. The prediction results are combined to give an overall mapping $g(\cdot)$ from the input entity state vectors $\tilde{\boldsymbol{s}}(k)$ to the estimated future entity state $\boldsymbol{x}(k+q+1)$ as shown in (5). The difference vector between the predicted future state and the current entity state is then normalized and scaled by the current speed to give the NR velocity vector as in (6). When an update is triggered by the event of error threshold violation, instead of the current velocity, an NR velocity is calculated and transmitted over the network to the remote host as a part of the standard update packet.

$$\begin{aligned} \text{NR mapping} \ : \ & \boldsymbol{x}(k+q+1) = g(\tilde{\boldsymbol{s}}(k)) \\ & \tilde{\boldsymbol{s}}(k) = (\boldsymbol{x}(k), \boldsymbol{v}(k), \ldots, \boldsymbol{v}(k-d), \boldsymbol{\theta}(k), \ldots, \boldsymbol{\theta}(k-d)) \end{aligned} \tag{5}$$

$$\text{NR velocity} \ : \ \hat{\boldsymbol{v}}(k) = \frac{\boldsymbol{x}(k+q+1) - \boldsymbol{x}(k)}{|\boldsymbol{x}(k+q+1) - \boldsymbol{x}(k)|} \cdot |\boldsymbol{v}(k)|. \tag{6}$$

As complicated as it might seem, NR only involves replacing the velocity prediction procedure on the local host, which means it is a self-contained component that can be easily integrated into standard DR: from the perspective of the remote host, the operations of receiving and extrapolating synchronization messages remain exactly the same as the standard 1^{st}-order DR.

It should be remarked that neural networks, like other statistical learning techniques [Zukerman and Albrecht 2001], require large amounts of prior information derived from training datasets in order to extract patterns in user behavior, and is thus application-dependent (meaning that the neural network predictors must be re-trained for different application scenarios). However, the idea of employing predictive velocities to comply with long-term movement trends applies well in general DVEs.

In the next section, concepts from information theory are used to reinterpret the operation of PCMs. From this perspective, the performance of different prediction algorithms are measured by their ability to make use of the temporal dependence in the entity dynamic to speculate future entity state. Information analysis on the content of the messages (state derivatives) has been recently introduced [Zhang et al. 2008, 2009]. Here we provide a considerable extension to the information model by investigating how information included in the messages is utilized by the prediction algorithms in building the remote model, and the three prediction models are then compared under the information metric.

3. INFORMATION MODEL

We begin with a brief review of concepts in information theory such as entropy and mutual information and their estimation from practical datasets. Using these concepts, we then introduce the new perspective where PCMs are treated as a form of lossy information compression, transmission and decompression.

3.1 Information Basis Overview

It is the temporal dependence in user behavior that provides predictability of the future entity state and enables PCMs to extrapolate state approximation from the current motion status. For example, standard DR directly uses current state derivatives to predict future entity states assuming the entity keeps the current motion status over a short period. The underlying motive of the information model presented here is to find a general and quantified measure of the predictability in the entity behavior and to investigate how this predictability is utilized by a prediction scheme in building the remote model. In information theory, such predictability is measured by Mutual Information [Cover and Thomas 2006].

In information theory, the general knowledge about a variable is measured by the reduction of uncertainty, instead of spatial precision. Consider a random variable X with m possible states $\{x_1, x_2, \ldots, x_m\}$, each with probability $p(x_i)$. To measure the uncertainty of the variable state, or how unpredictable its state is, the entropy $H(X)$ of the variable is defined as [Cover and Thomas 2006]:

$$H(X) = -\sum_{i=1}^{m} p(x_i) \cdot \log p(x_i). \tag{7}$$

Entropy measures the degree of complexity and unpredictability of the variable. Intuitively, entropy is the "quantity of surprise when a measurement specifies one particular value x_i." If the binary logarithm is used, $H(X)$ indicates the average number of binary questions needed to determine the value of X, namely the data in bits to fully describe the variable. For a completely deterministic variable X, there is some state x^* such that $p(x^*) = 1$, which means that X is completely predictable since it is fixed at x^* with absolute certainty. The entropy is $H(X) = 0$. On the other hand, for a completely random variable Y with a uniform probability $p(y_i) = \frac{1}{m}$ for all the possible states, its entropy is the maximum $H(Y) = \log m$, and there is no way to predict its value since every state is equally possible. In general, variables with larger entropy are more unpredictable and need more data to describe.

For two interdependent variables X and Y with possible states $\{x_1, x_2, \ldots, x_m\}$ and $\{y_1, y_2, \ldots, y_n\}$ respectively, knowing Y may also gives some knowledge about X and thus reduces the uncertainty of the latter. The remaining uncertainty of X, given the full knowledge about Y, is defined as conditional entropy:

$$H(X|Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i, y_j) \cdot \log p(x_i \mid y_j), \tag{8}$$

where $p(x_i, y_j)$ denotes the joint probability and $p(x_i|y_j)$ denotes the conditional probability. For arbitrary variables, we always have $H(X) > H(X|Y)$. The difference between the two is the amount of reduced uncertainty of X due to the knowledge given by Y. Thus, the predictability about X provided by Y is defined as the mutual information $I(X;Y)$:

$$I(X;Y) = \sum_{x_i, y_j} p(x_i, y_j) \cdot \log \frac{p(x_i, y_j)}{p(x_i) \cdot p(y_j)} = H(X) - H(X \mid Y). \tag{9}$$

Notice that entropy is the auto-mutual-information between the variable X and itself, that is $I(X;X) = H(X)$. The mutual information in (9) is also called cross-mutual-information [Jeong et al. 2001].

All the concepts mentioned here involve knowledge about respective probability functions, which are normally not known in practice. Estimating probability and mutual information in general PCMs could be difficult because of the potentially large state space. For the purpose of this article, we use a Kernel Density Estimation (KDE) approach to estimate the probabilities and mutual information from experimental data. Consider two sequences $x(k)$ and $y(k), k = 1, 2, \ldots, N$ as collections of N samples of X and Y at different time instances k. Let $\boldsymbol{r}(k) = [x(k), y(k)]$ be a two-dimensional variable representing the combined sample. The KDE estimation of the joint probability using a Gaussian kernel $K(\cdot)$ is given by [Moon et al. 1995]:

$$\hat{p}(\boldsymbol{r}) = \frac{1}{N} \sum_{k=1}^{N} K\left(\frac{[\boldsymbol{r} - \boldsymbol{r}(k)] \boldsymbol{S}^{-1} [\boldsymbol{r} - \boldsymbol{r}(k)]^T}{h^2} \right), \tag{10}$$

$$K(w) = \frac{1}{(2\pi)^{\frac{d}{2}} h^d det(\boldsymbol{S})^{\frac{1}{2}}} \exp\left(\frac{-w}{2} \right), \tag{11}$$

$$h = \left[\frac{4}{d+2} \right]^{\frac{1}{d+4}} N^{\frac{-1}{d+4}}. \tag{12}$$

where d is the dimension of $\boldsymbol{r}$ ($d = 2$ in this case), h is the kernel bandwidth, and $\boldsymbol{S}$ is the covariance matrix on $\boldsymbol{r}(k)$. When estimating the probability at $\boldsymbol{r}$, each observation $\boldsymbol{r}(k)$ gives a weight based on the kernel function and the distance w between $\boldsymbol{r}$ and $\boldsymbol{r}(k)$, and the estimated probability is the local weighted average of the neighborhood. The bandwidth h controls the range of the neighborhood and the optimal value for the Gaussian kernel in (11) is shown in (12). It can be seen that every sampled data point contributes to the estimated probability located in its neighborhood and thus KDE is particularly advantageous with small datasets (with very low sampled occurrence frequencies) in generating reasonably accurate estimations [Moon et al. 1995; Steuer et al. 2002].

3.2 Information Model

Our information model takes a novel perspective towards understanding the updating scheme of PCMs. Instead of seeing the packets as carrying the most up-to-date entity motion parameters to meliorate modeling precision, the information model views the effect of the messages as improving the information quality of the derivative vector used by the predictors to increase the predictability of the future entity states, so that they can be estimated with higher certainty. This philosophy redefines the illustration in Figure 1 into a flow diagram in Figure 3. Here the entity dynamic is represented by a discrete time series $x(k) = \{x(1), x(2), \ldots\}$, where $x(k)$ is the entity state (position) at the k^{th} simulation step. The remote model $\hat{x}(k) = \{\hat{x}(1), \hat{x}(2), \ldots\}$ is the state approximation simulated by the remote host using PCMs.

From the information perspective, PCMs can be seen as a process of information transfer in which information about the entity dynamics is encapsulated in update packets at the local host and remotely

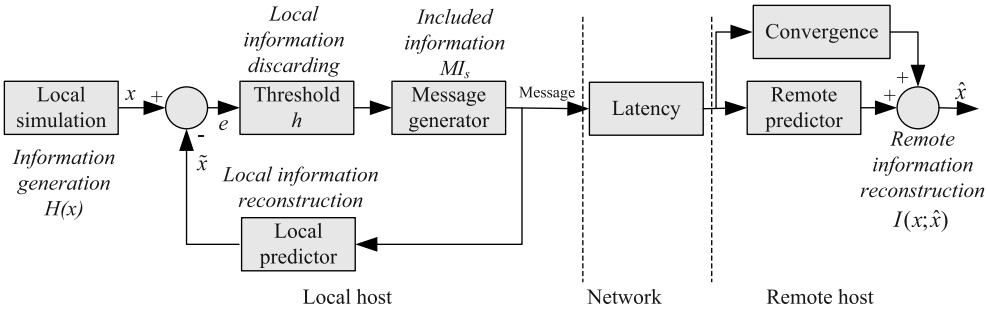

Fig. 3. Information model of PCMs. Information generated by the local dynamic x is included with loss into the messages, from which the remote model $\hat{x}$ is simulated. The extrapolated dynamic $\tilde{x}$ is compared to the real dynamic to trigger update transmission.

extracted to reconstruct the remote model. Each packet, as an information carrier, holds full information about the instantaneous entity state and some information about future states. The information about the current state is the entropy $H(x)$ of the dynamic $x(k)$. It is also the average information rate required for a perfectly accurate approximation to be reconstructed.

Extrapolation uses the knowledge about future states included in messages for prediction. Consider predicting the entity state $x(k+\tau)$ from the message $\boldsymbol{u}(k)$ generated τ steps earlier. For a certain prediction span τ, the available knowledge or predictability that can be utilized by the extrapolation model is measured by mutual information $I(\boldsymbol{u}; x_\tau)$ between $\boldsymbol{u}(k)$ and $x(k+\tau)$:

$$\begin{aligned} I(\boldsymbol{u}; x_\tau) &= I(\boldsymbol{u}(k); x(k+\tau)) \\ &= \sum_{\boldsymbol{u}(k), x(k+\tau)} p_{\boldsymbol{u}x}(\boldsymbol{u}(k), x(k+\tau)) \cdot \log \frac{p_{\boldsymbol{u}x}(\boldsymbol{u}(k), x(k+\tau))}{p_{\boldsymbol{u}}(\boldsymbol{u}(k)) p_x(x(k+\tau))}. \end{aligned} \tag{13}$$

The hypothetical update packet series $\boldsymbol{u}(k)$ consists of prediction parameters (vary for different prediction models) at every simulation tick, which would be sent out with update packets if necessary. This mutual information is the average amount of predictability in a message that can be used to predict the entity state τ steps later than its generation. Here we define the functioning period T_f of a message as the period during which this message is referenced by the prediction scheme to generate modeled states. The local functioning period T_{Lf} is the time interval between successive update generations, and the remote functioning period T_{Rf} is between message arrivals. On average, the information MI_s stored in a message for extrapolating one modeled state is the mutual information averaged over the functioning period:

$$MI_s = \frac{\sum_{\tau \in T_f} I(\boldsymbol{u}; x_\tau)}{T_f}. \tag{14}$$

The stored information MI_s in (14) is mostly the characteristic of the entity dynamic in that it characterizes the capacity of the derivative states to carry information for extrapolation. An update packet could have full information about all the future states in deterministic dynamics, or none in cases of completely random motions.

With the information stored in the messages as the input, only part of the information is actually extrapolated by the prediction scheme in building the remote model as the output. To measure the information utilization, we first consider how much information in the messages is used to predict the entity state at a particular prediction span τ. Any prediction scheme, from something as simple as 1^{st}-order DR to those as complicated as NR, is essentially a function or mapping $\hat{x}(k+\tau) = g(\boldsymbol{u}(k), \tau)$ that

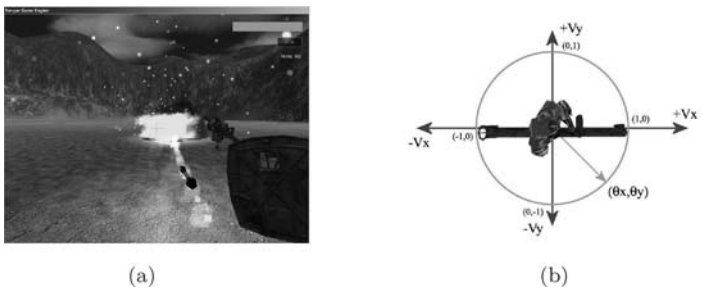

Fig. 4. (a) A Screen-shot of the DVE scenario. (b) The entity state space of the DVE.

relate the content of the packet $\boldsymbol{u}(k)$ to the future entity state estimation $\hat{x}(k+\tau)$ at different prediction span τ. We apply this mapping to the whole packet series $\boldsymbol{u}(k)$ for a single prediction span τ, and build a modeled path $\tilde{x}_\tau(k) = \{g(\boldsymbol{u}(1),\tau), g(\boldsymbol{u}(2),\tau),\ldots\}$ for $k > \tau$ that consists of entity states estimated from the packet series for the same prediction span. Then the mutual information $I(x;\tilde{x}_\tau)$ measures the average information about the true entity motion that is utilized by the prediction scheme in producing the prediction result from the messages at the given prediction span.

Following the same logic as in (14), the utilized information MI_u of a message is obtained by averaging the mutual information $I(x;\tilde{x}_\tau)$ for all the prediction spans within the functioning period as in (15). Notice that compared to T_{Lf}, the remote functioning period T_{Rf} is delayed by the network latency.

$$MI_u = \frac{\sum_{\tau \in T_{Rf}} I(x;\tilde{x}_\tau)}{T_{Rf}}. \tag{15}$$

It is apparent that information utilization is bounded by the stored information. Given the same messages, better algorithms could utilize a higher percentage of the stored information to produce better prediction with higher certainty.

The spatial analysis of the remote inconsistency in (1) can also be redefined from this information perspective. By allowing for local model error within the threshold, PCMs prune the original information generation rate $H(x)$ of the entity dynamic to the storing rate MI_s of the messages. The local information loss is controlled by the threshold. The network latency endured by data transmission adds further information loss and only a reduced information rate MI_u is reconstructed remotely.

In the next section, we apply this model to a motion dataset from a FPS game [McCoy et al. 2007], and compare the three extrapolation methods under the information model. The scenario settings are representative of general DVEs.

4. RESULTS AND ANALYSIS

4.1 Overview

To demonstrate our information model, experiments are conducted under a representative DVE environment based on the Torque game engine [Lloyd 2004]. The simulation scenario, developed by McCoy et al. [2007], is shown in Figure 4(a). In the simulation, players who controls the avatar using a keyboard (for translation movements) and a mouse (for rotations) in first-person perspective are born at randomly assigned "spawn-points" as starting locations. The goal of the players in the simulation is to be the first to reach a prespecified "hit-point" score-limit by attacking others using projectile-based weapon with unlimited ammo, and "disabling" the opponents. Disabled players are reborn at one of the spawn-points. At the point of the score-limit being reached by one of the players, the simulation can be repeated. This "deathmatch" scenario involves typical user behavior in general DVEs, such as

Table I. Derivative Information in an Update Packet

PCMs	Derivative information
1^{st}-order DR	$x(k_u), \dot{x}(k_u) = v_x(k_u)$
2^{nd}-order DR	$x(k_u), \dot{x}(k_u) = v_x(k_u), \ddot{x}(k_u) = \frac{v_x(k_u) - v_x(k_u - 1)}{\delta}$
NR	$x(k_u), \dot{x}_N(k_u) = \hat{v}_x(k_u)$

Table II. Ranges and Occurrences of State Values in the Dataset

	x	$\dot{x}$	$\ddot{x}$	$\dot{x}_N$	$(x, \dot{x})$	$(x, \dot{x}, \ddot{x})$	$(x, \dot{x}_N)$
Range	456	7	11	39	3169	6143	16895
OCC (95%)	201	5700	2310	2005	13	5	3

exploration, chasing, circumvention and engagement. We will submit such a DVE to an analysis from our information perspective.

Figure 4(b) illustrates the entity state space in the simulation. At the simulation step k, an entity is modeled by its position $\boldsymbol{x}(k)$, velocity $\boldsymbol{v}(k)$ and orientation $\boldsymbol{\theta}(k)$, each of which is a 2-dimensional vector. The entity position and velocity are coordinates and speed along the x-axis and y-axis respectively, based on game units. The orientation is the sine and cosine coordinate pair of the positive angle from the x-axis to the line of the orientation. For the convenience of illustration, the presented results are based on the x-coordinate of the entity position, which ranges from -52 to 403 game units. Multidimensional analysis can be conducted in a similar manner. The simulation is run at the rate of 20Hz, with the constant simulation step of $\delta = 50$ms.

The PCMs we examine here are standard DR with 1^{st} and 2^{nd}-order predictors, and NR. For 1^{st}-order DR, the x-component of the entity velocity vector is directly taken as the 1^{st}-order derivative in (3). The 2^{nd}-order derivative in (4) has to be estimated since it is not recorded during the simulation. For NR, we use the neural network predictors trained and validated under the same game scenario in McCoy et al. [2007], where the maximum prediction time-delay d and the maximum prediction horizon q were set at $d = 3$ and $q = 10$ to give the best prediction accuracy. Therefore, the NR velocity vector is predicted based on the motion descriptions of the most recent four steps and pointing through a predicted position at 0.55s in the future. Similar to the case of 1^{st}-order DR, the x-axis component $\hat{v}_x(k)$ of the NR velocity $\hat{\boldsymbol{v}}(k)$ is used in extrapolating the one-dimensional entity dynamic. The derivatives in a packet sent at the step k_u is summarized in Table I.

In the experiment, the human-user is asked to repeatedly play the game and the entity state trajectories of the human-user are then combined to form a true entity motion with the length of $N = 102053$ steps. Details of the recorded dataset in terms of the number of possible values and occurrences of entity states, derivatives, and their combination in each type of packet are summarized in Table II. The "OCC (95%)" row denotes that at least 95% of the possible combinations are sampled with occurrences higher than the shown value. It is then clear that the entity motion is within a small subset of the whole possible space and low occurrence is very rare. From other application domains where the KDE approach has been successfully deployed [Moon et al. 1995; Steuer et al. 2002], this dataset is more than sufficient to give decent results for the purpose to demonstrate the concept to evaluate PCMs using information measurement. In practical DVE systems where huge amounts of recorded data are easily available, even more accurate information estimation for larger state spaces can be obtained using our approach. The user motion is prerecorded and then applied to the three mechanisms to evaluate their performance under varying thresholds and simulated fixed latencies.

Figure 5 shows examples of the probability estimation for the 1^{st}-order DR experiment. The contrast of the unconditional probability $\hat{p}(x_{\tau=1})$ and the conditional probability $\hat{p}(x_{\tau=1}|\boldsymbol{u} = [29, 1])$ in Figure 5(b) reveals that given the knowledge in parameters in the packet, the next entity state became highly

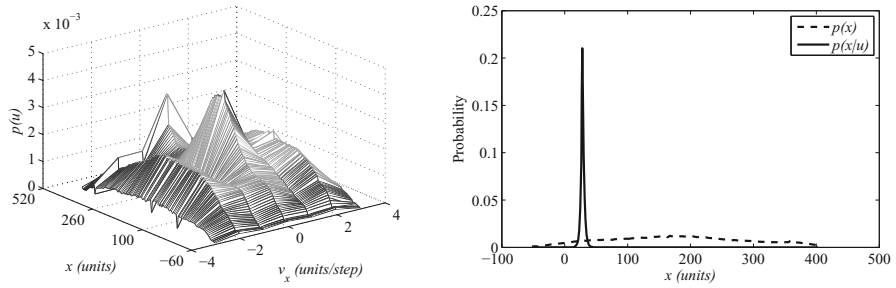

Fig. 5. (a) Probability of the update parameters in the $x - v_x$ plane. (b) Conditional probability $\hat{p}(x_{\tau=1}|\boldsymbol{u} = [29, 1])$ and marginal probability $\hat{p}(x_{\tau=1})$ from the 1^{st}-order DR experiment. The occurrence of $u = [29, 1]$ is 76.

Table III. Number of Packets Generated by Each of the Three Predictive Contract Mechanisms

Local threshold (units)	1^{st}-order DR	2^{nd}-order DR	NR
2	11799	12312	10431
4	7586	7929	6507
8	4943	5173	4358
16	3165	3307	2962

predictable compared to the situation without the update. Figure 5(b) illustrates how information could be used to reduce uncertainty.

4.2 Packet Generation and Spatial Inconsistency Results

We first present a performance comparison of the three techniques based on traditional metrics, namely packet generation and spatial inconsistency. Shown in Table III are the numbers of packets generated by the two DR algorithms and NR under a series of increasing local error thresholds h, which is measured in terms of games units. The size of the entity avatar is approximately 5.7 units. Across the range of thresholds, NR outperforms the two standard DR methods, yielding packet generation reduction ranging from 4% to over 15% against either 1^{st} or 2^{nd}-order DR. Interestingly, 1^{st}-order DR slightly outperforms 2^{nd}-order DR for the thresholds simulated. This is because the 2^{nd}-order predictor imposes more inertia into the modeled motion by considering acceleration, however entity velocity (or acceleration) changes are rapid and frequent in FPS games [Pantel and Wolf 2002]. The fact that 2^{nd}-order DR and NR both take into consideration more derivative information but result in the contrary performance compared to 1^{st}-order DR indicates that the performance of a PCM depends on how well the prediction model fits the pattern in user behavior so that the resources in the packets can be efficiently utilized. This will be further investigated with our information perspective in a quantified manner.

Remote inconsistency under varying fixed latencies is compared in Table IV. The remote inconsistency RI is measured as the average absolute modeling error (in game units), that is, $RI = \frac{\sum_{k=1}^{N} |\hat{x}(k) - x(k)|}{N}$. Once again, NR outperforms standard DR by giving the smallest remote inconsistencies. It should be noted that since NR generates the least updates (Table III), the NR models are corrected less often, which gives the remote model more time to diverge from the true motion. However, NR still presents the best inconsistency results. This is because instead of relying on the current derivative state information, which changes rapidly, the predictive NR velocity compensates and averages out the potential changes in the entity motion, and gives a less dynamic but more stable entity state model.

Table IV. Remote Inconsistency Results (in Units)

Latency (ms)	Local threshold $h = 2$ units			Local threshold $h = 4$ units		
	1^{st}-order DR	2^{nd}-order DR	NR	1^{st}-order DR	2^{nd}-order DR	NR
50	1.19	1.21	0.99	1.95	2.96	1.58
100	1.56	1.62	1.22	2.36	2.41	1.89
150	2.02	2.16	1.59	2.86	2.98	2.33
200	2.56	2.80	1.96	3.42	3.62	2.76
250	3.15	3.52	2.38	4.03	4.35	3.24
500	6.77	8.35	5.56	7.74	9.07	6.64
750	11.24	14.96	9.75	12.28	15.36	10.99
Latency (ms)	Local threshold $h = 8$ units (Graphical representation in Figure 7)			Local threshold $h = 16$ units		
	1^{st}-order DR	2^{nd}-order DR	NR	1^{st}-order DR	2^{nd}-order DR	NR
50	3.34	3.36	2.80	6.04	6.03	5.27
100	3.84	3.89	3.18	6.63	6.65	5.75
150	4.40	4.51	3.70	7.27	7.34	6.40
200	5.01	5.21	4.20	7.95	8.09	6.97
250	5.67	5.97	4.74	8.68	8.90	7.60
500	9.59	10.75	8.43	12.86	13.71	11.67
750	14.28	16.83	12.97	17.73	19.55	16.51

4.3 Information Measurement Results

Our information model focuses on the "life cycle" analysis of the synchronization messages, which measures how information about the true entity dynamic is included in the messages with the derivative states, and then extrapolated to reconstruct the remote model.

Firstly, an entropy of $H(x) = 8.81$ bits is given by (7) for the true entity motion, and that is the average amount of information or data per simulation step required to reconstruct the entity motion with the perfect fidelity, assuming every entity state is transmitted to the remote host under ideal network conditions. Instead, PCMs make use of the content of one message to extrapolate future entity states at multiple steps, to reduce update frequency. But uncertainty arises because only an estimated state model, rather than the definite value, can be reconstructed from the reduced amount of updates.

The delayed-mutual-information in (13) measures the information between a message and the true entity state at a certain prediction span. The message holds the full information for the current state ($\tau = 0$), because the current state itself is literally in the message. While for further future states ($\tau > 0$), the information contained in the message is generally partial (except for deterministic movements) because the potential changes in the entity motion adds uncertainty, making the future entity states more and more unpredictable. Figure 6(a) compares the information capacity of the three types of packets for the future entity states at increasing prediction spans. For standard DR, the information capacity of both the 1^{st} and 2^{nd}-order packets decrease with increasing prediction span. Throughout all the prediction spans, the 2^{nd}-order packet embraces more information about the future dynamics since an additional acceleration value is included.

The NR packet exhibits much slower information drop compared to the standard DR packets. Notice that the NR packet has exactly the same structure as the 1^{st}-order DR packet (Table I). However, the value of the NR velocity, calculated by neural networks based on the knowledge from 17 referencing values (two-dimensional velocity and orientation vectors for each of the 4 most recent steps, and the current entity position), has considerable advantage over the standard DR packets in carrying information for extrapolation. That is why the NR velocity can accurately capture and compensate for the expected changes of the entity state over a time interval, and outperforms standard DR with significant packet transmission reduction.

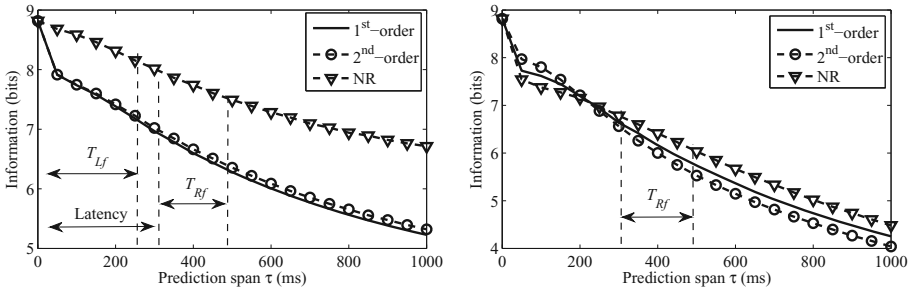

Fig. 6. (a) Information capacity of update packet for the three reckoning techniques. (b) Extrapolated information at increasing prediction span for the three reckoning techniques.

Table V. Included Information on the Local Host

Local threshold (units)	T_{Lf} (ms)			Included Information (bits)		
	DR		NR	DR		NR
	1^{st}	2^{nd}		1^{st}	2^{nd}	
2	432	414	489	7.47	7.54	8.24
4	672	643	784	7.02	7.12	7.88
8	1032	986	1171	6.50	6.62	7.51
16	1612	1543	1723	5.91	6.05	7.15

By allowing for modeling error within human perception (controlled by the local threshold error), PCMs make use of the incomplete information about the future entity dynamic in the messages to trade consistency for less network traffic. Generally, a larger threshold makes an update packet responsible for more future states. In other words, a larger threshold means a longer functioning period, but with more information loss and thus looser inconsistency bounding. On the local host, the local functioning period of the message T_{Lf} begins from the time this message is generated and ends at the generation of the next update. As illustrated in Figure 6(a), all the information within the local function period is the information included in the packet and ready for transmission. The average local functioning period and included information per simulation step $MI_{s,local}$, as in (14), are summarized in Table V. Due to the significant advantage over the other two types of packets, the NR packet carries the most information.

On the remote host, only outdated information can be employed to build the remote model because the message experiences network transmission delay. The remote functioning period of the message is shifted by the length of network latency (Figure 6(a)). This shifting further compromises information quality in the message. The remaining information within the remote functioning period T_{Rf} is the amount of information available for remote extrapolation. The remote functioning period is of the same length of time as the local functioning period, only with the delay of network latency. By applying T_{Rf} to (14), the remote available information $MI_{s,remote}$ in the three types of messages is compared in Table VI. Due to the relatively higher quality of information capacity, the information in the NR packet is less vulnerable to network latency than the DR packets.

With the available information on the remote host, only part of the available information is actually utilized by PCMs in reconstructing the remote model. Figure 6(b) compares the extrapolated information $I(x;\tilde{x}_\tau)$ over increasing prediction spans τ for the three prediction schemes. In spite of the high values of available information, NR does not exhibit that much advantage in utilizing the information over standard DR. This is because the extrapolation equation in use under NR is as simple as linear

Table VI. Remote Available Information (in Bits)

Latency	Local threshold $h = 2$ units			Local threshold $h = 4$ units		
(ms)	1^{st}-order DR	2^{nd}-order DR	NR	1^{st}-order DR	2^{nd}-order DR	NR
50	7.20	7.28	8.11	6.80	6.90	7.76
100	7.02	7.10	7.98	6.63	6.74	7.64
150	6.84	6.92	7.84	6.48	6.59	7.53
200	6.66	6.76	7.72	6.32	6.44	7.43
250	6.49	6.59	7.60	6.18	6.29	7.32
500	5.81	5.91	7.10	5.59	5.70	6.91
750	5.33	5.44	6.76	5.27	5.29	6.62
Latency	Local threshold $h = 8$ units			Local threshold $h = 16$ units		
(ms)	1^{st}-order DR	2^{nd}-order DR	NR	1^{st}-order DR	2^{nd}-order DR	NR
50	6.32	6.44	7.41	5.78	5.92	7.07
100	6.19	6.31	7.32	5.67	5.81	7.00
150	6.05	6.18	7.23	5.57	5.71	6.93
200	5.93	6.06	7.14	5.38	5.61	6.86
250	5.81	5.94	7.06	5.38	5.52	6.80
500	5.32	5.45	6.73	5.00	5.14	6.53
750	4.97	5.09	6.49	4.72	4.86	6.34

extrapolation after all, as the result of the simplicity of integrating NR into standard reckoning architecture. The 1^{st}-order equation cannot fully interpret the delivered information, therefore the compensated trajectory loses details of the entity's dynamics (Figure 2) and the huge information advantage of the NR velocity is wasted. Overcoming this requires a more complicated extrapolation algorithm. For example, the remote host could employ a reverse process of the NR velocity prediction, by making use of the neural network predictors trained on the local site to reconstruct the intermediate state changes within the maximum prediction horizon based on the included information. However, such an extrapolation algorithm will compromise the transparency of NR to the remote host. Also, it should be noted that NR does utilize more information than standard DR at large prediction spans, when the prediction span approaches and go beyond the maximum prediction horizon of 550ms, the time scale at which the NR vector extrapolates through the estimated future state. The information measurement captures, in a quantified way, the NR preference towards long-term and averaged motion trend over instantaneous derivatives, and how this preference improves the NR performance in reducing update packets.

Although the 2^{nd}-order DR packet provides more information to the remote reconstruction than the 1^{st}-order DR packet (Table VI), it only has a slight advantage in extrapolating entity states over 1^{st}-order DR over short periods, due to the limited inertia implemented by the game engine (in this example). Such inertia only has impact over short ranges of time as the player changes motion status frequently. For most of the prediction spans, 2^{nd}-order DR is no better than 1^{st}-order DR. This result shows that higher throughput (as 2^{nd}-order DR has larger packet size) does not guarantee better consistency or information quality. The information in the message has to be interpreted properly and efficiently. Our information metric shows that with the information available, there may exist some way, other than the 2^{nd}-order equation, to better utilize the information hidden in the acceleration value in the 2^{nd}-order DR packet.

Finally, the extrapolated information MI_u within the remote function period, as in Figure 6(b) and (15), is the information about the true entity state that is reconstructed in the remote model (Table VII). From the information perspective, PCMs can be viewed as a form of lossy information processing, where by allowing for local model error within the threshold, the original information rate $H(x)$ of the entity dynamic is reduced to the local storing rate $MI_{s,local}$. The network latency endured by

Table VII. Remote Reconstructed Information (in Bits)

Latency (ms)	Local threshold $h = 2$ units			Local threshold $h = 4$ units		
	1^{st}-order DR	2^{nd}-order DR	NR	1^{st}-order DR	2^{nd}-order DR	NR
50	6.93	6.98	6.86	6.40	6.40	6.33
100	6.71	6.71	6.69	6.20	6.15	6.17
150	6.48	6.43	6.52	5.99	5.91	6.00
200	6.25	6.15	6.34	5.79	5.67	5.83
250	6.03	5.89	6.16	5.59	5.45	5.67
500	5.09	4.89	5.29	4.77	4.59	4.90
750	4.40	4.21	4.58	4.16	3.98	4.26
Latency (ms)	Local threshold $h = 8$ units (Graphical representation in Figure 7)			Local threshold $h = 16$ units		
	1^{st}-order DR	2^{nd}-order DR	NR	1^{st}-order DR	2^{nd}-order DR	NR
50	5.76	5.71	5.73	4.99	4.92	5.03
100	5.58	5.51	5.57	4.85	4.77	4.90
150	5.40	5.31	5.43	4.71	4.61	4.78
200	5.24	5.12	5.28	4.58	4.46	4.65
250	5.07	4.94	5.13	4.45	4.32	4.53
500	4.38	4.21	4.46	3.89	3.75	3.97
750	3.85	3.69	3.91	3.47	3.33	3.50

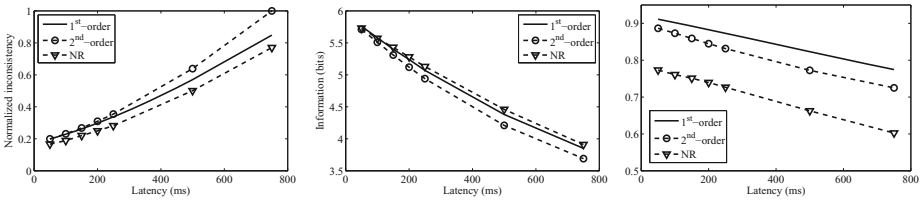

Fig. 7. Graphical representations and comparison of inconsistency and remote information for the threshold of 8 units. (a) Normalized inconsistency. (b) Remote information. (c) Information utilization efficiency.

the transmission adds further information loss and only an information rate of $MI_{s,remote}$ arrives on the remote host, which is finally reconstructed at the rate of MI_u. The reduced information rate indicates a corresponding reduction in network traffic and inconsistency of the reconstructed remote model. In this way, the information model treats the Consistency-Throughput trade-off as a form of lossy media compression.

Graphical representations in Figure 7(a) and (b) compare the remote spatial inconsistency and remote mutual information for increasing latencies for the local threshold $h = 8$ units. Results for other threshold values are similar. The remote inconsistency is normalized by the largest inconsistency in the shown data. Notice that 2^{nd}-order DR exhibits worse consistency than 1^{st}-order DR at larger latencies, but their information metrics are approximated equal. This indicates that although 2^{nd}-order DR diverges farther from the true motion, the true entity state is equally predictable from the 2^{nd}-order DR extrapolation as it is from the 1^{st}-order DR extrapolation, because the information model views the problem from the perspective of predictability rather than spatial similarity. Again, some better extrapolation equation might be able to properly make use of the information in the 2^{nd}-order DR packet to give better spatial consistency.

Figure 7(c) shows the information utilization efficiency (by scaling the reconstructed information by the corresponding available information in Table VI) of the three mechanisms for the threshold of 8 units. Beyond the inconsistency performance shown by the traditional perspective, our information

analysis reveals the potential of NR messages to give further improvement in remote inconsistency control, as only the lowest portion among the three mechanisms is used by the simple linear extrapolation in the current NR. 1^{st}-order DR, which is the most widely deployed prediction scheme, emerges as the most information-efficient extrapolation equation investigated here.

On a final remark, the experiment in this article is subject to the particular scenario since NR predictors have to be trained to capture patterns in a specific user behavior. Training NR predictors for a new application scenario is a nontrivial work and beyond the scope of this article. However, the NR process is effective for general DVE scenarios where specifically trained neural network predictors are available [McCoy et al. 2007]. Therefore insights arising from the comparison between NR and standard DR in our work can be generalized to general DVE scenarios.

5. CONCLUSIONS AND FUTURE WORK

In our information model, aspects of consistency maintenance, namely entity dynamics, prediction algorithms, threshold, and network latency are analyzed using an integrated information metric, and factors causing inconsistency are viewed as a form of information loss or reduction in information rate. Such analysis facilitates a novel understanding of the Consistency-Throughput Trade-off, and suggests improvement in application design. Although our information model is presented through analysis of threshold-based mechanisms, it is also suitable for rate-based methods, in which case, the function period is the constant updating interval.

A performance comparison of three prediction schemes, namely 1^{st} and 2^{nd}-order DR, and NR, is conducted under a novel information model perspective, in which we use information theory to reframe PCMs as a form of lossy compression. Our results show that NR achieves a reduction in the number of update packets compared to standard DR, because it employs a huge amount of information in producing the predictive NR vector. On the other hand, the simple linear extrapolation for the purpose of transparency of integrating NR into standard DR framework hinders good utilization of the available information. NR could be further improved by some advanced extrapolation method. 1^{st}-order DR is the most information-efficient extrapolation in that most of its delivered information is used to reconstruct the remote model. 2^{nd}-order DR does not properly utilize the additional information in the acceleration parameter into building the remote model, and is thus unsuitable for the game scenario investigated here. Our information perspective not only facilitates new distinctions in the relative performance of these PCMs in controlling remote inconsistency, it also reveals the potential of PCMs by providing an analytical measure of the information capacity of the messages.

Our future work will include: investigating different types of entity dynamics from various DVE application scenarios, facilitating the pursuit of an application-dependent optimum prediction algorithm; designing remote extrapolation techniques that can efficiently make use of the information available; and relating the information rate and network traffic.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Aaron McCoy for providing the game scenario and user motion data used in the experimental study.

REFERENCES

CALVIN, J., DICKENS, A., GAINES, B., METZGER, P., MILLER, D., AND OWEN, D. 1993. The SIMNET virtual world architecture. In *Proceedings of the Virtual Reality Annual International Symposium*. 450–455.

CAPPS, M., MCGREGOR, D., BRUTZMAN, D., AND ZYDA, M. 2000. NPSNET-V: a new beginning for dynamically extensible virtual environments. *IEEE Comput. Graph. Appl. 20*, 5, 12–15.

CAPPS, M. AND STOTTS, D. 1997. Research issues in developing networked virtual realities: Working group report on distributed system aspects of sharing a virtual reality. In *Proceedings of the 6th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. 205–211.

CHEN, L. AND CHEN, G. 2005. A fuzzy dead reckoning algorithm for distributed interactive applications. *Proceedings of the 2nd International Conference on Fuzzy Systems and Knowledge Discovery*. Lecture Notes in Computer Science, vol. 3614, 961–971.

COVER, T. M. AND THOMAS, J. A. 2006. *Elements of Information Theory* 2nd Ed. John Wiley & Sons, Inc., Hoboken, NJ.

DELANEY, D., WARD, T., AND MCLOONE, S. 2003. On reducing entity state update packets in distributed interactive simulations using a hybrid model. In *Proceedings of the 21st IASTED International Multiconference on Applied Informatics*.

DELANEY, D., WARD, T., AND MCLOONE, S. 2006. On consistency and network latency in distributed interactive applications: A survey—part I. *Presence: Teleoper. Virtual Envir. 15*, 2, 218–234.

FRÉCON, E. AND STENIUS, M. 1998. DIVE: a scaleable network architecture for distributed virtual environments. *Distrib. Syst. Engin 5*, 91–100.

HANAWA, D. AND YONEKURA, T. 2005. On the error modeling of dead reckoned data in a distributed virtual environment. In *Proceedings of the International Conference on Cyberworlds (CW '05)*. IEEE Computer Society, Los Alamitos, CA, 279–288.

HANAWA, D. AND YONEKURA, T. 2006. A proposal of dead reckoning protocol in distributed virtual environment based on the taylor expansion. In *Proceedings of the International Conference on Cyberworlds (CW '05)*. IEEE Computer Society, Los Alamitos, CA, 107–114.

IEEE. 1998. IEEE standard for distributed interactive simulation - application protocols. IEEE Std 1278.1a-1998.

IEEE. 2000. IEEE standard for modeling and simulation (M&S) high level architecture (HLA)—framework and rules. IEEE Std 1516-2000, i–22.

JEONG, J., GORE, J. C., AND PETERSON, B. S. 2001. Mutual information analysis of the EEG in patients with Alzheimer's disease. *Clinical Neurophys. 112*, 5, 827–835.

KENNY, A., MCLOONE, S., AND WARD, T. 2009. Controlling entity state updates to maintain remote consistency within a distributed interactive application. *ACM Trans. Internet Techn. 9*, 4, 1–25.

KUSHNER, D. 2002. The wizardry of id. *IEEE Spectrum, 39*, 8, 42–47.

LEE, B.-S., CAI, W., TURNER, S. J., AND CHEN, L. 2000. Adaptive dead reckoning algorithms for distributed interactive simulation. *Int. J. Simul. Syst. Sci. Techn. 1*, 1–2, 21–34.

LLOYD, J. 2004. The torque game engine. *Game Devel. Mag. 11*, 8, 8–9.

MCCOY, A., DELANEY, D., MCLOONE, S., AND WARD, T. 2005. Dynamic hybrid strategy models for networked multiplayer games. In *Proceedings of the 19th European Conference on Modelling and Simulation (ECMS '05)*. 727–732.

MCCOY, A., WARD, T., MCLOONE, S., AND DELANEY, D. 2007. Multistep-ahead neural-network predictors for network traffic reduction in distributed interactive applications. *ACM Trans. Model. Comput. Simul. 17*, 4, 16.

MELLON, L. AND WEST, D. 1995. Architectural optimizations to advanced distributed simulation. In *Proceedings of the 27th Winter Simulation Conference (WSC '95)*. IEEE Computer Society, Los Alamitos, CA, 634–641.

MILLER, D. C. AND THORPE, J. A. 1995. SIMNET: The advent of simulator networking. *Proc. IEEE 83*, 8, 1114–1123.

MOON, Y.-I., RAJAGOPALAN, B., AND LALL, U. 1995. Estimation of mutual information using kernel density estimators. *Phys. Rev. E 52*, 3, 2318–2321.

ORIGIN SYSTEMS. 1997.

PANTEL, L. AND WOLF, L. C. 2002. On the suitability of dead reckoning schemes for games. In *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames '02)*. ACM, New York, NY, 79–84.

ROBERTS, D., ASPIN, R., MARSHALL, D., MCLOONE, S., DELANEY, D., AND WARD, T. 2008. Bounding inconsistency using a novel threshold metric for dead reckoning update packet generation. *Simulation 84*, 5, 239–256.

SADDIK, A., RAHMAN, A., ABDALA, S., AND SOLOMON, B. 2008. PECOLE: P2P multimedia collaborative environment. *Multimedia Tools Appl. 39*, 3, 353–377.

SINGHAL, S. AND ZYDA, M. 1999. *Networked Virtual Environments: Design and Implementation*, 1st Ed. Addison-Wesley, New York.

STEUER, R., KURTHS, J., DAUB, C. O., WEISE, J., AND SELBIG, J. 2002. The mutual information: Detecting and evaluating dependencies between variables. *Bioinformatics 18,* supplement 2, S231–S240.

YU, S.-J. AND CHOY, Y.-C. 2001. A dynamic message filtering technique for 3d cyberspaces. *Comput. Comm. 24*, 18, 1745–1758.

YU, Y., LI, Z., SHI, L., CHEN, Y.-C., AND XU, H. 2007. Network-aware state update for large scale mobile games. In *Proceedings of 16th International Conference on Computer Communications and Networks*. Z. Li, Ed., 563–568.

ZHANG, X., WARD, T., AND MCLOONE, S. 2008. Towards an information model of consistency maintenance in distributed interactive applications. *Int. J. Comput. Games Techn. 4*, 1–10.

ZHANG, X., WARD, T., AND MCLOONE, S. 2009. Exploring an information framework for consistency maintenance in distributed interactive applications. In *Proceedings of the 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT '09)*. 121–128.

ZIMMERMANN, R., CHEW, E., AY, S. A., AND PAWAR, M. 2008. Distributed musical performances: Architecture and stream management. *ACM Trans. Multimedia Comput. Comm. Appl. 4*, 2, 1–23.

ZUKERMAN, I. AND ALBRECHT, D. W. 2001. Predictive statistical models for user modeling. *User Model. User-Adapt. Interact. 11*, 1, 5–18.

Received February 2010; revised October 2010; accepted November 2010