NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

An Information-Theoretic Framework for Consistency Maintenance in Distributed Interactive Applications

by

Xin Zhang, M. Sc.

A thesis presented to

National University of Ireland Maynooth

in partial fulfillment of the

requirements for the degree of

Philosophiae Doctor

Department of Electronic Engineering

April 2011

Head of Department:	Dr. Seán McLoone
Supervisor:	Dr. Tomás Ward
Co-supervisor:	Dr. Séamus McLoone

Table of Contents

Abstract

Distributed Interactive Applications (DIAs) enable geographically dispersed users to interact with each other in a virtual environment. A key factor to the success of a DIA is the maintenance of a consistent view of the shared virtual world for all the participants. However, maintaining consistent states in DIAs is difficult under real networks. State changes communicated by messages over such networks suffer latency leading to inconsistency across the application. Predictive Contract Mechanisms (PCMs) combat this problem through reducing the number of messages transmitted in return for perceptually tolerable inconsistency. This thesis examines the operation of PCMs using concepts and methods derived from information theory. This information theory perspective results in a novel information model of PCMs that quantifies and analyzes the efficiency of such methods in communicating the reduced state information, and a new adaptive multiple-model-based framework for improving consistency in DIAs.

The first part of this thesis introduces information measurements of user behavior in DIAs and formalizes the information model for PCM operation. In presenting the information model, the statistical dependence in the entity state, which makes using extrapolation models to predict future user behavior possible, is evaluated. The efficiency of a PCM to exploit such predictability to reduce the amount of network resources required to maintain consistency is also investigated. It is demonstrated that from the information theory perspective, PCMs can be interpreted as a form of information reduction and compression.

The second part of this thesis proposes an Information-Based Dynamic Extrapolation Model for dynamically selecting between extrapolation algorithms based on information evaluation and inferred network conditions. This model adapts PCM configurations to both user behavior and network conditions, and makes the most information-efficient use of the available network resources. In doing so, it improves PCM performance and consistency in DIAs.

Declaration

I hereby declare that this thesis is my own work and has not been submitted in any form for another award at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Signature: _____ Date: 12/4/2011

Acknowledgments

First, I would like to thank my supervisors Dr. Tomás Ward and Dr. Séamus McLoone for their perceptive guidance and help throughout the years of my work that led to this thesis. Many thanks also to Dr. Aaron McCoy, Dr. Damien Marshall, and the rest of the DIA Group for their kind help throughout the development of my work.

Thanks to all the staff at Department of Electronic Engineering at NUI Maynooth. They have made my work and life here, as an international student, a lot easier.

Finally, I would like to express my gratitude to my family for their persistent support for me during the course of this work. Thanks especially to Nan, my beautiful and beloved wife, for her patience, understanding, and support during the last three years.

List of Published Contributions

Zhang, X., T. Ward, and S. McLoone (2008). Towards an Information Model of Consistency Maintenance in Distributed Interactive Applications. *International Journal of Computer Games Technology 2008*(4), 1–10, 2008.

Zhang, X., T. Ward, and S. McLoone (2009). Exploring an Information Framework for Consistency Maintenance in Distributed Interactive Applications. In *Proc. 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT '09)*, Singapore, October 2009, pp. 121–128.

Zhang, X., T. Ward, and S. McLoone (2011). Comparison of Predictive Contract Mechanisms from an Information Theory Perspective. *ACM Transactions on Multivideo Computing, Communications, and Applications (TOMCCAP)*. To appear.

Zhang, X., T. Ward, and S. McLoone (2011). An Information-Based Dynamic Extrapolation Model for Networked Virtual Environments. *ACM Transactions on Multivideo Computing, Communications, and Applications (TOMCCAP)*. To appear.

Chapter 1

Introduction

1.1 Introduction

This thesis proposes that concepts from a branch of applied mathematics called information theory can be utilized to understand the operation and efficacy of distribution mechanisms in a class of software systems known as Distributed Interactive Applications (DIAs). DIAs are a group of software systems that enable geographically distant users to collaboratively interact with each other in a simulated virtual environment. By utilizing the cutting-edge development of computer graphics, human-computer-interfaces, and networking technologies, DIAs offer the implementation of shared time, shared space, and shared presence through exchanging information across the network [Singhal and Zyda 1999]. The earliest form of a simulated environment dates back to the Head-Mounted Display in 1960's, which presents changing perspective images to users as they move their heads and gives the users the illusion of seeing three dimensional objects [Sutherland 1968]. In recent decades, DIAs have seen significant deployments in the areas of military simulations (e.g. SIMNET [Calvin *et al.* 1993; Miller and Thorpe 1995], DIS [IEEE 1998], and HLA [IEEE 2000]) and academic virtual networked communities (e.g DIVE [Frécon and Stenius 1998], NPSNET [Capps *et al.* 2000]). However, it is the world of online entertainment systems (e.g. Ultima-Online [Origin Systems 1997], Quake [Kushner

(a) (b) (c)

Figure 1.1 Examples of various DIA systems. (a) SIMNET [Cosby 1995]. (b) DIVE [Frécon 2004]. (c) Quake [Kushner 2002].

2002], Xbox Live [Microsoft 2005b]) that has seen the greatest proliferation of DIAs in commercial applications. Figure 1.1 illustrates scenes from representative DIA systems from each of the three application areas.

By no means an exhaustive list, key milestones DIA systems to be introduced in the following sections encompass most of the efforts and primary software systems that have contributed to DIAs development. These systems all showed some emphasis on innovations in supporting scalable interactive virtual environments, since communication of state changes of the shared virtual environment among a massive number of participants has been a key design challenge throughout the evolution of DIAs.

1.1.1 Military Simulations

In the early years of DIAs, primary research efforts had been devoted to the military domain, focusing on developing large-scale simulations (hundreds of entities). During the years between 1983 and 1990, the US Defense Advanced Research Projects Agency (DARPA) and the US Army sponsored the development of the SIMNET (SIMulator NETworking) program with the purpose of enhancing collective skills training in a dynamic and free-play environment [Miller and Thorpe 1995]. As the first successful implementation of a large-scale, real-time, human-in-loop military simulation system, SIMNET defined core concepts and principles relating to the

management of a shared environment among distributed participants. It provided the foundation for a new generation of battlefield simulations and other DIAs. SIMNET employed the *Object/Event Architecture*, in which the virtual world is modeled as a collection of objects interacting with each other through a series of events. Each simulation node representing an object is completely autonomous, in that the receiving nodes have the responsibility to determine the effect of an event on them, and not the simulation node that initiates the event. Dead reckoning algorithms are used to minimize communications processing, whereby a simulation node only transmits an entity update message when the true state it represents diverges from a calculated state model by more than a pre-determined threshold. The receiving nodes extrapolate remote entity states from the position, orientation, and velocity information in the latest update message. SIMNET also applied the *selective fidelity* principle, where simulation fidelity varies among components with different interaction needs [Miller and Thorpe 1995]. The most significant applications of SIMNET simulators and protocols include Forward Area Air Defense System (FAADS), Combat Vehicle Command and Control (CVCC), Nonline-of-Sight (NLOS) Missile, and Counter Target Acquisition System (CTAS) [Atwood *et al.* 1991; Miller and Thorpe 1995].

In an attempt to build SIMNET to a consensus standard for distributed simulations, all of its essential elements were incorporated in the Distributed Interactive Simulation (DIS) standards [IEEE 1998]. The DIS standards defined Protocol Data Unit (PDU) structures and protocol families for various functionalities required by DIS-compliant applications, such as CCTT (Close Combat Tactical Trainer) [Johnson *et al.* 1993] and STOW (Synthetic Theater of War) [Calvin *et al.* 1995], as shown in Figure 1.2. These functionalities include Entity Information/Interaction, Warfare, Logistics, Simulation Management, Distributed Emission Regeneration, Radio Communications, Entity Management, Minefield, Synthetic Environment, Simulation Management with Reliability, Live Entity Information/Interaction, and Non-Real Time Protocol. Dead reckoning and collision detection algorithms are also specified to reduce network traffic.

As the most recent family of standards for Distributed Interactive Applications, the

(a) (b)

Figure 1.2 Examples of distributed military simulation systems. (a) CCTT. (b) STOW.

High Level Architecture (HLA) provides a general framework to support distributed simulation applications [IEEE 2000]. The HLA defined the Object Model Template (OMT), which is a standard documentation of the data to describe a particular model, as a basis for reusability. Simulation applications (termed *federates*) are connected and coordinated through a generic communication interface, called the Federate Interface Specification, to address interoperability. Operations and data exchange are supported by the Run-Time Infrastructure (RTI) software. Although the HLA aims at flexibility of the simulation architecture, it lacks the support for dynamic extension and composition of the object model. The Extensible Run-Time Infrastructure (XRTI) has addressed this problem by defining the Reflection Object Model (ROM) to dynamically extend the object model [Kapolka 2003]. The XRTI also defines a common message protocol for its run-time infrastructure. The RTI-integrated STOW is one of the significant HLA-compatible platforms [Calvin *et al.* 1999].

1.1.2 Academic Virtual Networked Communities

Academic research on DIA applications pays closer attention to the on-screen representation of participants, as avatars, and user collaborations. The early systems were mainly designed for local use and only supported a small number of users. One of the first such works is Reality Built for Two (RB2) which, as the name suggests,

supported up to two users [Blanchard *et al.* 1990].

As the applications scaled up, academic research focused on developing network software architectures for large-scale (more than 1000 participants) virtual networked environments. The NPSNET (Naval Postgraduate School Networked Virtual Environment) series developed by the NPSNET Research Group is the longest continuing academic research effort in networked virtual environments. The first version NPSNET-1 was demonstrated in 1991, and the following NPSNET-2 and NPSNET-3 improved graphic and database access. NPSNET-Stealth supports SIMNET data and protocols. NPSNET-IV (as shown in Figure 1.3(a)) in 1993 compiled with DIS protocols and incorporated dead reckoning algorithms. It also developed the first 3D virtual environment suitable for dispersed users over the Internet using IP protocols [Macedonia *et al.* 1994; Macedonia 1995; Macedonia *et al.* 1995]. The most current incarnation NPSNET-V constructs a shared base-level component architecture to fulfill the infrastructure requirements of a large-scale, consistent online virtual world [Capps *et al.* 2000], namely:

- **Run-time extensibility.** NPSNET-V defines a Dynamic Behavior Protocol to allow run-time addition and interpretation of new application components. Protocols that map network messages to behaviors in the virtual world are also developed.
- **Composability.** NPSNET-V offers an infrastructure that supports composition of heterogeneous contents and applications during run-time by using the Extensible Markup Language (XML) to form a structured data description.
- **Scalability in complexity and number of participants.** NPSNET-V uses the Lightweight Dictionary Access Protocol (LDAP) to build a hierarchical distributed database. The data is replicated across several servers to avoid a single bottleneck. Inter-entity communications preferably use a multicast-based network architecture to reduce traffic load.

Aside from the NPSNET series, many other research efforts have contributed in

(a) (b) (c)

Figure 1.3 Examples of typical academic DIAs. (a) NPSNET-IV [Macedonia *et al.* 1995]. (b) SPLINE. (c) MASSIVE [Greenhalgh *et al.* 2000].

the development of general software architectures for Distributed Virtual Environments (DVEs). The PARADISE distributed simulation system developed by the Distributed Systems Group at Stanford University employed an improved reckoning algorithm called the "position history-based dead reckoning". It also introduced the "projection aggregations" technique that only transmits summary information of a group of distant or uninterested remote entities, which do not merit the local viewer's experience at high detail [Singhal 1996; Singhal and Cheriton 1996]. The SPLINE (Scalable Platform for Large Interactive Network Environments) system shown in Figure 1.3(b) introduced the concept of breaking a virtual space into independently processed regions (or locales), to which different multicast addresses are assigned [Barrus *et al.* 1996; Waters and Anderson 1997]. The DIVE (Distributed Interactive Virtual Environment) system provides a flexible application-dependent partitioning mechanism by introducing an application-level backbone to connect different locales (also termed as worlds) [Carlsson and Hagsand 1993; Frécon and Stenius 1998; Frécon 2004]. The Chiba system adopts the idea of region-based communication and uses SpaceFusion technique to achieve a dynamical partitioning of the virtual world [Sugano *et al.* 1997]. Information from multiple regions is fused and presented to the users. The MASSIVE (Model, Architecture and System for Spatial Interaction in Virtual Environments) system (Figure 1.3(c)) incorporates the concept of spatial trading, in which the virtual world is structured, or divided, based on the user's range of awareness or interest [Greenhalgh and Benford 1995a,b;

Greenhalgh 1998; Greenhalgh *et al.* 2000].

An incomplete list of other notable developments in collaborative virtual environments includes MR Toolkit [Shaw *et al.* 1993], AVIARY [Snowdon and West 1994], BrickNet [Singh *et al.* 1995], PaRADE [Roberts *et al.* 1995], RING [Funkhouser 1995], Workbenches [Krüger *et al.* 1995], CAVERN series [Leigh and Johnson 1996; Roussos *et al.* 1997; Leigh *et al.* 1997; Park *et al.* 2000], Community Place (formally known as CyberPassage) [Lea *et al.* 1997], NetEffect [Das *et al.* 1997], VLNET [Sunday Pandzic *et al.* 1997], Virtual Society [Lea *et al.* 1997], AGORA [Harada *et al.* 1998], Living Worlds [Wray and Hawkes 1998], Virtual Playground [Schwartz *et al.* 1998], WorldToolKit [Rahn 1998], CIAO [Sung *et al.* 1999], COVEN [Babski *et al.* 1999], DEVA3 [Pettifer *et al.* 2000], DVECOM [Choukair and Retailleau 2000a,b], Urbi et Orbi [Verna *et al.* 2000], Virtual Park [Joslin *et al.* 2001], CyberWalk [Ng *et al.* 2002; Chim *et al.* 2003], VELVET [de Oliveira and Georganas 2003], DIVIPRO [Marsh *et al.* 2004] and ATLAS [Lee *et al.* 2007], MACVE [Lin *et al.* 2007; Zhang and Lin 2007]. DIP [Zimmermann *et al.* 2008] and PECOLE [Saddik *et al.* 2008].

1.1.3 Networked Multiplayer Games

Due to the widespread availability of the Internet and high-performance PCs and game consoles (such as the XBox [Microsoft 2005a]), networked multiplayer computer games have been the most rapidly developing area in DIAs during the last decade. Currently, Massively Multiplayer Online Games (MMOGs) support thousands of users simultaneously interacting with each other in the simulated virtual worlds [GameSpy 2003].

Although it is the last decade that has seen the huge popularity of MMOGs, the history of this game genre can be traced back to the earliest networked mainframe in the 1970s. There is a general agreement that MMOGs grew out of MUDs (Multi-User Dungeons), which was completed in 1978. In 1985, LucasFilm created a virtual online world called Habitat that supported up to 64 users on a private network named

(a) (b)

Figure 1.4 Scenes from typical Online Games. (a) Doom [Kushner 2002]. (b) Second Life [Linden Laboratories 2003].

QuantumLink [Farmer *et al.* 1994]. The first persistent virtual world considered to be "large-scale" was Meridian 59 in 1996 [GameSpy 2003]. It paved the way for the later burst of MMOGs in lots of ways such as the gaming interface, price model, text overhead, etc.

First Person Shooter (FPS) games such as Doom [id Soft 1993] and Quake [id Soft 1996] (as shown in Figure 1.4(a)) gained global popularity in 1990's. They gave players a first-person view of the virtual world. While Doom simply communicates updates at constant rates without any remote prediction scheme, Quake takes the advantage of unreliable transport for continuous communication and a variety of data compression methods to reduce game traffic. Quake was also featured with full 3D graphic rendering [Kushner 2002].

In 1997, Origin Systems released Ultima-Online [Origin Systems 1997], a true pioneer in the area of role-playing games, which introduced the concept of dividing the entire virtual world into separate shards ran by different server-clusters. It then spawned Lineage [NCsoft 1998] who had 4 million subscribers in 2002 [Castronova 2002]. The popularity of these games has reached the point where virtual social structures involving a massive number of interacting players have emerged [Seay *et al.* 2004]. A more recent example of such "on-line society" is the persistent online virtual world Second Life [Linden Laboratories 2003], which provides an ever-evolving 3D virtual community built by the player themselves in real-time (Figure 1.4(b)). This virtual

world has achieved a full virtual economy in which virtual treasure can be traded for real world currency.

1.1.4 Networking Issues

To achieve real-time interaction among multiple simultaneous users from geographically distant locations, a significant amount of information must be transmitted across the underlying communication network to maintain consistency between views of the shared virtual environment simulated on different hosts in a DIA. Therefore application layer protocol design in DIAs has to take into account various factors such as available network bandwidth, latency, lower layer protocol performance, and overall communication architectures.

1.1.4.1 Network Bandwidth and Latency

The two primary inhibiting factors to large-scale deployment of DIAs are finite network bandwidth and non-zero network latency [Capps and Stotts 1997]. Network bandwidth refers to the rate at which data can be transmitted along the inter-host connection. It can also be referred to as throughput per time unit. Despite the lack of a commonly accepted definition, latency, throughout this thesis, refers to the time taken to communicate data between the application layers of the two participating nodes [Delaney *et al.* 2006a].

Network latency can be decomposed as the sum of two types of delay from different sources: processing delay and propagation delay. The processing delay refers to the time taken to parse the packet and manage the transmission from source node to the destination, through the intermediate route. This includes application processing (such as compressing/decompressing and encrypting/decrypting) and operations on the relaying network nodes (such as packet framing/deframing, buffering, queuing, flow/congestion control). The propagation delay refers to the time required for the whole packet to be transmitted from the source to the destination node. This

includes the delay associated with the physical signal transmission speed and the delay related to the data transmission rate defined by the network bandwidth of the inter-node connection [Delaney *et al.* 2006a].

The limited network bandwidth restrains the frequency of the information exchanges between the participants. If the data transmission rate exceeds the bandwidth of the network link, the latency will increase due to queuing of the packets and loss of data may eventually occur. The effect of high network latency harms the participants' sense of shared space, time and presence. It also affects user performance and behavioral strategies [Vaghi *et al.* 1999; Park and Kenyon 1999; Gutwin 2001; Armitage 2003; Henderson 2003; Meehan *et al.* 2003; Sheldon *et al.* 2003; Beigbeder *et al.* 2004; Yasui *et al.* 2005; Claypool 2005; Claypool and Claypool 2006]. For different types of interaction, Table 1.1 summarizes user tolerances to network latency before notable performance declines.

Due to the heterogeneous and packet-relaying nature of the Internet, packets may go through different routing paths, router queuing lengths and buffer times, which results in a different latency for each transmitted packet. This unpredictable variation in network latency is referred to as jitter [Blow 1998; Smed *et al.* 2002b]. Jitter may cause more severe consistency problems than latency itself since it impedes users to adapt their strategies to network latency. With a low average latency of 10 ms and jitters up to 500 ms, a collaboration environment is almost as bad as one with a 200 ms latency but no jitter [Park and Kenyon 1999]. Also, jitter might twist the causal-effect chain of the events, since the packet notifying an event may arrive after the effect it triggered [Zhou *et al.* 2007].

1.1.4.2 Communication Architectures and Transmission Protocols

A DIA consists of a number of connected computers. Careful design of the communication architecture protocols to organize all the participating nodes and efficiently disseminate information among them is crucial to the implementation of the shared environment.

Table 1.1 Latency tolerances for different interactions

Application	Latency tolerance
Telephone conversation	100 ms [Cheshire 1996]
Web browsing	Tens of seconds [Bhatti *et al.* 2000]
DIS Military simulations	100–300 ms [IEEE 1996]
Target tracking in Virtual Reality	225 ms [MacKenzie and Ware 1993]
Warcraft III	500–800 ms [Claypool 2005; Sheldon *et al.* 2003]
Virtual RC car racing	200 ms [Pantel and Wolf 2002a]
Madden NFL football	500 ms [Nichols and Claypool 2004]
XBlast	139 ms [Schaefer *et al.* 2002]
Unreal Tournament 2003	60–100 ms [Beigbeder *et al.* 2004; Quax *et al.* 2004]
Quake	150–180 ms [Armitage 2003]
Everquest 2	500 ms [Fritsch *et al.* 2005]

Generally speaking, there are two basic architectural network topologies: client-server (C-S) and peer-to-peer (P2P). In the C-S architecture, all end-users (clients) are connected to a central server node. The server is responsible for collecting all the up-to-date entity states from the controlling clients, maintaining a definitive view of the environment and distributing entity information to the related clients. Examples of systems employing the C-S architecture include RB2 [Blanchard *et al.* 1990], RING [Funkhouser 1995], VLNET [Sunday Pandzic *et al.* 1997], NetEffect [Das *et al.* 1997], AGORA [Harada *et al.* 1998], DVECOM [Choukair and Retailleau 2000a,b], BrickNet [Singh *et al.* 1995] and MASSIVE-2 [Greenhalgh 1998]. In the P2P architecture, the participating hosts play equal roles and directly exchange information with each other. Examples of the P2P architecture include SIMNET [Calvin *et al.*

1993], MASSIVE-1 [Greenhalgh and Benford 1995a], DIVE [Carlsson and Hagsand 1993], MR Toolkit [Shaw *et al.* 1993] and NPSNET [Macedonia *et al.* 1994].

It is relatively easier to maintain consistency in a C-S system since the server acts as a central authority that controls the data distribution. However, the problem is that the server may become a bottleneck when the quantity of clients and data in the application scale up. The distributed structure of the P2P architecture improves local responsiveness and reduces the system's dependence on centralized control, but global consistency is difficult to maintain since the replicated versions of the world vary among users. These two basic topologies can be combined to build hybrid architectures where the responsibilities of object database maintenance and communication control are shared by clients and servers. In typical hybrid systems like SPLINE [Waters and Anderson 1997] and PaRADE [Roberts *et al.* 1995], groups of clients are assigned to different servers. Intra-group clients communicate in the C-S mode, while inter-group communications are relayed between servers in a P2P manner. The concept of the three architectures is shown in Figure 1.5.

Aside from network architectures, communication protocols guarantee that data transfer is proceeded between the network nodes following well-defined procedures so that each message has its exact meaning and provokes intended response. These protocols can work at different layers of OSI (Open System Interconnection) or TCP/IP reference model [Tanenbaum 1998]. Most DIA applications support IP (Internet Protocol) protocol at the network layer, and vary in protocol choices at the application layer and the transport layer.

The two most commonly employed transport layer protocols by DIAs are the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP) [Tanenbaum 1998; Roehl 1995]. TCP provides a connection-orientated stream communication that guarantees all packets are received intact and in the correct order. However, this best-effort nature of TCP comes with the cost of a packet header up to 64 bytes [Tanenbaum 1998], which translates into latency. UDP, on the other hand, disregards these sophisticated control schemes, which means that packets may occasionally get lost or arrive in an incorrect order. As a result, UDP packets are more

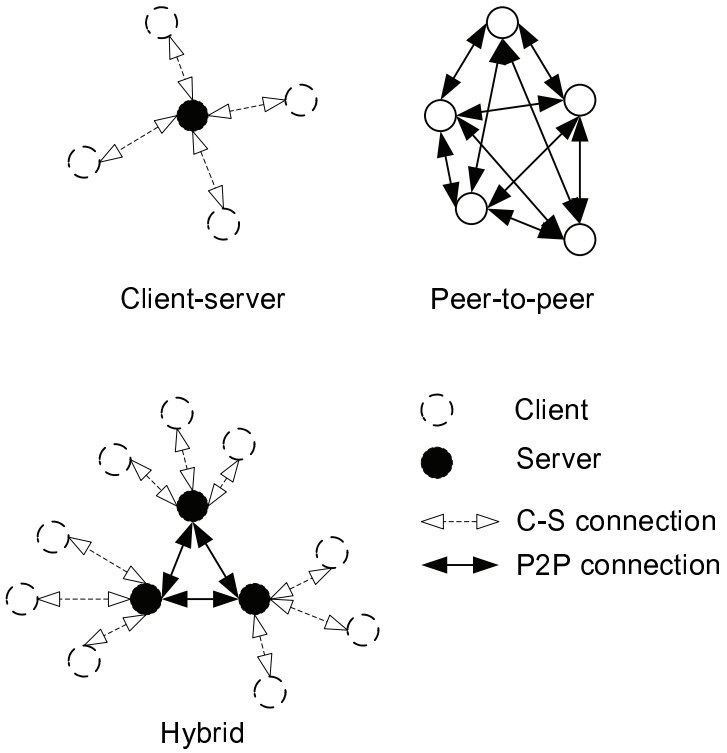


Figure 1.5 Different network architectures.

lightweight and preferred in communicating time-sensitive data such as entity state update information. Reliability if required is usually left to the application layer when using UDP.

Except for the transport layer protocols that are designed for general-purpose communications, there are also application layer protocols that are optimized specifically for large-scale distributed interactions. The Real-Time Application Level Protocol for Distributed Interactive Video (RTP/I) regards events, states, state changes and state queries as the four basic data types for DIAs communication, and is thoroughly optimized to meet the demand of DIA functionalities [Mauve *et al.* 2001]. A good overview of other application-specific protocols can be found in Delaney *et al.* [2006b].

Each computer or host machine in a DIA runs the DIA software and maintains a local instance of the virtual environment, which is presented to the human-user who controls a virtual *entity/object* through this *local host* (or owner). Such an

entity is the representative of the user in the virtual world. The *state* of the entity provides a complete description of this entity at a moment of time. The local host also observes the states of other entities controlled by *remote hosts*. In order to maintain consistency and enable real-time interactions in the face of the networking issues, a careful design (in terms of balancing performance and network resource consumption) of entity state distribution schemes is required. Typically, effects of the human interactions and operations are managed, based on the entity state replica in this world instance, to maximize local responsiveness of the application. The true state of the entity controlled by the local host machine is presented to the user with high fidelity. The changes of the local entity state are transmitted periodically in Entity State Updates (ESUs) for the remote host machines to maintain their own version of this local entity state with lower fidelity.

ESU packets constitute the majority of the network traffic during a DIA simulation. Therefore the regulation of ESU transmission is of prime importance in consistency maintenance in DIA deployment. One important group of consistency maintenance mechanisms, collectively known as Predictive Contract Mechanisms (PCMs) have been proposed and are widely used to reduce the amount of data transmission required to maintain consistency in DIAs [Calvin *et al.* 1993; Mellon and West 1995; Singhal and Zyda 1999; Frohnmayer and Gift 2000; Pantel and Wolf 2002b; Delaney *et al.* 2006a; McCoy *et al.* 2007; Yu *et al.* 2007]. PCMs introduce a form of *controlled inconsistency* by reducing the number of ESUs transmitted across the network in order to minimize latency caused by an overloaded network. The reduction of update transmission is governed by predetermined parameters on the local host, such as an inconsistency threshold or a constant update packet rate. On the remote hosts, the intermediate entity motion between two updated states is extrapolated using some prediction algorithm that, at some level, reflects the user behavior pattern. In spite of the inconsistency introduced by the reduced number of updates, the reduction in network traffic would in theory result in a reduced load on the network link and hence reduced latency, which, would in turn, ameliorate the overall inconsistency.

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 2002b]. Standard DR algorithms are simple and readily applicable to most DIA scenarios, but ignore the contextual dependence of the behavior of an entity that makes its future states predictable. There has been significant research focused on improving prediction accuracy by employing statistical modeling techniques to generate better prediction models that capture and reflect more human behavioral patterns [Zukerman and Albrecht 2001; Delaney and Ward 2004; McCoy *et al.* 2007]. These approaches attempt to produce accurate entity motion models to reduce the data transmission required to maintain a certain level of consistency. 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. Also, there has been no method to measure the predictability of the user behavior and how the predictability is used by these predictive techniques. There is clearly a need for a method to quantify and analyze the utilization of such predictability by PCMs for further optimization of PCM operation.

Adaptive user modeling approaches have been proposed to account for changes in user behavior. These techniques consist of parallel entity state models predicted by a pool of extrapolation algorithms. Only the state model estimated by the best performing algorithm, which is dynamically selected according to certain local accuracy criteria, is presented to the users [Lee *et al.* 2000; Delaney *et al.* 2003; McCoy *et al.* 2005]. However, the impact of changing the extrapolation model under use on the remote inconsistency is not clear, since the data transmission over the underlying network is not considered.

Choosing suitable values for the error threshold or a constant update rate in order to balance the trade-off between the number of update packets and consistency is also an important performance issue [Lee *et al.* 2000; Yu and Choy 2001; Chen and Chen 2005; Roberts *et al.* 2008; Kenny *et al.* 2009]. Consider rate-based PCMs for example. A low update rate reduces the network load and inconsistency caused by latency, but at the same time introduces an extra level of inconsistency by reducing the amount of state information transmitted. The remote host has to use the previously received ESU for a long time interval before a new update arrives, which leads to inconsistency

in the extrapolated remote entity model. On the other hand, a very high update rate increases the amount of data transmitted between participants and has negative impact on remote inconsistency due to the increased latency induced if the traffic exceeds the network bandwidth. The increases and decreases of inconsistency from the application layer and network layer perspectives can be jointly summarized as the *Consistency-Throughput Trade-off*, which states that [Singhal and Zyda 1999]:

> *It is impossible to allow dynamic shared state to change frequently and guarantee that all hosts simultaneously access identical versions of that state.*

Consequently, it is difficult to ascertain the overall effect of modifying the inconsistency control schemes. The suitability of the PCM configurations in the application must be examined with respect to network conditions in order to maximize utilization of the underlying network resources [Marshall *et al.* 2008].

1.2 Objectives of this Thesis

The focus of this thesis is a theoretic investigation into the predictability of representative types of user behaviors within DIAs, and how this predictability is exploited by the operation of PCMs in maintaining consistency with reduced data transmission. The development of an information-based mechanism to improve consistency maintenance is also proposed.

In the first part of this thesis, a novel information model for PCMs will be proposed to facilitate analysis of consistency maintenance mechanisms in DIAs. Attributes of entity motions relating to the temporal dependence in their state-evolutions, and the efficiency of PCMs in utilizing this dependence for building the remote entity state model will be quantified and analyzed using concepts and measurements derived from information theory. This analysis will show that the performance of a PCM in consistency control depends on the amount of information about the user behavior

it extrapolates. This thesis will demonstrate that the proposed information model provides new perspective in understanding the *Consistency-Throughput Trade-off* and the optimization of the entity state dissemination protocols. In particular, the operation of PCMs can be interpreted as a form of information reduction and video compression.

In the second part of this thesis, the design of a novel Information-Based Dynamic Extrapolation Model for rate-based PCMs will be presented. The proposed information model enables an evaluation of the performance of a PCM that combines the application and network layer perspectives. This thesis will demonstrate that by using the Information-Based Dynamic Extrapolation Model, the operation of PCMs implemented within a DIA can be dynamically adapted to match the user behavior and the network in which they operate, thus allowing both the application and network level performance to be improved.

1.3 Original Contributions of this Thesis

A number of original contributions are presented throughout the work contained in this thesis. The major contributions of this work are as follows:

1. A novel information model for Predictive Contract Mechanisms is developed. The information model employs concepts and techniques derived from information theory to quantify and analyze the efficiency of PCMs in communicating reduced state information in return for controlled inconsistency. The proposed information theoretic approach is a model-independent framework that can be applied to any predictive schemes employed by a PCM.

2. Based on the information model, the operation of PCMs are modeled as an information generation, reduction, transmission and reconstruction process. The entity state distribution in a DIA is seen as an information flow. The information perspective interprets consistency maintenance using PCMs as

a lossy video compression and presents novel insights into the *Consistency-Throughput Trade-off* that DIAs face.

3. A dynamic extrapolation framework, referred to as the Information-Based Dynamic Extrapolation Model, is developed to allow the operation of PCMs to be aware of the application layer user behavior and network layer performance. An analysis of the performance of the Information-Based Dynamic Extrapolation Model demonstrates that the dynamic extrapolation framework can make the most information-efficient use of the available network resources to deliver the highest information about the local entity state to the remote entity state model and thus optimize consistency in a DIA.

During the course of the development of each of these key contributions, a number of other important contributions are also made:

1. Various entity state evolutions of representative user behaviors within DIAs are analyzed from the viewpoint of information theory. Information measurements present a general and quantified description of the statistical dependence in entity motions, and highlight their predictability characteristics that enables the use of PCMs to maintain consistency with reduced ESU transmission.

2. An analysis of a number of representative PCMs, including standard DR and a recently proposed PCM (Neuro-reckoning [McCoy *et al.* 2007]), with respect to their utilization of the information available for remote extrapolation, is conducted through experimental studies. It is demonstrated that the performance of an extrapolation model depends on its efficiency to exploit the information encapsulated in the received ESUs. This analysis also implies further improvements for the current techniques.

3. The trade-off between packet rate and packet size modification within the Information-Based Dynamic Extrapolation Model is examined to achieve an efficient usage of the available network resources.

4. An analysis of inter-entity interaction within a First-Person Shooter (FPS) style networked multiplayer computer game is conducted using the information measurements. Windowed Cross-Mutual-Information presents a quantified and visual description of the time-evolution of the state-dependence among interacting users. Such an analysis reveals the possibility of exploiting cross-entity predictability to further reduce the data transmission required for maintaining consistency in a DIA.

1.4 Outline of this Thesis

The remainder of this thesis is organized as follows:

- **Chapter 2** gives an extensive review of concepts and techniques relating to consistency maintenance in DIAs. Predictive Contract Mechanisms are also explored in this chapter to provide a foundation for the work presented throughout this thesis. A discussion of general video compression procedures points out the analogy between video compression and consistency control in DIAs, and reveals that the basic functionalities in managing state sharing in DIAs can be treated as a form of video compression.

- **Chapter 3** reinterprets Predictive Contract Mechanisms from an information theory perspective. Following a brief review of the concepts and techniques in information theory, the information model is formalized based on an information analysis performed on each of the components of PCMs. Inconsistency arising from the execution of PCMs is modeled as information loss.

- **Chapter 4** presents and discusses the results from applying the information model to different type of user motions. The predictability of the user behaviors are quantified, and then the operation of PCMs to utilize this predictability is evaluated through comparative studies. Improvements to the current extrapolation schemes are also suggested.

- **Chapter 5** proposes the Information-Based Dynamic Extrapolation Model. The results from the simulation employing this dynamic extrapolation model to maximize bandwidth utilization efficiency are also discussed.
- **Chapter 6** offers final concluding remarks and potential future directions that arise from the work presented throughout this thesis.

Chapter 2

Background and Related Works

2.1 Introduction

In this chapter, the existing literature relating to the work presented throughout this thesis is explored. The chapter is divided into two main parts.

The first part of the chapter deals with the issues surrounding consistency and consistency maintenance in DIAs. Following definitions of common terms and terminology used in the area, Distributed Interactive Applications are categorized based on the nature of the different types of interactions involved. The key elements that define consistency, the main objective of DIAs, and consistency measures are then explored. A specific class of consistency maintenance techniques known as Predictive Contract Mechanisms, such as the well-known Dead Reckoning and various extensions to it, are extensively reviewed in terms of network traffic reduction in order to overcome network limitations to provide a consistency view of the virtual environment. This provides a necessary foundation for the work presented in this thesis.

The second part of the chapter deals with the area of video compression. The operations of general video compression are first introduced, with particular focus on the relationship between the contextual dependence in the raw video and the compression. Finally, the link between video compression and consistency maintenance

in DIAs is pointed out, providing the motivation for the work presented throughout this thesis.

2.2 Distributed Interactive Applications (DIAs)

2.2.1 Terminology

A Distributed Interactive Application is a networked software system that seeks to maintain global consistency when responding to multiple simultaneous non-deterministic inputs [Delaney *et al.* 2006a]. DIAs have been referred to by many different terms as summarized in Table 2.1. These terms reflect various research emphases of this application class, such as education, gaming, group editing, collaborative performing, and etc. In this thesis, we adopt the term Distributed Interactive Application [Diot and Gautier 1999; Lee *et al.* 2000] because it is a generic term that embraces the two essential elements of these networked applications, namely *distributed* (participants of the system are located so far away from each other that message transmission among them involves non-negligible delay compared to the time between events in a single process [Lamport 1978]) and *interactive* (the systems is at least partially driven by inputs from human users through a human-computer-interface and feeds back with appropriate response [Delaney *et al.* 2006a]):

> *A Distributed Interactive Application is a networked software system that seeks to maintain global consistency when responding to multiple simultaneous non-deterministic inputs.*

For the convenience of discussion in the following sections and chapters, common terms deriving from the development of DIAs and repeatedly used in this thesis will now be defined.

- **Entity/Object:** a virtual component in a DIA that can in some way change

Table 2.1 Various terms that overlaps with DIAs.

Computer-Supported Cooperative Work	[CSCW 1986]
Groupware System	[Ellis and Gibbs 1989]
Distributed Virtual Environment (DVE)	[Stytz 1996; Lui 2001], [Frécon and Stenius 1998], [Lee *et al.* 2007]
Shared Virtual Environment (SVE)	[Waters and Barrus 1997]
Distributed Interactive Simulation (DIS)	[IEEE 1998]
Collaborative Virtual Environment (CVE)	[Benford *et al.* 1994; Chris 1999], [Greenhalgh and Benford 1995a], [Babski *et al.* 1999], [Park and Kenyon 1999], [Vaghi *et al.* 1999]
Networked Virtual Environment (NetVE)	[Singhal and Zyda 1999]
Distributed Interactive Media (DIM)	[Mauve *et al.* 2001]
Networked Interactive Entertainment (NIE)	[Diot and Gautier 1999], [Qin 2002]

its state [Stytz 1996; Roehle 1997]. An entity can be *active* or *passive* [Singhal and Zyda 1999]. A passive entity either remains entirely static as a part of the scene designed or, such as an autonomous **Bot** controlled by software alone [Roehle 1997], moves deterministically. The behavior of an active entity controlled by a human user is non-deterministic, thought it may be at some level predictable due to environmental constraints or the behavior of the participant controlling.

- **Avatar:** a dynamic entity controlled either by a participant or automatically by the system [Diot and Gautier 1999]; an entity controlled by a human user [Roehle 1997] and conveys the identity, presence, location, and activities to others [Capin *et al.* 1997; Benford *et al.* 2001]; the visual embodiment of a user and the means of interaction with the others through expressing his/her presence and actions [Yura *et al.* 1999; Yu and Choy 2001].

- **Agent:** an autonomous entity controlled by software alone [Roehle 1997]; a software process that has autonomous behavior. It does not necessarily have to be represented by a graphics entity [Capin *et al.* 1997].

- **Virtual Environment/World:** the set of information needed to render a participant's view of an application's time-constant state [Mauve *et al.* 2001]; the simulated software system in which entities interact and communicate with one another, and the integrated set of algorithms, attributes, elements, and structures that comprise the system [Singhal and Zyda 1999].

- **State:** a complete description of an entity at a single moment in time [Churchill *et al.* 2001]. The *dynamic state* or *dynamic state vector* is the collection of information that describes an (active) entity in full [Roehle 1997]. The current location, velocity, orientation, and any other attributes of an entity that change over time are part of its state. The information relating to state changes should be shared among the participants. This information is also referred to as *dynamic shared state* [Singhal and Zyda 1999]. *Static state* of an (passive) entity, on the other hand, remains fixed over time.

- **Participant:** a real person who participates in the networked virtual environment and is represented by an avatar [Capin *et al.* 1997].

- **Event/Action:** the state of an entity in a DIA may change for two reasons: *event* or *passage of time*. An event is what causes a state change that is not a fully deterministic function of passage of time [Mauve *et al.* 2001]. An event may be *external* or *internal* [Mauve 2000b]. External events are user interactions or user-initiated operations, such as the user changes the direction of the entity movement. Internal events are non-deterministic state changes

within the application itself, e.g. the generation of a random number. Also, an event can either be *deterministic* or *non-deterministic* according to the predictability of its occurrence [Roberts and Sharkey 1997; Sharkey *et al.* 1998].

- **Host:** a computer system within a DIA that controls and inserts entities into the DIA and observes the actions of other entities, whether machine or human controlled [Stytz 1996]. To a specific entity, the *local host* is the host that controls that object, and the *remote hosts* are all the other hosts that observe and interact with it.

- **Node:** A general term denoting either a switching element in a network or a host computer attached to a network [IEEE 1996]. A node can be a source machine, router, destination machine, or any other device that processes data [Delaney *et al.* 2006a].

Other terms will be defined as they are encountered.

2.2.2 Classification of DIAs

A DIA consists of interactive entities controlled by distributed participants connected by a network. The features of DIAs suggest two categorizations of DIAs: A first categorization is by how the participants are distributed in terms of network topology, which is described in section 1.1.4.2; and a second categorization is focused on the different types of interactions that a DIA tries to provide to the users.

An interaction between entities in the context of DIAs is a communication that causes a change of their states [Natrajan and Reynolds 1999]. Hence, interactions, and the corresponding DIAs, can be classified based on two characteristics of the mechanisms that an entity in the environment can change its state. The first characteristic is the way that entity state changes are driven: *discrete* or *continuous* [Mauve 2000b; Mauve *et al.* 2001; Zhou *et al.* 2001]. In discrete applications, the entities change their states only as a result of events initiated either by the human users or the application itself. The entity states between the events remain static. The

shared document in a collaborative editing system is an example of a discrete entity. In continuous applications, entities change their states in response to the passage of time, in addition to non-deterministic events. State changes of a continuous entity between the events are governed by state evolution laws relating to the specific entity dynamics implemented by the application. An aviation simulation system is an example of such continuous applications. The pilot of an airplane may change the speed and orientation from time to time, generating user initiated actions. The airplane changes its position between the actions according to the simulated physics. It is worth noting that discrete state changes are a necessary element of a continuous DIA, because a system composed of pure "time-driven" entities, which do not accept events and evolve deterministically by the passage of time, is considered non-interactive.

The second characteristic in this classification is the means by which the participants take actions to an object in the environment: *turn-based* or *concurrent*. In turn-based applications such as board games, users make their moves in turn so that only one of them can interact with the object at one time instance. In concurrent applications, users may take simultaneous actions to the same object at the same time. The effect of such actions is resolved by concurrency control algorithms [Ellis and Gibbs 1989; Sung *et al.* 1999]. Most of the DIAs discussed in Chapter 1 are concurrent systems. Based on these two characteristics, DIAs can be generally classified into four categories [Zhou *et al.* 2001]. Figure 2.1 illustrates this general taxonomy in two dimensions.

- **Continuous Concurrent Applications:** the entity states evolve as a joint result of discrete events and the passage of time. Participants my take their action concurrently. Most DIAs such as SIMNET, DIS, MMOGs are examples of such applications.
- **Continuous Turn-Based Applications:** such applications consist of continuous entities, but participants can only take their actions in turn. Examples include tennis games where the ball and players moves continuously, but the players must take turn in generating events to the object, i.e. hitting the ball,

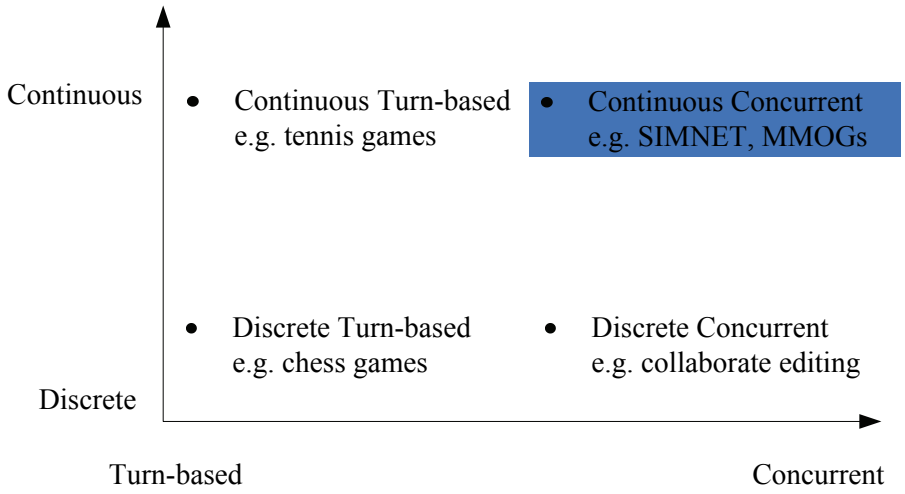


Figure 2.1 General taxonomy of DIAs based on the types of interactions involved in the applications.

as implied by the rule of the game.

- **Discrete Concurrent Applications:** participants may take actions concurrently, but the entities only accept events as the reason to change entity states. Examples include collaborative editing systems.

- **Discrete Turn-Based Applications:** in such applications, entities in the environment only change their states in response to discrete events initiated by the application or human users, who make their moves in turn. An example is online chess games.

It is important to make a classification to DIAs before discussion on consistency maintenance because the meaning and requirement of consistency may vary for different interaction contexts. For discrete applications, the entity state changes are solely driven by non-deterministic events and are independent of the passage of time. Therefore the correct order of the events, rather than the exact times of their occurrences, is sufficient to guarantee a correct entity state evolution. For continuous systems, on the other hand, the exact times when the non-deterministic events take place become necessary since entity states also evolve with the passage of time.

Another classification based on the management of time divides DIAs into two classes [Lamport 1978; Delaney *et al.* 2006a; McCoy 2007; Marshall 2008]:

- **Logical/Causal/Virtual Clock Applications:** in such applications, a clock is just a way of assigning a number to an event. Time is regarded as a sequence of ordered events, which stands still if no event happens. Logical clocks are sufficient for discrete DIAs.
- **Physical/Absolute/Wall Clock Applications:** in such applications, time is based on a periodic clock which is in most of the cases consistent with the real world clock. In the Internet, Physical clock is synchronized to coordinated universal time (UTC) by a network of servers using Network Time Protocol (NTP [Mills 1991]). Continuous DIAs is generally run by physical clocks.

Throughout this thesis, DIAs to be explored are assumed to consist of distributed participants organized using either the peer-to-peer or client/server architectures. Each participant, through a host machine, controls an interactive entity with real-time, continuous, and concurrent state variables. Such DIAs try to embrace the following features that are necessary in synchronizing entity states to mimic real-world interactions [Bouillot and Gressier-Soudan 2004]:

- **Causality:** the well-known "happening before" relation [Lamport 1978]. Preserving the temporal relation of the events guarantees the correct causal-effect relationship, which is vital to the users' understanding to the environment.
- **Concurrency:** multiple users can take actions on the same object at the same time.
- **Simultaneity:** if two, or more, actions performed by different participants are perceived by one user as simultaneous, then any participant will perceive the actions as simultaneous.
- **Instantaneity:** the time between the taking place of an action and the play-out of its effect from a user's perspective is negligible. This requires that the system's response time must be short.

Now, the issue of maintaining consistency in a distributed continuous DIA will be explored in the next section.

2.3 Consistency

Conceptually, a DIA consists of a group of distributed users who have access to a distributed database that contains dynamic shared entity states [McCoy 2007]. Changes made to the shared states by the human users interacting within the environment must be appropriately reflected within the distributed database to build the sense of shared time, space and presence. Therefore, maintaining consistency in a DIA not only provides a consistent view of the virtual world to the users, but also directly affects the level of interactivity offered by the application. However, under a realistic network environment, some of the four features needed to imitate the real world interactions are contradictory. For example, achieving both simultaneity and instantaneity is impossible in the face of non-zero network latency, because instantaneity requires that an action is operated on the local host immediately, while the remote hosts could only receive the update of this event after some time relating to the latency, which makes simultaneity impossible. This is also known as the *Consistency-Responsiveness Trade-off* [Bhola *et al.* 1998; Bouillot and Gressier-Soudan 2004].

The requirements to maintain consistency are specified and analyzed using consistency models. Consistency models put constraints on the temporal and spatial relations of entity state versions for different users. Consistency models can be distinguished by the types of clock and event-playout schemes utilized [Bouillot and Gressier-Soudan 2004]:

- **Wall-clock** based models use synchronized wall clock time to define consistency criteria.
- **Logical-clock** based models use logical time stamps to define consistency criteria.

- **Ultimate** based models order data before playing the action out to the users. "Ultimate" means the operation of an action can be delayed for "a non-deterministic amount of time". Ultimate consistencies provide very bad responsiveness and may break the temporal order of the events, and are thus particularly inappropriate for continuous applications. Simultaneity is not supported by ultimate consistency. Logical-clock based ultimate consistency preserves the order of the events playout and are used in some discrete applications such as chatting and whiteboard systems with additional concurrency resolution schemes [Strom *et al.* 1997; Chen and Sun 1999; Galli and Luo 2000; Li *et al.* 2000; Sun and Chen 2000].

- **Deadline** based models express a wall-clock relationship between the initiation and playout of an action. This notion requires that an action is operated on all the sites no later than a specific delay after its generation (reading on time), or at an exact time after its generation (reading at the specific time). Among all consistency models, Perceptive Consistency has the strictest consistency criterion, because it requires that any action is played simultaneously (in terms of an exact instance in wall clock time) to everyone interacting. In this way, it provides simultaneity, concurrency management and causality. The drawback of this model is that the local operations of the events must be delayed to be synchronized with the remote operations [Gautier and Diot 1998; Mauve 2000a; El Saddik *et al.* 2003]. The responsiveness is thus constrained by the local delay. Perceptive Consistency is the closest to the "What You See Is What I See" model [Stefik *et al.* 1987] and is formally defined as *absolute consistency* in Qin [2002]. Late Consistency [Qin 2002] is a relaxation of perceptive consistency in that the local actions are executed immediately. The remote operations of the actions are still delayed and synchronized as in perceptive consistency. Late Consistency trades simultaneity for a good responsiveness. It is worth noting that in perceptive and late consistencies, the operation of an action is delayed to be executed at the same time on all the remote hosts, which means that this delay must be no less than the worst transmission latency. This implicates that a remote host who receives the

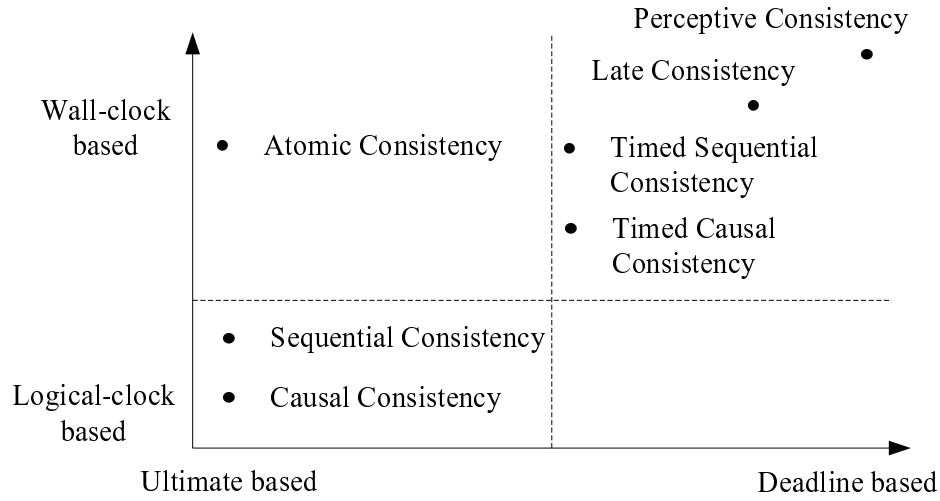


Figure 2.2 General taxonomy of consistency models [Bouillot and Gressier-Soudan 2004]. Perceptive Consistency imposes the strictest consistency constraints.

action early has to wait until the last remote host receives this action. The time delay of the action operation can be determined as a constant response time [Gautier and Diot 1998; Mauve 2000a], or based on the network Round Trip Time (RTT) [Akyildiz and Yen 1996; El Saddik *et al.* 2003].

Figure 2.2 illustrates a number of consistency models in terms of the two dimensions discussed above. Notice that consistency models based on wall-clock and deadline criteria generally impose stronger constraints. Throughout this thesis, the following definition of consistency proposed by Delaney [2004] is used:

> *Consistency is the maintenance of a uniform dynamic shared state across all participants in a DIA.*

The limitations of the network negatively impact on consistency maintenance in a DIA and cause inconsistency that manifest itself in a number of ways [Sun *et al.* 1998; Vaghi *et al.* 1999; Zhou *et al.* 2001]:

- **Divergence/Presentation Consistency:** the different users' views of the

virtual world may vary because events may be executed at different times or in different orders.

- **Causality Violation/Interaction Consistency:** due to the non-deterministic latency in update transmission, events may be operated out of their cause-effect order. For example, in a shooting game involving three users on different hosts, player A opens fire and then player B is killed as a result. Due to an unreliable connection, the event of B's death might arrive earlier than its cause, namely A firing, on an observing host C. The cause-effect order of this event pair is broken.
- **Expectation/Intention Violation:** occurs when the actual effect of an event by the initiating user is different from the intended effect.

The focus of this thesis is the divergence dimension of inconsistency, namely the difference between the views of the entity states on the local and remote host machines. In the next section, common metrics quantifying inconsistency in DIAs are reviewed.

2.3.1 Consistency Metrics

Defining a measure for the degree of consistency in a DIA is an important issue for the detection and resolution of the arising inconsistency as the system operates. Most consistency measurements in the literature generally focus on the spatial or temporal divergence between the local and remote versions of the same entity state variable.

The most popular spatial metric of consistency is the Euclidean distance between the positions of the local and remote versions of an entity in the environment, which is also referred to as drift distance [Gautier and Diot 1998]. The shared state in the DIA is considered inconsistent when the spatial inconsistency is large compared to a pre-determined threshold, which is usually related to the size of the avatar. The Exported Error [Aggarwal *et al.* 2004] examines the state difference of an entity

between different hosts at the time a state update arrives at the remote host. This difference is due to the remote host using the previous updated information to model the entity state during the time when the latest update is still in transmission.

Temporal consistency metric, or Phase difference [Lui 2001], considers the difference in times of the operations of the same event on the different sites, and is closely related to network latency.

While spatial consistency metrics only consider the instantaneous divergence of the shared state and temporal consistency focuses on the duration of an inconsistency but not the effect of applying events at diverged time instances, a time-space consistency metric has been proposed by Zhou *et al.* [2001, 2004] based on the premise that both the magnitude of an inconsistency and its duration affect the human perception of the inconsistency. This metric combines the time and space metrics into a single inconsistency metric as shown in (2.1):

$$\Omega = \begin{cases} 0 & , \quad \text{if } |\Delta(t)| < \varepsilon, \\ \int_{t_0}^{t_0+\tau} |\Delta(t)| \; dt & , \quad \text{if } |\Delta(t)| \geq \varepsilon, \end{cases} \tag{2.1}$$

where

$\Delta(t)$ is the instantaneous spatial difference of the entity state on different hosts,

t_0 is the time when the difference starts,

τ is the duration that the difference lasts,

ε is the minimum distance that a human user can discern.

In this definition, the time-space inconsistency Ω represents the area under the graphical evolution of the spatial inconsistency over a specific period of time for a particular entity. If the spatial inconsistency $\Delta(t)$ is negligible compared to the human perception limit throughout the time period of interest, we have $\Omega = 0$, and the Absolute Consistency is achieved [Zhou *et al.* 2001] since different users have the same view of the entity all the time. An equivalent metric is proposed by Li *et al.* [2004].

Inconsistency is defined as an information cost that combines spatial deviation, uncertainty and communication overhead [Wolfson *et al.* 1998]. Uncertainty by

concept means the area where an entity can possibly reside at a specific moment, but is however not formally defined in the literature.

2.3.2 Consistency Maintenance Mechanisms

For continuous DIAs, the dynamic nature of the entity states makes it impossible to achieve absolute consistency, because the local entity states change during the latency caused by transmitting any update to the remote host. Still, a number of techniques have been developed to help maintain sufficient consistency in DIAs under constrained network conditions. Based on their handling of the *Consistency-Responsiveness Trade-off*, these techniques fall into one of two categories [Jefferson 1990; Bhola *et al.* 1998; Vaghi *et al.* 1999; Fujimoto 2001; Cronin *et al.* 2004]:

- **Optimistic/Aggressive:** An optimistic mechanism promotes the user's perception of local interaction by performing speculative computation of the entity state before being confirmed by update packets or synchronized with other hosts in the application. The speculated state saves local response time if subsequently determined correct. The inconsistency arising from incorrect speculations must be rolled back by some repair scheme such as Time Warp [Jefferson 1985].
- **Pessimistic/Conservative:** A pessimistic mechanism does not indulge any speculative computation. The system only updates the entity state when it is synchronized with all other nodes. Such precaution avoids violation of causality and ensures consistency at the expense of interactivity.

Optimistic mechanisms prefer responsiveness over consistency, and vice versa for pessimistic mechanisms. Therefore, optimistic mechanisms are generally more favorable to real-time interactive applications from the perspective of a participant.

Various consistency maintenance mechanisms can also be classified into three classes [Delaney *et al.* 2006a]:

- **Information Management Techniques** reduce the amount of data that has to be transmitted over the network to compensate for the effect of network limitations in an DIA. Examples of information management techniques include Predictive Contract Mechanisms [Mellon and West 1995; Delaney *et al.* 2006a], relevance filtering [Van Hook *et al.* 1994; Bassiouni *et al.* 1997] or interest management [Morse *et al.* 2000; Boulanger *et al.* 2006], packet bundling [Liang *et al.* 1999], and data compression [Gutwin *et al.* 2006].

- **Time Management Techniques** manipulate the passage of time in the applications to mask the effect of network latency. Examples of such techniques include bucket synchronization or local-lag [Gautier *et al.* 1999; Mauve *et al.* 2004], local perception filtering [Sharkey *et al.* 1998], predictive time management [Roberts and Sharkey 1997], and lockstep synchronization [Blow 1998].

- **System Architecture Techniques** seek to improve the efficiency of processing and disseminating data throughout a DIA. Examples include communication protocols and network architectures, and sufficient Quality-of-Service (QoS) [Greenhalgh *et al.* 1999; Choukair and Retailleau 2000a,b].

The three categorizations discussed here are not mutually exclusive within their respective angles. Hybrid approaches can be designed to suit the particular demand. In addition, extensive reviews of aspects of networking and consistency in DIAs can be found in published works [Stytz 1996; Macedonia and Zyda 1997; Roehle 1997; Benford *et al.* 2001; Smed *et al.* 2002a,b; Joslin *et al.* 2003, 2004; Delaney *et al.* 2006a,b].

The focus of this thesis is information management techniques, which try to optimize consistency by tuning the amount of data transmission across the network for a reduction in latency. This would in turn improve overall consistency. The operation of one important group of information management techniques, namely Predictive Contract Mechanisms, is now discussed in the next section.

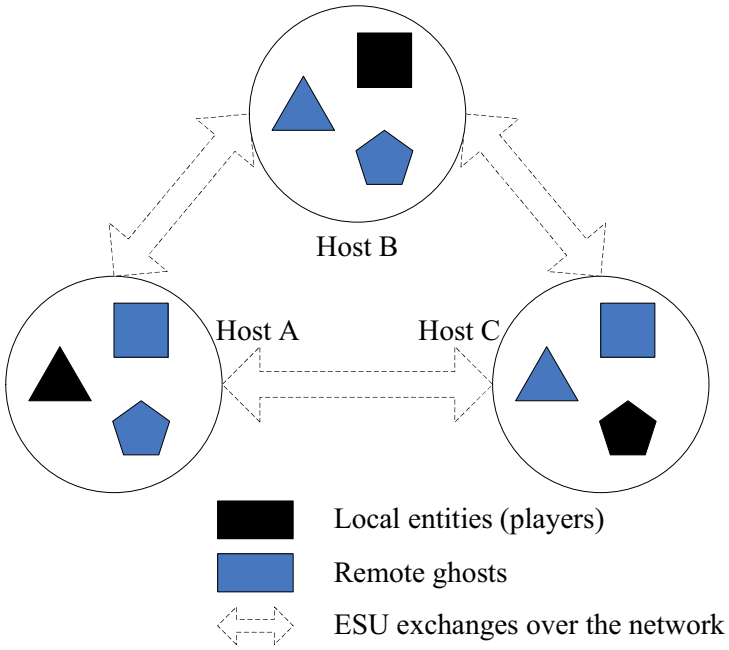


Figure 2.3 The *Players and Ghosts* paradigm in Predictive Contract Mechanisms.

2.4 Predictive Contract Mechanisms (PCMs)

In order to enable real-time interaction, Predictive Contract Mechanisms typically distribute the dynamic shared entity states in a *Players and Ghosts* paradigm [Blau *et al.* 1992]. During a simulation, the entity controlled by the human user through the local host machine is called a *Player* and maintains accurate and instantaneous state information in response to the user's inputs. An associated *Ghost* entity is maintained by each remote host as an approximation to the true entity state of the player for remote entity modeling and interaction. The local host transmits Entity State Updates about changes to the true entity state of the player at given instances in simulation time so that the remote hosts can maintain sufficient level of consistency of the shared state. The concept is explained in Figure 2.3.

By employing the ghost representation, PCMs maximize responsiveness because the effect of the local user action is rendered immediately, providing the sense of real-time interaction. Therefore, consistency of the shared state must be to some extent compromised because of the *Consistency-Responsiveness Trade-off*. PCMs employ

a form of *controlled inconsistency* by reducing the number of ESUs in order to minimize inconsistency caused by latency for transmitting data across an overloaded network.

PCMs can be classified into two classes, namely rate-based and threshold-based PCMs, relating to the regulation of ESU transmission. In a rate-based PCM, synchronization messages are transmitted by the local host at a lower rate than the entity states are simulated. On the remote host, extrapolation methods are used to make entity state speculation from previous ESUs when the newest update is not available. ESUs are transmitted constantly in rate-based PCMs, even if the predicted entity state model is sufficiently close to the associated local state. In threshold-based PCMs, on the other hand, the local host employs the same extrapolation method as the remote host to model the entity states, and ESUs are only generated when necessary, that is when the divergence of the local state model from the true entity state violates a limit specified by a pre-determined threshold metric. The most widely-used threshold based PCMs is known as Dead Reckoning (DR) [Lin and Schab 1994; Miller and Thorpe 1995; IEEE 1998; Pantel and Wolf 2002b]. Due to its popularity, except in Chapter 5, threshold-based PCMs are assumed when PCMs are referred to in the discussions throughout this thesis unless otherwise stated. Issues to be discussed in the following sections and chapters can be easily modified to be applied to rate-based PCMs.

2.4.1 Dead Reckoning (DR)

Dead reckoning, as the name suggests, originated from the ancient navigation technique that estimates one's location based on a starting point and velocity. It was introduced as a motion prediction mechanism in SIMNET [Calvin *et al.* 1993; Miller and Thorpe 1995] and formally defined as a data transmission reduction technique in IEEE DIS standards [IEEE 1998]. DR operates on the premise that there is no need to transmit an update as long as the true and extrapolated entity states are sufficiently close and the difference between them falls in a tolerable limit.

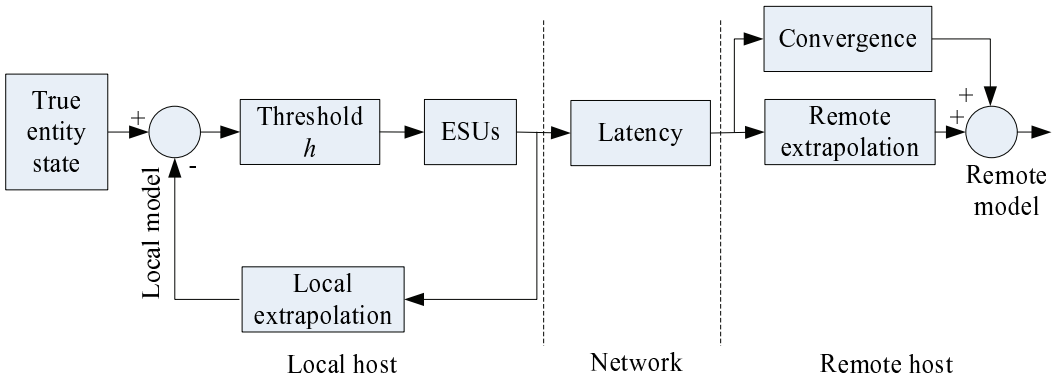


Figure 2.4 The framework of DR as a representative of general PCMs.

The framework of DR is shown in Figure 2.4. In dead reckoning, a local controlling host runs two parallel models for the entity under its control — a high-fidelity motion that represents the true entity state and a low-fidelity local model estimated from contextual dynamics by some extrapolation method. The low-fidelity model represents the remotely approximated entity state and is constantly compared to the true entity motion. An ESU (typically containing the current derivative information of the true motion such as position and velocity) is generated by the local host and transmitted to the remote host when the difference between the high-fidelity and low-fidelity models violates a given threshold. The same extrapolation method is also applied at every remote host to produce a remote model of the entity state from the most recently received packet. Using the same extrapolation model and ESUs as the remote host, the local model is produced for the local host to monitor the inconsistency arising from the reduced number of synchronization messages. It is however worth noting that due to network latency, the remote model keeps diverging during the period of ESU transmission while the local model has been corrected. Therefore, the difference between the true entity state and the remote model (namely remote inconsistency) will be different from local inconsistency between the high-fidelity motion and the local model.

The operation of PCMs is commonly divided into two main components: prediction and convergence [Singhal and Zyda 1999], which will be further discussed in detail in Section 2.4.2 and 2.4.3, respectively.

- **Prediction:** The prediction/extrapolation algorithms estimate future entity states from state derivatives included in the latest available ESUs. Prediction algorithms define how future entity state can be obtained from the current information so that redundant data transmission is saved. The current majority of DR mechanisms use polynomial extrapolation equations to predict entity state from state derivatives [Lin and Schab 1994; Cai *et al.* 1999; Lee *et al.* 2000].

- **Convergence:** Convergence algorithms define 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 [Lin and Schab 1994; Singhal and Zyda 1999; McCoy *et al.* 2007]. It should be noted that the convergence is the post-operation taken for the sole purpose of better perceptual experience for the human-users [Singhal and Zyda 1999]. It does not affect update packet generation or bandwidth consumption.

In addition to ESUs relating to local threshold violation, most systems send out an update if there is no ESU sent within a timeout period to maintain consistency in the face of update packet lost and to detect removal or disconnection of an entity from the environment. Any entity that has not sent an update for a number of timeout periods is considered deleted [Srinivasan 1996]. Such updates are also called *heartbeat* or *keep alive* packets. These updates are relatively rare compared to the regular ESUs and are not related to the shared dynamic state, thus they are not considered in the following sections and chapters of this thesis.

2.4.2 Extrapolation Equations

The polynomial equations used in standard DR can be generally described in the form of a truncated Taylor expansion of a time-dependent, real-value continuous function $f(t)$ [Lin and Schab 1994]. In the following discussion, time series $x(k) =$

Table 2.2 The first three orders of DR extrapolation equations.

Extrapolation order	Extrapolation equation	Truncation error
Zero$^{\text{th}}$-order	$x_\tau = x_0$	$\mathcal{O}(\tau\delta)$
First-order	$x_\tau = x_0 + \dot{x}_0 \cdot \tau\delta$	$\mathcal{O}((\tau\delta)^2)$
Second-order	$x_\tau = x_0 + \dot{x}_0 \cdot \tau\delta + 0.5\ddot{x} \cdot (\tau\delta)^2$	$\mathcal{O}((\tau\delta)^3)$

where: x_τ is the dead reckoned value of x at time $t = t_0 + \tau\delta$,

x_0 is the current value of x at time t_0,

$\dot{x}_0$ is the first-order time derivative of $f(t)$ at time t_0,

$\ddot{x}_0$ is the second-order time derivative of $f(t)$ at time t_0.

$f(t_k), t_k = k\delta, k = 1, 2, \ldots$ is a general representation of any state variable suitable for extrapolation (such as positional coordinates) in a DIA, with a constant simulation time-step δ. This assumption is considered reasonable for most DIAs where the shared dynamic states simulate the real-world object movement. Given state derivative information of f at step t_0, the n^{th}-order extrapolated state at a prediction span of τ steps from t_0 in the future is predicted as in (2.2) by expanding f around the point $t_0 + \tau\delta$:

$$f(t_0 + \tau\delta) = \sum_{i=0}^{n} \frac{f^{(i)}(t_0)}{i!}(\tau\delta)^i, \tag{2.2}$$

where $f^{(i)}(0)$ is the i^{th} derivative of $f(t)$ at time t_0. The truncation error is estimated as $\mathcal{O}((\tau\delta)^{n+1})$, which means the prediction error is no more than a constant multiplied by $(\tau\delta)^{n+1}$. The first three orders of dead reckoning extrapolation equations are elaborated on in Table 2.2.

The $zero^{th}$-order equation represents the situation where no extrapolation is used and the remote model just remains at the last updated value until the next update comes. The first-order equation represents the case of a linear extrapolation as the extrapolated model falls in a linear path starting from the last updated value with the constant initial velocity. Similarly, the second-order extrapolation assumes a

constant acceleration and is also referred to as quadratic extrapolation.

Dead reckoning relies on different orders of entity state derivatives to extrapolate future states. In DIA practice, some applications can read the current derivative information. For example, the second-order derivative can be accessed by Newton's law in those applications that translate the user input to the force applied to the entity. There are also alternative polynomial equations that estimate state derivative information using a short history of updates (Position History-Based Dead Reckoning [Singhal 1996]), i.e. the updated values $[x_0, x_{-1}, x_{-2}, \ldots]$, $[\dot{x}_0, \dot{x}_{-1}, \dot{x}_{-2}, \ldots]$, and etc. in the ESUs arrived at time $[u_0, u_{-1}, u_{-2}, \ldots]$, in the order from the most recent (current) update to the older ones. The update intervals are denoted as $[\Delta u_{-1} = u_0 - u_{-1}, \Delta u_{-2} = u_{-1} - u_{-2}, \ldots]$.

The most popular estimation of the first-order derivative of x at the point t_0 requires two successive values of x, namely x_0 in the most recent (current) arrived update and x_{-1} in the last update arrived at time u_{-1}. The estimation is the average velocity over the update interval Δu_{-1} as shown in (2.3) [Lin and Schab 1995; Singhal 1996; Hanawa *et al.* 2005; Hanawa and Yonekura 2005, 2006]:

$$\dot{x} \approx \frac{x_0 - x_{-1}}{\Delta u_{-1}}. \tag{2.3}$$

Like in the first-order case, the second-order derivative information can also be estimated in a series of approximations using parabolic interpolation [Lin and Schab 1995; Singhal 1996] or a Lagrange polynomial [Pantel and Wolf 2002b; Hanawa *et al.* 2005; Hanawa and Yonekura 2005, 2006]. Alternative second-order extrapolations requires up to three most recently arrived updates. Some approximation equations of the first and second-order derivatives of x at time t_0 for the second-order extrapolation (Table 2.2) are given as follows:

$$\ddot{x_0} \approx \frac{\dot{x}_0 - \dot{x}_{-1}}{\Delta u_{-1}}; \tag{2.4}$$

$$\ddot{x}_0 \approx 2\left(\frac{x_{-1} - x_0}{\Delta u_{-1}^2} + \frac{\dot{x}_0}{\Delta u{-1}}\right); \tag{2.5}$$

$$\ddot{x}_0 \approx x_0 \frac{2}{\Delta u_{-1}(\Delta u_{-1} + \Delta u_{-2})} - x_{-1} \frac{2}{\Delta u_{-1} u_{-2}} + x_{-2} \frac{2}{\Delta u_{-2}(\Delta u_{-1} + \Delta u_{-2})}, \tag{2.6a}$$

$$\dot{x}_0 \approx x_0 \left(\frac{1}{\Delta u_{-1}} + \frac{1}{(\Delta u_{-1} + \Delta u_{-2})} \right) - x_{-1} \left(\frac{1}{\Delta u_{-1}} + \frac{1}{\Delta u_{-2}} \right) + x_{-2} \frac{\Delta u_{-1}}{\Delta u_{-2}(\Delta u_{-1} + \Delta u_{-2})}. \tag{2.6b}$$

Mathematically, adding higher-order derivatives to the extrapolation equation improves prediction accuracy for a very close future. 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 DIA simulation, estimation error related to numerical approximation of high-order derivatives makes them unreliable. The computation complexity of the approximation algorithms imposes higher implementation overhead. For those applications which have direct access to accurate instantaneous derivatives, extra network load is needed to transmit the additional derivative information to the remote host, leading to negative impact on consistency. Consequently, high-order predictors are seldom used, and first and second-order DR are the most commonly deployed PCMs in DIAs [Singhal and Zyda 1999]. For the same reasons, the second-order extrapolation does not always outperform the first-order extrapolation in DIA practice [Lin and Schab 1995; Hanawa and Yonekura 2005; McCoy 2007].

2.4.3 Convergence Equations

At the moment an ESU arrives at the remote host, the most straightforward correction of the modeled entity state is to put the received new state into display immediately at the time of receipt. By doing this, the inconsistency arising from the inaccurate remote extrapolation is instantly corrected. However, the sudden jump in the displayed entity may negatively impact the user perception provided by the DIA. Hence, the preferred method is to gradually correct the entity state along a modeled path over a period of time [Lin and Lin 1994]. Such mechanisms are called smoothing techniques. The instant correction, also the simplest convergence

method, is called the $zero^{th}$-order convergence or *snap* equation [Singhal and Zyda 1999].

Generally, Starting from a starting position x_s (usually the modeled state before the arrival of the update), the converging state moves along a path specified by the particular convergence equation and ends with a desired final position of smoothing x_f over a period of c simulation steps. Then the remote model continues with extrapolation, as discussed in the last section.

The first-order convergence defines a simple linear smoothing path that directs the converging state to a future extrapolated position along a straight line. The equation is given by (2.7) [Lin and Lin 1994]:

$$x_i = x_s + (x_f - x_s)\frac{i}{c}, i = 1, \ldots, c, \tag{2.7}$$

where x_i is the i^{th} converging position. The linear convergence is also employed in the IEEE DIS standards [IEEE 1998].

Lin and Lin [1994] defines a cubic-spline convergence equation that corrects the remote modeled state along a smooth curve to a future extrapolated point. The cubic-spline convergence is relatively complicated compared to the linear smoothing equation, but the trajectory preserves continuity in velocity, which is more perceptually desirable. The cubic-spline convergence equation is given by (2.8):

$$\begin{aligned} x_i \quad = \quad & [c\delta(\dot{x}_s + \dot{x}_f) + 2(x_s - x_f)]\frac{i^3}{c^3} \\ & + [-c\delta(2\dot{x}_s + \dot{x}_f) + 3(x_f - x_s)]\frac{i^2}{c^2} \\ & + i\delta\dot{x}_s + x_s, i = 1, \ldots, c, \end{aligned} \tag{2.8}$$

where $\dot{x}_s$ and $\dot{x}_f$ are the DR velocities in the previous and the newly arrived ESUs, respectively.

Figure 2.5 illustrates the operation of DR with the first-order extrapolation and linear convergence as an example of typical PCMs. When the local model error reaches the threshold h, a new update is generated and the local model is corrected to the true entity state immediately. Hence the local threshold puts an upper bound

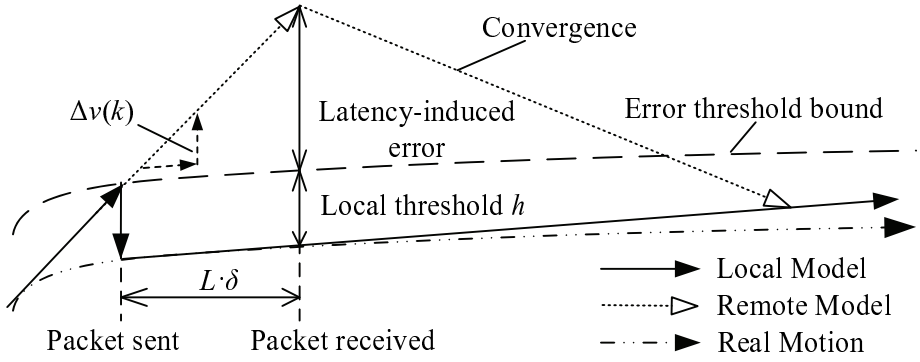


Figure 2.5 Visual illustration of DR as a typical PCM. In this example, the first-order extrapolation and linear convergence is employed.

on local model deviation. But the remote model keeps diverging from the real dynamic motion because transmitting an ESU from the local host to the remote host endures the inevitable network latency. The now out-of-date ESU received earlier is still in use on the remote site over the period of the network latency of L simulation steps. Therefore the remote model suffers from additional latency-induced error beyond the local error threshold. Consider the time of simulation step $k = 0$ when the local host generates a new update. By the time the ESU arrives at the remote host, the accumulation of the remote inconsistency R_e is described as [Zhang *et al.* 2009]:

$$R_e = h + \sum_{k=0}^{L-1} \Delta v(k)\delta = h + L\delta\overline{\Delta v}, \tag{2.9}$$

where $\Delta v(k)$ is the difference in velocity between the real dynamic and the remote extrapolation (i.e. the previously received reckoning velocity) 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 (2.9) identifies the three aspects that determines the remote inconsistency: the local threshold, network latency, and the inaccurate velocity. Among these factors, the local threshold is deterministic, the network latency is uncontrollable, and the difference in velocity depends on how the prediction model fits the entity dynamic. Therefore, the extrapolation algorithm is the core of the whole mechanism.

2.4.4 Extensions to Dead Reckoning

Under limited network conditions, PCMs such as dead reckoning rely on extrapolation techniques to maintain consistency with a reduced number of transmitted ESUs. There has been much previous work focusing on improving the standard DR technique in two general directions: managing the value of local threshold that controls ESU generation; and exploring advanced extrapolation techniques that give better accuracy in predicting future entity state.

The first category of techniques attempt to reduce the number of ESUs by adapting the local threshold based on user behavior. Cai *et al.* [1999], Lee *et al.* [2000], and Shim and Kim [2001] differentiate the local threshold values based on the distance between the objects. As the distance between two entities increases, the possibility of the two interacting becomes less and there is no need to send as many update messages between the hosts that control the entities. A fuzzy dead reckoning algorithm that takes all properties of entity (e.g. position, size and view angle etc.) into consideration when adjusting the level of threshold is proposed by Chen and Chen [2005]. This algorithm employs fuzzy correlation degree to measure the relationship between entities and determine the level of threshold under use. Behavior complexity of entity based on the frequency of rotation behavior is used to determine local threshold level in Yu and Choy [2001]. The motivation for this technique is that for the same threshold value, complex behavior tends to generate more update messages than smooth behavior, therefore the local threshold must be increased during erratic behavior periods to reduce network load of the DIA. The "Pre-reckoning" algorithm identifies the sudden changes in the user behavior, such as a rapid change of direction, when breaches of the threshold are likely to take place [Duncan and Gracanin 2003]. In such situations, update packets are transmitted before the threshold violation. Roberts *et al.* [2005], Marshall *et al.* [2006], and Roberts *et al.* [2008] examine the suitability of employing different consistency metrics as the local threshold. The traditional spatial threshold is suitable for user behaviors with a high *curvature* (meaning that the entity diverges from previous direction rapidly), while the *time-space* threshold works well in low curvature cases where the orientation of the entity

motion changes slowly. A hybrid threshold scheme that evaluates both metrics simultaneously as the simulation runs was also proposed. Under this scheme, an ESU is generated as soon as either of the metrics reaches the threshold. Kenny *et al.* [2006] uses linguistic user feedback to determine an appropriate threshold value based on psycho-perceptual measures of the user perception of the environment. The results tentatively suggest that a larger threshold for a fast paced application could elicit a similarly acceptable user experience as a smaller threshold for a slow paced scenario. As stated by the *Consistency Throughput Trade-off*, these techniques reduce network traffic at the cost of a less accurate remote model of the true entity state, or vice versa.

Adjusting local threshold value indirectly controls the frequency of ESU generation (and also the data transmission rate) of the DIA. A direct transmission rate control scheme for rate-based PCMs based on the distance between the objects has been proposed [Shim and Kim 2001], where update packets are sent at a higher rate for closer entities. Marshall *et al.* [2006] examine the issue of optimizing the update packet rate under constrained network bandwidth. As shown in Figure 2.6, if the network is heavily loaded, a reduced data rate could reduce inconsistency caused by the latency for transmitting the update packets through the network that connects the hosts. However, if the network is not heavily loaded, the insufficient update information is the main contributing factor of inconsistency. Reducing the amount of data transmission now contributes little in reducing latency and simply results in worse inconsistency due to the lack of updates sent to the remote host [Marshall *et al.* 2006]. The *Consistency Optimization (CO)* algorithm adjusts the update packet rate for a rate-based PCM so that the data rate matches the available network bandwidth estimated from latency trends [Marshall *et al.* 2008].

The second category of extension techniques attempt to explore better extrapolation models to improve dead reckoning based on the premise that an accurate entity state model can reduce the number of updates needed to be transmitted to the remote host, and can also improve consistency between the arrivals of the ESUs. Generally, these techniques employ *Predictive Statistical User Modeling* to build a user behavior model from collected data and then infer user actions, behaviors, goals, intentions,

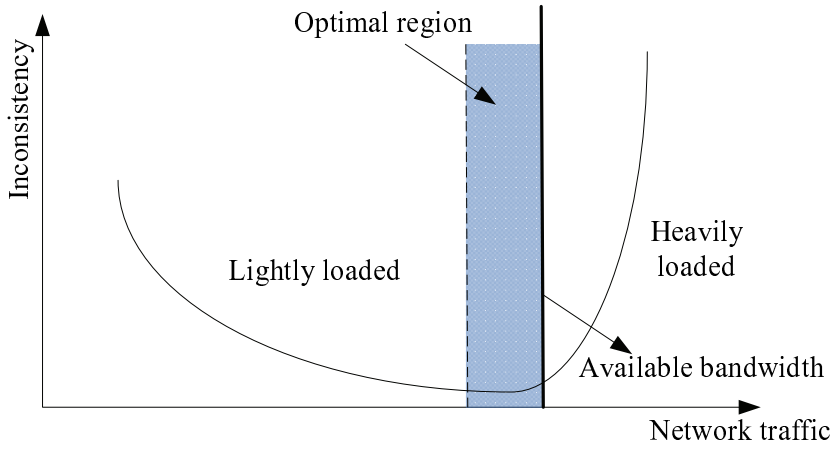


Figure 2.6 Inconsistency in different network conditions. The optimal region for data transmission rate is near the available network bandwidth.

and/or preferences in both real and virtual environments [Zukerman and Albrecht 2001]. The two main approaches adopted to build predictive statistical models are *content-based* and *collaborative*. Content-based models assume repeatability of the user behavior under similar circumstances and use the past behavior (the collected data) of an individual user as a reliable indicator of its future actions. Collaborative approaches estimate user behavior model from the historical behavior of a group of similar users based on the assumption that a user behaves similarly to other users sharing common attributes such as the same goal or preferences. The collaborative approaches are suitable for modeling a user identified as a member of a known group encounters a new scenario, where information about the user for a reliable individual model is not available. The primary forms of predictive statistical models are: linear models, neural networks, Kalman Filter, Artificial Potential Field, Term Frequency Inverse Document Frequency (TFIDF) models, Markov models, Bayesian networks, classification and rule-induction methods [Zukerman and Albrecht 2001; Delaney and Ward 2004; McCoy *et al.* 2006, 2007; Shi *et al.* 2009]. The *Neuro-reckoning* technique [McCoy *et al.* 2006, 2007] that uses Artificial Neural-Networks to build user behavior models to further reduce entity state updates in DIAs will be further reviewed in Section 2.4.5.

Prediction models can also be dynamically adapted to improve the performance of

extrapolation under different circumstances. In Position History-Based Dead Reckoning, a period of rapid changes in the moving direction of the entity invokes the first-order predictor, while the second-order equation is used when the entity is moving towards stable orientations [Singhal 1996]. Lee *et al.* [2000] adaptively select the extrapolation formula according to the current type of motion identified as one of the three classes: smooth (circular), bounce, and jolt. Delaney *et al.* [2003] reveals that within DIAs such as racing games, the participants interact with the environment with fixed goals and follow specific trajectories to achieve their goals. These trajectories are identified, stored at each host, and then used for extrapolation if the entity is reckoned to be following one of them. The employment of the pre-determined trajectories significantly reduce the number of ESU transmissions and prediction quality. This Hybrid Strategy Model (HSM) mechanism switches to standard DR for short-term deviation if the entity diverges from the long-term trajectory. The Dynamic Hybrid Strategy Model (DHSM) extends HSM by recalculating extrapolation models at run-time as users pursue their dynamic goals [McCoy *et al.* 2005, 2006].

2.4.5 Neural-Reckoning (NR)

Neuro-reckoning extends the first-order predictor in standard DR in two respects [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 in the future. Secondly, NR employs Artificial Neural-Networks to learn behavioral patterns in the entity motion from contextual dynamics to estimate the future velocity changes. The basic philosophy is illustrated in Figure 2.7. At the time of an error threshold violation, the local host employing standard first-order DR would transmit the instantaneous position and velocity information for the remote host to extrapolate the remote entity state model using the first-order equation. Under the NR mechanism, however, the local host 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

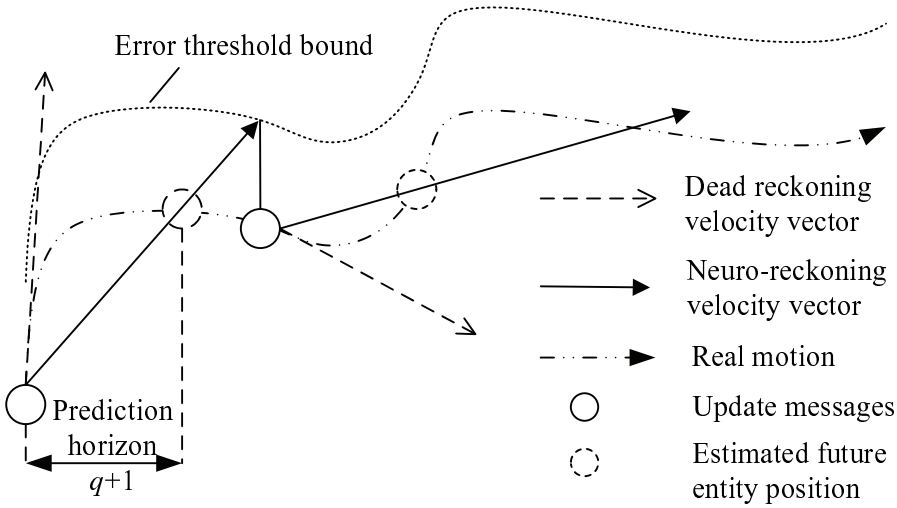


Figure 2.7 Visual illustration of Neuro-reckoning.

the current velocity in first-order DR (DR velocity) with an "NR velocity" that extrapolates from the current entity state through the estimated future position. By distributing the NR velocity as a predictive derivative that implicitly embraces expected changes in the entity motion, NR directs the local and remote model towards more likely, and thus more stable, directions. The ultimate goal is to further reduce the number of ESUs exchanged among the host machines due to the violation of the local error threshold. Consequently, the modeled motion is less dynamic or detailed since NR overlooks transient changes in the entity motion and only manifests an averaged overall effect of the object movement.

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 2.7) 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 implemented using a collection of static Multi-Layer Perceptrons (MLPs), which have been shown to have good mapping capabilities for modeling entity behaviors in a number of related application domains, including autonomous robot motion [Behnke *et al.* 2004], Distributed Interactive Simulation [Henninger *et al.* 2000, 2001], MMOGs [Thurau *et al.* 2003, 2004], Networked Virtual Environments [Garrett *et al.* 2002; Aguilar *et al.* 2003; Sas *et al.* 2003], and Adaptive Hyper-media applications [Frías-Martínez *et al.* 2005]. As

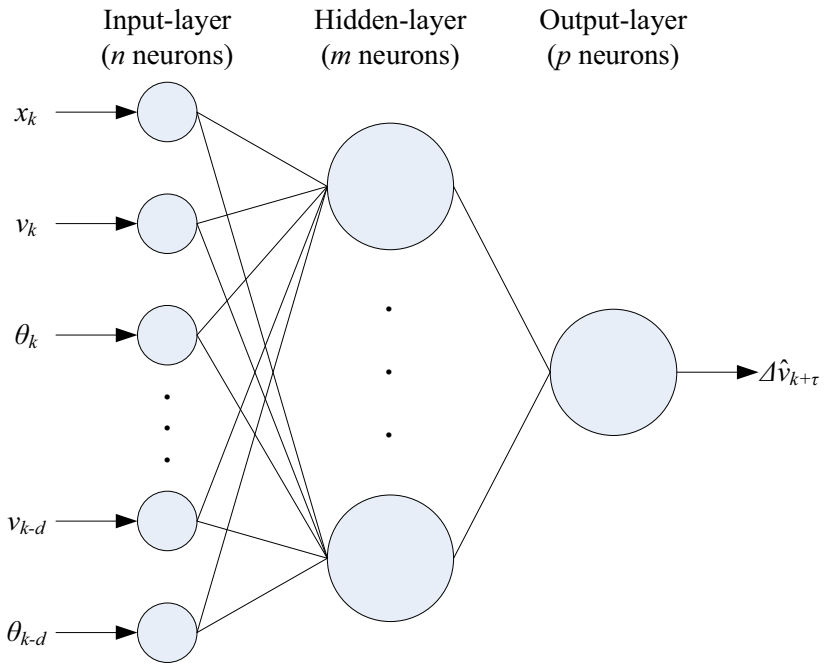


Figure 2.8 Diagram illustration of a neural-network predictor. A total of q such neural-networks (one for each prediction span τ) are required to predict successive changes in the entity's velocity [McCoy *et al.* 2007]

shown in Figure 2.8, each MLP contains a single input-layer with m neurons, a single hidden-layer with n neurons, and a single output-layer with d neurons, arranged in a fully connected m-n-p feed-forward architecture. Each MLP is trained using the standard Levenberg-Marquardt Back-Propagation (LM-BP) algorithm based on the mean squared error (MSE). Equation (2.10) – (2.13) shows the calculation of the NR velocity. In NR, each neural-network predictor is responsible for approximating a functional mapping $g_\tau(\cdot)$ that relates the input entity state vectors $\tilde{\boldsymbol{s}}_k$ to the future change in velocity $\Delta\tilde{\boldsymbol{v}}_{k+\tau}$ within the maximum prediction span q. Based on the predicted changes in velocity, a series of estimated intermediate entity state vectors are iteratively calculated for increasing values of τ. The difference vector between the final predicted future state $\hat{\boldsymbol{x}}_{k+q+1}$ and the current entity state $\boldsymbol{x}_k$ is then normalized and scaled by the current speed $|\boldsymbol{x}_k|$ to give the NR velocity vector as shown in (2.13). When an update is triggered by the event of error threshold violation, instead of the current velocity, the NR velocity is calculated and transmitted over the network to the remote host as a part of the standard update packet.

$$\text{Input entity state vectors: } \tilde{\boldsymbol{s}}_k = [\boldsymbol{x}_k, \boldsymbol{v}_k, \ldots, \boldsymbol{v}_{k-d}, \boldsymbol{\theta}_k, \ldots, \boldsymbol{\theta}_{k-d}]\,, \tag{2.10}$$

$$\text{Neural-network predictors: } \Delta\hat{\boldsymbol{v}}_{k+\tau} = g_\tau(\tilde{\boldsymbol{s}}_k), \tau = 1, \ldots, q, \tag{2.11}$$

$$\text{State vector extrapolation: } \begin{cases} \hat{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_k\delta, & \\ \hat{\boldsymbol{v}}_{k+\tau} = \boldsymbol{v}_k + \Delta\hat{\boldsymbol{v}}_{k+\tau}, & \tau = 1, \ldots, q, \\ \hat{\boldsymbol{x}}_{k+\tau+1} = \hat{\boldsymbol{x}}_{k+\tau} + \Delta\hat{\boldsymbol{v}}_{k+\tau}\delta, & \tau = 1, \ldots, q, \end{cases} \tag{2.12}$$

$$\text{NR velocity: } \hat{\boldsymbol{v}}_k = \frac{\boldsymbol{x}_{k+q+1} - \boldsymbol{x}_k}{|\boldsymbol{x}_{k+q+1} - \boldsymbol{x}_k|} \cdot |\boldsymbol{v}_k|\,. \tag{2.13}$$

As complicated as it might seem, neuro-reckoning only involves replacing the velocity prediction procedure on the local controlling 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 standard first-order DR.

On a final remark, a neural network, like other statistical learning techniques, requires large amount of prior information from training datasets 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 DIAs.

2.5 DIAs as a Media Class

From the perspective of communication, Distributed Interactive Applications store and deliver information to communicate between humans, as well as between humans and machines, in a way that enables real-time interactions among dispersed

participants. Therefore DIAs can be viewed as a form of *media*, or more specifically *multimedia* because the communications in DIAs often incorporate multiple forms of information content and processing including plain and formatted text, graphics, images, sound, animations, video, and interactive virtual reality objects [Kinsner 2002].

DIAs and general forms of media have the common fundamental functionality of sharing dynamic states in different time and space. The term *state* means the entity state in DIAs as discussed in this chapter, and refers to the specific form of representation of the content in a particular media, such as sound wave signals in audio media and successive frames in a video. DIAs and media both reproduce the time-evolution of the original dynamic state (the true entity state in DIAs or the raw media) as precise and exact as possible. The key characteristic that differentiates DIAs from the traditional media is that DIAs attempt to provide real-time interaction to the users and, thus, impose strong constrains on the timeliness of the shared state. The user's perception of the virtual world and their performance in various interactive tasks are more subject to time differences in displaying the same entity state for different users. Traditional media, on the other hand, only require relative temporal relationship between the states. They can be stored or buffered before play out and are less sensitive to delay.

Indeed, the link between DIAs and media has not gone unnoticed within the literature. Comerford [1996] noted the emergence of *Interactive Media* that use the Internet as the new communication channel to carry traditional media communication such as phone, radio and television. Such interactive media include Internet chatting, radio and video conferencing, etc. Mauve *et al.* [2001] defined a media class called *Distributed Interactive Media* or DIM (Table 2.1) which allows a set of spatially separated users to interact synchronously with the medium itself. The DIM model consists of *Shared State* and *State Changes* caused by *passage of time* or by *events*. An interactive medium is often partitioned into several *sub-components*, which break down the complete state of the medium into more manageable parts and allow the participants of a session to track only the states of those sub-components in which they are actually interested. A substantial part of an interactive medium

usually remains constant over the course of a session and is called the *environment*. By its definition, DIM covers most application domains of DIAs. However, aside from two equivalent terms, the existing work does not explore the link between DIAs and media much further.

Besides similar functionalities, DIAs and media also share a common challenging issue. The amount of data used to share the dynamic state must be reduced or compressed to fit into the available communications channels, namely the underlying network with limited bandwidth. In DIAs, PCMs make use of user behavior models to predict expected future entity state and reduce the amount of data transmissions for maintaining consistency. By employing the local error threshold, inconsistency within human perception tolerance is discarded. Following a similar philosophy, video compression models the raw media through statistical or dictionary models and transforms it to eliminate redundant and irrelevant information to reduce the size of compressed video. In the next two sections, general procedures of video compression techniques (the focus is video or motion picture compression for the ease of illustration and comparison) is briefly reviewed and compared to the PCM operation, showing that PCMs can been seen as a form of video compression that reduces the amount of data required to represent the dynamic virtual world at the cost of a controlled level of inconsistency.

2.5.1 Video Compression

Video compression can be generally separated into lossless and lossy approaches. Lossless compressions reduce the size of the video data without any information loss so that the video can be reconstructed exactly the same as the original. Lossless compressions remove redundant information in the raw video. For example, in a video clip of a moving car, the constant background (trees and the lane) is redundant and unnecessary to be restored in all frames of the clip. The background can be "borrowed" from a previous frame and combined with the new position of the car to recover the exact current frame. Lossy compressions remove irrelevant information in the raw video for a further reduced data size, meaning that the reconstructed

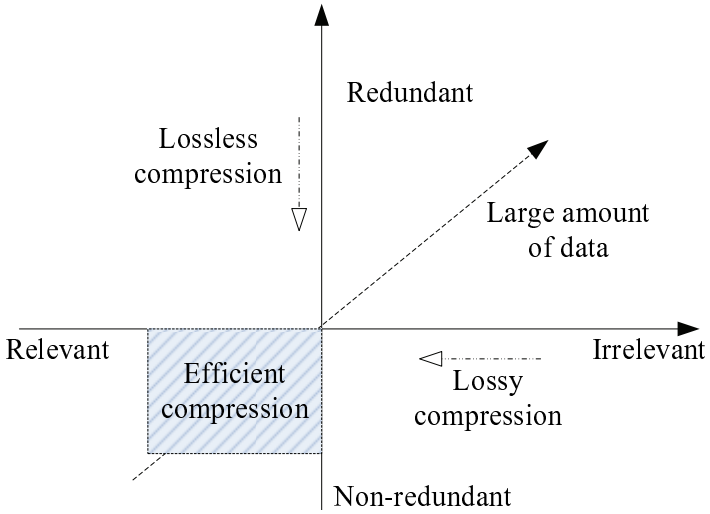


Figure 2.9 Redundancy reduction and irrelevancy removal [Kinsner 2002].

video would be inaccurate to some negligible extent. In the moving car video example, the movement of the car is the critical content of the video and has to be accurately delivered, while the subtle movements of the leaves on the trees are relatively irrelevant and thus can be ignored without causing much noticeable difference. Figure 2.9 illustrates the two compression approaches. General video compression techniques employ both lossless and lossy approaches to achieve a high compression ratio.

In video compression, the aim of the compression is to reduce the amount of data (in digital bits) needed to store a video, or the *bit-rate* used to play the video for a unit time. The raw video consists of a series of frames representing the static images of the video at particular time instances. Each frame contains pixels of data which are arranged in a spatial order to form the whole image. A coder compresses the raw video by storing the original frames in an encoded form that removes redundant and irrelevant information. The redundant information is re-inserted to the frames by a decoder to reconstruct the video. But the irrelevant information loss is irreversible. A coder and decoder pair are referred to as a "codec". In video signals, bit-rate reduction generally comes from *spatial and temporal redundancy* and *psychovisual irrelevancy* [Tudor 1995]. Spatial and temporal redundancy is based on the fact that pixel values are dependent and correlated with their neighbors both within the same

frame (spatial) and across frames (temporal). To some extent, the value of a pixel is predictable from the values of neighboring pixels. Psychovisual irrelevancy comes from the limited response of the human eye to fine spatial detail. Controlled impairments introduced to detail near object edges or around "shot-changes" (changes in the scene content of a video) are unlikely to be visible to a human observer.

In this section we take MPEG (Moving Pictures Expert Group) codec as a representative example of video compression. MPEG has developed several versions of video compression standards since 1988 (MPEG-1, MPEG-2, MPEG-4, etc.), which share similar codec architecture since MPEG-2. Two key techniques employed in an MPEG codec are intra-frame Discrete Cosine Transform (DCT) coding and Inter-frame motion compensation prediction [Tudor 1995; Koenen 2002].

A two-dimensional DCT transforms the pixel matrix representing an image into DCT coefficients that indicates the contributions of particular orthogonal spatial frequencies and thus removes spatial redundancy, in a similar way as a Fourier Transformation removes redundancy in a temporal signal. In an MPEG codec, DCT is performed on small 8×8 or 16×16 pixel blocks of the image to produce blocks of DCT coefficients. For typical blocks from natural scenes, pixel values change smoothly among neighboring area. So the DCT coefficients would consist of many zeroes or near-zeroes for higher frequencies. A Variable-Length Coding (VLC) is used to further reduce the code length. Both DCT and VLC are lossless.

Motion compensation exploits temporal redundancy by attempting to predict the frame to be coded from a previous "reference" frame, because within a tiny time step, it is very likely that only parts of the whole scene get moved and each moving part changes its position as a whole object, instead of individual pixels. So all the pixels in that part share the same motion vector. Again, in the moving car example, if a complete frame is transmitted to the decoder as a reference frame, then all the information needed for the next frame is a motion vector indicating the distance and direction that car moves over the time interval between the two frames. The simplest and basic method of detecting such a motion is a "block-matching" search in which blocks in the frame to be coded and the reference frame are compared and the

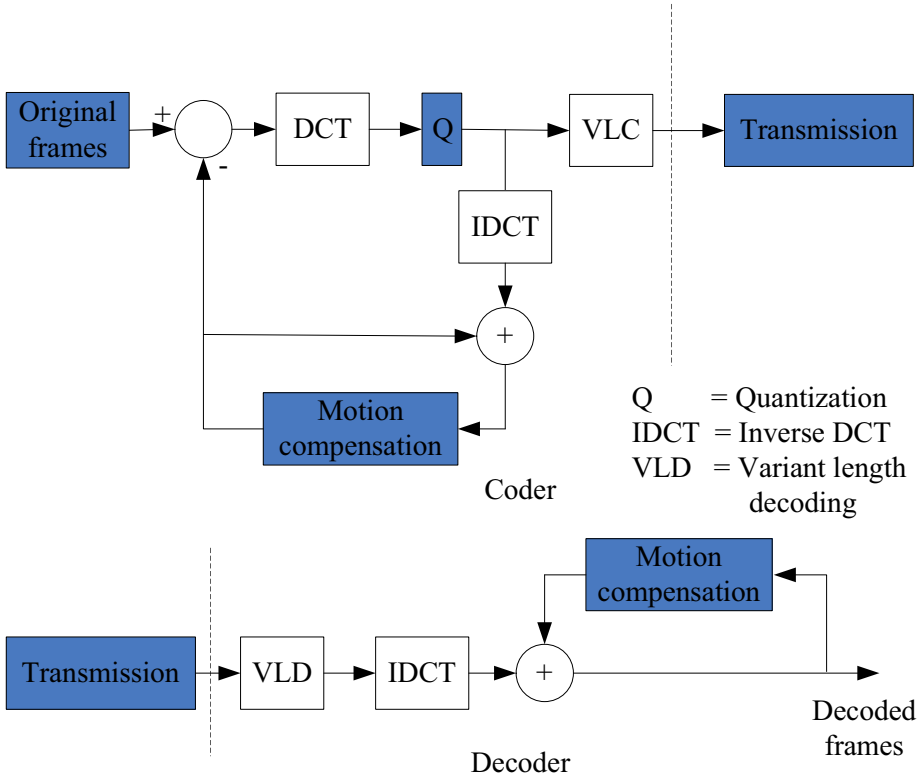


Figure 2.10 Basic block diagram of MPEG video codec [Tudor 1995; Koenen 2002].

"best" offset is selected on the basis of the minimum error. Motion Compensation is also lossless.

Figure 2.10 shows the codec structure (without further details) of an MPEG system. The coder subtracts the motion-compensated frame from the original to form a "difference" image, which is transformed with DCT. The coefficients are quantized to discard spatial irrelevancy. These quantized values are coded using VLC and transmitted to the decoder combined with motion vectors. In the decoder, the quantized DCT coefficients are decoded and inverse transformed to produce the difference image. This is added to the motion-compensated prediction generated from previously decoded frames to produce the reconstructed output frame. The information loss in this process comes mainly from the quantization.

The motion compensation procedure makes some of the encoded frames dependent on the reference frame and impossible to be decoded independently. In addition,

the motion vector of the block to be coded can also be "backward" detected from a future frame. In MPEG-2, three types of frames are defined. *Intra* frames (I-frames) are coded without reference to other frames. Only moderate compression is achieved by reducing spatial redundancy, but not temporal redundancy. I-frames are used periodically as access points where decoding can begin. *Predictive* frames (P-frames) can use the previous I or P-frame for motion compensation and may be used as a reference for frames afterwards. P-frames offer better compression compared to I-frames by reducing both spatial and temporal redundancy. *Bidirectionally-predictive* frames (B-frames) can use the previous and next I or P-pictures for motion compensation, and offer the highest degree of compression. The backward motion compensation requires the coder to reorder the natural display order of the frames so that the B-frames are transmitted after the frames they reference, which introduces an extra reordering delay. Such delay is acceptable for traditional video.

2.5.2 PCMs as a Video Compression

Compare Figure 2.4 and Figure 2.10, and it is obvious that PCMs in DIAs and video compression follow the same basic framework and working philosophy. All highlighted components in Figure 2.10 have their counterparts in PCMs:

- **The original frames and true entity state** represent the content that is intended to be shared for the users. In video compression, each original frame can be seen as a set of variables of the pixel values that represent the instantaneous state of the video content at a specific time instance, just like the time-variant state variable in DIAs.
- **Motion compensation and extrapolation model** represent the method intended to remove temporal redundancy between states/frames at different times so that the real content can be transmitted with lower network traffic. The temporal dependence is embodied by the motion compensated vectors in video compression or state derivatives in PCMs. Predictive statistical modeling techniques can be used to facilitate extracting such dependence [Koenen

2002]. In theory, motion compensation and extrapolation are both lossless and the removed redundancy can be re-inserted to accurately rebuild the original state/frame.

- **Quantization and local threshold** represent the lossy compression operation that removes irrelevant information for a further reduced bit-rate. The difference between the predicted state/frame and the true state is rounded up so that negligible details are discarded. Such removal is irreversible and causes information loss.

Note that DCT and VLC, aiming to reduce the spatial redundancy within a frame, are typically unnecessary for the relatively simple state variables in DIAs. It is then apparent that PCMs can be seen as a form of video compression, which, unlike traditional compression, imposes stronger constraints on timeliness to provide interactivity. That is also why backward prediction is not used in PCMs.

From the perspective of video compression, the *Consistency Throughput Trade-off* that PCMs try to deal with can be seen as the trade-off between the reconstructed quality and bit-rate, or *Rate-Distortion Trade-off* [Cover and Thomas 2006]. This is also the core philosophy in the work presented throughout this thesis. To further investigate the trade-off, a quantified metric of the redundancy and irrelevancy identified and removed by the extrapolation and local threshold is needed. This metric must be generic and does not imply any particular form of extrapolation scheme. However, no consistency metric discussed in this chapter has given such measurement of the redundancy (as such that future states can be extrapolated from the current state) and perceptual irrelevancy (as such that inconsistency below the local threshold is assumed insignificant to human perception and is ignored).

2.6 Concluding Remarks

In the first part of this chapter, concepts and issues relating to DIAs and consistency are reviewed as they apply towards the work presented throughout this thesis. Common terms and a taxonomy of DIAs are defined. PCMs are extensively discussed in the context of consistency maintenance within a distributed, real-time system. Dead reckoning and its extensions as outlined in detail within this part.

In the second part of this chapter, general video compression approaches are briefly reviewed in the context of MPEG video compression. The link between PCMs and video compression is demonstrated by pointing out that they work in the same basic framework and philosophy and share similar components. PCMs can then be seen as a form of video compression. Although entity state variables in DIAs are simpler than the frame data in video, the real-time interaction provided by DIAs imposes very strict constraints on time and delay. The *Consistency Throughput Trade-off* in DIAs can be seen as the trade-off between the reconstructed quality and bit-rate. This analogy provides the motivation for and underlines the work presented throughout this thesis, and forms the basis for the proposal of a novel model of PCMs in the later chapters of this thesis.

In the next chapter, concepts and techniques in information theory is briefly introduced and reviewed, based on which an information model of PCMs is proposed. This information model uses statistical predictability to measure the temporal redundancy captured by the extrapolation models. The irrelevancy discarded by the local threshold is measured by information loss. Such an information metric is general and applies to any form of predictive extrapolation.

Chapter 3

An Information Model For Predictive Contract Mechanisms

3.1 Introduction

In the previous chapter, consistency maintenance issues are reviewed surrounding the two fundamental trade-offs in Distributed Interactive Applications, namely the *Consistency Throughput Trade-off* and *Consistency Responsiveness Trade-off*. To deal with these trade-offs, Predictive Contract Mechanisms, including the well-known dead reckoning defined in the IEEE DIS Standards [IEEE 1998], attempt to maximize local responsiveness and reduce network throughput by allowing for a controlled inconsistency. Network traffic is reduced through non-periodic Entity State Update transmission. The remote consistency is, to some extent, maintained by using extrapolation formulas to remotely estimate the entity dynamics in the short-terms. Inconsistency arising from the combination of a reduced number of ESUs and an inaccurate prediction is (indirectly) controlled by the local error threshold.

The re-interpretation of PCMs as a form of video compression demonstrates that it is the temporal dependence in the evolution of entity behavior that enables the employment of predictive extrapolation methods to speculate entity dynamics over

short time scales. Prediction accuracy depends on how much temporal dependence is utilized by the specific extrapolation model under use. Standard DR mechanisms, for example, sacrifice remote predictive accuracy for low computational complexity, low network traffic overhead and ease-of-implementation by employing simple polynomial prediction equations that ignore further useful dependence in contextual dynamics aside from instantaneous state derivatives [Singhal and Zyda 1999]. On the other hand, the extensions to standard DR develop advanced prediction models that take (application-dependent) patterns in user behaviors into consideration and improve prediction accuracy at higher costs, which could come from higher computational complexity, higher network traffic loads, or loss of generality.

Following the philosophy of works reviewed in the last chapter, one can always introduce some new predictive scheme that has not been used in extrapolating entity state and investigate the effectiveness of this scheme in reducing ESU transmission. It is then natural to ask the question of *how* to design an optimal extrapolation method that uses as much contextual information as possible and provides the highest prediction accuracy so as to reduce the amount of synchronization messages to the minimum. Unfortunately, such a question cannot be answered using current approaches, not only because there may not be a universal optimal extrapolation mechanism that is effective for all kinds of user behavior, but also because there has not been any existing measure of *what* contextual information is there to be utilized for extrapolation in the first place. In other words, there is a lack of a general metric of the amount of the temporal dependence in entity dynamics. In a continuous motion the future state is related to the current state and such redundancy (or predictability) enables extrapolating future state given the current entity state.

In this chapter, we propose an information model that measures the temporal dependence as predictability of the entity state to be extrapolated from the contextual dynamics. The predictability is measured as a reduction in uncertainty about the entity state to be predicted and is related to the data size (number of bits) required for transmission. The information model takes a novel perspective towards PCMs and provides a quantified analysis to the *Consistency Throughput Trade-off*.

In the first part of this chapter, concepts such as Entropy and Mutual Information and the techniques used to estimate them from practical data are reviewed, providing a necessary foundation for the subsequent proposal of the information model.

In the second part of this chapter, the information model for PCMs is formalized. The operation of PCMs is modeled as information generation, compression, transmission and reconstruction. Inconsistency arising from the local threshold, latency and inaccurate extrapolation is measured as information loss. In this way, the information model interprets PCMs as a lossy video compression.

In this respect, the emphasis throughout this chapter is placed on defining and specifying the information model of PCMs, rather than deriving or proposing specific predictive behavioral models.

3.2 Information Theory Basis

3.2.1 Entropy and Mutual Information

In this section we introduce the basic definitions required for subsequent development of the ideas described later in this thesis. All the definitions and methods discussed here are given in discrete terms, as state variables of all entities in a virtual environment are finite and discrete.

In information theory, the predictability of one variable provided by another is measured by the reduction of uncertainty by knowing the latter, instead of the absolute difference between the values of the two. We first introduce the concept of *entropy*, which is a measure of the uncertainty of a random variable. 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]:

$$H(X) = -\sum_{i=1}^{m} p(x_i) \cdot \log p(x_i). \tag{3.1}$$

If the base of the logarithm is 2, the entropy is expressed in *bits*. If the log is to the natural logarithm base e, the entropy is measured in *nats*. Throughout this thesis, all entropies will be measured in bits. An equivalent definition interprets the entropy of X as the expected value of the random variable $\log \frac{1}{p(x)}$ as shown in (3.2) [Cover and Thomas 2006]:

$$H(X) = E\left[\frac{1}{\log p(x)}\right], \tag{3.2}$$

where $E[\cdot]$ denotes expectation.

Entropy measures the degree of complexity and unpredictability of 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 (such as a fair dice roll) 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, a variable with more possible states or the probability is more evenly distributed has larger entropy and is more complex and less predictable.

The entropy of a variable is also a measure of the amount of information (or data) required on average to describe the variable. Imagine determining the result of a fair dice roll, which can be described as $p(x_i) = \frac{1}{6}, x_i = 1, 2, \ldots, 6$, by the minimum number of binary questions. An efficient first question "is it no larger than 3?" splits the probability in half. Then the question "is it 1?" following a positive answer to the first question may give the final result with $\frac{1}{6}$ chance. Otherwise the result can be determined by the answer to the third question "is it 2?". Similar questions can be asked if the answer to the first question is "no". Therefore we can have the result with no more than three questions. It is shown that the minimum average number of binary questions required to determine a random variable X lies between $H(X)$ and $H(X) + 1$ [Cover and Thomas 2006]. In the case of dice roll, it is somewhere between $\log 6$ and 3.

If the answers to a series of binary questions are appropriately encoded, the concept of entropy can answer the question "what is the average length of the shortest

description of the variable?" or "what is the shortest code length for a lossless compression of the variable?" [Cover and Thomas 2006]. This is especially important in network communications. Imagine transmitting a deterministic variable to a remote receiver at a fixed frequency. The only necessary transmission is a packet with the static state x^*, which is negligible over time. However, it takes a data unit of average size of $H(Y) = \log m$ bits each time to inform the receiver of the state of the totally random variable, if the data unit is properly encoded in binary [Cover and Thomas 2006]. Generally, more complex variables with higher entropy require larger amount of information or data to fully describe them.

The definition of the entropy of a single random variable can be extended to a pair of inter-dependent variables X and Y with possible states $\{x_1, x_2, \ldots, x_m\}$ and $\{y_1, y_2, \ldots, y_n\}$ respectively. Considering (X, Y) as a single vector-valued random variable, the *joint entropy* $H(X, Y)$ is defined as

$$H(X,Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i, y_j) \cdot \log p(x_i, y_j), \tag{3.3}$$

where $p(x_i, y_j)$ denotes the joint probability that X is in the state x_i and Y is in the state y_j.

The *conditional entropy* $H(X \mid Y)$ of the variable X given Y is defined as the expected value of the entropies of the conditional distributions, averaged over the conditioning variable Y:

$$H(X|Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i, y_j) \cdot \log p(x_i \mid y_j), \tag{3.4}$$

where $p(x_i \mid y_j)$ denotes the conditional probability that X is in the state x_i, given Y is in the state y_j. The *chain rule* in (3.5) shows the natural relationship among the three entropies defined above: the joint entropy of a pair of random variables (X, Y) is the entropy of X combined with the conditional entropy of Y given the conditioning variable X, and vice versa.

$$\begin{aligned} H(X,Y) &= H(X) + H(Y \mid X) \\ &= H(Y) + H(X \mid Y). \end{aligned} \tag{3.5}$$

For arbitrary variables, we always have

$$H(X) \geq H(X|Y), \tag{3.6}$$

with equality if and only if X and Y are independent. Equation (3.6) highlights that *conditioning reduces entropy* or *information cannot hurt* [Cover and Thomas 2006]. It states that knowing Y may also give some knowledge about X and thus reduces the overall uncertainty of the latter. Thus, *mutual information* $I(X;Y)$ is defined as the reduced uncertainty of X due to the knowledge about Y:

$$\begin{aligned} I(X;Y) &= \sum_{x_i,y_j} p(x_i,y_j) \cdot \log \frac{p(x_i,y_j)}{p(x_i) \cdot p(y_j)} \\ &= H(X) - H(X \mid Y). \end{aligned} \tag{3.7}$$

By symmetry, it also follows that:

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y \mid X) \\ &= I(Y;X). \end{aligned} \tag{3.8}$$

The mutual information in (3.7) is also called cross-mutual-information [Jeong *et al.* 2001], and the unconditional entropy of X is the auto-mutual-information between the variable X and itself:

$$I(X;X) = H(X). \tag{3.9}$$

Mutual information can also be defined through the concept of *relative entropy* [Cover and Thomas 2006]. Relative entropy is a measure of the difference between two distributions. Given two probability mass functions $p(x)$ and $q(x)$, the relative entropy is defined as:

$$D(p\|q) = \sum_{x_i} p(x_i) \cdot \log \frac{p(x_i)}{q(x_i)}. \tag{3.10}$$

with equality if and only if $p(x)$ and $q(x)$ are the same. Relative entropy is also called *discrimination information* or *Kullback-Leibler distance* [Cover and Thomas 2006]. Although the relative entropy does not satisfy symmetry and triangle constraints required by the ordinary definition of a "distance", it is very helpful to consider it as a distance, because relative entropy measures the knowledge obtained if the description

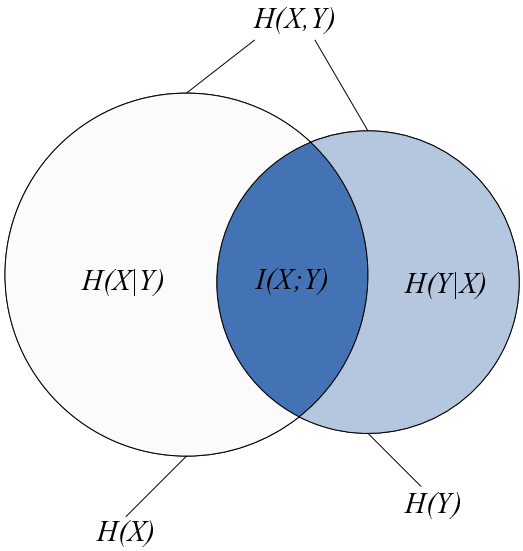


Figure 3.1 Venn diagram of mutual information and the entropies.

of the random variable X is changed from q to p. Typically p represents the true or verified distribution of data after observation or experiment, while q represents an approximated or assumed model of X. In this case, the relative entropy measures the information about X obtained from the observation or experiment.

From this perspective, the interdependence between the variables X and Y can be measured as the distance between the true joint probability $p(x, y)$ and the product probability $p(x)p(y)$ under the independence assumption. Therefore another definition of mutual information is given by (3.11):

$$I(X;Y) = D\left[p(x,y) \| \left(p(x)p(y)\right)\right] \tag{3.11}$$

$$= \sum_{x_i,y_j} p(x_i, y_j) \cdot \log \frac{p(x_i, y_j)}{p(x_i) \cdot p(y_j)}. \tag{3.12}$$

The equivalence of the two definitions is shown in (3.12).

The relationship between mutual information and the entropies can be illustrated by set operations in the Venn Diagram in Figure 3.1. Notice that mutual information represents the intersection of the two single entropies.

Mutual information measures the interdependence between the variables X and Y.

This information increases the predictability of X by making some of its states more probable than the case before Y is known.

3.2.2 Numerical Estimation of Information

3.2.2.1 Naive Algorithm

All the concepts introduced in the previous section involve knowledge about respective probability functions, which are normally not known in practice. To apply information theory to DIA in practice, the probability functions involved must be estimated from discrete data, which is a non-trivial process that requires significant research efforts [Steuer *et al.* 2001; Paninski 2003]. In this section, we start with a simple approach to estimate probability 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 time instances k. Let r_i be the number of cases that $x(k) = x_i$, and r_{ij} the number of occurrences that $x(k) = x_i$ and $y(k) = y_j$ at the same time. In the simple algorithm, the mass and joint probabilities are estimated as the frequencies of occurrences [Steuer *et al.* 2002]:

$$\hat{p}(x_i) = \frac{r_i}{N}, \qquad \hat{p}(x_i, y_j) = \frac{r_{ij}}{N}. \tag{3.13}$$

The straightforward probability estimation in (3.13) is directly derived from the plain understanding of discrete probability. Such an approach is easy to implement but relies on the availability of very large sample sizes to produce reasonably accurate results. In a small dataset, the randomness of the sampling process would manifest and lead to biased probability estimation. Figure 3.2 presents a simple example to illustrate this "Finite-Size Effect". In this example, the simple algorithm is used to estimate the probability of having "1" when tossing a dice repeatedly for N times. When $N = 30$ (a sample size considered to be small for the state space of 6), it is very unlikely that each of the 6 possible results are sampled exactly 5 times. Hence the estimated probability is diverged from the true value. Only in very large sample

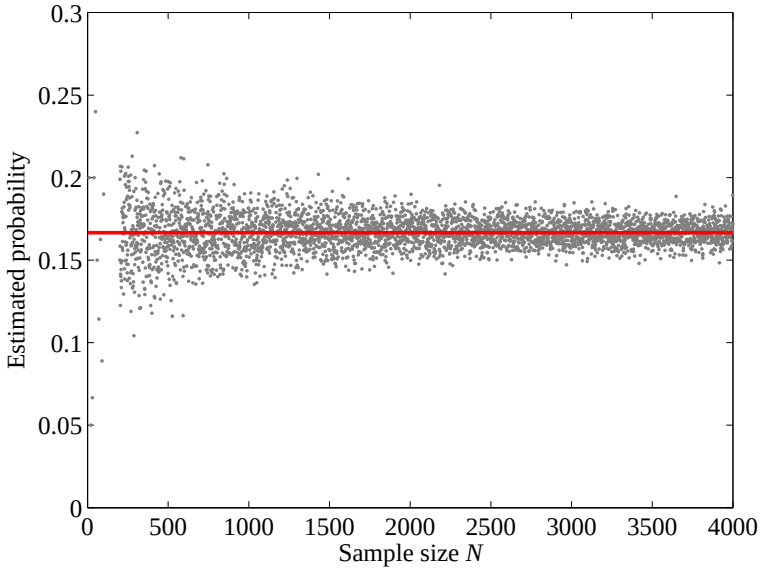


Figure 3.2 Estimating the probability of having "1" when tossing a dice, using the simple algorithm. The estimated probability in each sample size N is marked by a dot in the figure, and the true probability value $\frac{1}{6}$ is indicated by the solid line.

sizes does the estimated probability converge to the true value indicated by the solid line in Figure 3.2.

To address the Finite-Size Effect when using the simple probability estimations in information calculation by (3.7), an additional correction term has to be applied to the original definition of mutual information in (3.7) to eliminate the systematic bias [Steuer *et al.* 2002]:

$$I(X;Y) \approx \sum_{x_i,y_j} \hat{p}(x_i,y_j) \cdot \log \frac{\hat{p}(x_i,y_j)}{\hat{p}(x_i) \cdot \hat{p}(y_j)} - \frac{m_{xy} - m_x - m_y + 1}{2N}. \tag{3.14}$$

Here, m_x, m_y denote the number of unique state values in $x(k)$ and $y(k)$, and m_{xy} is the number of unique values of $[x(k), y(k)]$ combined as a vector variable. It is then apparent from the correction term that the sample size N must be considerably larger than the number of possible state combinations to constrain the Finite-Size Effect. For all possible state values, $r_{ij} \geq 10$ (r_{ij} defined as in (3.13)) is considered the rule of thumb for the simple algorithm to provide a good estimation [Roulston 1999]. Unfortunately, in general DIA practice, where the entity movement may pass through a large space, the requirement of the sample size could become so large rendering it impractical to use the simple algorithm to estimate probabilities and

information in experimental studies. In such cases, a better algorithm which can generate accurate estimation results in small datasets is necessary.

3.2.2.2 Kernel Density Estimation Algorithm

In order for improved information estimations in small datasets, an alternative (and improvement) to the simple algorithm based on Kernel Density Estimation (KDE) was suggested by Moon *et al.* [1995] and was found to be superior to the naive estimation in terms of a better convergence of the estimate to the underlying density [Moon *et al.* 1995; Steuer *et al.* 2002]. Instead of simply counting the occurrence of the variable whose probability is to be estimated, KDE employs a form of kernel function to calculate the contribution of a sampled data point to the probabilities of its neighborhood. 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 multivariate Gaussian kernel $K(\cdot)$ is given by Moon *et al.* [1995]:

$$\hat{p}(\boldsymbol{r}) = \frac{1}{N}\sum_{k=1}^{N} K\left(\frac{[\boldsymbol{r}-\boldsymbol{r}(k)]\boldsymbol{S}^{-1}[\boldsymbol{r}-\boldsymbol{r}(k)]^T}{h^2}\right), \tag{3.15}$$

$$K(w) = \frac{1}{(2\pi)^{\frac{d}{2}} h^d det(\boldsymbol{S})^{\frac{1}{2}}} \exp(\frac{-w}{2}), \tag{3.16}$$

$$h = \left[\frac{4}{d+2}\right]^{\frac{1}{d+4}} N^{\frac{-1}{d+4}}. \tag{3.17}$$

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 $\boldsymbol{r}(k)$. The estimated probability is the local weighted average of the contributions of the sampled data points in the neighborhood. The bandwidth h controls the range of the neighborhood and the optimal value for the Gaussian kernel in (3.16) is shown in (3.17).

Figure 3.3 gives a graphical illustration of KDE algorithm using a very simple one-dimensional example. In this example, two state values are sampled with occurrences proportional to the heights of the vertical bar at each of the values. The solid curves denotes the kernel functions for the two sample values. It can be seen that the

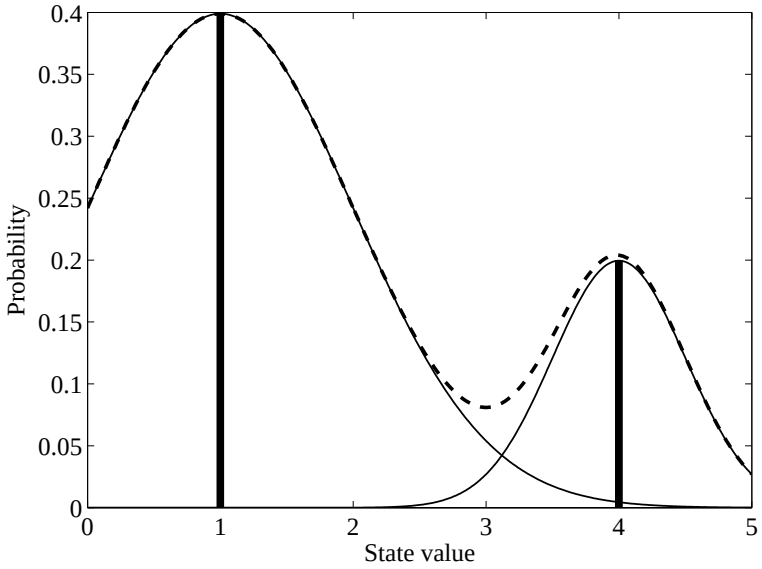


Figure 3.3 A graphical illustration of KDE algorithm. The estimated probability (dashed curve) is the sum of each kernel function (solid curve) at the point to be estimated. The height of the vertical bars are proportional to the frequencies of the observed data points.

Gaussian kernel places Gaussian "bumps" at the observed positions. The probability estimation (the dashed curve in Figure 3.3) is then given by the sum of the "bumps" at the state value to be estimated. Notice that the vertical bars in Figure 3.3 also indicate, in a relative sense, the frequencies of the sampled data. When estimating the probability at the state value of 3, the simple algorithm would produce a zero probability for lack of an observation at this position. In a small dataset, such an absence of observation could be the consequence of the insufficient sample size rather than the characteristic of the variable itself. The estimated impossibility would result in misleadingly reduced uncertainty of the variable and overestimated information values. However, in KDE algorithm every sampled data point (at position 1 and 4 in this example) contributes to the estimated probability located in its neighborhood and produce a non-zero probability at position 3, even if it is not observed in the sample. KDE is not subject to observed sample data and has been demonstrated particularly advantageous with small datasets (with very low sampled occurrence frequencies) in generating reasonably accurate estimations [Moon *et al.* 1995; Steuer *et al.* 2002].

Although KDE algorithm generates accurate probability and information estimations in small sample sizes, it is much more complicated than the naive algorithm and causes increased computation complexity (especially for the calculation of the kernel function). Therefore, for the purpose of this thesis to demonstrate that information measures are a valid means for examining and understanding PCMs, we will in the following chapters and sections use the simple estimation algorithm where possible. The KDE algorithm is employed in cases where large datasets are not available.

3.3 Information Model

As discussed in Chapter 2, PCMs rely on extrapolating short periods of future entity state from the received entity state updates to maintain a sufficient level of consistency while minimizing network traffic. The underlying motive and philosophy of the information model for PCMs presented here is that update packets can be used to extrapolate future entity states because of the temporal dependence among states at different times, therefore contextual entity state holds some knowledge that enables PCMs to locate potential entity states in the future. For example, standard DR directly uses the current state derivatives to predict future entity states assuming the entity motion status remains the same; NR references a history of recent entity motion to produce an estimate of future state, and directs the local and remote models along a path through that predicted future state. This is based on the assumption that the true dynamic of the entity during the prediction horizon does not diverge very much from the expected averaged path.

Naturally, to further improve the performance of PCMs, a quantified investigation of the temporal dependence in entity behavior is necessary. In the proposed information model, such dependence is measured by mutual information between the entity states at different time instances. The mutual information measures the amount of knowledge in the contextual dynamics that can be used to reduce the uncertainty in predicting a future entity state. Figure 3.4 explains the uncertainty reduction

using a simple illustrative example, in which users control objects moving around the limited space in a virtual environment. From the perspective of the remote host, the object controlled by another local host could be anywhere within the area before any update information arrives. In terms of probability, this little knowledge about the true entity state is represented by a uniform distribution over the whole space, which indicates that the entity state is unpredictable because any position is equally possible from the viewpoint of the remote host. The uncertainty of the entity state is very high. Once an ESU for the object has arrived, the remote host would have some information, for example the updated position, about the true entity state. For a DIA that simulates realistic object movement, it is then very likely that the local entity only moved over a short distance during the period of network latency, and is now located within a small area near the update position. Therefore in the distribution under the condition of the update, some states now become more probable than others, making the true entity state now more predictable. The update information reduces the uncertainty of the entity state by making some of the state values more probable. The uncertainty can be further reduced if additional information (velocity, acceleration for example) is available. So the amount of information about the true entity state in the ESU determines how much predictability can be used by the remote host to estimate the future entity state. The proposed information model focuses on measuring such predictability in the contextual dynamics, and how this information is transmitted by the ESUs for the remote host to build a remote model of the entity state.

In a DIA, there are several sources of information about future entity states: physical laws such as limit of speed and acceleration, gravity; environmental constraints such as obstacles and fixed path or lanes; and repeatable patterns of structured behavior that indicates user interactions. The proposed information model focuses on measuring the contextual information in structured user behavior, namely the temporal redundancy in the contextual dynamic (i.e., the immediate history of the entity state to be extrapolated, which is very much related to the state), and how this information is transmitted by the ESUs for the remote host to build a remote model of the entity state.

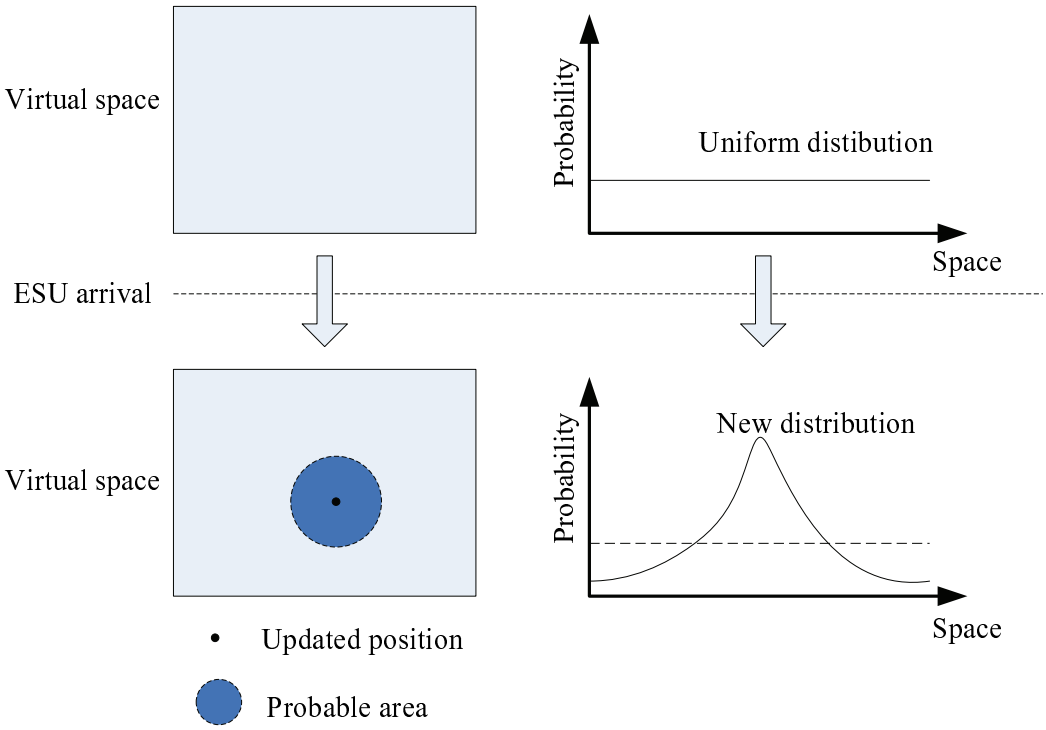


Figure 3.4 A simple illustration of how dependence can be used to reduce the uncertainty of the entity state.

Before going through a detailed discussion of the information model, it is worth noting that several measures of dependence other than the information metric (such as Euclidean distance, Pearson correlation, Windowed Cross-Correlation, etc. [McCoy *et al.* 2004]) are available. However, most of them only measure specific dependency patterns, such as linearity in the case of the Pearson's correlation, and consequently can only be of utility for a specific class of extrapolation methods that explore the particular pattern. In DIAs, predictability in user behavior is generally nonlinear and complicated because of its complex dependence on many factors such as user experience, knowledge of the shared virtual environment, user intentions and goals, environmental structures, layout and constraints, and common behavior exhibited by groups of similarly experienced users [McCoy 2007]. Using the existing methods to measure entity motion predictability could therefore be misleading as they will be ignoring useful temporal dependence in other forms than a particular pattern of entity movement (for example, an entity tends to remain roughly the same velocity

as the previous moment) [Li 1990]. Such situation is illustrated by a simple comparison shown in Figure 3.5. In this instance, the Pearson's correlation r_{xy} defined in (3.18) is used to measure the dependence between two sets of sampled data $x(k)$ and $y(k)$, each consists of $N = 41$ data points.

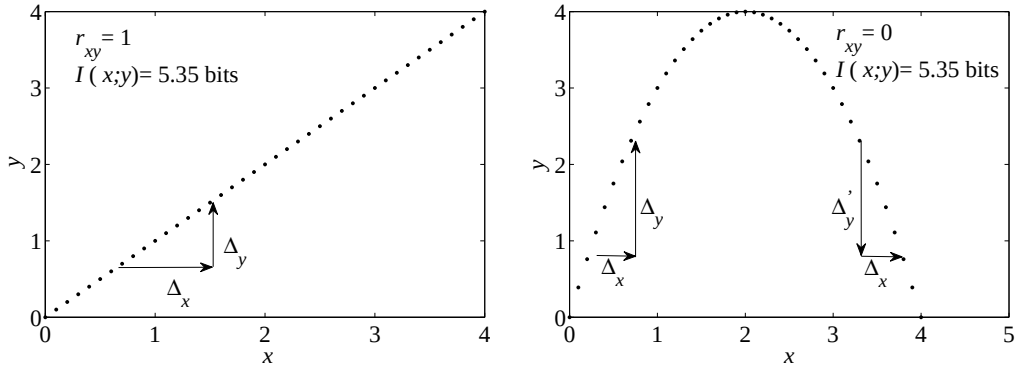
$$r_{xy} = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{x(k) - \mu_x}{\sigma_x} \right) \left(\frac{y(k) - \mu_y}{\sigma_y} \right), \tag{3.18}$$

where

μ_x and μ_y denote the sample means of x and y respectively,

σ_x and σ_y denote the standard deviations of x and y respectively.

In Figure 3.5(a), the Pearson's correlation gives the highest coefficient ($r_{xy} = 1$), indicating perfect linear dependence between x and y. Every increase Δ_x in x is corresponding to a proportional increase Δ_y. However, in Figure 3.5(b), due to the obvious non-linear relation between the two datasets, the same Δ_x leads to variations Δ_y and $\Delta_y^{'}$ with the same magnitude but in opposite directions. Therefore the effect of a variation in x on y is offset and the correlation coefficient is now zero. Such a vanished correlation does not imply that the two variables are independent, but merely suggests that there is no obvious linear relationship between them. In fact, the two datasets in Figure 3.5(b) can be perfectly determined from each other, just



(a) (b)

Figure 3.5 Using the Pearson's correlation and mutual information to measure the dependence between two datasets given by (a) $y(k) = x(k)$ and (b) $y(k) = x(k) \cdot (4 - x(k))$.

as the case in Figure 3.5(a). The Pearson's correlation fails to reflect the non-linear dependence between the two variables.

The mutual information, on the other hand, accurately captures the deterministic relationships and gives the highest mutual information ($\log 41 = 5.35$ bits) for both linear and non-linear dependence. By using probabilistic measures, the information metric is independent of any particular type of user behavior pattern and provides a general measurement of the temporal dependence in any possible form that can be used to extrapolate future entity state.

3.3.1 Information Generation

For the ease of illustrating the information model, the true entity dynamic on the local controlling host 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 modeling of other forms of entity state can be formalized in a similar manner. The value of $x(k)$ varies within a finite discrete set of entity state values $S = \{s_i\}$. The remote model $\hat{x}(k) = \{\hat{x}(1), \hat{x}(2), \ldots\}$ is the state approximation simulated by the remote host using PCMs.

Information generation refers to the evolution of the local entity state. From the information theory perspective, the information model regards the time-evolution of the true entity dynamic as constantly generating information about itself. At each simulation step, the determination of the current state of the local entity by the application simulation process fully removes the uncertainty of the current entity state before it is simulated. The amount of information generated by the rendering of this state depends on the complexity of the entity dynamic. For example, determining the current state of a random walking movement in the environment obviously gives more information than the state simulation for a motion with a constant velocity. As the simulation continues, the information generated at each simulation step constitutes a flow of information that accounts for the time-evolution of the entity state value. The average *information generation rate*, also the state uncertainty at each

step, can be measured by the entropy $H(x)$ of the entity trajectory:

$$H(x) = -\sum_{s_i} p_x(s_i) \log p_x(s_i), \tag{3.19}$$

where $p_x(s_i)$ is the probability function of the entity state. The set of entity state values S covers the complete state space. It is very likely that in a DIA an entity only travels over part of the whole space. The unexplored territory would produce zero probabilities in (3.19) and does not affect the information calculation.

The information generation rate in (3.19) is a memoryless entropy that measures the amount of information needed to locate one entity state with absolute accuracy without any knowledge about its previous dynamic, i.e., the amount of uncertainty about the entity state before receiving any update packet. Since PCMs operate solely based on the latest update packet rather than the entire motion history, this memoryless entropy is a suitable foundation for the information model. This information generation rate is also the minimum amount of data in number of bits needed to represent a complete description of the current entity state where no PCM is used in the DIA. In such a case, the complete description of the entity state at all the simulation steps are transmitted to the remote host. Assuming the same information is received by the remote host, a perfect remote model of the local entity state with the absolute accuracy can be rebuilt. Therefore, the information generation rate in (3.19) represents the ideal case of a complete replica of the local entity state dynamic on the remote host, and the highest consistency is achieved.

However, such complete state replication is not achievable in practical DIAs, meaning that it is impossible to have the necessary information to eliminate the state uncertainty directly through using complete descriptions. Instead, PCMs make use of extrapolation methods to explore contextual dependence in previously received ESUs for information about the entity state. The information model uses mutual information to measure the amount of the useful temporal dependence between an ESU and the entity state to be extrapolated.

Let $\boldsymbol{u}(k)$ denote the entity state update generated at the k^{th} simulation step. It should be first noted that the content of $\boldsymbol{u}(k)$ varies for different extrapolation

models under use. In the $zero^{th}$-order extrapolation, the ESU only contains the current position $x(k)$ of the entity. $\boldsymbol{u}(k)$ could also be a vector variable if additional information is included in the ESU, such as the velocity $v(k)$ and acceleration $a(k)$ if the first or second-order equation is used. Consider predicting a future entity state $x(k+\tau)$ at a particular prediction span τ from the ESU $\boldsymbol{u}(k)$. The available knowledge or dependence between the information source and the entity state to be extrapolated is measured by *Available Information* $I(\boldsymbol{u}; x_\tau)$:

$$\begin{aligned} I(\boldsymbol{u}; x_\tau) &= I(\boldsymbol{u}(k); x(k+\tau)) \\ &= \sum_{\boldsymbol{u}(k), x(k+\tau)} p_{\boldsymbol{u}x}(\boldsymbol{u}(k), x(k+\tau)) \\ &\qquad \cdot \log \frac{p_{\boldsymbol{u}x}(\boldsymbol{u}(k), x(k+\tau))}{p_{\boldsymbol{u}}(\boldsymbol{u}(k)) p_x(x(k+\tau))}. \end{aligned} \tag{3.20}$$

The update packet series $\boldsymbol{u}(k)$ is hypothetical in that it consists of prediction parameters at every simulation tick, which would be sent out with an update packet if necessary. This mutual information is the average amount of knowledge in an ESU that can be used to predict the entity state τ steps later than its generation.

The available information $I(\boldsymbol{u}; x_\tau)$ in (3.20) is the core of the information model for PCMs because it quantifies the amount of temporal dependence between the ESU and the entity state to be extrapolated. This dependence is the total information available for extrapolation models to speculate the entity state in future. The generality of the information metric guarantees that the measured mutual information $I(\boldsymbol{u}; x_\tau)$ takes all forms of user behavioral structures or patterns into consideration and sets an upper bound to the amount of information that can possibly be explored by any predictive extrapolation for the purpose of reducing the uncertainty of a future state value.

$I(\boldsymbol{u}; x_\tau)$ represents the information characteristic of the true entity state dynamic and evolves in two dimensions, namely the prediction span τ and ESU content $\boldsymbol{u}(k)$. Firstly, it is obvious from (3.6) that including more parameters in the ESUs would generally increase the available information about a future entity state and offer the extrapolation model more potential to give better predictions. This "information cannot hurt" explains why advanced predictive statistical models (such as

Neuro-reckoning) outperforms simple polynomial equations in prediction accuracy — they take advantage of better information sources which usually consist of longer referencing history of the entity dynamic. However, it is not guaranteed that the additional parameters in the ESUs bring more information. This depends on the characteristics of the dynamics. For example, transmitting an additional velocity value to extrapolate future states for a randomly walking entity does not bring any additional information than the simple positional update since the extra parameter is irrelevant, nor does including the acceleration value into the ESUs when extrapolating a constant-speed movement increase the amount of available information because the parameter is redundant for this particular entity dynamic under consideration. The information metric enables a discrimination of the parameters in the ESUs in term of their capacity in providing information about the entity state to be extrapolated.

$I(\boldsymbol{u}; x_\tau)$ also varies with prediction span τ. This time-evolution of $I(\boldsymbol{u}; x_\tau)$ with increasing prediction spans characterizes the regularity of the entity dynamics. For a deterministic entity motion (such as the state of a programmed *Bot*) whose actions in the future can be determined by its state history and environmental constraints, the current ESU, containing the appropriate parameters, could hold full information about all the entity state changes that follow. In this case the mutual information $I(\boldsymbol{u}; x_\tau)$ would remain at the full information rate $H(x)$ for all the prediction spans. In the other extreme, namely the completely random motion case, the future movements of the entity are independent of the current status and the information $I(\boldsymbol{u}; x_\tau)$ would drop to zero for any prediction span $\tau > 0$. Figure 3.6 illustrates the time-evolution of the mutual information $I(\boldsymbol{u}; x_\tau)$ with increasing prediction spans. Entity dynamics in practical DIAs are generally neither random nor deterministic. In the case where the prediction span $\tau = 0$, $I(\boldsymbol{u}; x_0)$ is essentially the auto-mutual information in (3.9), and the message holds enough information to eliminate all the uncertainty $H(x)$ of the current entity state. Hence the current state can be determined with absolute certainty, which is obvious because the state itself is updated. However, an ESU typically has partial information about the future entity states, which decays with increasing prediction spans since the future entity states become

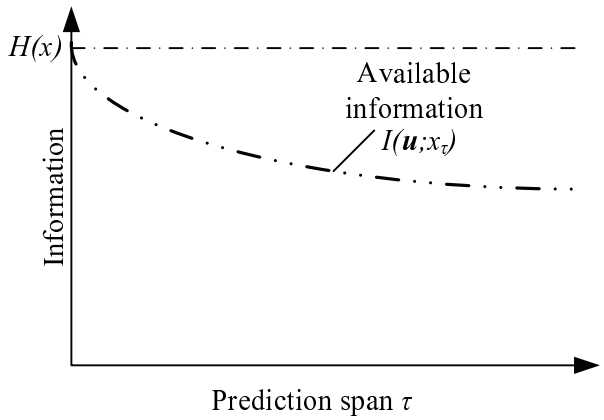


Figure 3.6 A simple illustration of the amount of the available information $I(\boldsymbol{u}; x_\tau)$ in an ESU for the entity state changes for increasing prediction spans τ. An ESU typically has the full information $H(x)$ about the current entity state and only partial information about the future states.

more and more independent of the current motion status (described by parameters such as position, velocity, etc.). This partial information enables the extrapolation models to predict the future entity states with some level of certainty and precision. The information metric provides a measurement of the information preservation of the ESUs: those ESUs whose mutual information $I(\boldsymbol{u}; x_\tau)$ drops more slowly with increasing prediction spans clearly hold greater potential in giving sufficient prediction accuracy for a longer period of time and allow for a further reduction in updates transmission under the same inconsistency requirement.

3.3.2 Local Compression

PCMs rely on employing extrapolation models to predict entity state at multiple steps in the future from one update message so that network traffic for transmitting the update messages can be reduced to go through the network connections without causing congestion. The reduction in network traffic is (indirectly) regulated by the local error threshold. It has been shown in the previous section that it is the mutual information residing in the ESUs that enables the speculation of the future states. However the partial information is not capable of providing completely accurate prediction and the extrapolation error grows with time as the mutual information

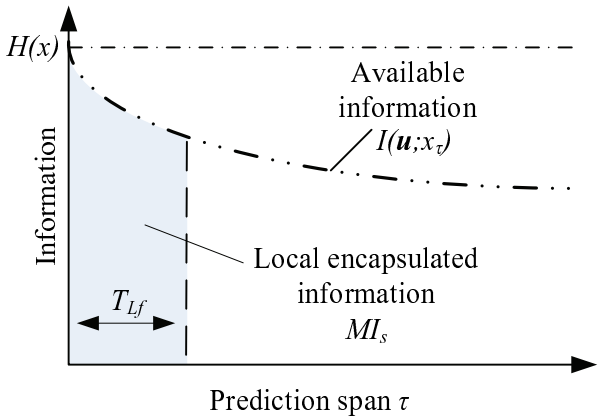


Figure 3.7 The sum of the available information for all the prediction spans within the local functioning period T_{Lf} is encapsulated in the ESU for future state extrapolation. The partial information included reduces the information rate, taking the form of a compression.

$I(\boldsymbol{u}; x_\tau)$ decays with increasing prediction spans. Therefore the state model estimated from an ESU becomes increasingly irrelevant to the true entity state, and the resultant inconsistency is more likely to exceed the desired level. A violation of the error threshold would trigger a new message update to the remote host to replace the outdated ESU with updated information. The information volume delivered by an ESU forms the foundation for controlling the inconsistency arising from extrapolating future entity states from the ESU.

To measure the amount of useful information in an ESU intended for extrapolation, the *Functioning Period*, T_f of an ESU is defined as the time period during which the ESU is being used as the information source for building an entity state model by the extrapolation method. On the local host machine, the inaccurate prediction, which triggers the generation of the ESU, is immediately corrected. The ESU is then extrapolated by the prediction scheme under use to build the local entity state model, until the prediction error exceeds the threshold again and a new ESU is generated. Therefore the *Local Functioning Period* T_{Lf} of an ESU is the time period between the generations of its own and the next ESU. As shown in Figure 3.7, the amount of *locally stored mutual information* in an ESU $\boldsymbol{u}(k)$ is the sum of the available

information $I(\boldsymbol{u}; x_\tau)$ for all the prediction spans within its local functioning period:

$$MI_s = \sum_{\tau \in T_{Lf}} I(\boldsymbol{u}; x_\tau). \tag{3.21}$$

In the continuing simulation of a DIA, the stored information in (3.21) can be averaged over the functioning period to give a *locally stored information rate* $R_{s,local}$, as shown in (3.22). $R_{s,local}$ characterizes the average amount of information provided by the local host machine for modeling of the entity state at each simulation step. Notice that an ESU only has partial information about the future entity states. Thus $R_{s,local}$ is a reduced information rate lower than the perfect information rate $H(x)$ under the complete replica scheme. The information loss causes inconsistency in the modeled entity state due to the remaining uncertainty.

$$R_{s,local} = \frac{MI_s}{T_{Lf}}. \tag{3.22}$$

For a given entity dynamic and ESU content, the locally stored information rate is regulated by the local error threshold. A large error threshold allows for the modeled entity state to diverge farther from the true entity state before sending a new update, and generally leads to a longer functioning period. According to (3.21), the longer functioning period would in turn increase the total amount of stored information in an ESU. However, the average information rate for extrapolating the modeled state at each step, due to the non-ascending feature of the available information $I(\boldsymbol{u}; x_\tau)$ for typical DIAs, would be compromised. The inconsistency is consequently more severe. From the information perspective, increasing the local threshold prefers a lower network load over a better modeled fidelity in the *Consistency Throughput Trade-off*. On the contrary, a smaller error threshold leads to a relatively shorter functioning period. The information in one particular ESU is reduced but the information rate is higher because the stored information is updated more frequently.

The information metric also provides mechanisms by which the value of the local error threshold can be selected based on the regularity of the entity dynamics. A quite slow information drop with increasing prediction spans can be expected for

entity motions such as a car-racing game, where objects mostly move along their assigned lanes. In this case, a large error threshold is preferred because the ESUs stay in effect longer without causing much further information loss, and network traffic is reduced. In other scenarios such as First Person Shooting (FPS) games, for example, where players shoot enemies out of random hiding places, mutual information would drop faster, and the ESU functioning period has to be short to include timely information and thus a small threshold is better.

The employment of the local error threshold allows entity states at multiple simulation steps to be modeled from one update packet. However, although an ESU has full information about the current state, it only bears partial information about the future states, and the rest of the information is lost. From the viewpoint of the information model, the local operation of PCMs can be seen as a lossy compression, just like in video compression where the negligible difference between the original frame and the motion-compensated frame is quantized and ignored for a shorter code length. Using one packet to predict multiple future states eliminates temporal redundancy in the user motion, and ignoring inconsistency below the local threshold reduces irrelevancy (assuming state difference below the threshold is perceptually tolerable). In this way, PCMs reduce the perfect high information rate $H(x)$ down to the lower stored information rate $R_{s,local}$. The irreversible information loss in the process causes inconsistency in the modeled entity dynamic.

3.3.3 Information Transmission

To implement a synchronized virtual world in a networked environment, the reduced information about the local entity dynamic must be transmitted to the remote host through the underlying network. Due to the limited network bandwidth, the network data transmission always comes with non-zero network latency. As shown in (2.9) and Figure 2.5, network latency introduces extra transmission error to the remote model. The remote host has to rely on the now out-of-date information in the previous received ESU to extrapolate the remote entity state model when the new update is still in transmission. The new ESU is only utilized by the remote host

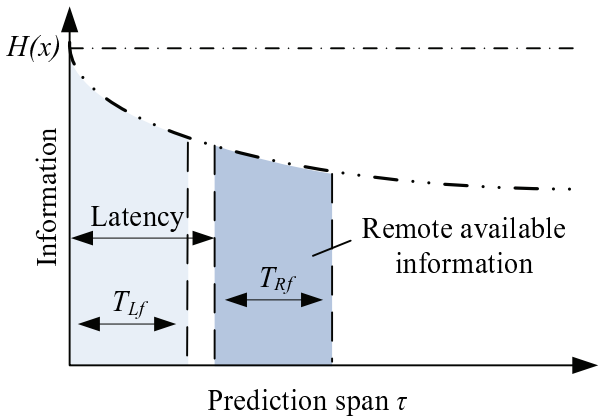


Figure 3.8 The impact of network latency on remote inconsistency is represented by the shifting of the local functioning period. This delay further reduces the available information for the remote host due to the decreasing feature of the temporal dependence in the entity dynamic.

after its arrival at the remote host.

From the perspective of the information model, the impact of network latency on consistency maintenance in DIAs is characterized by a shift of the functioning period of the ESU. As shown in Figure 3.8, the ESU is being extrapolated only from the time it arrives at the remote host (its generation time delayed by the period of network latency) until the next ESU comes. Consider the simple case of constant network latency for the ease of illustration. The *Remote Functioning Period* T_{Rf} of an ESU is obtained by shifting its local functioning period T_{Lf} by a period of the network latency towards increasing prediction spans. As a result of the descending feature of the mutual information $I(\boldsymbol{u}; x_\tau)$, this shift adds further information loss in two aspects. Firstly, the information stored for the extrapolation of states during the transmission period, with relatively high information quality, is wasted because the remote host is still extrapolating the previous ESU. Secondly, the current ESU must be extrapolated for further prediction spans, where the information quality is lower, when the next ESU is being communicated. Consequently, the available information for remote extrapolation is further reduced by the network latency. When the ESU arrives at the remote host after the network latency L (measured in simulation steps), the *remotely stored information rate* $R_{s,remote}$ is the average

information within the remote functioning period of the ESU:

$$\begin{aligned} R_{s,remote} &= \frac{\sum_{\tau \in T_{Rf}} I(\boldsymbol{u}; x_\tau)}{T_{Rf}}, \\ T_{Rf} &= T_{Lf} + L. \end{aligned} \tag{3.23}$$

The information model reveals that minimizing network latency is especially important for consistency maintenance since it improves the utilization of the high quality information for small prediction spans.

The latency-induced information loss does not concern traditional video compression because the video can be buffered before being played out without causing significant distortion in user perception of the video. However, in DIAs which impose constraints on timeliness of the data transmission, minimizing latency-induced information loss is a key factor that facilitates the implementation of real-time interactivity under limited network resources.

3.3.4 Remote Reconstruction

Remote reconstruction refers to the process whereby the remote host machine extrapolates the update information in the latest ESU to build the remote entity state approximation. The remote entity state is calculated from the parameters in the ESU by the extrapolation equation. From the information and compression perspective, the parameters in the ESU capture some of the dependence in the true entity dynamics, from which the redundancy in the original entity dynamics removed by the local compression is reinserted by the extrapolation equations. In the "PCM-Video Codec" analogy, remote reconstruction can be seen as the decoding stage where a full frame is estimated from the arrived I/P-frames, using motion compensation. Given the available information in the ESU, the fidelity of the reconstructed entity state model depends on the efficiency of the extrapolation equations in exploiting the information. A good prediction model that suits the patterns and structures in the original entity dynamics could extract and interpret a higher portion of the delivered available information so that the reconstructed remote model could reflect characteristics of the true entity state more precisely and exhibit better consistency.

The proposed information model measures the suitability of an extrapolation model for a particular entity dynamic from the viewpoint of uncertainty reduction. As discussed in the previous section, the information arriving at the remote host machine is typically incomplete. The information loss (as the gap between the highest entropy rate $H(x)$ and the available information in Figure 3.9) is the main lossy element of the compression scheme, and such loss is irreversible. The available information, supposedly utilized in full, only provides so much knowledge about the true entity state as to constrain the future state to be extrapolated to a few probable values. However, an extrapolation model, taking the form of a function or an equation, is a deterministic mapping $g(\boldsymbol{u}(k), \tau)$ from the parameters in an ESU $\boldsymbol{u}(k)$ to the modeled state $\tilde{x}(k+\tau)$. This deterministic relation ignores the potential possibilities of the future states other than the predicted value. Therefore, only part of the delivered information in the message is extrapolated. For a specific prediction span τ, the *extrapolated information* $I(\tilde{x}_\tau; x_\tau)$ is defined between the prediction result $\tilde{x}_\tau$ and the true entity state x_τ to measure the information utilization efficiency of the prediction model:

$$\begin{aligned} I(\tilde{x}_\tau; x_\tau) &= I(\tilde{x}_\tau(k); x_\tau(k)) \\ &= \sum_{\tilde{x}_\tau(k), x_\tau(k)} p_{\tilde{x}_\tau x_\tau}(\tilde{x}_\tau(k), x_\tau(k)) \\ &\qquad \cdot \log \frac{p_{\tilde{x}_\tau x_\tau}(\tilde{x}_\tau(k), x_\tau(k))}{p_{\tilde{x}_\tau}(\tilde{x}_\tau(k)) \cdot p_{x_\tau}(x_\tau(k))}, \\ \tilde{x}(k+\tau) &= g(\boldsymbol{u}(k), \tau), k = 1, 2, \ldots, \end{aligned} \tag{3.24}$$

where the *state speculation series* $\tilde{x}(k+\tau), k = 1, 2, \ldots$ consists of the prediction results extrapolated from the update packet series $\boldsymbol{u}(k)$ for a particular prediction span τ. It is then clear from (3.24) that the ESUs act as a information carrier that conveys information about the true entity state to the estimated state model.

The extrapolated information in (3.24) measures the average information from the messages utilized by the prediction model in producing the prediction result at the given prediction span, and it is apparently bounded by the available information in (3.20), as illustrated in Figure 3.9. The efficiency of utilizing the provided information indicates the suitability of the extrapolation model under use to exploit patterns in the user behavior. The additional information loss caused by inadequate

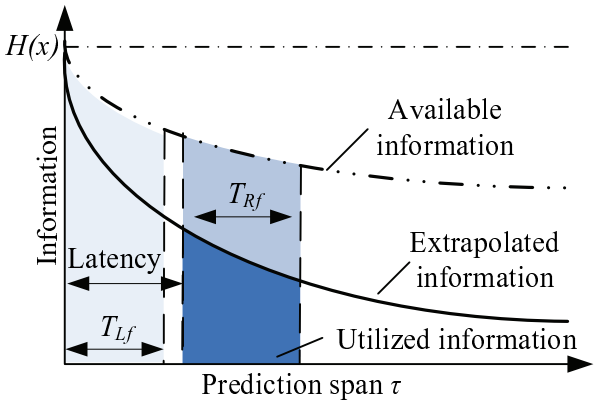


Figure 3.9 When the entity state model is extrapolated on the remote host, the utilized information is constrained by the imperfect prediction model, which can only explore part of the delivered available information about the true entity dynamic.

extrapolation (illustrated by the light blue area between the available and extrapolated information) is theoretically recoverable. The extrapolated information of a better prediction model would approach the available information. Following similar reasoning, the information rate that is actually *utilized information rate* $R_{u,remote}$ by the prediction model is calculated as in (3.25):

$$R_{u,remote} = \frac{\sum_{\tau \in T_{Rf}} I(\tilde{x}_\tau; x_\tau)}{T_{Rf}}. \tag{3.25}$$

$R_{u,remote}$ measures the amount of information about the true entity state that the remote state model contains. Such information goes through local compression, transmission and remote reconstruction. From the perspective of the remote user, a high information rate indicates that he/she is offered a rendered remote entity state model from which the true entity state could be determined with a high confidence level. This in turn, by providing a reliable entity state model, facilitates the remote user in responding to and interacting with the local entity.

Figure 3.10, with the notations shown, re-interprets the framework of PCMs in Figure 2.4 from the perspective of information theory and lossy compression. Through the local compression, transmission, and remote reconstruction, the information rate $H(x)$ generated by the local simulation of the entity state is gradually

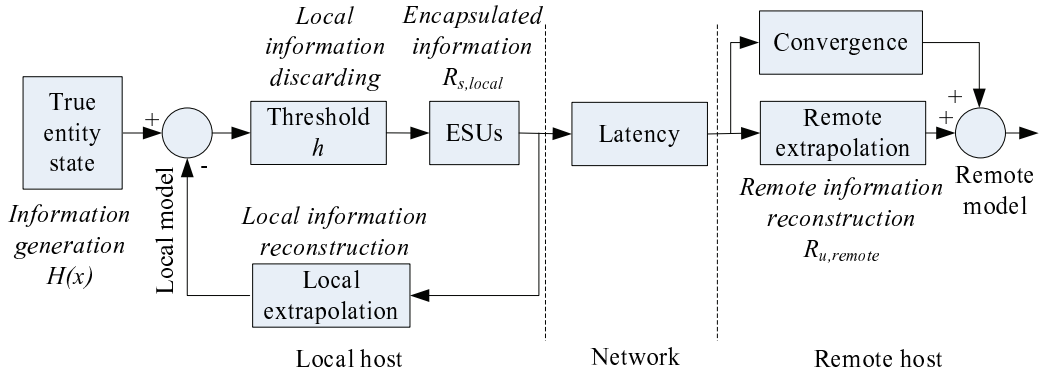


Figure 3.10 The framework of Predictive Contract Mechanisms based on information theory and lossy compression perspective.

reduced to $R_{u,remote}$ between the true and modeled entity states. The arising inconsistency from employing PCMs is measured as information loss during the process. The spatial analysis of dead reckoning in Figure 2.5 and (2.9) identifies three factors, namely the local threshold, network latency and inaccurate prediction models, which determine the remote inconsistency. In the information model, these elements of PCMs are seen as causing information loss and increasing the uncertainty of the remote entity state model. From an information perspective, each time the information is reduced by one of the steps in the PCM operation, the requirement on the minimal number of bits necessary for reconstructing the remote model is also reduced. In this way, the *Consistency Throughput Trade-off* in consistency maintenance in DIAs can be seen as a *Rate-Distortion Trade-off* in video compression. In this trade-off, PCMs optimizes consistency by lowering data rate at the cost of a certain level of distortion between the true and modeled entity states.

3.4 Concluding Remarks

Throughout the first part of this chapter, the concepts and technologies in information theory used in the information model is introduced and reviewed. The philosophy underlying the information theory approach towards PCMs is that a quantified analysis of the temporal dependence in entity dynamics, which enables the employment of predictive modeling schemes to speculate future entity states

instead of updating them through network communications, could bring a novel perspective and insight towards consistency maintenance in DIAs. Furthermore, it is a prerequisite for optimization of PCMs under constrained network resources.

In the second part of this chapter, an information model is proposed as a general framework for the analysis of information flow in PCMs. In the information model, PCMs can be viewed as an information processing scheme whose role is to discard irrelevant information within a tolerable perception limit (the local error threshold), store useful information using the minimal data units (the ESUs), and remotely reconstruct the entity state with sufficient fidelity. A set of information metrics is defined to measure the amount of information conveyed or lost at each step of the PCM operation. Finally, a re-interpreted framework for PCMs based on the information and compression perspective is presented.

In the next chapter, the information model is implemented and applied to several entity dynamics from various environments, demonstrating the utility of the information model to general DIA scenarios and exemplifying information analysis to PCMs. This work also provides a foundation for new techniques improving consistency maintenance in DIAs from the information perspective.

Chapter 4

Information Model Implementation

4.1 Introduction

In the previous chapter, the information model for PCMs is proposed as a general framework to facilitate quantifying the trade-off between consistency and throughput in DIAs based upon an information theory and video compression perspective [Zhang *et al.* 2008, 2009]. The information model enables measurement and analysis of the efficiency of PCM operation in exploiting the information about the user behavior in the reduced entity state updates.

This chapter aims to illustrate the utility of the proposed information model by conducting experimental studies and information analysis on various entity motions and PCM techniques. The information model is implemented at two levels. The first part of this chapter uses the information metrics to measure the predictability of a number of datasets derived from entity motions in experimental DIAs. This analysis is motivated by the information perspective that the predictability, or temporal dependence, in entity behavior enables the use of predictive schemes to extrapolate entity state models. The performance of a general PCM is therefore subject to the

predictability of the user behavior. Two types of motion data are considered in this study: motions simulated from explicit models that control the predictability of the generated trajectories, and datasets collected from user behaviors within realistic DIA scenarios.

The second part of this chapter implements the information model and analyzes the complete operation of PCMs. This analysis evaluates their performance in utilizing the available predictability from the user behavior when building the remote model of the entity. Three extrapolation models, namely first and second-order dead reckoning and neuro-reckoning, are employed to extrapolate an entity motion collected from human-user behaviors in a realistic DIA. In this case the DIA is represented by a First-Person Shooting (FPS) game. The suitability of the extrapolation models is then comprehensively studied and compared from the information perspective. This implementation of the information model reveals that the ability of an extrapolation scheme to provide sufficient consistency with reduced network traffic depends greatly on its efficient utilization of the predictability inherent to the entity's motion.

Finally, the concept is extended from intra-entity dependence to an inter-entity analysis. Specifically, statistical inter-dependence between state-evolutions across different entities in a DIA is measured using mutual information. The interaction between users exhibits drastic changes over time and such cross-entity dependence also provides predictability about the state of one object from the other one that is closely interacting with the former. From the viewpoint of video compression, such inter-entity predictability demonstrates *spatial redundancy* among different elements in the virtual environment, and could be exploited to further reduce the network traffic for maintaining a consistent view of the shared virtual world.

4.2 Measuring Motion Predictability

As discussed in Chapter 3, one of the primary factors influencing the successful deployment of PCMs is entity motion predictability. A simple entity dynamic exhibits strong temporal dependence, from which the extrapolation models can estimate the

future entity state with high certainty. Such entity states can be shared among the dispersed participants with low network traffic. A complicated entity motion, on the other hand, can be defined as that motion which presents frequent irregular changes in state and leaves little useful information in its behavior, making the accurate speculation of its future state very difficult. Maintaining a consistent view of the state of such an entity requires higher network throughput for transmitting the frequent synchronization messages.

In this section, to examine the information characteristics of various entity trajectories, motions simulated by explicit models are consider first. The motion models used here describe particular aspects commonly seen in user behaviors within DIAs. In particular, modeled motions include three deterministic motion models (*smooth*, *bounce*, and *jolt*) and a *complex* motion derived from the combinations of the three deterministic models. Such modeled motions exhibit controlled predictability and can be used to validate the results from the information measurements. Collected motion data from two realistic DIA scenarios, namely a Pong game and a virtual navigation environment, are then examined using the proposed information metrics. Once again, all the simulations and analysis in this chapter are based on one-dimensional motion data for ease of illustration and discussion. The same procedures can be easily generalized to multi-dimensional data with increased computation complexity.

4.2.1 Predictability Measurement Overview

As discussed in Chapter 3, the temporal dependence in an entity dynamic is embodied in the contextual dynamics of the object and can only be measured with respect to a certain information source that carries such dependence. In particular, the predictability of an entity motion x is measured by the amount of information about its future state. Such information is encapsulated in the combination of instantaneous motion status (position, velocity, etc.) in an ESU $\boldsymbol{u}$ for a particular extrapolation model. This is formalized by the available information $I(\boldsymbol{u}; x_\tau)$ in

(3.20) for increasing prediction spans τ. The utilization of this encapsulated predictability by the extrapolation model is measured by the extrapolated information $I(\tilde{x}_\tau; x_\tau)$ in (3.24). The measured mutual information is scaled to the entropy $H(x)$ of the entity motion to remove the effect of the range of the entity state space on the absolute information values. Therefore, the predictability in this section measures the degree to which the total uncertainty of an entity state in the future could be eliminated by knowing the current motion status. High predictability of a motion, if properly used by the extrapolation model, enables an accurate prediction of its future state and better consistency with lower network traffic. The predictability available in the motion and extrapolated by the extrapolation model is defined in (4.1) and (4.2), respectively:

$$\text{Available predictability} = \frac{I(\boldsymbol{u}; x_\tau)}{H(x)}, \tag{4.1}$$

$$\text{Extrapolated predictability} = \frac{I(\tilde{x}_\tau; x_\tau)}{H(x)}. \tag{4.2}$$

To measure the predictability of the entity motions, two kinds of information sources, the first and second-order update packets, are examined in terms of the amount of information about the future entity state evolution embraced in them. These extrapolation models are chosen due to their overwhelming popularity in DIA deployments. For an entity motion $x(k)$, the elements in an ESU $\boldsymbol{u}(k)$ and the prediction equations for the two extrapolation models are summarized in Table 4.1. The availability of the motion status varies among different datasets because of the different nature of motion generation in the various application domains. For modeled motions, the

Table 4.1 Elements in an ESU and the prediction schemes for the first and second-order extrapolation models.

Extrapolation order	ESU elements	Extrapolation equation
1^{st}	$x(k), \dot{x}(k)$	$\tilde{x}(k+\tau) = x(k) + \dot{x}(k) \cdot \tau\delta$
2^{nd}	$x(k), \dot{x}(k), \ddot{x}(k)$	$\tilde{x}(k+\tau) = x(k) + \dot{x}(k) \cdot \tau\delta$ $+\frac{1}{2}\ddot{x}(k) \cdot (\tau\delta)^2$

state derivatives can be directly recorded during the simulation, while for data collected form realistic DIAs they may have to be derived from other states that are available. The methods to acquire the prediction parameters will be respectively discussed in Section 4.2.3 for each particular motion type.

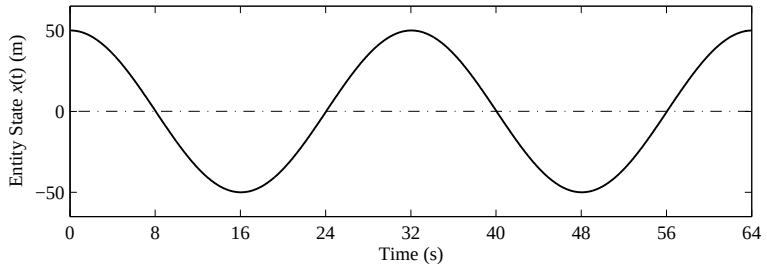
4.2.2 Data Collection

4.2.2.1 Deterministic Motions

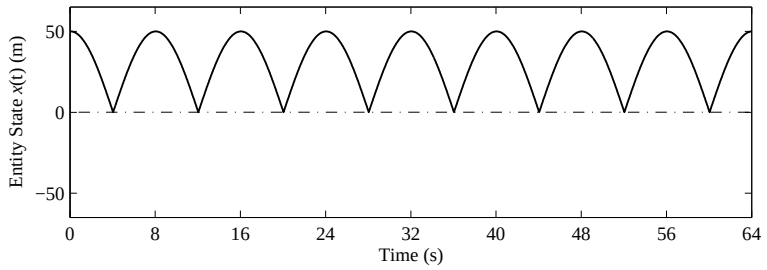
Assessing entity behaviors can be difficult as they are generally complicated and depend heavily on the type of application domains that the human users interact within. However, an entity's motion can at least locally be described by one of a number of representative curve classifications that define and describe the common characteristics of such motions [Singhal 1996]. That is to say, the complicated entity motions in DIAs can be modeled as combinations of several base "prototypes" that are much simpler and easier to study. In Lee *et al.* [2000], entity motions have been classified according to one of the following three descriptive types: *smooth*, *bounce* and *jolt*.

The smooth or circular motion, as shown in (4.3), can be produced by an entity moving around a fixed point in a two-dimensional virtual environment at constant angular velocity [Lee *et al.* 2000]. The one-dimensional representation of the smooth motion is a sinusoidal curve with the same period and amplitude. Circular motion describes the motion mode where the entity's velocity smoothly changes in direction rather than in magnitude [Singhal and Cheriton 1995], and the speed of the entity remains unchanged. Figure 4.1(a) shows an example of such a trajectory with a period of 32s and an amplitude (or radius) of 50m. The motion is simulated at a rate of 20Hz (i.e. a constant simulation time-step of $\delta = 50$ms)

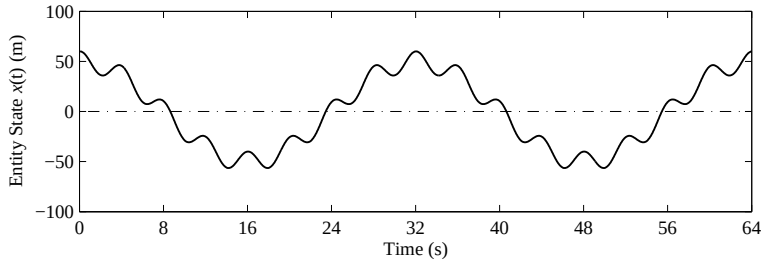
$$\begin{aligned} x(t) &= A\cos(\omega t), \\ t &= k\delta, \\ \omega &= \frac{2\pi}{T}, \end{aligned} \tag{4.3}$$



(a)



(b)



(c)

Figure 4.1 Examples of one-dimensional entity states for (a) *smooth* motion (period = 32s, amplitude = 50m), (b) *bounce* motion (period = 16s, amplitude = 50m), and (c) *jolt* motion (period = 32s / 4s, amplitude = 50m / 10m. All the modeled motions are simulated at a rate of 20Hz.

where:

k is the index of the current simulation step,

δ is the simulation step interval,

A is the amplitude (or radius) of the smooth motion,

T is the period of the smooth motion.

A bounce motion can be produced by an entity moving back and forth through a 90-degree arc (centered at a fixed point) at a constant angular velocity [Lee *et al.* 2000]. The entity's motion direction is reversed at each collision or "turn", but retains

a constant speed. Such a trajectory describes the motion characteristics where a smoothly moving entity occasionally exhibits sudden changes in entity motion direction, which are mostly caused by object collisions or circumventing an oncoming enemy [Singhal and Cheriton 1995]. As in (4.4), a bounce motion can be described as a variation of the smooth motion. Figure 4.1(b) shows an example of bounce motion with a period of 16s and an amplitude of 50m.

$$x(t) = A\left|\cos(\omega t)\right|. \tag{4.4}$$

As formalized in (4.5), the jolt motion describes a more complicated motion characteristic that arises as a result of frequent bounce motion. A jolt trajectory can be produced by an entity that spins itself while moving in a circle [Lee *et al.* 2000]. Figure 4.1(c) presents an example of jolt motion. In this instance a period of 32s and an amplitude of 50m for the inner (circular) motion, and a period of 4s and an amplitude of 10m for the outer (spin) motion are chosen.

$$\begin{aligned} x(t) &= A_1\cos(\omega_1 t) + A_2\cos(\omega_2 t), \\ \omega_1 &= \frac{2\pi}{T_1}, \\ \omega_2 &= \frac{2\pi}{T_2}, \end{aligned} \tag{4.5}$$

where:

A_1 and A_2 are the respective amplitudes of the inner and outer motions,

T_1 and T_2 are the respective periods of the inner and outer motions.

From a visual perspective, the three basic motion types introduced above exhibit increasing changes in motion status (described by entity velocity, acceleration, etc.). However they are all deterministic as the entity states only evolve with the passage of time once the parameters are set. From the information perspective, these deterministic motions leave no uncertainty for the future entity state and are among the simplest entity dynamics possible. It is clear that the bounce and jolt motions can be derived from variations of the simple smooth motion. Therefore, the smooth motion is considered representative of the deterministic motions and will be examined using the information model in the following sections. The bounce and jolt motions, together with the smooth motion, will be used to generate a non-deterministic motion

with controlled irregularity and randomness.

4.2.2.2 Motion with Controlled Randomness

In a virtual environment, an entity's motion rarely exhibits a single motion type (as described with the three deterministic motions) over its lifetime. As the entity interacts in the environment with other entities, its motion changes accordingly in response to the various stimuli or events such as collision with obstacles in the territory and engagements with other participants. As a crude approximation to such a more realistic entity dynamic, a *complex* motion that evolves with random factors in addition to the passage of time is introduced here to account for the non-deterministic state changes that characterize human-controlled behaviors in shared virtual world. The complex motion describes the unpredictable changes in motion type by randomly selecting among each of the three deterministic motion types (smooth, bounce, or jolt) for random intervals of time using random inputs for every motion. The model parameters (the amplitudes and periods) are restrained within minimum and maximum values.

Figure 4.2 shows an example of a complex motion that consists of 16 intervals of randomly selected deterministic motions. Each switching of the motion model is denoted by a vertical dashed line. This type of curve exhibits both smooth, rapid changes and sharp, unpredictable changes [Singhal and Cheriton 1995; Singhal 1996]. The random switching brings some level of irregularity to the entity state dynamic. Clearly, given a history of the entity motion, there would always be some uncertainty that remains about the future state. The predictability of motions generated by such a "white box" style model is controlled by the constrained intervals of allowed model parameters.

4.2.2.3 Pong Game Motion

The first, and very simple, application scenario uses motion data from a single-user "Pong-like" game developed in Java. Pong is among the very first multiplayer

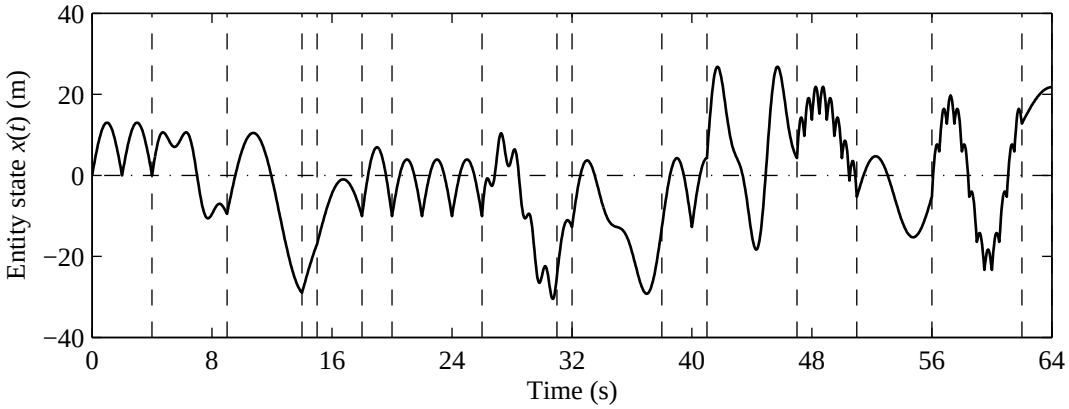


Figure 4.2 An Example of one-dimensional entity states for a *complex* motion. The period of the deterministic motion in each interval is randomly selected within 3 – 8s, the amplitude is selected within 8 – 20m, the period and amplitude of the outer circle for each jolt motion is selected within 1 – 3s and 4 – 10m, respectively.

video games and has achieved widespread popularity. By its nature, the behavior of the players are homogeneous, and thus the Pong-like application scenario developed here is simplified to the form of a single-player game. The game scenario is shown in Figure 4.3(a). The environment consists of a rectangle area surrounded by three "walls" and a moving paddle. A ball moves at constant speed around the area delineated by the walls and the paddle. The direction of motion of the ball is reversed when it hits the walls or the paddle, causing it to bounce back. The player tries to keep the ball from going outside by moving the mouse-controlled paddle on the right side. The paddle is designed to be controlled by a mouse, instead of a keyboard, to embrace features of the continuous movement of the human user's hand. The entity state is the position of the center of the paddle, which is a one-dimensional variable since the paddle only moves vertically. The geometrical structure of the application domain is shown in Figure 4.3(a). The possible entity states range from 40 to 200 pixels, and are sampled at a frequency of 100Hz, or a simulation step of $\delta = 10$ms. Figure 4.3(b) shows a typical trace of the paddle movement. The motion is highly structured and can be considered to consist of "awaiting phases" during which the paddle stops at a position expected on the trajectory of the on-coming ball, and the "chasing phases" when the user moves the paddle to the next expected ball position.

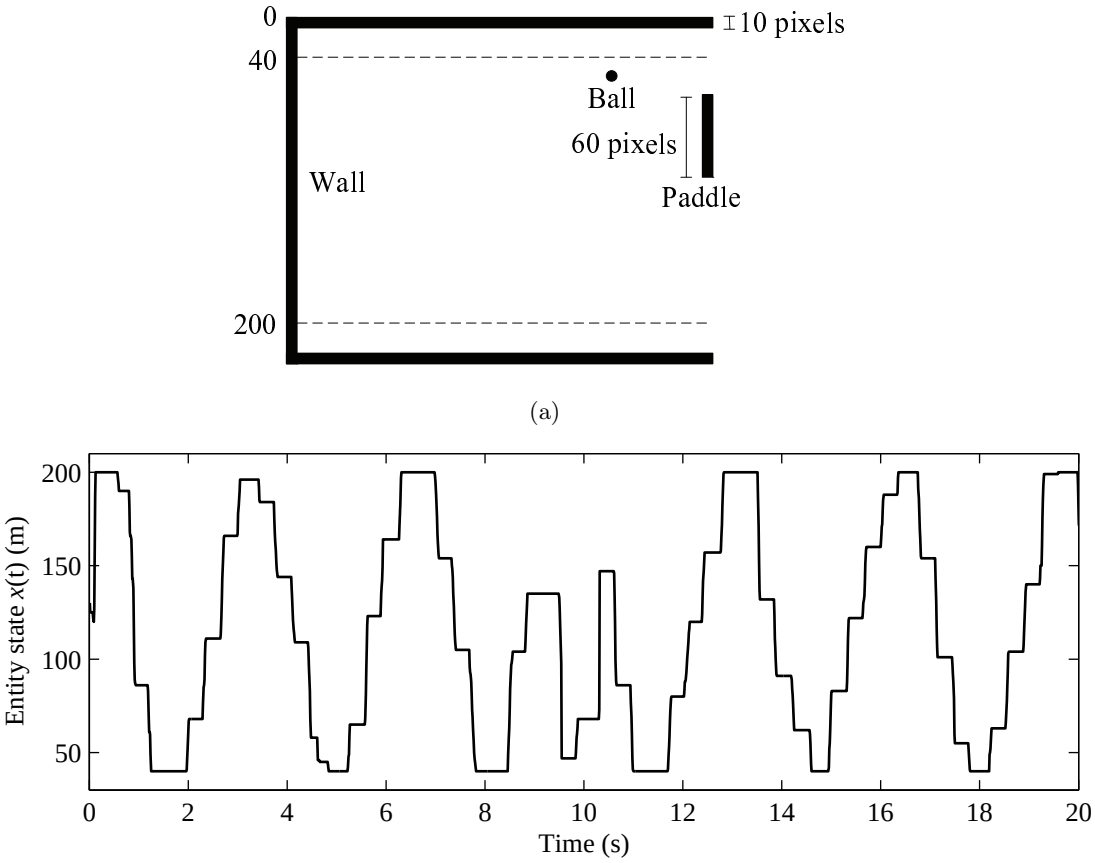


Figure 4.3 (a) The one-user Pong-like game scenario. The height of the paddle is 60 pixels, and the possible entity states range from 40 to 200 pixels. (b) An example of the pong game motion. The motion is highly structured visually and consists of a series of awaiting phases and chasing phases.

4.2.2.4 Navigation Motion

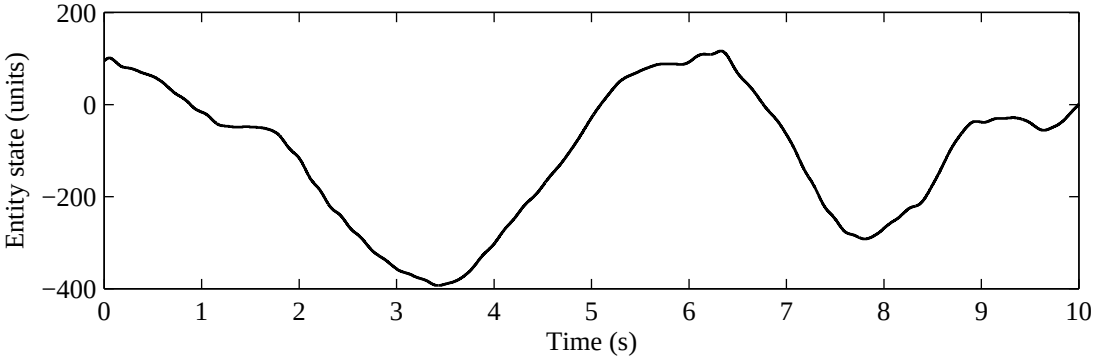
The second practical motion data used here is a head motion dataset from the Step World-In-Miniature (StepWIM) navigation environment, a hands-free multi-scale navigation in a semi-immersive virtual space [LaViola Jr. *et al.* 2001]. Figure 4.4(a) and (b) show the scenes of the navigation system. In the StepWIM, the 3D virtual environment is projected into a miniature version as a 2D road map placed beneath the human user's feet such that the actual position of the user in the virtual environment coincides with the approximate location of his/her feet in the miniature. The user can use the StepWIM to navigate to a specific place in the virtual environment

by simply walking to the desired location in the miniature. An animated view of the virtual environment is scaled up and shown to the user according to his/her location in the miniature.

A segment of the navigation motion is shown in Figure 4.4(c). The entity state is measured in application units. As in all the above sections, only one dimension coordinate is used as the entity state motion. The state is sampled at a frequency of 1607Hz, which is significantly higher than simulation rates in general DIAs. However, this high sampling rate bears no impact on the measurement of the information characteristics of the motion as the temporal dependence between entity states at two time instances is inherent to the entity dynamic and is independent of the intermediate samples.

(a)

(b)



(c)

Figure 4.4 Scenes in StepWIM system of (a) the miniature version of the virtual environment and (b) the scale-up virtual environment shown to the user [LaViola Jr. *et al.* 2001]. (c) An Example of the one-dimensional entity motion in the StepWIM navigation. The entity state is measured in application units and sampled at 1607Hz.

Compared to the pong game motion, the StepWIM navigation motion exhibits more smooth changes in the dynamic. Such a motion could be more predictable and embrace more temporal dependence in the time-evolution of the entity state, as the user's exploration of the virtual environment experiences little abrupt changes.

4.2.3 Results and Analysis

For the smooth and complex motions, the first and second-order derivatives, needed for the investigation of entity motion predictability, can be derived from their explicit motion models and directly recorded by the simulation process. However, for the pong game and StepWIM navigation motions, only positional state of the entity is recorded. Therefore the state derivatives in the two DR methods must be estimated from historical positions according to (2.3) and (2.4), respectively.

For ease of illustration, entity states in all the motions to be examined are presented in application units. The typical inconsistency measurement, *drift distance* D (as defined in (4.6)) is used as the spatial inconsistency measure.

$$D = \frac{1}{N} \sum_{k} |x(k) - \tilde{x}(k)| \,. \tag{4.6}$$

Details about the motion trajectories to be examined, including the simulation frequencies, sample sizes N, ranges of the entity state value combinations and the frequency of their occurrences in the dataset, are summarized in Table 4.2. To ensure sufficient sample sizes, a state sample occurrence statistics in the "OCC (95%)" row denotes that 95% of the possible combinations are sampled with occurrences higher than the shown value. More specifically, the entity in the smooth motion moves from -250 to 250 units, giving a range of 501 different state values, and 95% of these values are sampled at least 16 times in the simulation. In a DIA simulation, it is very likely that the entity passes by a particular position with different velocities, and Table 4.2 shows that there are 1000 different values for such a two-dimensional vector variable $[x, \dot{x}]$ in the smooth motion. Based on statistics in Table 4.2, the

datasets for the smooth, complex and pong game motions are considered sufficient for the simple algorithm to generate accurate information estimations since the majority of the state combinations are sampled at least 10 times (Section 3.2.2.1). However, some parameter combinations in the navigation dataset are only sampled for very low occurrences. Therefore the simple probability estimation cannot generate sufficiently accurate results. To measure the information characteristics of the navigation motion using the limited data, the Kernel Density Estimation algorithm introduced in Section 3.2.2.2 is employed.

4.2.3.1 Smooth Motion

Figure 4.5 and Figure 4.6 present the results of DR extrapolation and motion predictability measurement of the smooth motion, respectively. In Figure 4.5(a) and (b), and also the cases in which other motions are examined, the visual representations of the true and modeled motions under a local threshold of 16 units is shown to present typical situations under a moderate threshold value. Results for other threshold values are similar. Based on these results, observations and discussion

Table 4.2 Details about the motion trajectory data.

		x	$\dot{x}$	$\ddot{x}$	$(x, \dot{x})$	$(x, \dot{x}, \ddot{x})$
Smooth	Range	501	499	485	1000	1000
20Hz, N=16000	OCC (95%)	16	32	32	16	16
Complex	Range	1717	13	16	5712	6471
20Hz, N=100000	OCC (95%)	30	40	50	10	10
Pong	Range	161	15	16	853	1290
100Hz, N=114700	OCC (95%)	40	60	120	10	10
Navigation	Range	565	5	3	1020	1290
1607Hz, N=20695	OCC (95%)	8	539	3592	2	2

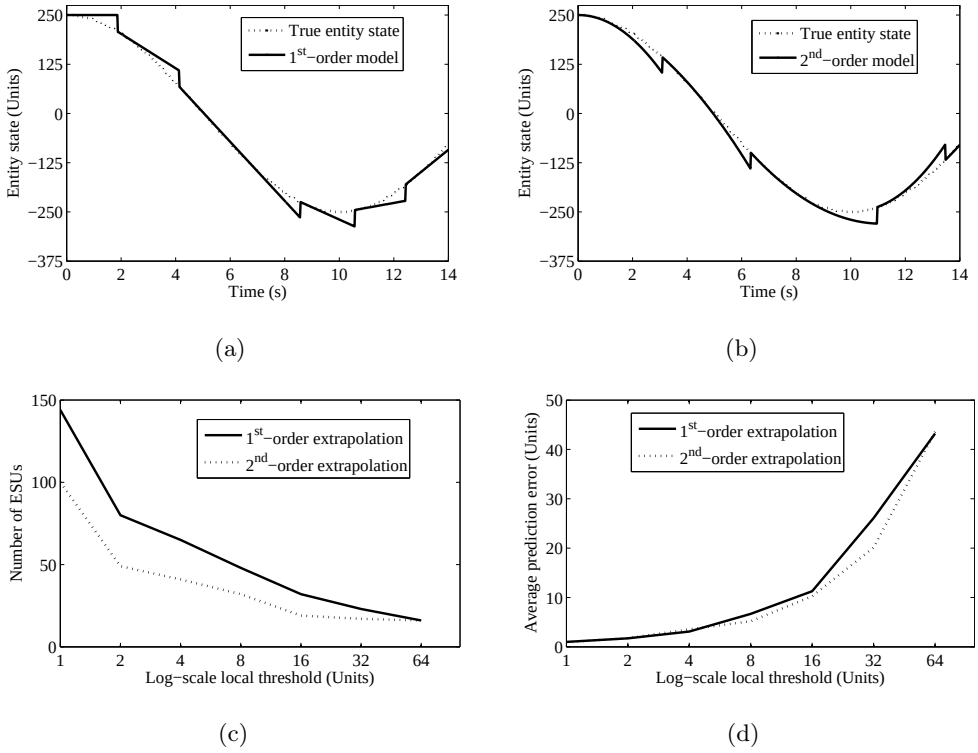


Figure 4.5 Results of extrapolating the smooth motion. Highlighted sections of the true motion and the local model extrapolated by (a) the first-order equation and (b) the second-order equation, at the local error threshold of 16 units. (c) The total number of ESUs generated by the two extrapolation models due to error threshold violations for increasing local error thresholds. (d) The accuracy metrics for the two PCMs for increasing local error thresholds. The logarithm of the increasing local error thresholds is used to highlight details at small values.

regarding the smooth motion are made as follows:

- It can be seen in Figure 4.5(a) and (b) that the second-order extrapolation model agrees more with the true entity motion in that it exhibits a curvature that complies with the smooth motion. This is especially obvious at the beginning of the simulation during the time 0–3s. The second-order derivative captured the trend of a decreasing velocity and the modeled motion consequently goes down with the true motion, while the first-order model goes along with the initial velocity and diverges from the true motion. The local error threshold is violated soon, generating the request of an ESU transmission.

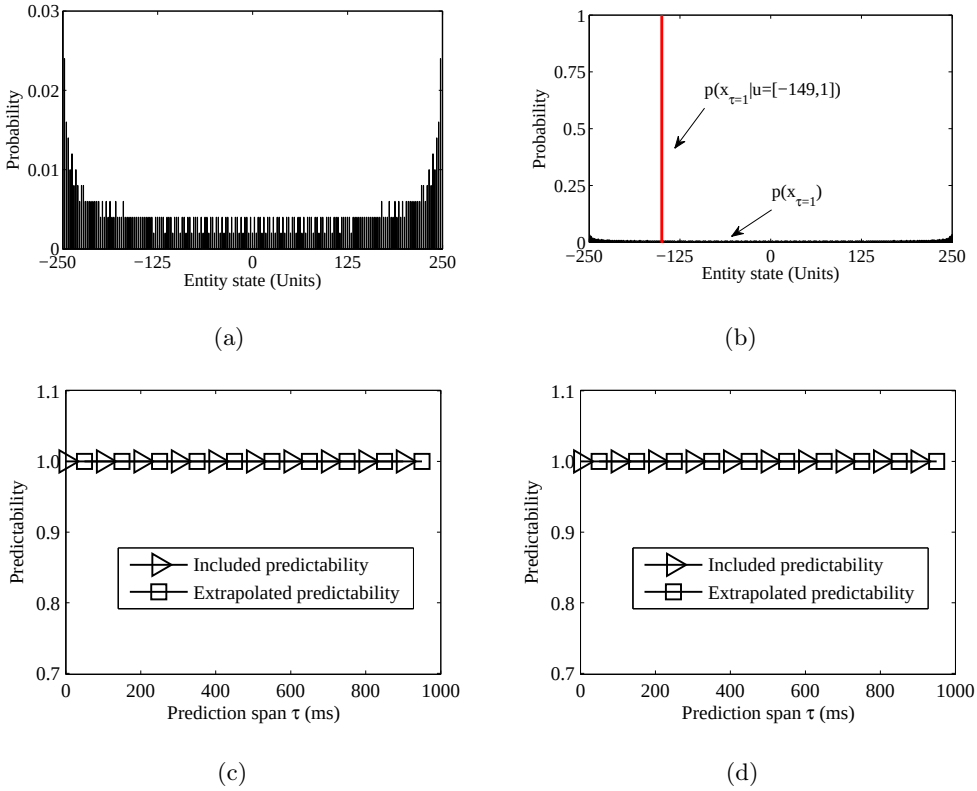


Figure 4.6 Results of measuring information characteristics of the smooth motion. (a) Probability distribution of the entity state. (b) Conditional probability distribution $p(x_{\tau=1}|\boldsymbol{u} = [-149, 1])$ and unconditional probability $p(x_{\tau=1})$. The amount of information encapsulated in and extrapolated from (c) a first-order ESU and (d) a second-order ESU for increasing prediction spans.

- The less jerky motion modeled by the second-order extrapolation is consistent with the results of ESU generations shown in Figure 4.5(c) and the inconsistency results shown in Figure 4.5(d). The second-order model maintains better locally estimated spatial consistency (with the exception of marginally larger inconsistency when the threshold is very tight) with less number of ESUs. From this perspective, the second-order extrapolation model is to be preferred over the first-order model.

- Statistics in Table 4.2 reveals the deterministic nature of the smooth motion. The sampled two-dimensional vector variable $(x, \dot{x})$ is a small subset of the whole possible space that scales up to 501×499. This indicates very high dependence between the two variables. Notice that including an additional

acceleration parameter does not expand the range of possible ESU values for second-order extrapolation. The acceleration value can be determined by the instantaneous position and velocity of the entity. Therefore the acceleration value is redundant for the smooth motion from the information perspective.

- The entity state in the smooth motion is distributed rather evenly, like a uniform distribution (Figure 4.6(a)). The entity state of the smooth motion appears slightly more often at the two ends of the possible state set, because the entity moves with very low speed near the peaks and valleys as illustrated in Figure 4.1(a). This broad distribution suggests that it is difficult to predict the entity state in a smooth motion without prior knowledge (such as knowing its deterministic nature and the motion model). The entropy $H(x) = 7.385$ bits indicates that the shortest average length of data required to describe the smooth motion state at each simulation step with absolute accuracy is 7.385 bits. Again, this required data rate does not consider any useful temporal dependence that could be extrapolated to speculate the future entity state.

- Figure 4.6(b) exemplifies how the information in an ESU is used to reduce the uncertainty of a future state. By receiving a first-order DR ESU $\boldsymbol{u} = [x = -149, v = 1])$, the uniform distribution $p(x_{\tau=1})$ of the entity state $x_{\tau=1}$ becomes a conditional probability $(p(x_{\tau=1}|\boldsymbol{u} = [x = -149, v = 1]))$. In this extreme case, the next entity state is "pinned" to one value by the information in the ESU because the smooth motion is deterministic. Generally, the knowledge in an ESU rules out a vast number of state values and restrains the value of the next state to a few possibilities.

- In Figure 4.6(c) and (d), the two ESUs both facilitate perfect predictability not only for the current entity state but also for all that follow at increasing prediction spans. This is not surprising considering the deterministic nature of the smooth motion. It can be easily derived from the smooth motion model in (4.3) that given the position and velocity of the entity, the phase ωt of the circular motion is determined, and so is the subsequent movement of the object. From this point of view, the instantaneous position and velocity in the first-order ESU are sufficient to store all the information needed to extrapolate

the entity motion precisely, and the acceleration parameter in the second-order ESU is redundant for extrapolating future entity state. The perfect information storage in Figure 4.6(c) and (d) is typical for all deterministic motions as long as the necessary parameters are included in the ESU.

- It is not expected, though, that both the first and second-order extrapolation models can make use of all the available information in building the entity model, as shown in Figure 4.6(c) and (d). At first glance, it might seem in conflict with the fact that neither of the two models is capable of maintaining absolute consistency under a non-zero local error threshold (Figure 4.5), while the perfect information encapsulation and utilization of the two extrapolation models suggest otherwise. This apparent contradiction originates from the philosophy of the information metric to measure the relationship between the true and modeled entity dynamics based on uncertainty reduction, instead of spatial similarity. The situation is explained using a simple example in Figure 4.7, which displays a full dataset of 5 simulation steps for a trivial simulation used for illustrative purposes. Although the initial extrapolation

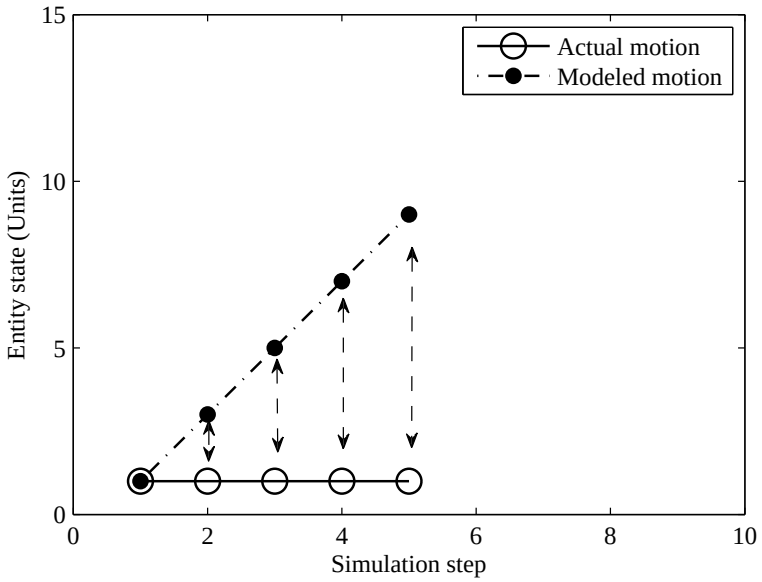


Figure 4.7 A simple illustration of the difference between mutual information and spatial inconsistency. Though spatially diverged, the modeled path still has full information about the true entity motion because of the absolute certainty between the two motions.

vector is not perfect and the modeled path diverges from the actual motion, the mutual information is still perfect, as in the situation with the smooth motion, because the actual motion can be determined with absolute certainty from the modeled path based on the one-on-one relationship between the two motions. Therefore the modeled path has full information about the true motion, but it is not perceptually presented in the best way. In the case of the smooth motion, the second-order extrapolation model exhibits better consistency than the first-order model because the second-order derivative, even if it brings no more information, helps interpret the extrapolated information in a visually more appropriate form. The contradiction between the information metric and the spatial measurement suggests that given an appropriate transformation scheme, the first and second-order extrapolation models are capable of providing the same highly accurate entity state model of the deterministic smooth (or other) motion.

4.2.3.2 Complex Motion

Based on the results of DR extrapolation in Figure 4.8 and motion predictability measurement in Figure 4.9, a number of observations and discussion regarding the complex motion can be made:

- As with the smooth motion, the second-order model outperforms the first-order model by generating significantly less ESUs (Figure 4.8(c)) and maintaining a lower inconsistency (Figure 4.8(d)). The curvature the second-order extrapolation exhibits complies with the complex motion (obvious examples can be identified during the time 90 – 100s and 115 – 120s in Figure 4.8(a) and (b)), giving a perceptually more favorable entity state model. From this perspective, the second-order extrapolation model, again, is to be preferred over the first-order model, which is not unexpected given that the complex motion is a random combination of motion types that are derived from the smooth motion. Therefore the complex motion shares the common preference towards second-order DR as the smooth motion.

- A number of "peak states" with relatively higher probabilities can be found in Figure 4.9(a). Such peak states indicate the places where the entity moves slowly, or changes direction of movement. Therefore, the entity state in the complex motion is unevenly distributed, compared to the smooth motion. Without any prior knowledge of the motion (such as previous motion status), such a probability distribution means that the entity state could be predicted more easily since the peak values are more probable than others. However, the entity state is more broadly distributed since the entity movement is randomly assigned to one of the three motion models and is therefore no longer periodic.

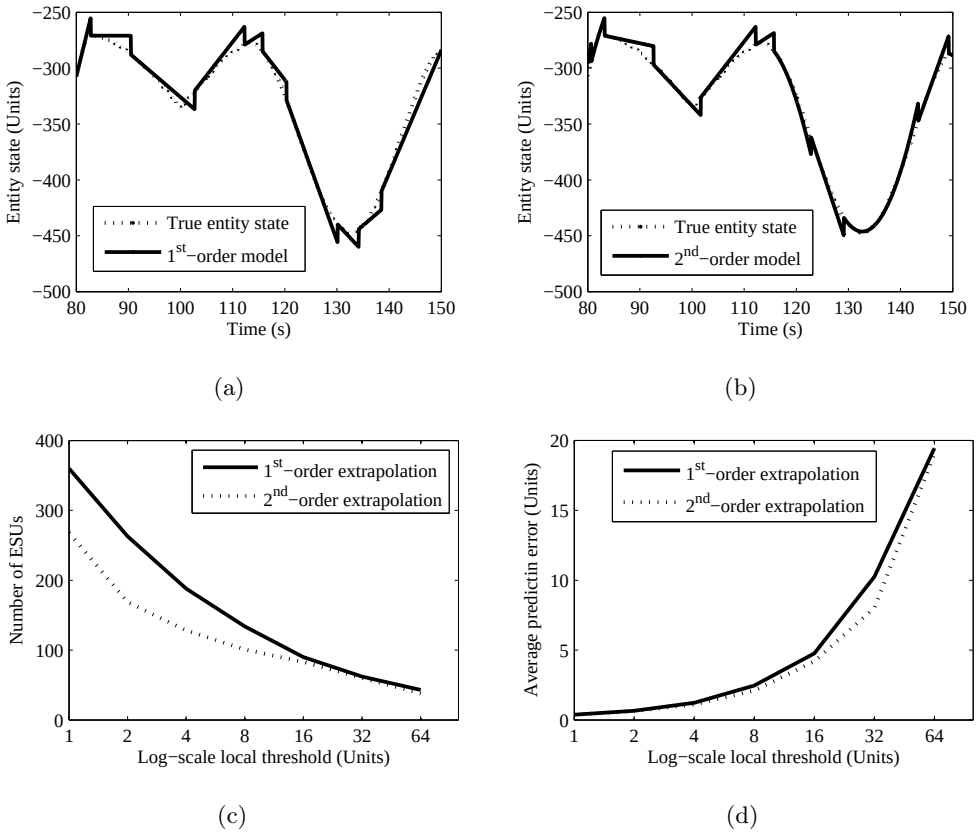


Figure 4.8 Results of extrapolating the complex motion. Highlighted sections of the true motion and the local model extrapolated by (a) the first-order equation and (b) the second-order equation, at the local error threshold of 16 units. (c) The total number of ESUs generated by the two extrapolation models due to error threshold violations for increasing local error thresholds. (d) The accuracy metrics for the two PCMs for increasing local error thresholds. The logarithm of the increasing local error thresholds is used to highlight details at small values.

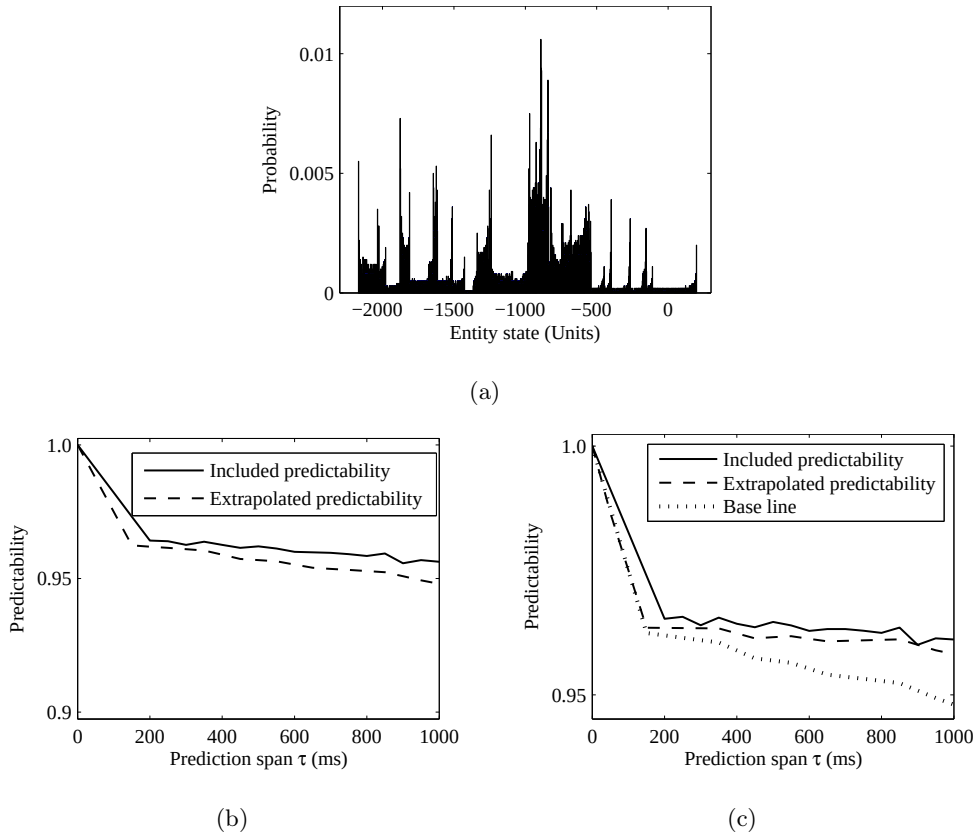


Figure 4.9 Results of measuring information characteristics of the complex motion. (a) Probability distribution of the true motion, giving an entropy $H(x) = 10.47$ bits. The amount of information encapsulated in and extrapolated from (b) a first-order ESU and (c) a second-order ESU for increasing prediction spans. The extrapolated predictability in a first-order ESU is re-drawn as a base line for comparison in (c).

The entity could go through a quite large space over time. The wide range of possible states results in low values of the probabilities, even for those peak states, and a higher entropy $H(x) = 10.47$ bits.

- Unlike the situation in the smooth motion, neither of the two ESUs could preserve perfect predictability for increasing prediction spans (Figure 4.9(b) and (c)), because of the randomness introduced in the complex motion model.

  Firstly, this incomplete information implies that based on the parameters in the ESUs, it is impossible to predict the future entity state with absolute accuracy under a non-zero local error threshold, regardless of any form of prediction scheme used.

Secondly, as the introduced randomness is purely "external" (governed by a random number generation by the simulation process rather than the entity behavior), any set of prediction parameters based on historical motion status is incapable of providing sufficient information for building an absolute accurate entity state model.

Finally, the available predictability in an ESU decays with increasing prediction spans rather slowly, because the entity motion is still deterministic within each interval between the random motion type switches. The temporal dependence between the current motion status and future states is still strong, if not perfect, as most of the information is encapsulated in the ESU. Although the complex motion only introduces limited uncertainty to the motion, the decaying available information characterizes non-deterministic entity motions that make state update transmissions necessary to maintain consistency in DIAs.

- The second-order ESU exhibits no significant advantage over the first-order ESU in conveying available information. As discussed in the smooth motion case, within each deterministic motion interval, the second-order ESU includes a redundant parameter and is just as good as the first-order ESU in storing available information. The random motion type switches, on the other hand, is independent of the entity motion status. Thus the second-order ESU conveys no additional information about the random element of the motion. This invariant information content in the two type of ESUs is mostly due to the external randomness introduced by the particular complex motion model.

- Due to the randomness introduced in the complex motion, the available predictability in the ESUs is not fully utilized by the extrapolation models when building the entity state model (Figure 4.9(b) and (c), dashed curves). It is apparent that the perfect mapping illustrated in Figure 4.7 is compromised. However, the two extrapolation models utilize most of the available information as a result of the piecewise-deterministic nature of the entity dynamic. It is also worth noting that the second-order model extrapolates more information, though not much more, than the first-order model (the extrapolated predictability in a first-order ESU is drawn as a base line for comparison, and

the ordinate in this figure is reduced to the range of interest). The extrapolated entity motion modeled by the second-order equation reflects the highly curved feature of the complex motion, which agrees with the results presented in Figure 4.8.

Due to the similarity between the motion elements that compose the complex motion and the smooth motion, the information measurement of the complex motion exhibits similar characteristics to the smooth motion, such as high efficiency in conveying and utilizing entity motion predictability. From the information perspective, the updated parameters in the two types of ESUs manifest potential for better spatial consistency. The available information in an ESU decays with increasing prediction spans because of the randomness introduced by the motion type switches in the entity dynamics. But this randomness is governed by an external factor, namely the random number generation that controls the motion parameters for each section of the motion. This controlled uncertainty differentiates the modeled motions from the motion data collected from practical DIA applications, where the uncertainty in the time-evolution of the entity state is largely driven by the inherent factor of user behavioral patterns. The predictability in entity motions that embrace the *human-in-the-loop* factor is investigated in the following sections by implementing the information model on the collected entity dynamics.

4.2.3.3 Pong Game Motion

The following observations and discussion regarding the pong game motion can be made based on the results of DR extrapolation in Figure 4.10 and motion predictability measurement in Figure 4.11:

- Unlike the situation with the modeled motions, where the second-order model is favorable for extrapolation, Figure 4.10(c) and (d) show that the first-order equation maintains comparable consistency to that afforded by the second-order model, but with a significantly reduced number of entity state updates.

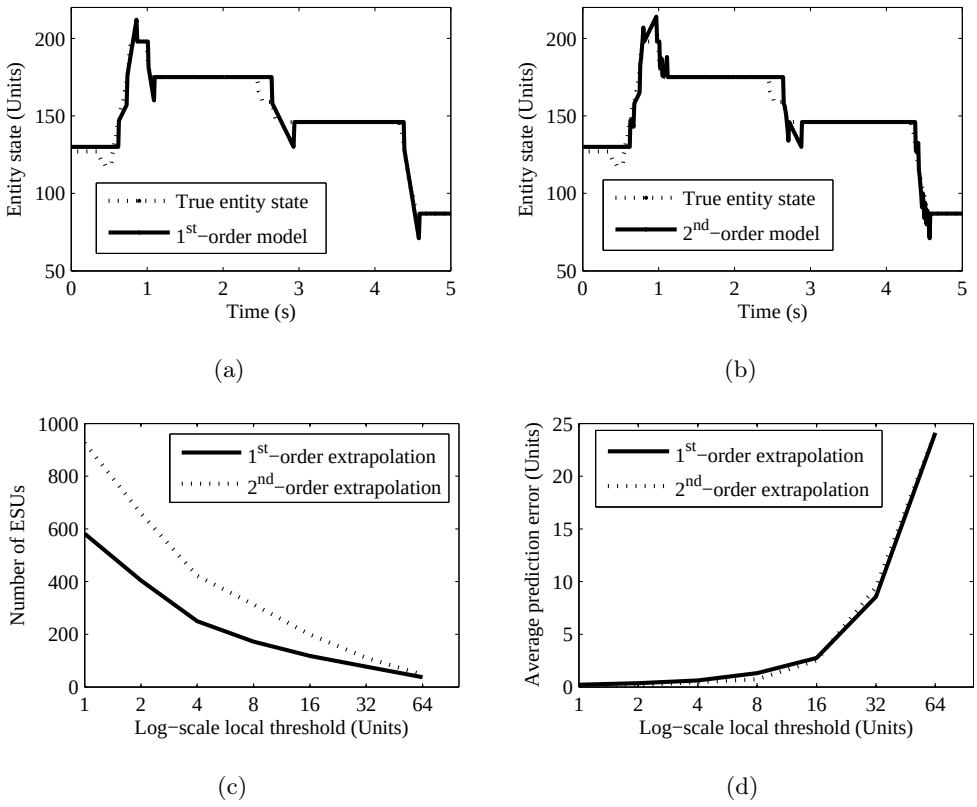


Figure 4.10 Results of extrapolating the pong game motion. Highlighted sections of the true motion and the local model extrapolated by (a) the first-order equation and (b) the second-order equation, at the local error threshold of 16 units. (c) The total number of ESUs generated by the two extrapolation models due to error threshold violations for increasing local error thresholds. (d) The accuracy metrics for the two PCMs for increasing local error thresholds. The logarithm of the increasing local error thresholds is used to highlight details at small threshold values.

This advantage is especially evident at smaller threshold values, where the average prediction error of the first-order model is even lower.

- Further scrutiny of Figure 4.10(a) and (b) reveals that the advantage of the first-order model over the second-order equation is due to the particular pattern of the user behavior in this case.

  It seems that the entity (namely the paddle in this game scenario) remains occasionally stationary at the positions where the moving ball is expected to hit the paddle and bounce back. When the object is about to enter this "static phase", speed decreases for a short period. This decreasing speed causes a

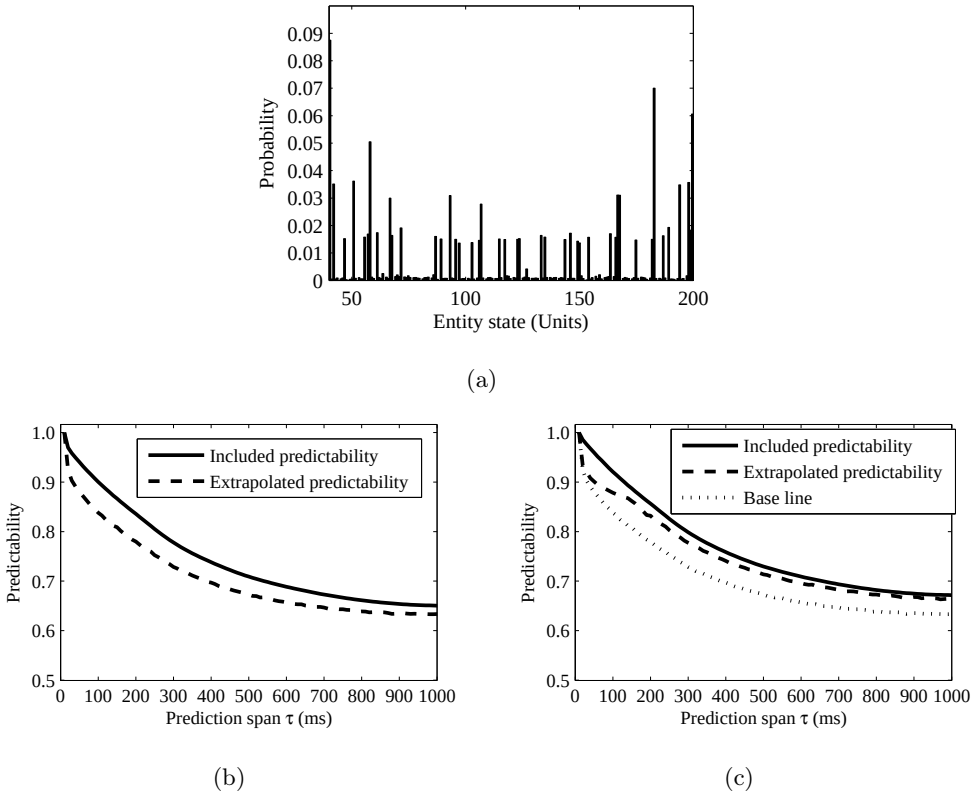


Figure 4.11 Results of measuring information characteristics of the pong game motion. (a) Probability distribution of the true motion, giving an entropy $H(x) = 5.80$ bits. The amount of information encapsulated in and extrapolated from (b) a first-order ESU and (c) a second-order ESU for increasing prediction spans. The extrapolated predictability in a first-order ESU is re-drawn as a base line for comparison in (c).

state update as a result of a threshold violation as the modeled path diverges from the true motion. In the first-order extrapolation, this update conveying a zero velocity state update indicates and directs first-order motion model into the static phase. However, in the second-order model, where the acceleration parameter adds additional inertia, the modeled path would diverge faster from the true motion and the ESU generation would in many cases be triggered before the entity becomes static. This ESU with non-zero velocity and acceleration consequently leads to another error threshold violation during the static phase, after which the second-order motion model converges to the true motion. The extra spikes of the second-order extrapolation near the start of the static phases is obvious around 1s and 3s in Figure 4.10(b) and

account for most of the additional ESUs of the second-order model shown in Figure 4.10(c). It is unlikely to find extra spikes when the paddle starts moving after hitting the ball because the entity motion status is well synchronized during the static phases.

- It is very clear in Figure 4.11(a) that the entity motion is a combination of static phases with significantly higher probability state values and moving phases with low probability states. Unlike the modeled motions, the static phases are distributed rather evenly through the entity space. Such highly structured movement is expected to be regular and predictable. The entropy of the entity state is 5.80 bits.

- In contrast to the situation for the modeled motions, the amount of predictability provided by the ESUs decreases rather fast as the prediction span increases (Figure 4.11(b) and (c)): about 40% of the full information is lost during the interval of 1000ms. Compared to the available information in a first-order ESU, the second-order ESU includes more information by taking an additional state derivative (information cannot hurt), even though the extra parameter does not make significant difference because of the simple and inaccurate estimation of the second-order derivative and the particular user behavior pattern as explained above.

- As can be expected, the available predictability in the ESUs is not fully utilized by the extrapolation models. Nevertheless, the first-order extrapolation model utilized 90% of the information provided (Figure 4.11(b)), since the entity, for most of the time, either stays still, awaiting the approaching ball, or moves smoothly towards the next expected contact position.

- The predictability extrapolated by the second-order equation exhibits more interesting behavior (Figure 4.11(c)). During the static or transition stages in the entity dynamic, the acceleration parameter of the entity is mostly at very small values such that it takes time to manifest its effect on the entity movement. Therefore, the extrapolated information is at first just as good as the information extrapolated by the first-order equation (the extrapolated

predictability in a first-order ESU is drawn as a base line for comparison). However, when the prediction span grows, the second-order extrapolation exhibits significant advantage over the first-order extrapolation in utilizing predictability.

Although the second-order equation does not translate the extra information into better perceptual fidelity (Figure 4.10(d)), which is explained in the additional spikes situation, the information advantage suggests that a better, and possibly more complicated, extrapolation scheme could present an improved consistency result. For example, using the acceleration parameter, an extrapolation scheme could detect the decreasing trend in entity speed and recognize the coming static phases. The spikes before the static phases could then be avoided by converging the state model to the static motion directly, instead of extrapolating the state model using the second-order equation all the time.

4.2.3.4 Navigation Motion

The results of DR extrapolation and motion predictability measurement for the navigation motion are presented in Figure 4.12 and Figure 4.13, respectively. A number of observations and discussion are made as follows:

- For most part of the true and modeled trajectories shown in Figure 4.12(a) and (b), the two DR models exhibit similar visual extrapolation performance. One exception is that in the period of 9 – 10s, the second-order model is impacted by the transient fluctuation in the entity motion, and produces a jerky path due to the sensitivity of the acceleration parameter. Such a fluctuation is filtered or ignored by the first-order model by giving an averaged but less dynamical prediction. In Figure 4.12(c) and (d), the second-order extrapolation generates a few more ESUs and maintains a slightly better consistency than the first-order extrapolation. The difference between the two extrapolation models for this navigation motion is not significant.
- In Figure 4.13(a), the probability of the entity states is distributed rather

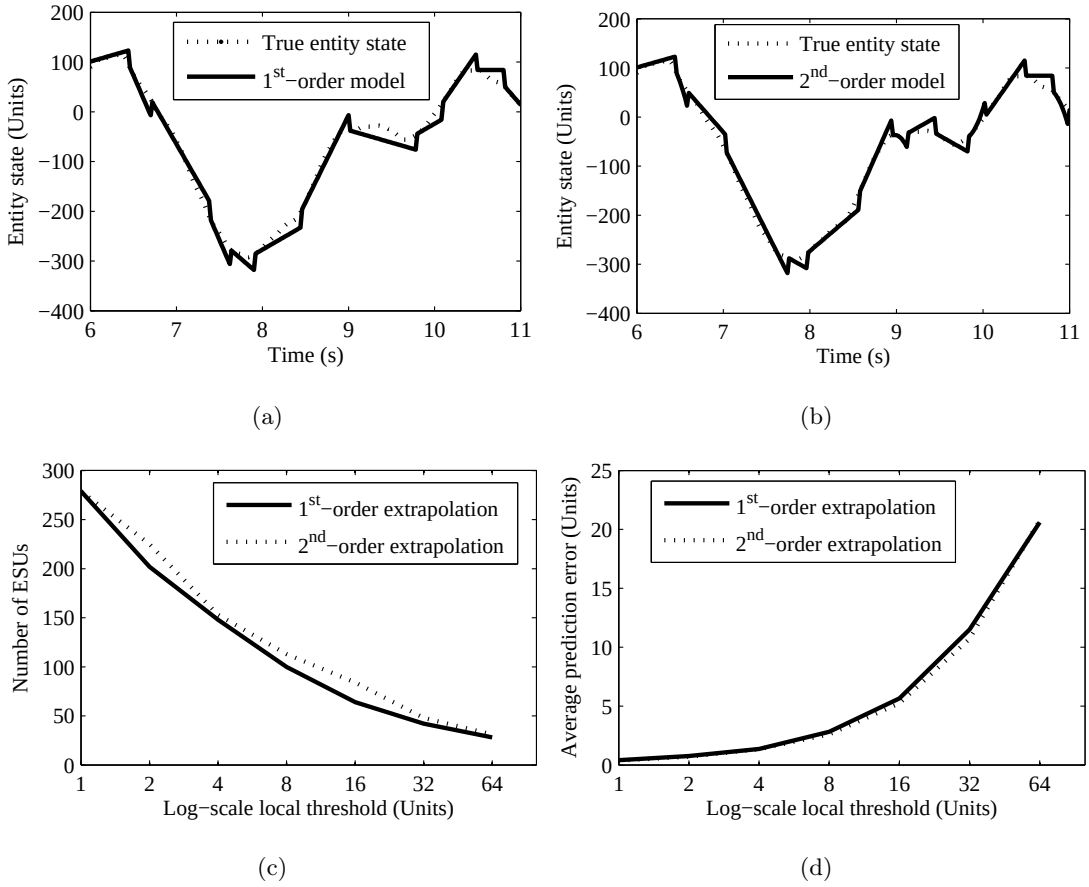
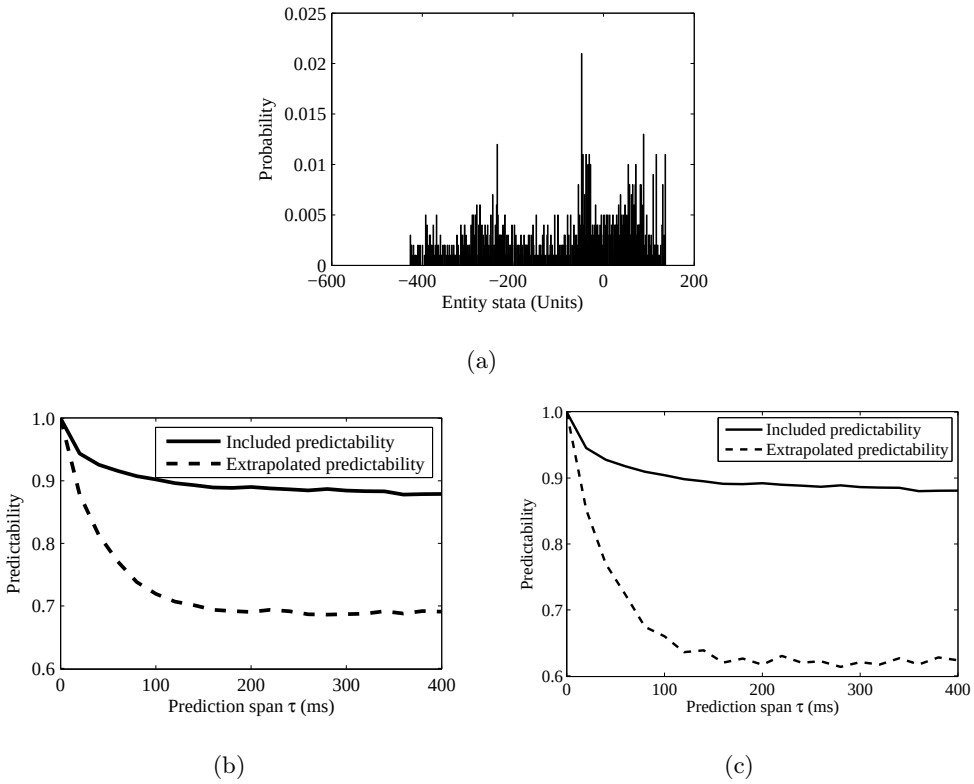


Figure 4.12 Results of extrapolating the StepWIM navigation motion. Highlighted sections of the true motion and the local model extrapolated by (a) the first-order equation and (b) the second-order equation, at the local error threshold of 16 units. (c) The total number of ESUs generated by the two extrapolation models due to error threshold violations for increasing local error thresholds. (d) The accuracy metrics for the two PCMs for increasing local error thresholds. The logarithm of the increasing local error thresholds is used to highlight details at small threshold values.

evenly. Most of the state values only appear with probabilities under 1%. This probability observation is consistent with the navigation behavior where the user would typically explore an unfamiliar virtual space with steady movement. The entropy of the entity state is 7.61 bits.

- It is obvious by comparing Figure 4.13(b) and (c) that the two types of ESUs provide almost the same information about the future entity state. In



(a)

(b)

(c)

Figure 4.13 Results of measuring information characteristics of the StepWIM navigation motion. (a) Probability distribution of the true motion, giving an entropy $H(x) = 7.61$ bits. The amount of information encapsulated in and extrapolated from (b) a first-order ESU and (c) a second-order ESU for increasing prediction spans.

Table 4.2, there are only three possible velocity and acceleration values sampled in the dataset, which indicates that the object is exploring a rather limited space with slow and stable velocities. The navigation motion is so smooth that the additional acceleration parameter in the second-order extrapolation model appears to be irrelevant to the entity motion. Such redundant information is misinterpreted and leads to lower information utilization values for the second-order extrapolation model. This observation agrees with the spatial inconsistency results in Figure 4.12. Therefore, the first-order extrapolation model is preferred for the navigation motion for presenting at least equal ESU generation and consistency performance without transmitting any redundant information.

- The steady exploration behavior is rather regular compared to the pong game

motion in that about 90% of the full information is preserved by the updated parameters over a prediction span of 400ms. This indicates that the navigation motion itself is potentially highly predictable. However, only 60% of the available information is extrapolated by the prediction models, which indicates that it is a better extrapolation scheme that is necessary to improve the performance of PCMs for this particular motion, rather than more data.

Throughout the discussion of the results from measuring the predictability of various entity dynamics using the information metrics, the following general observations can be made:

1. The entropy of an entity motion denotes the minimal amount of data in bits required to describe the entity state at each time instance. The entropy value depends on a number of factors such as the scale of the virtual space and the user movement. A high entropy alone does not indicate a complicated movement as information from contextual dynamics could be explored by extrapolation methods to effectively reduce uncertainty of the future entity state so that the entity state can be shared with reduced data transmission.

2. Entropy measures the average uncertainty of a single entity state without consideration of its relationship with preceding entity dynamics (see the introduction of $H(x)$ in (3.19)). For example, the state uncertainty of 7.385 bits for the smooth motion could be completely removed because the future entity state could be determined with absolute certainty based on the full information, if properly interpreted, in the first and second-order ESUs. The predictability of the entity motion, from the predictive perspective, is the reduced uncertainty about the future entity state after the information in the contextual dynamics is taken into consideration.

3. The predictability of an entity motion can be measured by the mutual information between the currently known motion status and the future entity dynamics for increasing prediction spans. The amount of information available in the update data consequently depends on the selection of parameters,

i.e. entity position, velocity, etc. In general, a larger set of update parameters could always improve the information capacity.

4. The available predictability encapsulated by a specific set of motion parameters is characteristic of the entity dynamic itself. The performance of a PCM depends on how much information can be utilized by the extrapolation model under use to build the entity state model. A bad extrapolation model, even if provided with more information, could mis-interpret the available information in the entity state updates and lead to poor information delivery performance and inadequate consistency.

5. For deterministic motions, perfect information could typically be conveyed by appropriate update parameters and extrapolation models. However, a high extrapolated predictability figure does not always guarantee a modeled path with high fidelity, because the information measurement focuses on certainty relationships between two variables instead of spatial similarity. In the case of a higher information and low fidelity, a transformation of the modeled path can always be found to re-interpret the modeled entity motion with high perceptual accuracy (Figure 4.7).

6. For non-deterministic motions, the available predictability provided by the updated parameter based on the current motion status decays with increasing prediction spans. More complicated motions exhibit rapid information drop as the temporal dependence among contextual dynamics fades quickly with time. The extrapolated predictability is bounded by the available predictability, and whether or not additional parameters can improve the information utilization depends very much on how much the user behavior and the motion pattern implied by the extrapolation model comply with each other.

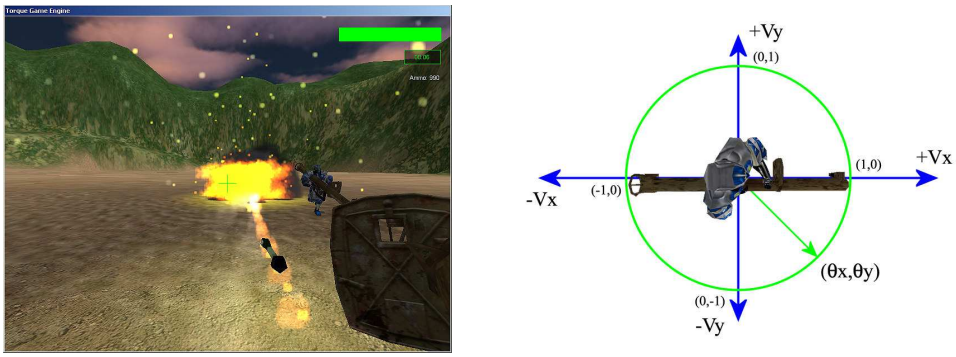
The results presented above measure the predictability of the entity dynamics. In the next section, the impact of PCM operation and network latency on utilizing such predictability to build the remote entity model is examined through a complete "life cycle" information analysis of the ESUs.

4.3 Comparison of PCMs using the Information Model

In this section, to show how the predictability in user behavior is locally encapsulated, transmitted and remotely extrapolated by PCMs to maintain a sufficient level of consistency with reduced network traffic, the information model proposed in Chapter 3 is implemented for a motion dataset collected from a realistic FPS (First Person Shooting) game application. The inconsistency caused by discarding prediction errors within the threshold limit is measured as the information loss, and the effect of latency on the remote inconsistency is also measured from an information perspective. Analysis of the information model is presented through a comparison of the standard polynomial DR algorithms and the Neuro-reckoning technique (Section 2.4.5), demonstrating that the information approach applies to both standard and novel PCMs.

4.3.1 Overview

To demonstrate the information model, experiments are conducted under a representative DIA environment based on the Torque Game Engine (TGE) [Lloyd 2004], a commercial game engine that enables the development of a real-time, interactive,



(a) (b)

Figure 4.14 (a) A screenshot of the FPS game scenario. (b) The entity state space of the DIA.

continuous, concurrent distributed application. As such, the TGE-based game application is a very capable representation for the many types of real-world DIAs characterized by similar attributes, and the data collected can be considered representative of what one would expect to observe in a commercial networked multiplayer computer game or some similar applications. The simulation scenario, developed by McCoy *et al.* [2004], is shown in Figure 4.14(a). In the simulation, players who control the avatar using a keyboard (for translation movements) and a mouse (for rotations) in first-person perspective are "born" at one of a collection of randomly assigned "spawn-points" as starting locations. The goal of the players in the simulation is to be the first to reach a pre-specified "hit-point" score-limit by attacking others using projectile-based weapon with unlimited ammo, and "disabling" the opponents. A health-meter of each entity, which quantifies the amount of damage left before being disabled, is viewable to the controlling player in real-time. Disabled players are reborn at one of the spawn-points and re-join the game with full health-meters. This "deathmatch" scenario involves typical user behaviors in general DIAs, such as exploration, chasing, circumvention and engagement. Such a DIA is now analyzed from the information perspective.

Figure 4.14(b) illustrates the entity state space in the simulation. The entity controlled by a human participant during the game play is allowed to move in a 3D virtual space. However, it is observed that the z-coordinate of the entities almost remain invariant throughout the simulation (as they rarely jump), and is removed from the data to be examined. 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 two-dimensional vector. The entity position and velocity are coordinates and speed along the x-axis and y-axis respectively, in terms of game units. The orientation, which describes the facing direction of the player, is the sine and cosine coordinate pair of the positive angle from the x-axis to the line of the orientation. Again, as throughout this thesis, the presented results are based on the x-coordinate of the entity position for the convenience of illustration. In the standard polynomial DR techniques, it is straightforward to treat the entity motion as a one-dimensional movement as the prediction is conducted within each dimension separately. In neuro-reckoning, on

the other hand, the prediction of the x-coordinate of an entity state also involves overall state values from the other dimension as it is considered to affect the x-coordinate dynamic. Therefore it should be noted, as shown in (2.10)–(2.13), that the neuro-reckoning uses information from two-dimensional states to extrapolate the one-dimensional motion examined in this section. The measurement presented here can be easily generalized, at the cost of increased computation, to multi-dimensional scenarios by treating the states as vector variables. The game is simulated at the rate of 20Hz, with the constant simulation step of $\delta = 50$ms.

The PCMs examined here are standard DR with the first and second-order predictors, and neuro-reckoning. For the first-order predictor, the x-component of the entity velocity vector is directly taken as the first-order derivative. The second-order derivative in the second-order extrapolation, which it is not recorded during the simulation, is estimated as the velocity change during two consecutive steps (as in (2.4)). NR uses 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 as 3 and 10 simulation steps respectively to give optimized 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 first-order DR, the x-axis component $\hat{v}_x(k)$ of the NR velocity $\hat{\boldsymbol{v}}(k)$ (see (2.13)) is used in extrapolating the one-dimensional entity dynamic. The derivatives contained in an entity state packet generated at simulation step k are summarized in Table 4.3.

Table 4.3 Derivative information in an update packet

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

In the experiment, the human-users are asked to repeatedly play the game for about 85 minutes, and the entity state trajectories of the human-users are then combined to form a true entity motion with the length of $N = 102053$ steps. The DR and NR algorithms are then ran on this true entity motion with varying local thresholds and simulated fixed latencies.

Details of the recorded dataset in terms of the number of possible values and occurrences of entity states, derivatives, and their combinations in each type of packet are summarized in Table 4.4. It is then clear that parameters in an NR ESU provide significantly larger possible value space and potentially higher capacity for carrying information about the entity behavior. However, such a large sample space makes the available entity trajectory insufficient for the simple algorithm to generate decent probability estimation. Therefore, KDE estimation algorithm is used later in the information measurement.

4.3.2 Packet Generation and Spatial Inconsistency Results

Firstly, the performance comparison of the three reckoning techniques based on traditional metrics, namely packet generation and spatial inconsistency, is presented. Table 4.5 shows the number of packets generated by each of the PCM schemes, under a series of increasing local error thresholds measured in terms of games units. To put the threshold values into some perspective, the size of the entity avatar is approximately 5.7 units. The lowest simulated error threshold of 2 units represents a fairly tight threshold, about one third of the entity size. In contrast, the largest simulated error threshold of 64 units represents a very relaxed threshold. Network

Table 4.4 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

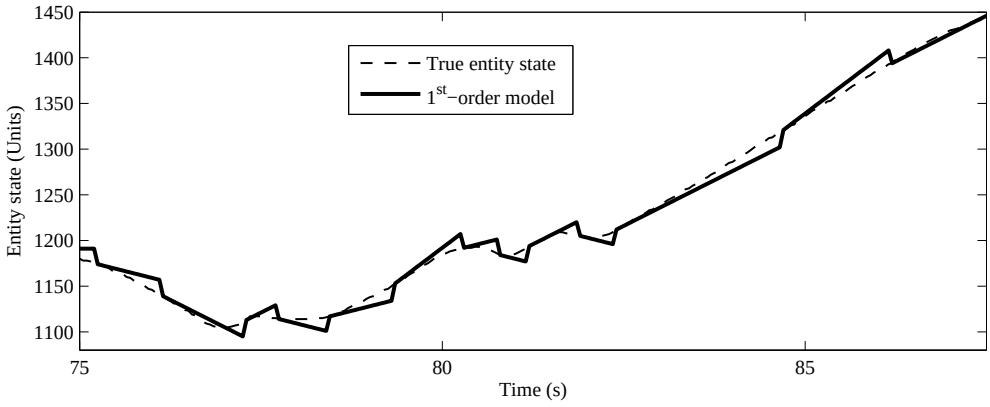
Table 4.5 Number of packets generated by each of the three PCMs

Threshold (units)	1^{st}-order DR	2^{nd}-order DR	NR	NR reduction (%)
2	11799	12312	10431	13.11
4	7586	7929	6507	16.58
8	4943	5173	4358	13.42
16	3165	3307	2962	6.82
32	2035	2108	1948	4.47
64	1222	1270	1191	2.60

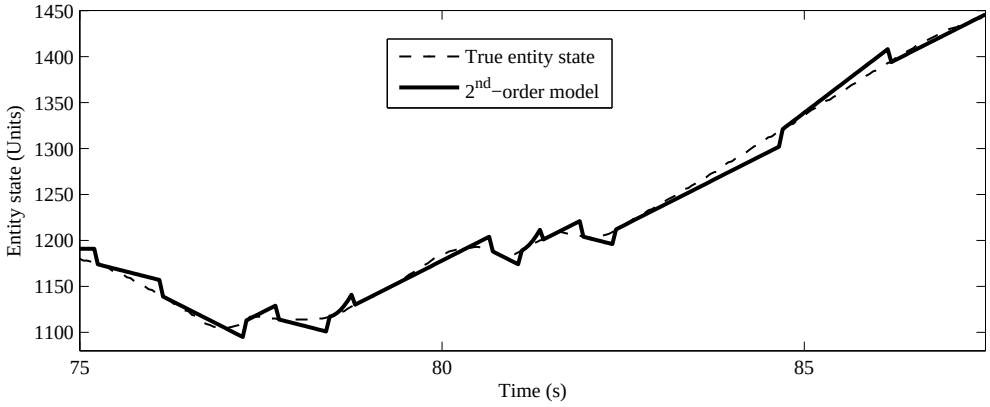
conditions are simulated by increasing fixed latencies. In the last column of the table, the heading "NR reduction" refers to the percentage reduction in the number of entity state update packets generated against either the first or second-order DR, whichever generates less packets for that particular error threshold. A positive values indicates a superior performance by the NR technique.

Across thresholds, NR outperforms the two standard DR, giving packet generation reductions ranging from 2.6% up to over 16% against the best performing standard DR model. This is more apparent from inspection of Figure 4.15 which provides a comparison of the locally reconstructed state trajectories by the three reckoning techniques. It can be observed around the time of 85s how NR compensated for expected changes in the future entity trajectory and avoided unnecessary ESUs. From the visual perspective presented in Figure 4.15, especially at 77 – 80s, NR "cuts through" corners by extrapolating towards a future estimated point on the entity's trajectory and produces an averaged state model [McCoy 2007].

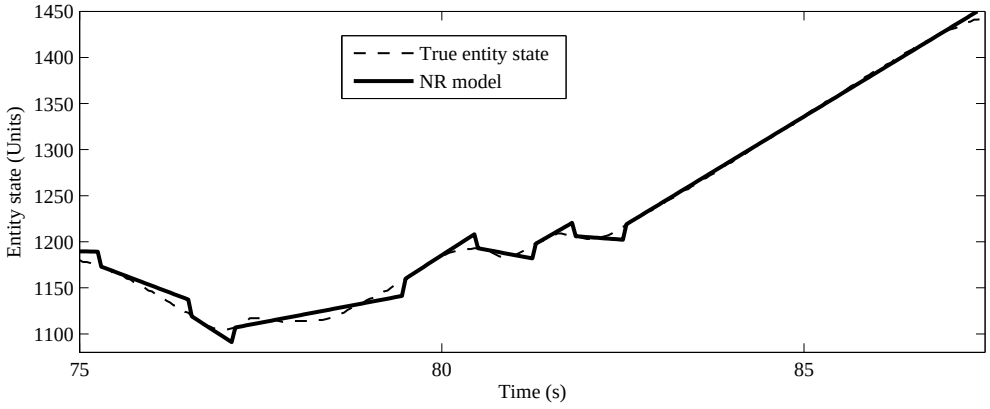
In addition, first-order DR slightly outperforms second-order DR for all the thresholds simulated as a consequence of the rapidly changing entity velocity (or acceleration) that is representative in First-Person Shooter (FPS) application domains [Pantel and Wolf 2002b]. In this case, the second-order predictor imposes too much



(a)



(b)



(c)

Figure 4.15 Results of extrapolating the Torque FPS motion. Highlighted sections of the true motion and the local model extrapolated by (a) the first-order equation, (b) the second-order equation, and (c) the NR technique at the local error threshold of 16 units.

inertia into the modeled motion by considering acceleration. It is interesting that second-order DR and NR both take into consideration more derivative information other than the current velocity but result in contrary performance compared to first-order DR. This demonstrates that the performance of any PCM depends on how well the prediction model fits the pattern in the user behavior so that the update parameters in the packets can be efficiently utilized. This will be further investigated from the information perspective in a quantified manner.

Figure 4.16 presents the remote inconsistency, measured as drift distance, arising from extrapolating the original entity motion using the three reckoning techniques for varying local error thresholds and latencies. The effects of latency are included to examine the impact of limited network conditions on the performance of PCMs. Once again, NR outperforms standard DR, yielding the lowest remote inconsistencies for varying thresholds and latencies. It should be noted that since NR generates the least updates (Table 4.5), 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. Second-order DR leads to higher remote inconsistency than first-order DR, especially at small error thresholds, because the second-order model diverges from the true entity trajectory faster and rises to the error threshold more frequently.

4.3.3 Information Model Results

The information model in Chapter 3 focuses on a "life cycle" analysis of the synchronization messages. The messages act as a information carrier that includes and conveys information about entity dynamics, with inevitable loss, to the remote entity state model.

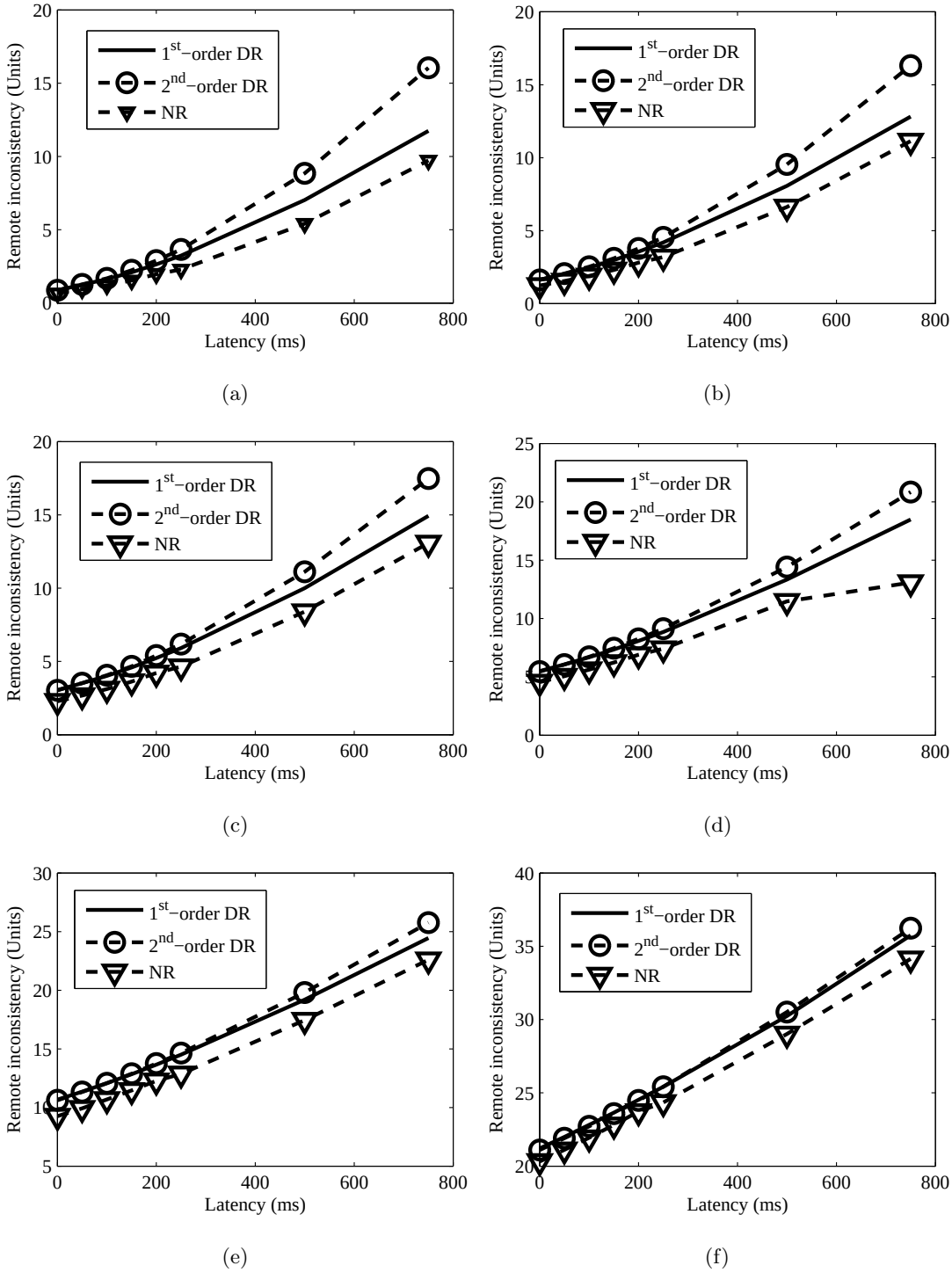


Figure 4.16 The remote inconsistency, measured as drift distance, arising from the extrapolations by the three PCMs for increasing fixed latencies under local thresholds equal to (a) 2, (b) 4, (c) 8, (d) 16, (e) 32, and (f) 64 units.

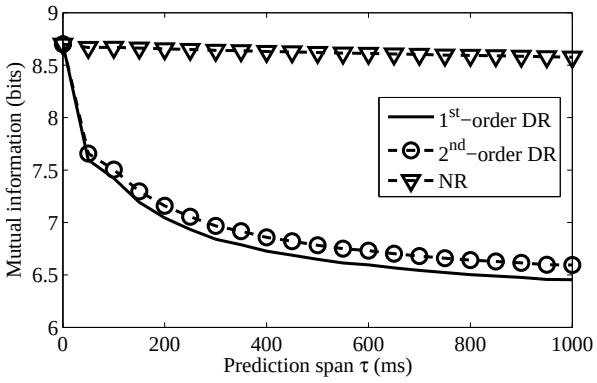
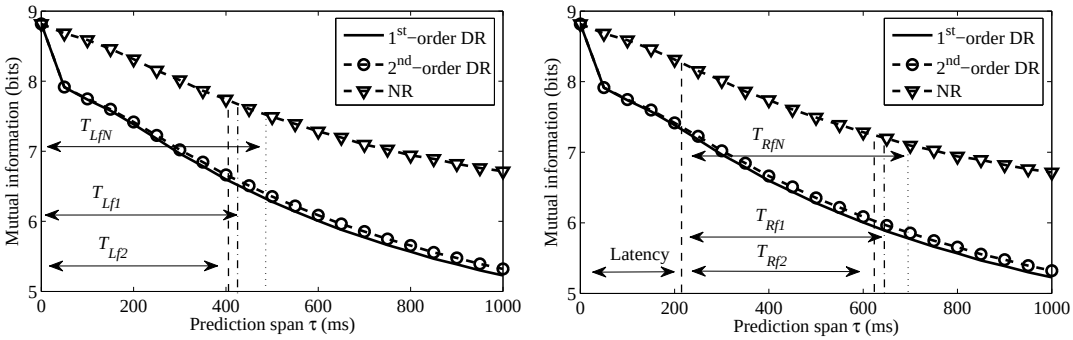


Figure 4.17 Results of measuring available information $I(\boldsymbol{u}; x_\tau)$ in the ESUs of the three reckoning techniques using the simple probability estimation algorithm.

For comparison purposes, the information capacity measured by the simple estimation algorithm (Section 3.2.2.1) is shown in Figure 4.17. The included information for the first and second-order extrapolation ESUs decay for increasing prediction spans as expected. However, the information capacity of the NR ESU stands out as it almost has perfect information capacity for future entity states, suggesting that a future entity state as far as 1s from the current time instance can be nearly perfectly determined from the current motion status. This high information value is intuitively suspicious. Consider the low occurrence of the sampled NR ESUs in Table 4.4. There are many ESU values that are sampled only once, which would in turn correspond to only one value of future entity state and result in a deterministic relationship in estimating information capacity. Such a deterministic relationship, however, is the consequence of the insufficient sample size rather than the entity behavior. Therefore the simple estimation algorithm can lead to overestimated information values in Figure 4.17.

To generate sufficiently accurate probability and information estimations from the limited dataset, the KDE algorithm is employed and the information capacity results for the three types of ESUs for increasing prediction spans are presented in Figure 4.18. Through a comparison of the different results in Figure 4.17 and Figure 4.18, it is obvious that the KDE algorithm corrects the overestimated high information value of the NR ESU, which now exhibits a noticeable information drop



(a) (b)

Figure 4.18 Results of measuring available information $I(\boldsymbol{u}; x_\tau)$ in the ESUs of the three reckoning techniques using the KDE probability estimation algorithm. The amount of information available for extrapolation in an ESU within (a) the local functioning period of the first-order ESU (T_{Lf1}), second-order ESU (T_{Lf2}), NR ESU (T_{LfN}), and (b) the remote functioning period of the first-order ESU (T_{Rf1}), second-order ESU (T_{Rf2}), NR ESU (T_{RfN}).

at large prediction spans. Consequently, all the following results are based on the improved information calculation using the KDE algorithm.

Figure 4.18 presents the available information about future entity motion encapsulated in each type of ESU for increasing prediction spans. Firstly, an entropy of $H(x) = 8.81$ bits is given by the true entity motion, and that is the average amount of information or data per simulation step required to reconstruct the entity motion with perfect fidelity, assuming every entity state is transmitted to the remote host under ideal network conditions. However, PCMs make use of the content of just one message to extrapolate future entity states at multiple steps, to reduce update frequency.

The delayed-mutual-information in Figure 4.18(a) and (b) compares the information capacity of the three types of packets for future entity states at increasing prediction spans. For standard DR, as can be expected for any non-deterministic entity motion data collected from practical DIAs, the information capacity of both the first and second-order packets decrease rather rapidly with increasing prediction spans. In the FPS game scenario where user's motion changes fast, instantaneous derivatives of the entity state bear little knowledge that could account for the user's future

behavior in long term. It can then be expected that the entity state must be updated more frequently under the first and second-order DR extrapolations. Throughout all the prediction spans, the second-order packet embraces more information about future dynamics since an additional acceleration value is included. This information advantage of the second-order ESU over the first-order packet is however marginal compared to the NR update message, which is now discussed.

The NR packet exhibits much lower information loss compared to the standard DR packets. Notice that the NR packet has exactly the same structure as the first-order DR packet (Table 4.3). This high information capacity is a result of two designing philosophies of NR. Firstly, the value of the NR velocity in the update packet is 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) including state derivatives from the other dimension of the entity trajectory, while first and second-order DR employ 2 and 3 values respectively from contextual dynamics of a single dimension. The high volume of referenced historical data gives NR considerable advantage over the standard DR packets in carrying information for extrapolation. Secondly, unlike the instantaneous derivatives in the standard DR packets which describe the current motion status of the entity, the calculated NR velocity is aimed at a predicted future position over a relatively long time interval (10 simulation steps in this experiment), which gives the NR velocity a greater tendency to keep information for long term prediction. Figure 4.18 explains, from the information perspective, why the NR velocity can accurately capture and compensate for expected changes of the entity state over a long time interval, and outperforms standard DR with significant packet transmission reductions.

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. In other words, each ESU is responsible for modeling multiple steps of future entity states within its functioning period. Generally a larger threshold means a longer functioning period and additional information loss due to the outdated content in

Table 4.6 Included information on the local host (sender)

Local threshold (units)	T_{Lf} (ms)			Included information rate $R_{s,local}$ (bits/step)		
	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
32	2507	2421	2619	5.34	5.47	6.77
64	4176	4018	4284	4.76	4.84	6.35

the previous ESU, which leads to a looser inconsistency bound. 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. The information within the local functioning period is the information included in the packet and ready for transmission. The average local functioning period and included information per simulation step $R_{s,local}$, as in (3.22) are summarized in Table 4.6. Figure 4.18(a) illustrates the local information encapsulation under the strictest local error threshold of 2 units. Due to the significant advantage over the other two types of packets, the NR packet carries the most information. The information in the NR packet is less vulnerable to the negative impact of extended functioning period compared to the DR packets, due to the larger amount of information it carries. The second-order ESU has an extra information source, and thus contains more information than the first-order packet.

On the remote host, only outdated information can be employed to build the remote model because the message has to go through the network and hence experiences transmission delay. The remote functioning period can be seen as the local functioning period delayed by a period of the network latency (an example at network latency

of 100ms is shown in Figure 4.18(b)). This delay further compromises information quality in the message because the mutual information decreases as the prediction span grows. The remaining information within the remote functioning period T_{Rf} is the amount of information available for the remote extrapolation. Based on (3.23), the remote available information $R_{s,remote}$ in the three types of messages is compared in Figure 4.19. It is then obvious from the information perspective that the information in the NR packet is less vulnerable to the negative impact of network latency compared to DR packets, due to the high quality of information capacity.

On the remote host, only part of the available information is actually utilized by PCMs in reconstructing the remote model. Figure 4.20 compares the extrapolated information $I(\tilde{x}_\tau; x_\tau)$ over increasing prediction spans for the three prediction schemes. It is interesting that in spite of the high values of the available information, NR does not have that much advantage in utilizing the information over the standard DR methods. The information extrapolated by NR extrapolation exhibits noticeable decays with increasing prediction spans. This is because the extrapolation equation in use under NR is as simple as the linear extrapolation after all, as a result of the simplicity of integrating NR into the PCM architecture. The first-order NR equation cannot fully interpret the delivered information. Therefore the compensated trajectory loses details of the entity's dynamics (Figure 2.7) 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. In addition, it should be noted that NR does utilize more information than standard DR over long time scales, when the prediction span approaches and goes beyond the maximum prediction horizon of 550ms. This is also 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

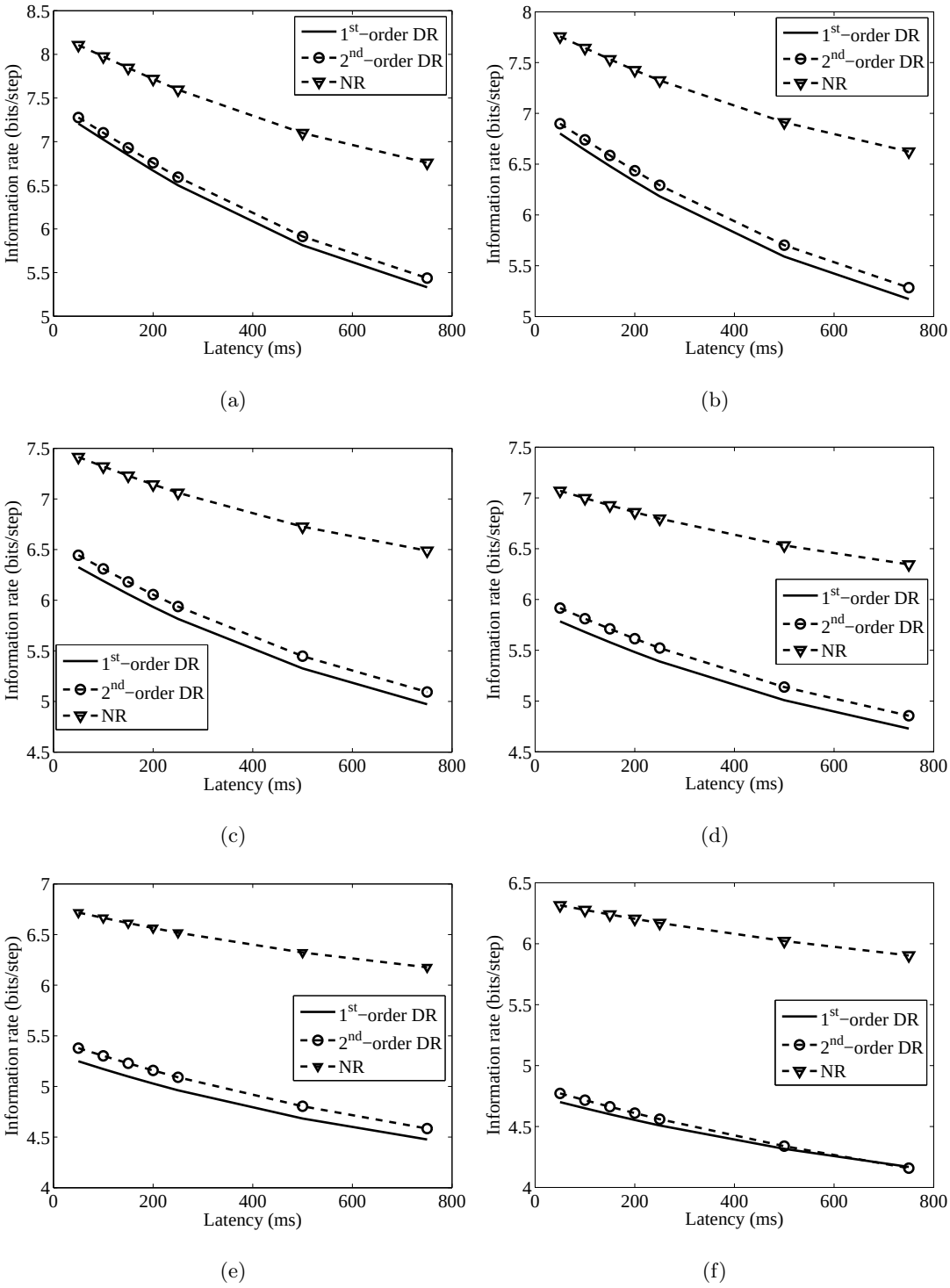


Figure 4.19 The available information rate $R_{s,remote}$ on the remote host for the three PCMs for increasing fixed latencies under local thresholds equal to (a) 2, (b) 4 , (c) 8, (d) 16, (e) 32, and (f) 64 units.

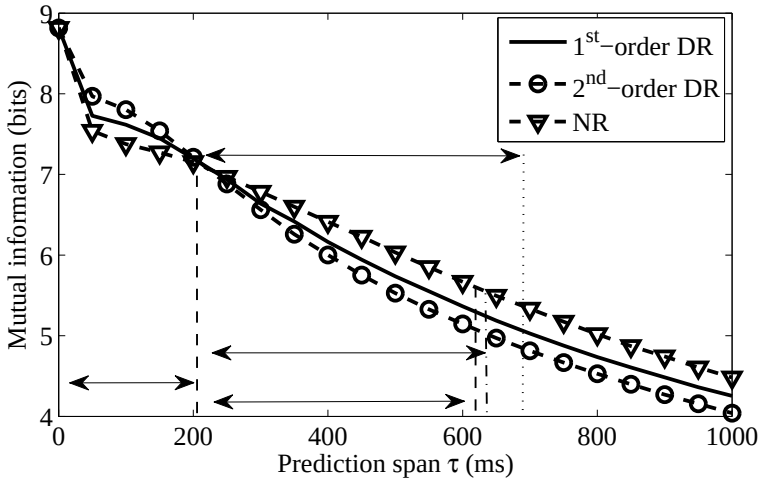


Figure 4.20 Results of measuring extrapolated information $I(\tilde{x}_\tau; x_\tau)$ in the ESUs of the three reckoning techniques, and the amount of information utilized from an ESU within the remote functioning period of the first-order ESU (T_{Rf1}), second-order ESU (T_{Rf2}), NR ESU (T_{RfN}).

preference improves the NR performance in reducing update packets.

Although the second-order DR packet provides more information to the remote reconstruction than the first-order DR packet (Figure 4.19), it only has a slight advantage in extrapolating entity states over first-order DR on very short time scales, due to the sensitivity to the instantaneous motion of the acceleration parameter. For most of the prediction spans, second-order DR is no better than first-order DR. Combined with the results presented in Figure 4.16, this result shows that higher throughput (as second-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. The information metric shows that with the information available, there may exist some other way, other than the second-order equation, to better utilize the information contained in the acceleration information included in the second-order DR packet.

Finally, the extrapolated information $R_{u,remote}$ within the remote functioning period, as in (3.25), is the information about the true entity state that is reconstructed in the remote model (an example at network latency of 100ms is shown in Figure 4.20). Based on (3.25), the remotely extrapolated information $R_{u,remote}$ in the three types

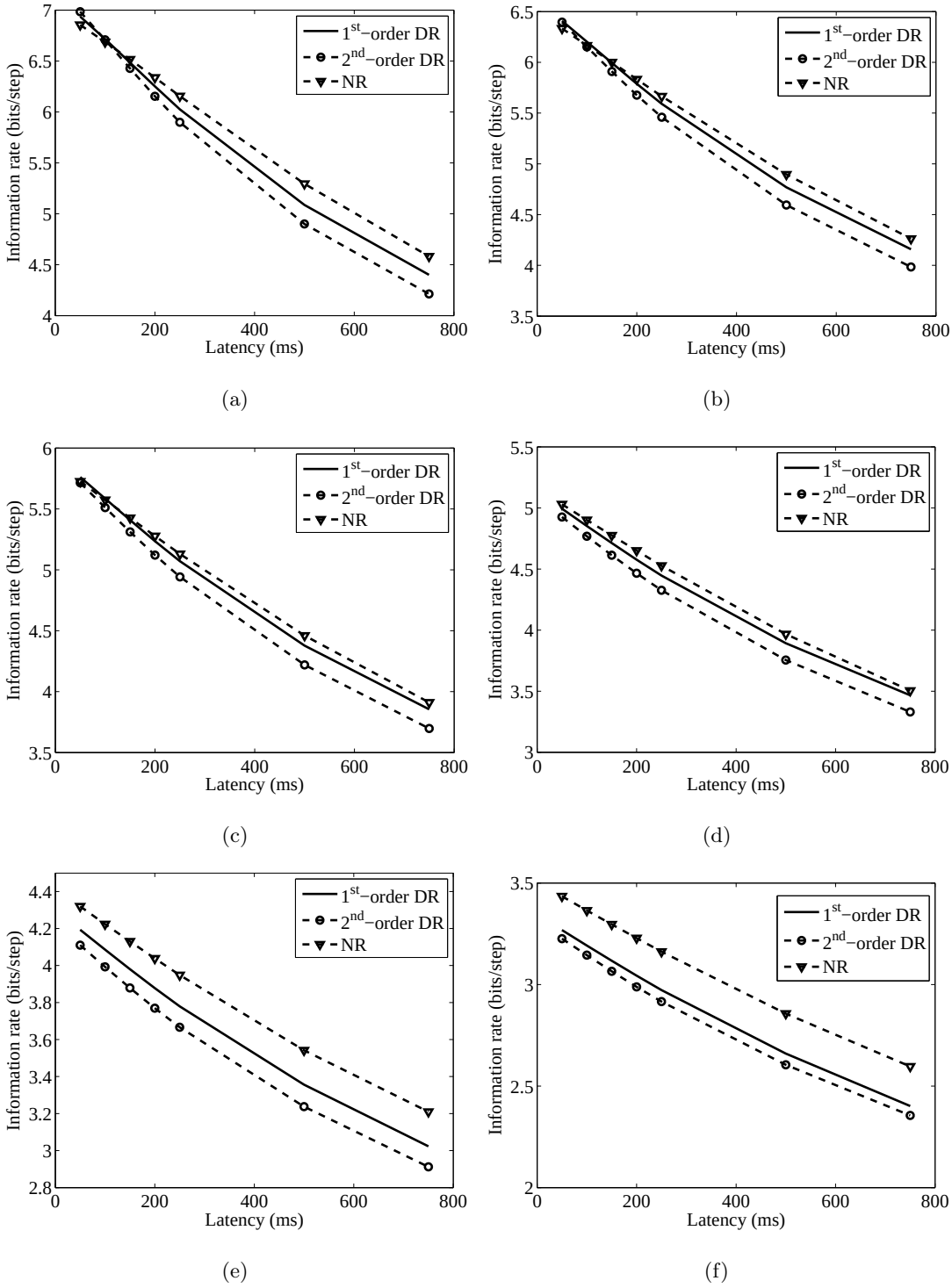


Figure 4.21 The extrapolated information rate on the remote host for the three PCMs for increasing fixed latencies under local thresholds equal to (a) 2, (b) 4, (c) 8, (d) 16, (e) 32, and (f) 64 units.

of messages is compared in Figure 4.21 for varying network latencies and local error thresholds. Due to the imperfect extrapolation equation, NR only exhibits moderate advantage over the standard DR methods at large thresholds or latencies. NR does not deliver higher information rate at small thresholds and latencies because of its preference towards a steady motion model that considers long-term state changes. In such situations, the ESU are generally sent very frequently and the two standard DR have the advantage in short-term predictions.

Comparing the remote information results in Figure 4.21 and the remote inconsistency results in Figure 4.16, it should be noted that second-order DR exhibits worse consistency than first-order DR at larger latencies, but their information metrics are approximated equal. This indicates that although second-order DR diverges farther from the true motion, the true entity state is almost equally predictable from the second-order DR extrapolation as it is from the first-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 second-order DR packet to give a better spatial consistency.

Figure 4.22 shows the information utilization efficiency (by scaling the reconstructed information by the corresponding available information in Figure 4.19) of the three mechanisms for a threshold of 8 units. Beyond the inconsistency performance shown by the traditional perspective, the information analysis reveals the potential of the NR message 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 algorithm. First-order DR, as the most widely deployed prediction scheme, is the most information-efficient extrapolation equation investigated here.

Through performance comparison of three prediction schemes, namely first and second-order DR, and NR, under a novel information model perspective, the following insights can be observed:

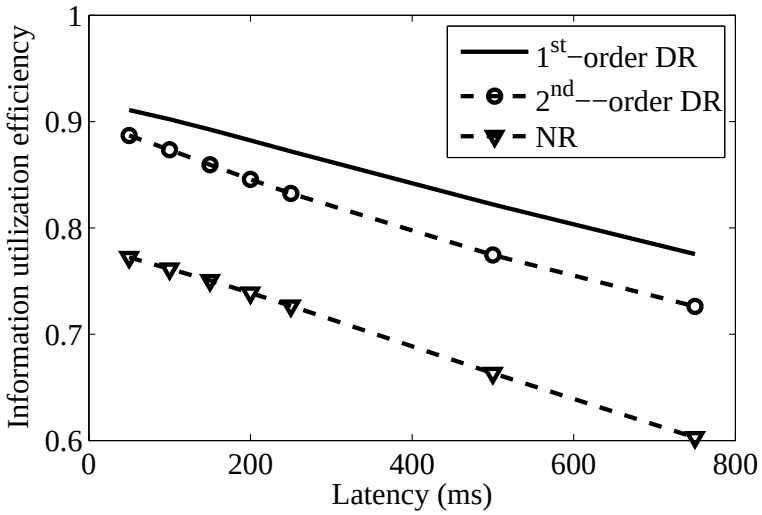


Figure 4.22 Information utilization efficiency at a local threshold of 8 units.

1. NR achieves a reduction in the number of update packets compared to standard DR, because it employs a large amount of information in producing the predictive NR vector.

2. 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.

3. First-order DR is the most information-efficient extrapolation in that most of its delivered information is used to reconstruct the remote model.

4. Second-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.

In the information model, aspects of consistency maintenance, namely entity dynamics, prediction algorithms, threshold, and network latency are analyzed using an integrated information metric. Factors causing inconsistency are viewed as information loss or reduction on information rate. From the information perspective, PCMs can be viewed as a form of lossy information processing where, by allowing

for local modeling error within the threshold, the original information rate $H(x)$ of the entity dynamic is reduced to the local storing rate $R_{s,local}$. The network latency endured by the transmission adds further information loss and only an information rate of $R_{s,remote}$ arrives on the remote host, which is finally reconstructed at the rate of $R_{u,remote}$. 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 lossy video compression problem.

4.4 Measuring Cross-Entity Dependence using the Information Metrics

In the previous chapters and sections, the information model for PCMs is proposed and implemented as a novel framework for measuring the utilization of the predictability in contextual dynamics of an entity to model the future entity state with reduced network traffic. This approach focuses on the philosophy that drives *content-based* predictive models where the past behavior of an individual object is an indicator of its future actions (see Section 2.4.4).

Aside from the individual contextual information, the interactions among two or more entities can also be explored to estimate user behavior. If several entities are engaging with each other closely, the actions they take are highly coupled, or dependent, to the behaviors of the others. Therefore, the state of one entity can also be modeled from other entities' behavior using *collaborative approaches* (Section 2.4.4). In this section, this cross-entity dependence is investigated and measured using the information metric. Such an analysis has important potential implications for research into entity state update (ESU) mechanisms within DIAs. Firstly, it potentially allows for predictive models of human-user behavior to be developed and used for remote entity extrapolation in the absence of transmitted ESU packets, reducing network bandwidth usage and improving the consistency of a DIA. For example, the state of a remote entity, once identified as closely interacting with

the local object in a particular pattern, could be estimated from the local object with high certainty without the state updates. Unlike the temporal dependence measured in previous sections, such an approach can be seen as the intra-frame prediction technique in video compression, where the value of a pixel is estimated from neighboring pixels by exploring the spatial redundancy of the raw video (see Section 2.5.1). Secondly, the information analysis of the cross-entity dependence also expands the scale for the general use of artificial intelligence (AI) in DIAs, particularly in networked multiplayer computer games [Laird 2002].

4.4.1 A Simple Taxonomy for Cross-Entity Behavior

The DIA scenario where the experiment is conducted is the same FPS game application developed under TGE, only with some modification of the rules. Here, a special "tag" item is located in the center of the map when the games starts, and it remains in this location until picked up by one of the players. The goal of the game for each player is to search for the tag and hold it for as long as possible, during which time he/she is not able to fire his weapon. The other player could achieve the possession of the tag by disabling the tag owner by firing weapon on him/her.

Based on research in related fields such as robotics and the multi-agent system (MAS) domain [Balch 2000; Lungarella and Pfeifer 2001; Cohen *et al.* 2002], three high-level categories of behavior with respect to the particular FPS application domain for the experiment, are defined: [McCoy *et al.* 2004; McCoy 2007]

- Attack/Pursuit: The goal of a player in this behavioral pattern is typically to "disable" the opponent, as quickly as possible, in order to gain possession of the "tag". Players in such a state will most typically exhibit aggressive actions, such as active chasing of the opponent coupled with a high degree of weapon firing.
- Defend/Evade: The goal of a player in this behavioral state is to remain alive for as long as possible in order to prevent the opponent from gaining possession

of the "tag" from him/her. Players in such a state will usually present passive strategies, such as attempting to escape from an on-coming opponent or hiding behind an available obstacle for an extended period of time.

- Wander/Search: The goal of a player in this behavioral state is exploration of the environment, either searching for an item of interest (typically another player or the "tag") or to simply get familiar with layout of the virtual environment.

4.4.2 Windowed Cross-Mutual-Information

The intensity of interaction between two players in the FPS game scenario is measured using mutual information between the state dynamics of the two entities. Again, the information metric is implemented based on one dimensional motions, namely the x-coordinates of the two entity dynamics, $x(k) = \{x(1), x(2), \ldots, x(N)\}$ and $y(k) = \{y(1), y(2), \ldots, y(N)\}$.

Due to the nature of the FPS application domain, the association between the two variables over time is unlikely to be stationary. In other words, a set of variant statistical properties could be expected during the dynamic interaction among two human-users. It is then necessary to use a "windowed cross-mutual-information" for the analysis of two time-dependent variables. The proposed algorithm is based on the mutual information defined in (3.7), and uses a sliding window to extract samples from the two time-series datasets. Delayed mutual information values are then calculated between the two windows of samples over a range of lags. The windowed cross-mutual-information produces a matrix of information values that can then be used to examine temporal evolution of dependence among the two data series, and allows for a greater and more intuitive insight into the interaction between the variables.

The operation of the windowed cross-mutual-information is shown in Figure 4.23. For the two single-dimensional motions $x(k)$ and $y(k)$ with the same length of N samples, consider a window size W, a time lag l on the integer interval $l_{min} \leq$

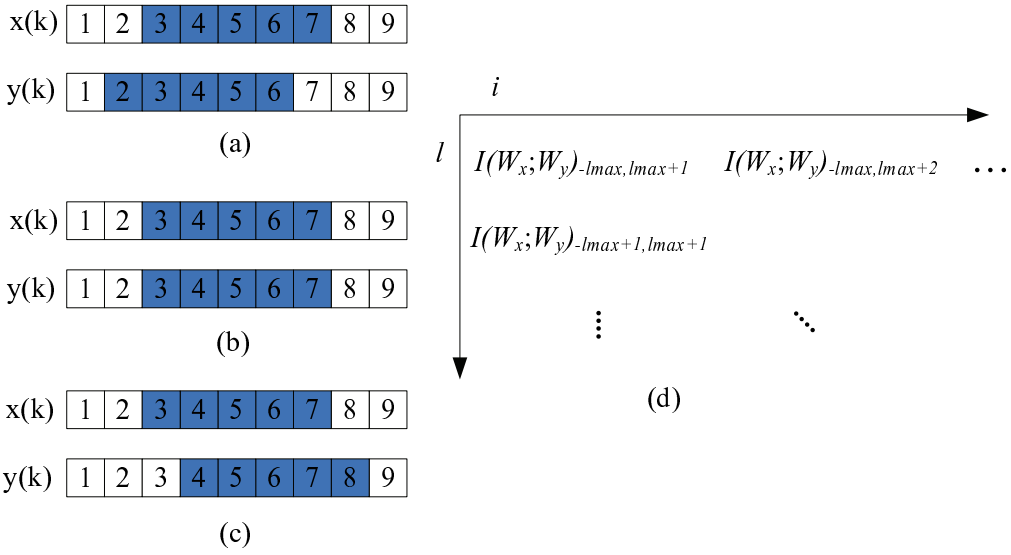


Figure 4.23 Selecting pairs of windows W_x and W_y from two time-series for $W = 5, i = 3$ and (a) $l = -1$, (b) $l = 0$, and (d) $l = 1$. (d) The results of the mutual information between each pair of windows are stored into the results matrix, whose columns represent the relative lag of the two windows and whose rows represent the starting time of the window selected from $x(k)$.

$l \leq l_{max}$ and an elapsed time index i from the beginning of the dataset. For each $i = l_{max} + 1, l_{max} + 2, \ldots, N - l_{max} - W + 1$, a pair of windows W_x and W_y can be extracted from the original motions x and y respectively:

$$\begin{aligned} W_x &= \{x(i), x(i+1), \ldots, x(i+W-1)\} \\ W_y &= \{y(i+l), y(i+l+i), \ldots, y(i+W+l-1)\} \end{aligned} \tag{4.7}$$

The windowed cross-mutual-information $I(W_x; W_y)_{l,i}$ between the two windows W_x and W_y can then be calculated using (3.7).

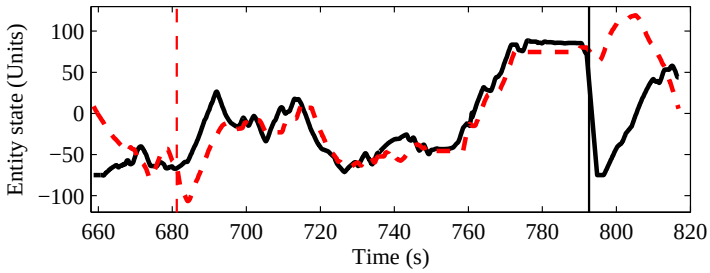
As shown in Figure 4.23(d), the results of the mutual information between each pair of windows are stored into a matrix, whose columns represent the relative lag of the two windows and whose rows represent the starting time of the window selected from $x(k)$. Each element in the matrix stands for the measured dependence between the two entities around the particular starting time i for the given lag l. The result matrix represents the time evolution of the interaction between the two users. The symmetry of mutual information in (3.8) guarantees that the result matrix of cross-mutual-information will contain the same values, only in reverse order, when the

variables x and y are swapped.

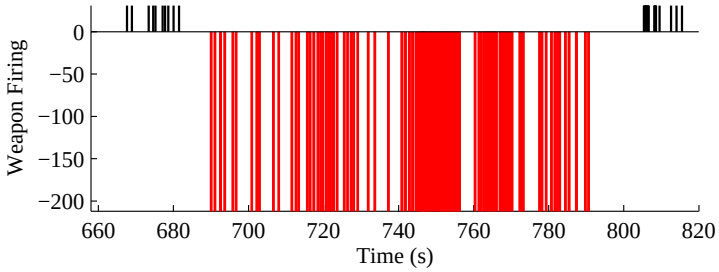
4.4.3 Results and Analysis

Figure 4.24 and 4.25 present the results of a windowed cross-mutual-information analysis conducted on two representative periods during the game play involving two interacting players, numerically labeled as "X" and "Y". Parts (a) within each figure graph the two horizontal components of the spatial state for both users (graphed as solid for player X, dashed for player Y). Solid and dashed vertical lines are superimposed for player X and Y respectively to denote the instances of time when the corresponding user is "disabled" due to exhausted health meter from damage through weapons firings, which are graphed in parts (b) within the two figures. The vertical lines in the weapon firings graphs show instances of time when the players fired their weapons. Player X's weapon firings are indicated by vertical lines graphed above the origin of the graph while the vertical lines graphed below represent the firings of player Y. The total number of their firing events is indicated by the vertical height of the lines — for example, in Figure 4.24(b), player A fired his weapon for 31 times in total, while player Y fired 210 times.

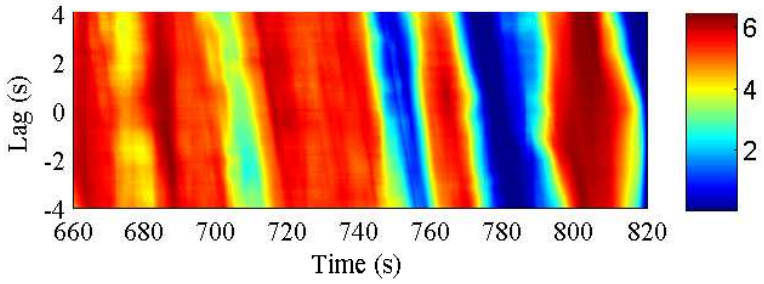
Parts (c) and (d) within each figure present the results of the windowed cross-mutual-information between the motions for both users along the horizontal components of the spatial state. In each case, the ordinate axis stands for the relative lag in seconds between the pairs of windows extracted from motions of player X and Y. The abscissa axis represents the starting time in seconds of the current window W_x. To assist visual inspection, the value of the measured cross-mutual-information between the two extracted windows for the particular point in simulation time and the particular relative lag is toned by a color scheme shown by the side of each graph (dark red for a high information result and dark blue for a low mutual information value). Therefore, the colored density plot is a graphical representation for the results matrices for windowed cross-mutual-information. These density plots characterize the time-evolution of dependence between the two users' behavior. The operation of the windowed cross-mutual-information procedure is conducted using the window



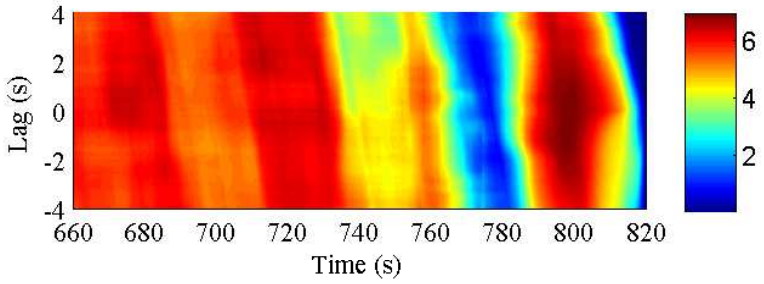
(a)



(b)

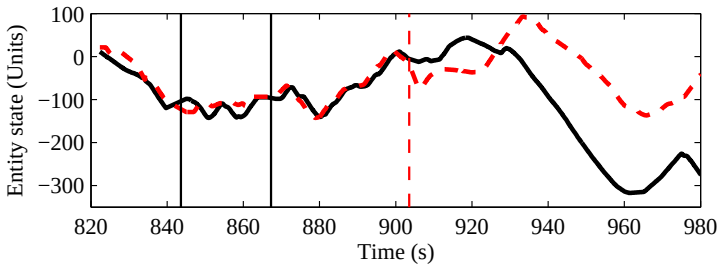


(c)

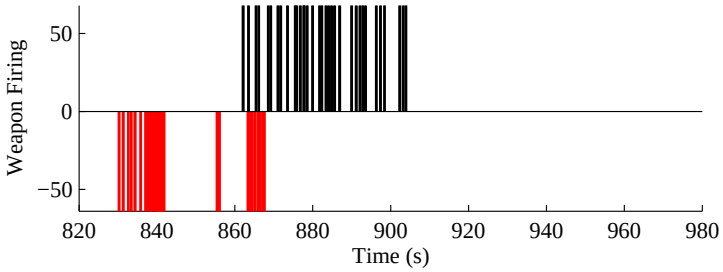


(d)

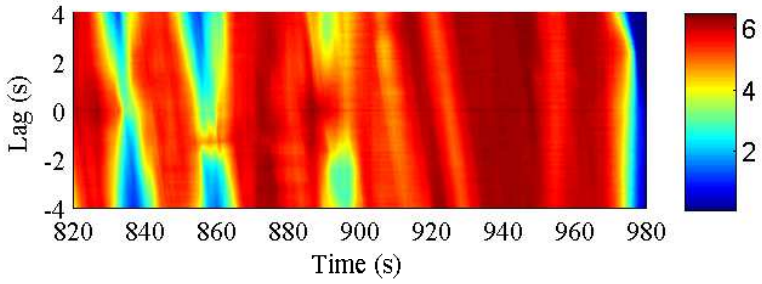
Figure 4.24 Windowed cross-mutual-information analysis performed over a period of 160 seconds of a game involving two opposing players. (a) The two interacting entity motions with death events. (b) Weapon firing events. Cross-mutual-information with window sizes (c) W=10s and (d) W=20s.



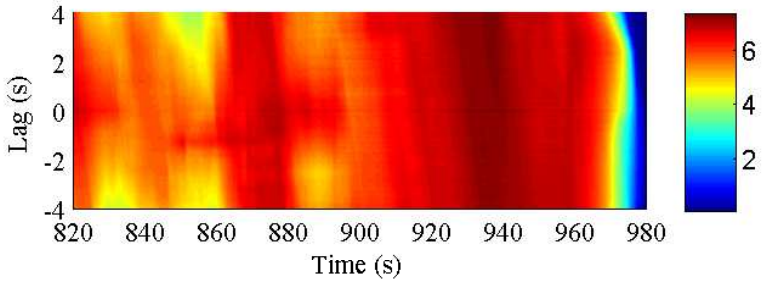
(a)



(b)



(c)



(d)

Figure 4.25 Windowed cross-mutual-information analysis performed over a period of 160 seconds of a game involving two opposing players. (a) The two interacting entity motions with death events. (b) Weapon firing events. Cross-mutual-information with window sizes (c) W=10s and (d) W=20s.

size W=10s in parts (c) and 20s in parts (d). Such window sizes give very small sample sizes (about hundreds of data points) for information estimation. Therefore the KDE algorithm is employed to generate sufficiently accurate results. The maximum lag l_{max} is set at 4s throughout the experiment. In order to enable the calculation of a cross-mutual-information value at every sample-index in the data, the two time-series are padded with zeros at both the beginning and the end. Such an action may reduce the absolute value of the mutual information due to the static element introduced. It should be noted that the variation of the motions is preserved, and the zero-padding does not impose much significant impact upon the purpose of a qualitative investigation of cross-entity interaction of the work presented in this section.

Visual inspection of the state and events data graphed in the figures reveals a large number of intensely engaged sections of behavior between the two players. In particular, pronounced chase/evade patterns can be noticed in the following period:

- between frame 680s and 780s in Figure 4.24(a)
- between frame 820s and 840s in Figure 4.25(a)
- between frame 880s and 900s in Figure 4.25(a)
- between frame 940s and 980s in Figure 4.25(a)

Here, a player in possession of the tag is being actively pursued by the opponent. Such interactions are usually coupled with intense weapon firings from the chasing player. In general, the attack (pursuit) and the defend (evade) behaviors are seen to occur in a coupled fashion. Such an observation is partly due to the "death match" scenario governed by the rule of the FPS game, where the player in possession of the tag cannot fire his weapon and typically takes a defensive strategy when engaged with the opponent. Such an observation implies that the actions of the chasing player is highly predictable from the evading trajectory of the player being pursued.

Sections of convergent behavior, possibly coupled with co-occurring weapon firing events, can be found in time between frame 660s and 680s in Figure 4.24(a) and

frame 850s and 870s in Figure 4.25(a). In such situations, both players are attempting to gain the possession of the tag item.

The windowed cross-mutual-information analysis provides easy-to-understand visual information on the evolution of the dependence between the interacting players. From an initial inspection of these density plots, as shown in parts (c) and (d) of Figure 4.24 and Figure 4.25, abrupt changes occur among successive vertical sections or slices of mutual information values (i.e. across the temporal dimension). The mutual information in these areas varies over time, but remains relatively stable over the set of relative lags. Such an observation indicates a notable change in the intensity of the dependence calculated between the two players as time progresses, implying that the interactions between them are clearly non-stationary because the behavior of the players is highly coupled. Such vertical slices are pronounced at times between frames 740s and 820s in Figure 4.24(c) and frames 820s and 880s in Figure 4.25(c). It should be noted that the vertical slices pattern of the cross-mutual-information, identified as blocks of constant color-zone, coincides with the interaction section changes captured by visual examination discussed above (for example, chasing/evading, approaching the tag, or exploring the space after rebirth). Many of such sections denote time intervals that imply strong and stable dependence between the two players that continues for an extended period of time. indicating the information metric is capable of measuring cross-entity dependence and reflecting the time-evolution of user interactions in real-time.

Variations of the cross-mutual-information along varying relative lags could also be observed during many of the density plots. At the frames around 750s in Figure 4.24(c), the mutual information between the two extracted windows of motions changes from blue (indicating a low mutual information less than 2 bits) at the minimum lag $-l_{max}$ to red (indicating a high information about 6 bits) at the maximum lag l_{max}. While at the frames around 860s and 900s in Figure 4.25(c), higher mutual information is identified around the zero-lag area (indicated by the red color). Such pattern suggests that if the relative lag that produces the strongest dependence between the players could be discerned, one may model the coupled behavior of one player in a predictive manner by using the value of the other. Parts (c)

and (d) of each figure does not appear to identify any significant difference, implying that the dependence evolution between the two players remains largely invariant over a temporal-interval of length 10s – 20s. However, it is also apparent that the colored density plots generated under the longer window size are smoothed versions of those generated for the shorter window size.

On a final remark, the work presented in this section provides preliminary results in demonstrating the concept of employing the information measurements to investigate cross-entity predictability. Such an idea is not a core contribution of the thesis and shall be extended in future work.

4.5 Concluding Remarks

In this chapter, the information model for PCMs proposed in Chapter 3 is implemented for various entity motions one may expect in DIAs. The philosophy underlying this analysis is that predictability within contextual entity dynamics forms the foundation for the estimation of a remote entity state using predictive schemes. The importance of a quantified measure of such predictability and how this predictability could be transmitted and utilized by PCMs for both understanding and improving PCMs cannot be overestimated. The investigation of PCMs using the information model and metrics bring novel insights about consistency maintenance and the trade-off between throughput and consistency.

In the first part of this chapter, the information characteristics of various entity motions, including synthetic motion models and collected datasets, are examined by the information metrics. The analysis of these entity motions reveals that contextual entity dynamics contain predictability about user behavior. Parameters describing the entity motion status (for example, the estimated state derivatives) could be used for extrapolating future entity states. Generally more parameters could embrace more predictability and make the future entity motion more predictable. For deterministic motions, a sufficient amount of contextual information could determine the entity state in the future with complete certainty, while for non-deterministic motions only

part of the uncertainty can be removed by exploring the temporal dependence. The time-decaying feature of non-deterministic motions indicates the necessity of employing state update schemes to maintain consistency among dispersed participants in DIAs.

In the second part of this chapter, the operation of PCMs is reorganized based on an information analysis of the ESUs that synchronize the local and remote views of a virtual world in which users interact. PCMs are considered a form of lossy video compression under this perspective. Consequently, the information rate provided by the ESUs is reduced during the local modeling, network transmission, and remote extrapolation. The inconsistency arising in the process is due to the information loss. A comparison among three PCMs suggests that NR outperforms standard DR in that it takes the advantages of a large amount of historical data and the learning ability of neural networks to produce an NR velocity that preserves high volume of information for long-term prediction. In the meantime, the information analysis also indicates that given the available information in the ESUs, there is a need for some better extrapolation method that could utilize the information more efficiently and further improve perceptual consistency without additional state update information.

Finally, the information metrics are used to measure cross-entity dependence during user interactions in DIAs. Results from the windowed cross-mutual-information analysis of entity motions from two interacting players demonstrate that the information measurements is capable of reflecting the time-evolution of the dependence between two entities due to changes of interaction patterns during the application. Such quantified measure of cross-entity dependence validates the employment of collaborative user behavior modeling techniques that estimated entity state model in a predictive manner based on the actions of other entities that are closely interacting. Such approaches can be seen as exploring spatial inter-entity redundancy in video compression.

In the next chapter, the concept is applied to a novel dynamic extrapolation model. In this extrapolation framework, different extrapolation schemes are assessed using the information model and the one that can deliver the highest information rate

to the remote host under changing network conditions is dynamically selected as the active model under use. The simulation results presented verify the utility of such a dynamic framework for the further reduction of remote inconsistency when compared to the use of a fixed DR approach.

Chapter 5

An Information-Based Dynamic Extrapolation Model for DIAs

5.1 Introduction

Chapter 4 presented the reinterpretation and evaluation of the operation of Predictive Contract Mechanisms using the information model. Concepts derived from information theory are demonstrated to capture the general predictability exhibited by the entity behavior and measure the efficiency of the extrapolation model to deliver and utilize this predictability to build the remote entity state model. Inconsistency arising during the execution of a PCM is caused by information loss introduced by the reduced number of ESUs, imperfect prediction, and limited network resources. From such a perspective, PCMs are regarded as performing lossy video compression. The chapter concludes with a discussion regarding the suitability of the extrapolation model. It is shown that an extrapolation model is considered better if it can utilize more of the information encapsulated by the received ESUs and hence further reduce the uncertainty of future entity states.

Motivated by this analysis, a novel approach to manage data transmission for rate-based PCMs, known as the Information-Based Dynamic Extrapolation Model, is

proposed in this chapter. This approach presents the design of a conceptual framework that minimizes inconsistency by optimizing the usage of the available network bandwidth. The proposed model manages the data transmission to deliver to the remote host as much information about the true entity state as possible. The information model is employed to measure the information utilization efficiency of a set of extrapolation models. The proposed framework then dynamically switches the extrapolation model and the packet rate to make the most information-efficient usage of the available bandwidth. By doing this, the dynamic extrapolation framework allows consistency control mechanisms to be network-aware, and thus combines both the application and network layer factors in a joint optimization of consistency maintenance for DIAs using PCMs.

Finally, the dynamic extrapolation model is implemented on two representative DIA scenarios within simulated network conditions. The results show that this approach can help optimize consistency under constrained and time-varying network conditions.

5.2 Overview of the Information-Based Dynamic Extrapolation Model

The proposed framework is designed for an optimized management of rate-based PCMs, where ESUs are sent by the local host at a constant update frequency. Such PCMs have direct control over the data transmission rate arising from state changes communication. The information model and analysis (previously developed for threshold-based systems) can be adapted to rate-based PCMs through the observation that the functioning period of an ESU is now the constant interval between two update packets.

5.2.1 Design Rationale

As described in Section 2.4.4, PCMs must be carefully adapted to the participating user's behavior and the underlying network conditions. Otherwise they could have a negative impact on consistency maintenance for DIAs. Many existing works employ an adaptive approach to manage PCMs. However, this "tuning" is typically based on a single factor involving either the entity behavior in the application layer or data transmission in the network layer. In the former case, the existing adaptive extrapolation models dynamically choose an active extrapolation equation that best suits the specific user behavior from a pool of candidates [Lee *et al.* 2000; Delaney *et al.* 2003; McCoy *et al.* 2005]. The extrapolation model selection is solely based on some application layer criteria that evaluate the candidates' performance (such as the average local prediction error over a period of time in the past). Such application level parameters that control these adaptive approaches are independent of the network in which they operate. From the information perspective, an extrapolation model producing more accurate prediction makes better use of the predictability of the entity behavior and hence utilizes more information in the ESUs. Typically, different extrapolations require various amounts of data (such as entity position and velocity) to be included and transmitted in the ESUs. But the impact of this varying packet size and, in turn, the network traffic on the overall inconsistency is not clear, as the performance of the network connecting the local and remote hosts involved in the DIA is not considered. So, for example, an extrapolation model with higher local prediction accuracy may cause heavier network traffic, causing the network connection to become overloaded. The ESUs are then queued or even dropped, causing the remote host to rely on severely outdated information to estimate the entity state model. In this case, the extrapolation model causes more inconsistency than it solves.

In the latter case, in order to optimize consistency, Marshall *et al.* [2008] manages the update rate of a PCM such that the overall data transmission rate matches the available bandwidth of the underlying network. In this way, the bandwidth usage is maximized to control the inconsistency introduced by insufficient updates, and

in the meantime the inconsistency caused by increased latency and packet loss on overloaded network hardware is avoided. However, the maximized data throughput alone cannot guarantee an optimized consistency without considering the suitability of the prediction model for extrapolating the specific user behavior. If a poor extrapolation model is used, the network bandwidth, even though in maximal usage, may be wasted transferring meaningless data with little useful information that can be extrapolated to estimate the true entity behavior.

In light of this analysis, there is clearly a need for a scheme that takes into consideration both the application layer and network layer factors in optimizing inconsistency arising from a DIA. The information model, as demonstrated in the previous chapters, enables a quantified examination of the data transmitted with the ESUs with respect to their capacity of delivering predictability about the user behavior to be explored by a particular extrapolation model. The information metrics combine user behavioral characteristics in the application layer and data transmission over real networks. Such a measure provides the foundation for a joint optimization of the application and network layer performance of PCM operation.

From the information perspective, the full information rate of the true entity motion is reduced to the extrapolated information rate on the remote host by PCMs. The extrapolated information is influenced by features of the extrapolation model and limited network resources that introduce information loss and cause inconsistency: the imperfect extrapolation model only utilizes a part of the predictability in the user motion; non-zero network latency makes the information in the messages outdated by the time they arrive on the remote host; limited network bandwidth constrains the packet generation rate, and a low update rate implies that a message suffers from more information decay before the next message arrives. In Figure 3.9, the extrapolated information rate is the average information within the area constrained by the information utilization $I(\tilde{x}_\tau; x_\tau)$ and the remote functioning period T_{Rf} of the ESUs. This graphical illustration of the information model perfectly shows how the overall performance of a PCM is jointly subject to characteristics in both the application and network levels.

It has been demonstrated that for any single extrapolation model, the optimal update packet rate can be determined such that the overall data transmission rate matches the available network bandwidth [Marshall *et al.* 2008]. Transmitting ESUs above or below this optimal rate will introduce extra unnecessary inconsistency due to over or under utilization of the available network bandwidth. Such a maximized bandwidth usage also leads to a higher information rate as extra information loss due to an extended function period is avoided. However, an issue arises when the adaptive approach can choose from several prediction methods to optimize the delivered information rate. For a simple example, consider two popular polynomial extrapolation models, namely first and second-order dead reckoning. For many DIA scenarios examined in Section 4.2.3, the second-order extrapolation model produces higher information utilization than the first-order extrapolation. Generally, such an information advantage for an extrapolation model comes at the price of a larger volume of motion parameters in the resultant ESUs.

For a given available network bandwidth, the update rate of the second-order extrapolation must be reduced accordingly to maintain the optimal data rate under the increased packet size. Also, the larger packet size leads to higher delay in transmitting the ESUs through the network connection. Figure 5.1 illustrates the situation by comparing the functioning periods of the two extrapolation models. Although second-order DR extrapolates more information than the first-order equation, the extra functioning period and the higher latency make it suffer more from out-of-date and lost information while the first-order extrapolation has received a new packet with updated and higher information rate. Whether or not the faster update rate of the first-order extrapolation can compensate for the lack of the acceleration information in its ESUs depends on the information decay of the two extrapolations and their functioning periods. In changing network conditions, both the packet rate and the extrapolation model must be adapted in order to deliver the highest information rate to the remote entity state model.

In the next section, an information-based extrapolation scheme that dynamically adjusts both the extrapolation model and update packet rate to optimize the remote information rate is proposed.

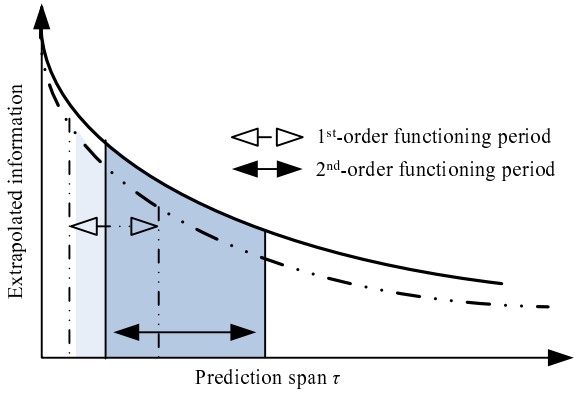


Figure 5.1 Illustration of the information trade-off between packet size and update rate. The larger packet of the second-order extrapolation model (due to the acceleration data) suffers from higher latency and a longer functioning period, which lead to further information loss that could offset the extra information brought by the additional parameter.

5.2.2 Information-Based Dynamic Extrapolation Model

The Information-Based Dynamic Extrapolation Model is designed to minimize the level of inconsistency that can arise in a peer-to-peer or client-server DIA using rate-based PCMs. In such scenarios, the last mile link of the user's internet connection is typically the bottleneck link that constrains the consistency maintenance the most [Jehaes *et al.* 2003; Dube *et al.* 2005]. The diagram in Figure 5.2 presents the operation of the proposed model. The model is a closed-loop consistency control technique that works in an *Observation-Feedback-Adjustment* manner. It operates by inferring the information utilization of the extrapolation models and the state of the bottleneck connection. Such information is then used in two ways. First, it is used to determine an optimal update packet rate for each extrapolation model in order to maximize the usage of the resources available on the underlying network connecting hosts. Second, it is used to select the extrapolation model that produces the highest information rate for the remote entity model, so that the data transmission between participants is more information-efficient.

The model consists of five major components:

(1) A pool of i candidate extrapolation models given by $M = \{m_1, m_2, \ldots, m_i\}$,

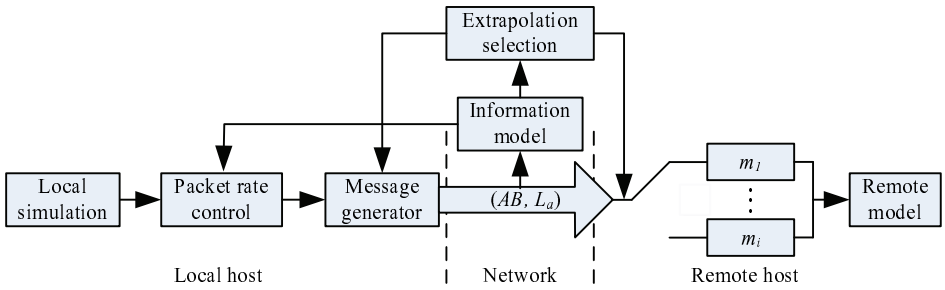


Figure 5.2 Illustration of the Information-Based Dynamic Extrapolation Model.

and update parameters (such as state derivatives) in their ESUs $U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_i\}$ used for these prediction models, respectively. The set of extrapolation models also define the update packet sizes $PS = \{ps_1, ps_2, \ldots, ps_i\}$ for the candidate models. A default active extrapolation model m_0 is chosen from M for the application to start with.

(2) A run-time estimation scheme that monitors network conditions on the application layer's "logical connection" for each user. The logical connection is an application layer connection that is maintained between the server and each client in a client-server architecture, or between each pair of peers involved in a peer-to-peer architecture. This is a common construct in DIAs that employ UDP as a transport protocol. Each logical connection represents one flow of data for one host and multiple logical connections may be aggregated to transmit data over the same physical link.

The network condition estimator is responsible for producing estimation of the available bandwidth AB of the logical connection on the bottleneck link and the one-way network latency L_a from the local host to the remote host. The network conditions estimator works as an interface for the dynamic extrapolation framework to the underlying network and is a relatively independent component in the proposed approach. Any estimation algorithm that gives the two indicative characteristics of the state of the network resources can be incorporated in the Information-Based Dynamic Extrapolation Model. The design of such an estimation scheme is beyond the scope of this chapter and is

not discussed in detail. A method of estimating the available network bandwidth using latency trends can be found in [Marshall *et al.* 2008]. It should also be noted that since the network conditions are estimated for each logical connection separately, the dynamic extrapolation model operates on a per-user basis. It allows for multiple hosts under the heterogeneous Internet to take different strategies to optimize consistency according to their own situations.

(3) An update rate control scheme based on the available bandwidth. The optimal update rate PR_i for each candidate is determined so that the bandwidth usage is maximized and in the meantime the available resources on the bottleneck link is not overloaded. Such an optimal update rate is determined by (5.1):

$$PR_i = \frac{AB}{ps_i}. \tag{5.1}$$

(4) An information model that characterizes the information utilization of each candidate extrapolation model. The information model is key to the operation of the dynamic extrapolation model, as it examines the capability of each prediction model to extrapolate information about the true entity motion to build the remote entity state model, and how this extrapolated information decays with time. The information model measures the suitability of using the extrapolation models for the specific user behavior, and combines the application layer and the network layer factors by measuring the amount of information about the true entity motion that is extrapolated from the transmitted data. Along with the estimation of the current network condition and the optimal update rate, the remote information rate $R^{(i)}_{u,remote}$ that each extrapolation model m_i can produce is estimated by (3.25), in which the functioning period $T^{(i)}_{Rf}$ of m_i is now the constant update interval determined by (5.2):

$$T^{(i)}_{Rf} = \frac{1}{PR_i}. \tag{5.2}$$

The information model is trained in an off-line manner from pre-recorded trajectory on the local host and is transparent to the remote host. Thus the framework is easily integrated with current PCMs. The active extrapolation model choices made on the local host must be notified to the remote host.

However, such switching decision communication is negligible compared to the normal state updates in the sense of network traffic caused, and is therefore not considered in the following discussion.

(5) An extrapolation selection, based on the estimated information rate, chooses whichever extrapolation model that produces the highest information rate under the current network conditions as the current active extrapolation m^*:

$$m^* = \arg\max_{m_i} R^{(i)}_{u,remote}. \tag{5.3}$$

Ideally, in the face of network condition changes, a switch of the active model will always occur when one of the inactive models is seen to be outperforming the rest. However, there are two concerns that have to be considered for smooth user perception in practice. Firstly, the extrapolation model change is deactivated within a time-out T_O after the previous model switch. Such an "inhibited" phase is designed to avoid excessive switchings in response to transient network condition fluctuations. This can also be done by applying a smoothing operator to the network condition measurement. Secondly, to avoid a sudden change in consistency level at the moment k_0 of extrapolation model switch, a gradual convergence algorithm described in (5.4) is defined for a smooth conversion to the new active model:

$$\hat{x}(k_0 + k) = (1 - \frac{k}{T_C}) \cdot m^-(u^-) + \frac{k}{T_C} \cdot m^+(u^+) \text{ for } k \leq T_C, \tag{5.4}$$

where T_C is the convergence period in simulation steps, $m^-(u^-)$ is the state predicted for the current step $k_0 + k$ by the previous model using the last received message before the switch, and $m^+(u^+)$ is the prediction made by the new active model using the most recently arrived message.

In the next section, for the purpose of illustrating the framework, the performance of the Information-Based Dynamic Extrapolation Model is evaluated through experiments run on two different user motions. The data used are representative of those entity motions for which higher-order extrapolations extrapolate more information, and there exists a trade-off between "small-packet-fast-update" and "large-packet-slow-update" schemes, as illustrated in Figure 5.1. The results presented

demonstrate that the proposed technique can accurately choose the extrapolation model that gives the best information rate and consistency under changing network conditions.

5.3 Model Implementation

For illustration purposes, the network conditions estimation is implemented merely as two control parameters, namely bandwidth and latency. In realist network environments, an actual estimator would be necessary, but the simple approach used here does not affect the applicability of the Information-Based Dynamic Extrapolation Model to the real Internet environment. The information model just needs the available bandwidth and latency to measure the information loss in transmitting data over a constrained network connection. Any specific estimation algorithm, however, is independent of the operation of the dynamic extrapolation model and is therefore not discussed in detail.

5.3.1 Experiment Set Up

The two entity motion datasets to evaluate the proposed framework are generated by a First-Person Shooter (FPS) game [McCoy *et al.* 2007] and a racing game [Marshall *et al.* 2006] developed using the same Torque Game Engine as in Section 4.3, but with different scenarios and rules. The plan views of the two game environments are shown in Figure 5.3. In the FPS game, the goal of the players is now to disable the opponent (a programmed robot) by firing their weapons. The disabled object (the player or robot) is reborn at a random location and resumes the game. The programmed robot is constrained to, and moves constantly along, the defined path shown in Figure 5.3(a). The reactive robot is provided with a "sensor field" with a pre-defined radius. If an approaching human player enters this field, then the robot will "attack" the player. The entity state is recorded at 10Hz for 1600s, and the motion is re-sampled at 50Hz by interpolating the recorded data using cubical spline interpolation. This higher sample rate is necessary for the experimental studies to

(a) (b)

Figure 5.3 Plan views of part of the environments for (a) the FPS experiment and (b) the racing experiment.

consider a wide range of update packet rates that are commonly seen in DIAs. This dataset is considered representative of those application scenarios in which the simulation of realistic physics plays a key role, since cubic interpolation provides natural continuity. In the racing game scenario, an entity's movement is constrained as a consequence of the defined racing course in the environment. The goal of the players is to be the first to past the finishing line. The racing motion is recorded in repeated game sessions at 50 samples per second for a total period of 1600s. As throughout this thesis, the results presented are based on the x-coordinate $x(k)$ of the motions.

The first half of each dataset is used to build an information model for each of the candidate extrapolation models using (3.24) to measure the amount of information about the user behavior that can be used by the extrapolation models. The second half of the datasets are used as test data to evaluate the dynamic extrapolation model.

The network conditions in the experiment are set up for a simplified but representative network environment for DIAs transporting synchronization messages using UDP. Figure 5.4 shows an overview of the architecture of a current day WiMax access network, as a representative wireless network environment which covers a wide rage of area and allows for a fairly large number of participants (namely human-users).

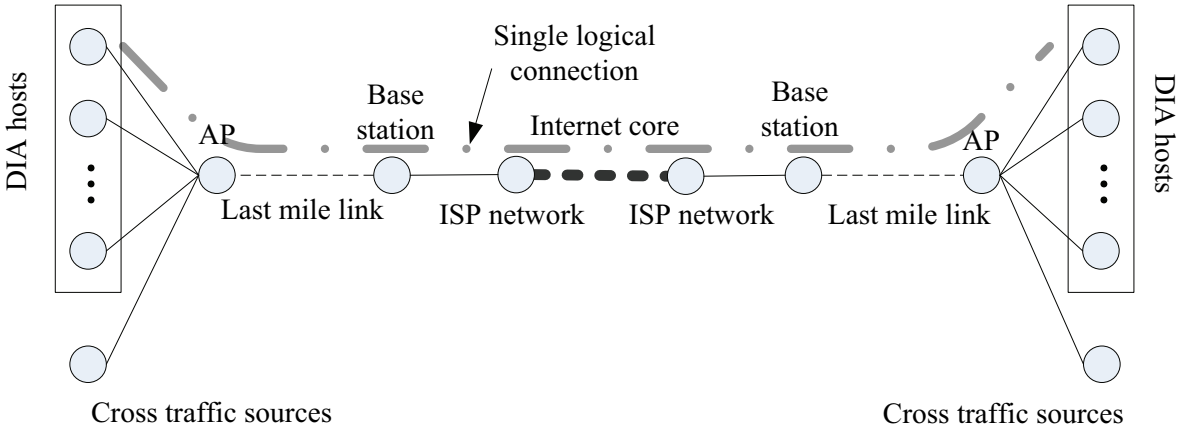


Figure 5.4 An overview of the network topology emulated in the experiment.

A DIA host represents a separate host machine through which a DIA participant interact in the virtual environment. Each DIA host maintains a separate logical connection to another. Typically, each logical connection from a DIA host connected trough an Access Point (AP) and is then multiplexed to a base station over a single broadband connection, also known as the last mile link. The base station is then, through the ISP network, connected to the core of the Internet, which has a much higher bandwidth. Therefore the upstream link of the broadband connection, with the typical bandwidth value around 200kbps [Imagine 2010], is the bottleneck link of the network. Considering the possible multiple participants behind a last mile link, due to the large number of users involved in a DIA, and cross traffic from other applications such as Internet video, the available bandwidth AB for a single logical connection for a DIA host is set to be no more than 20kbps. The maximum update packet rate of the applications is 30 Packet Per Second (PPS). These settings are in line with general DIA traffic [Färber 2002; Feng *et al.* 2005; Zander and Armitage 2005; Harcsik *et al.* 2007].

As the dynamic extrapolation model operates by controlling the overall data rate to match the available bandwidth, there should be no queuing on the logical connection for the DIA instance on the bottleneck link. As shown in (5.5), the one-way network latency L_a from the local host to the remote host is modeled as the sum of packet transmission delay L_T on the bottleneck link and a delay L_f in passing the packets

Table 5.1 Network condition settings

Time (s)	800	880	960	1040	1120	1200	1280	1360	1440	1520
AB (kbps)	15	8	2	1	7	8	10	12	10	10
L_f (ms)	20	100	200	400	500	800	800	600	120	100

through the network nodes outside the last mile link. L_f refers to all other factors in the overall latency, which are independent of the bottleneck bandwidth changes. Although these elements are not constant values, they are collectively modeled as a fixed delay in the experiment.

$$L_a = L_T + L_f = \frac{ps_i}{AB} + L_f. \tag{5.5}$$

Table 5.1 shows the changing network conditions given by the two control parameters AB and L_f in the test path simulation, reflecting wide variation of the realistic network [Pantel and Wolf 2002a; Armitage 2003; Claypool 2005].

The rest of the experiment is set up as follows. The set of candidate extrapolation models consists of the two most widely employed PCMs: first and second-order DR. The state motion status parameters are entity position x, velocity v for the first-order extrapolation and an additional acceleration a for the second-order extrapolation, each recorded in a data unit of 8 bytes. Considering the overhead to these values in a UDP packet, the packet sizes for the two extrapolation models are 44 bytes and 52 bytes respectively and are representative of typical state updates in DIAs [Färber 2002; Feng *et al.* 2005; Harcsik *et al.* 2007]. For this particular one-dimensional example, the effect of the packet overhead is large. In a real world application with many more parameters (e.g., motion status from other dimensions), packet overhead would be a smaller fraction of total bandwidth usage and the change from the first-order to the second-order parameter would likely have a larger effect. The candidate

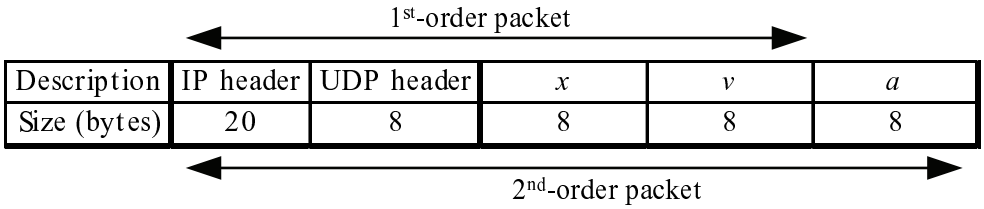


Figure 5.5 Packet structure for the two extrapolation models.

set is summarized in (5.6) and the packet structure is shown in Figure 5.5.

$$\begin{aligned} M &= \{m_1 = 1^{st}\text{-order DR}, m_2 = 2^{nd}\text{-order DR}\}, \\ m_0 &= m_2, \\ U &= \{u_1 = [x, v], u_2 = [x, v, a]\}, \\ PS &= \{ps_1 = 44 \text{ bytes}, ps_2 = 52 \text{ bytes}\}. \end{aligned} \tag{5.6}$$

The determination of the model switching time-out T_O mostly relies on the specific connection. In typical residence broadband connections, a short change that lasts for only seconds is regarded as transient [Marshall *et al.* 2008]. Therefore T_O is set to be 5 seconds in the experiment. The convergence period T_C is arbitrarily set to be 2 seconds in this conceptual experiment, since there is no well-accepted threshold for user perception of model convergence.

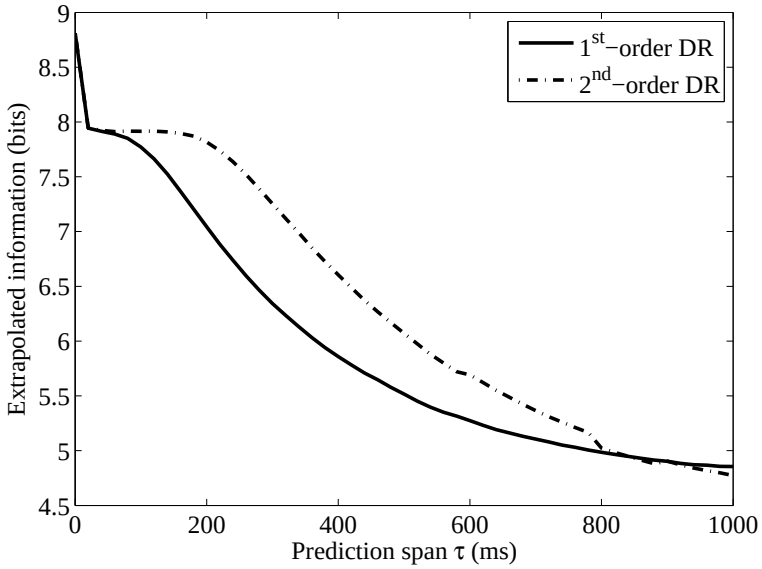
5.3.2 Extrapolation Selection Based on the Information Model

The key factor of the dynamic extrapolation model is the information model that serves as a performance measurement to select one of the candidate prediction models delivering the highest information rate to the remote host under changing network conditions. The information model characterizes how much of the predictability of the true motion is extrapolated from state derivatives by an extrapolation model. In the experiments, the training and test data each have a sample size of $N = 40000$. From the experimental studies in Section 4.3, such a sample size is unlikely to produce accurate information estimation using the simple algorithm. Therefore the

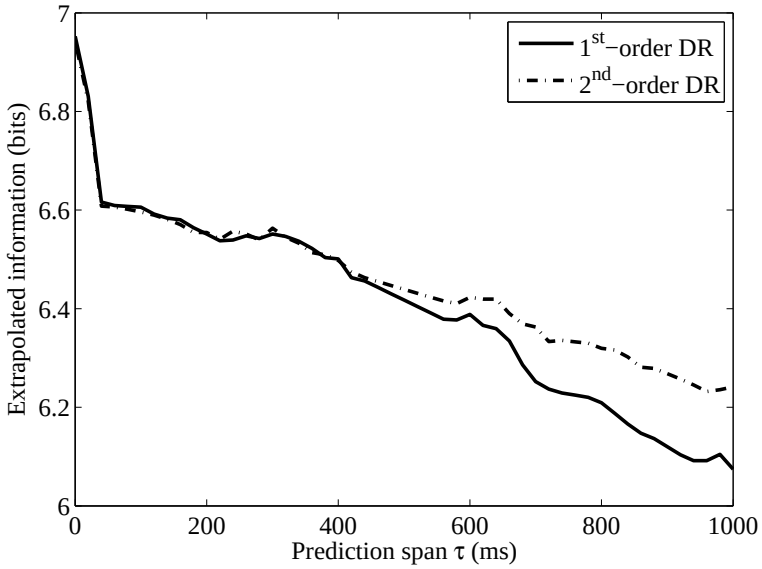
KDE algorithm is employed in this section to evaluate the information characteristics of the candidate extrapolation models.

Using (3.24), the extrapolated information by each of the two candidate models for increasing prediction spans is compared in Figure 5.6. For typical user motion from the FPS game (Figure 5.6(a)), an average amount of 8.8 bits data is needed to fully determine an entity state, of which about 7 bits can be extracted from the current second-order packet using standard second-order DR to estimate a future state at 200ms later. The future state can never be determined completely by the extrapolation equation because of the information that is missing. Consequently, inconsistency arises from uncertainty. As expected, including the additional acceleration does give the second-order extrapolation some advantage over the first-order equation. The second-order extrapolation utilizes more information in most of the prediction spans presented, indicating that the higher-order derivative provides a notable amount of knowledge about the entity states over short time periods. However, the information extrapolated by the second-order equation decays faster due to the sensitivity of the higher-order derivative to changes in motion status, so the first-order extrapolation is generally expected to be preferred for predicting future states on longer time scales. It is worth noting that the second-order extrapolation utilizes more information than the first-order method over an extended range of prediction spans, thus demonstrating that the interpolation (Section 5.3.1) has little effect on the overall motion since it only impacts the data in time scales under 100ms. The information metric results reflect the characteristics of human actions in the game.

From the measured extrapolated information in Figure 5.6(b), the motion in the racing game is generally smooth and stable since it requires only less than 7 bits data per simulation step to fully describe the entity state. In this scenario, players only move within the designed course and thus their motions are largely predictable. This feature is reflected by the information metric that first-order DR exhibits nearly equal predictability as second-order DR for short prediction spans. In contrast to the FPS motion, second-order DR exhibits information advantage over longer time scales. In the racing scenario, with a static finishing point and the constrained path,



(a)



(b)

Figure 5.6 Extrapolated information by the two candidate models for (a) the FPS motion and (b) the racing motion.

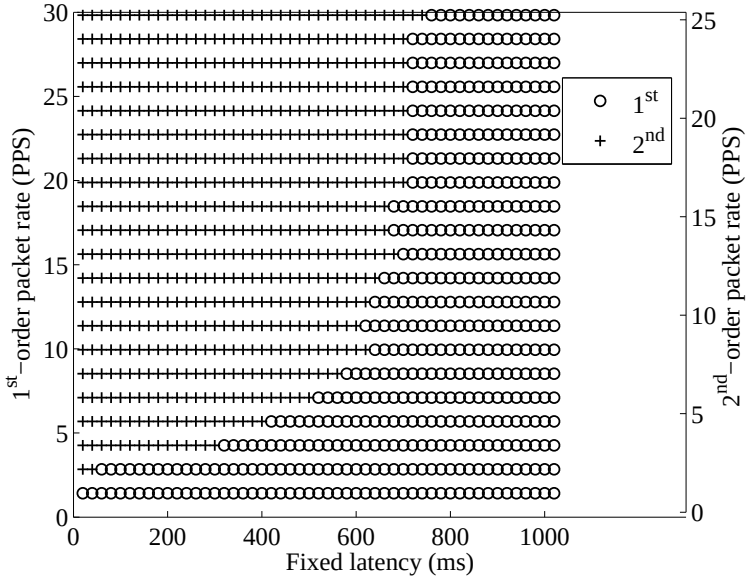
the entity motion would typically involve steady accelerations over extended periods.

In spite of the disadvantage in utilizing information at specific time instances, the first-order extrapolation model may compensate for the average information rate

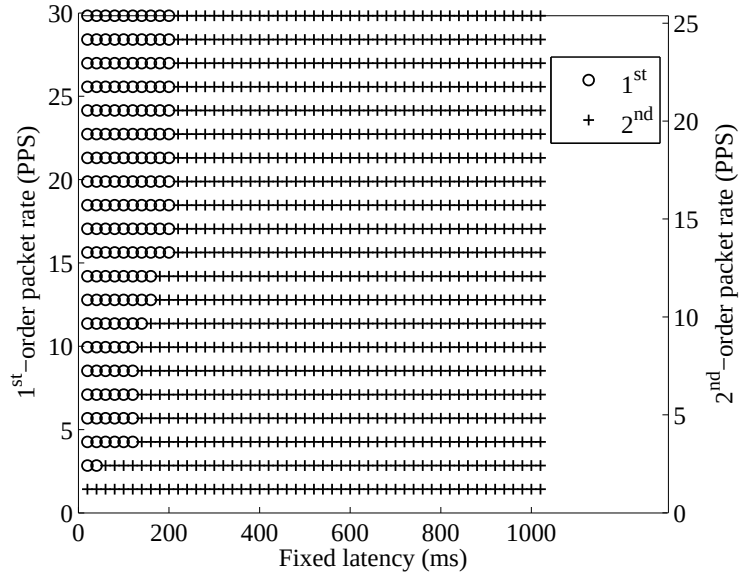
by faster update generation. The overall information rate provided by the synchronization messages and the corresponding extrapolation model also depends on the network conditions. For a given available bandwidth and fixed latency, the over-all information rate can be estimated from the information model, and the dynamic model switches the active extrapolation model to that which produces the higher information rate. Figure 5.7 shows the selected active models for varying bandwidth and latency settings based on the trained information models for the two motions. To put the data rate setting in some perspective, bandwidth is presented by packet rate (in PPS) instead of absolute bandwidth value.

For the FPS motion, the second-order extrapolation model is generally preferred in high bandwidth and low latency situations. The information advantage of the second-order update message, which diminishes with increasing prediction spans, comes with the cost of a larger packet size. Unless the packets are transmitted to the remote host in time, the additional information and traffic load will be wasted. In low bandwidth and high latency scenarios, the first-order extrapolation is preferred because it provides simple but stable motion trends that outperform the sensitive higher-order derivative information in longer term prediction. For the racing motion, on the other hand, second-order DR is generally the favorable model due to its extended information advantage over first-order DR. However a few exceptions can be found in situations with high bandwidth and low latency, where the functioning periods of the update messages reside in small prediction spans and the difference in information extrapolation between the two models is negligible. First-order DR is preferred for higher update rates. The contrast between the two game scenarios demonstrates that the information measurement is capable of capturing different types of user behavior and combining this factor with network conditions in making the decision on selecting the best extrapolation model. In the following experiments on the test motions, Figure 5.7 is used as look-up tables for selecting the best extrapolation model under different network conditions.

To evaluate the information model, the Information-Based Dynamic Extrapolation Model is carried out on the test datasets using the changing network settings listed in Table 5.1. The varying available bandwidth and latency reflect variations in



(a)



(b)

Figure 5.7 Extrapolation choices based on information rate under varying network conditions for (a) the FPS motion and (b) the racing motion.

network conditions due to players that share the same last mile link joining in and leaving, and/or cross-traffic from other applications. Figure 5.8 shows the impact on network conditions of transmitting either of the two types of ESUs over the bottleneck connection. For any given bandwidth, the overall data rates of the two

extrapolations are tuned at the same level as the available bandwidth to send as much data as possible and in the meantime avoid queuing delay. Therefore, it is obvious from Figure 5.8(b) that the second-order extrapolation always sends update packets at a rate about 20% lower than the first-order extrapolation because of its larger packet size, except in the cases where the bandwidths are high enough to allow both extrapolations to update at the highest rate of 30 PPS. Figure 5.8(c) shows the overall latency modeled in (5.5). The second-order extrapolation suffers from higher latencies than the linear extrapolation since it takes more time to transmit the larger packets through the bottleneck connection. The impact of including more data in one packet is thus clear from Figure 5.8. The second-order extrapolation faces negative impact on both update rate and latency as a result of the larger packet size. The overall performance of the two extrapolations under limited network resources depends on the trade-off between packet size and update rate.

The impact of varying the active extrapolation model and packet rate on the remote information rate and inconsistency is shown in Figure 5.9 and Figure 5.10. First, for each of the 10 sections with different network settings, an active extrapolation model is selected, based on the trained information model, to deliver the higher information rate (Figure 5.7). The selected model is shown in parts (a) of Figure 5.9 and Figure 5.10 with numbered notations (① for first-order DR and ② for second-order DR). Along with the average information rates rebuilt on the remote host by constantly using either of the two candidate models, the information rate of the dynamic model is also highlighted. The extrapolation model selection based on the trained information model (the numbered notations) is consistent with the actual information characteristics of the test data, and the dynamic model can accurately select the extrapolation model with the higher information rate under changing network conditions.

Parts (b) of Figure 5.9 and Figure 5.10 compare the remote inconsistency arising from using the two fixed extrapolation models and the dynamic model. Remote inconsistency (drift distance defined in (4.6)) is measured by simulation units, and averaged over every second. The interaction of the inconsistency that arises from employing either of the two fixed extrapolation models justifies the motivation of

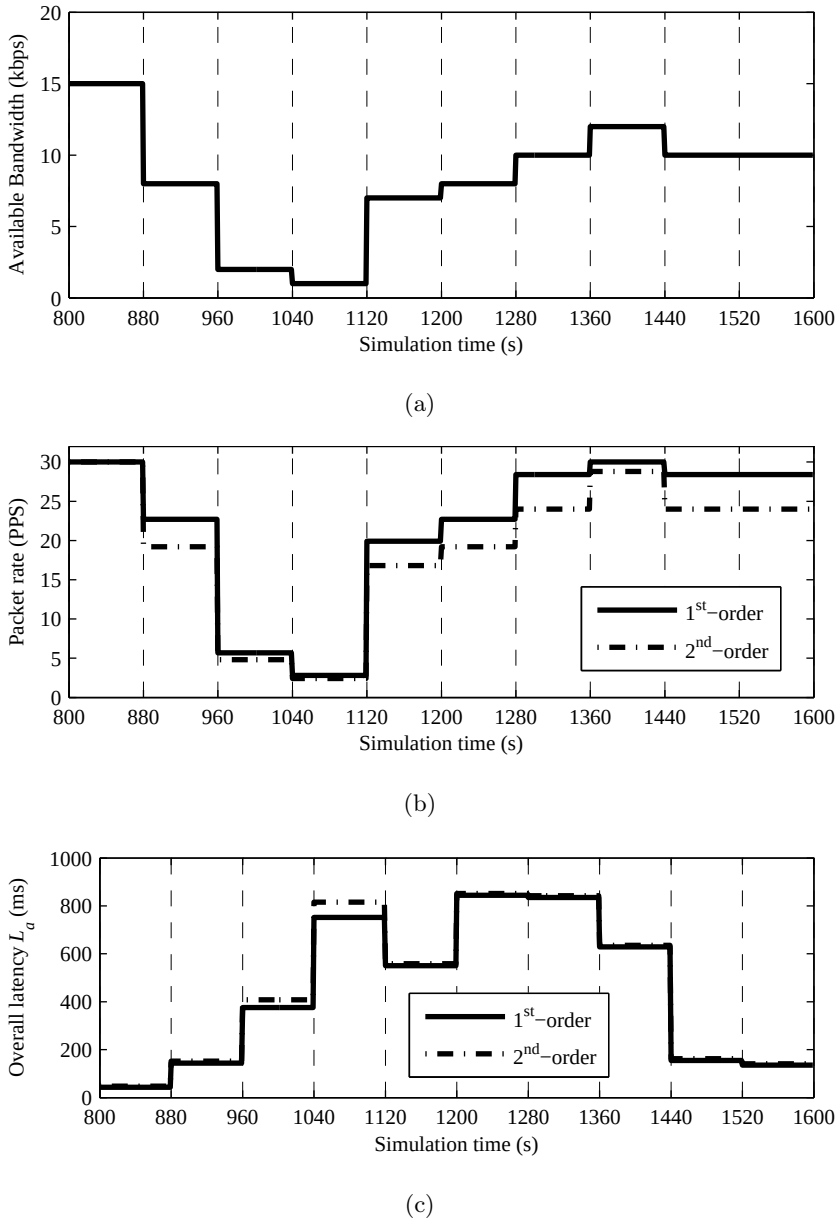


Figure 5.8 (a) Available bandwidth, (b) packet rate in PPS, and (c) overall latency as experienced by the test path simulation without the dynamic extrapolation model.

the dynamic extrapolation model because neither of the two fixed models minimizes remote inconsistency for all the network condition changes. Under changing network conditions, the 20% higher update rate of the first-order extrapolation could lead to contrary results in information rate and inconsistency compared to the second-order extrapolation. This also depends on the user behavior (evidence of such contrast

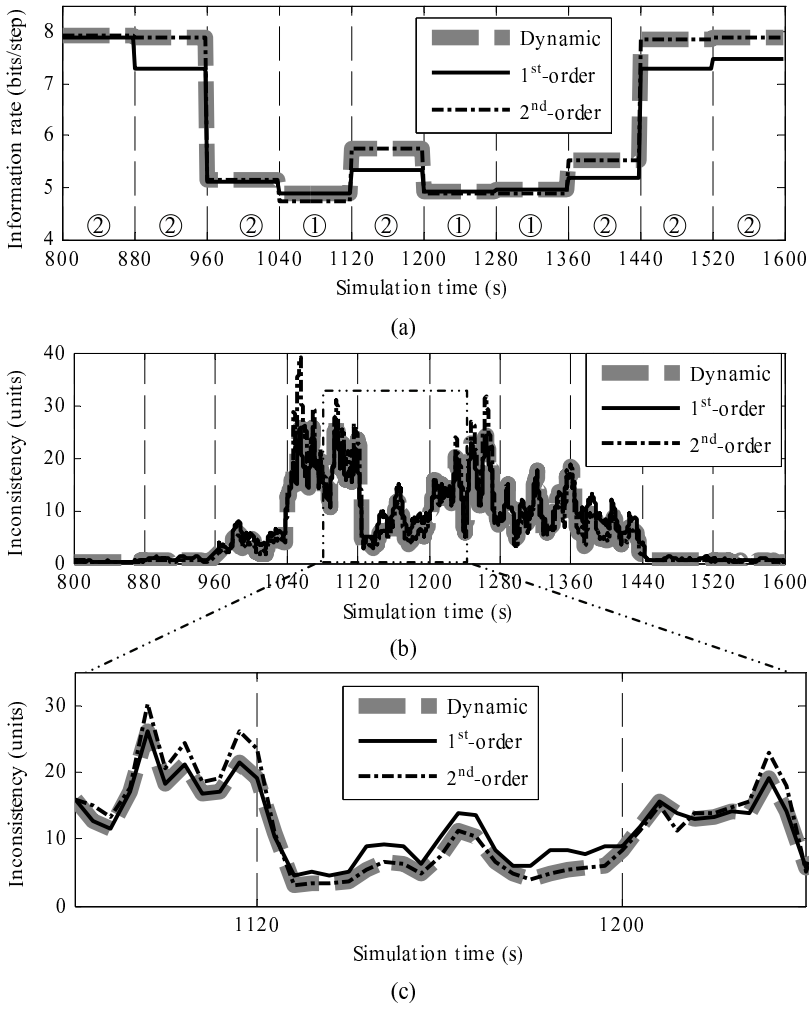


Figure 5.9 Comparisons of (a) average information rate, and (b) remote inconsistency among the two fixed extrapolation models and the dynamic extrapolation model for the FPS motion. A highlighted section of (b) is shown in (c).

can be found between frames 1040s and 1120s in Figure 5.9(b)). It is thus evident that by dynamically switching to the extrapolation model that produces the higher remote information rate, the proposed technique minimizes remote inconsistency for varying network conditions. The dynamic extrapolation model accurately chooses the extrapolation model that produces lower inconsistency and thus outperforms statically employing either of the two fixed extrapolation schemes. For the purpose of clarity, model switchings over highlighted periods are shown in parts (c) of Figure 5.9 and Figure 5.10.

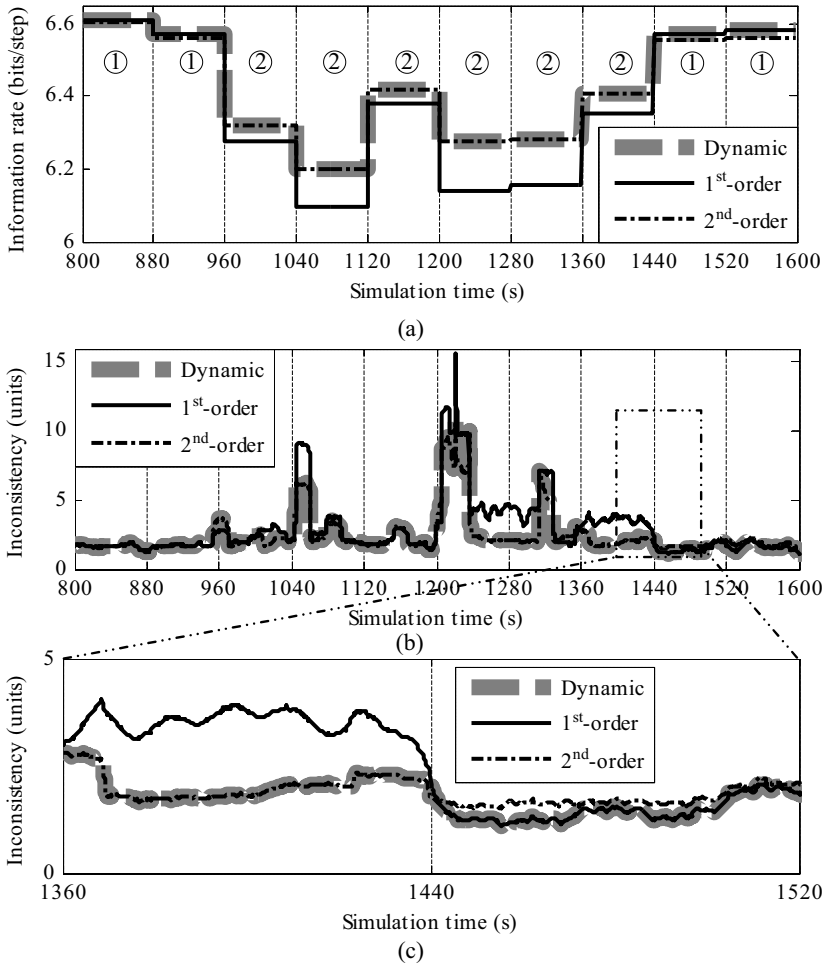


Figure 5.10 Comparisons of (a) average information rate, and (b) remote inconsistency among the two fixed extrapolation models and the dynamic extrapolation model for the racing motion. A highlighted section of (b) is shown in (c).

It is worth noting that in Figure 5.9(c), although the first-order extrapolation is selected to be the active model for the period starting from 1200 second, the second-order extrapolation gives lower inconsistency for a short period around 1210 second. The dynamic extrapolation model fails to pick this transient exception because the information-based model selection is the optimum in a statistical sense. It does not guarantee the best consistency at every time instance, but provides the most probable best model for long time implementation. Similar situations can also be found in Figure 5.10(b).

The core of the proposed Information-Based Dynamic Extrapolation Model is using

an information model to measure the remote information rate as a performance metric and to dynamically select the optimal extrapolation model for the underlying network conditions. The proposed information-based hybrid framework uses the information model to relate predictability in the user behavior and transmitted data, and hence combines both the application and network layer elements in adjusting system parameters. Such an approach guarantees that the usage of the available network bandwidth is maximized and the highest information rate about the true entity motion is reconstructed by the selected extrapolation method that best suits the user behavior.

Aside from minimizing the remote inconsistency, the Information-Based Dynamic Extrapolation Model also helps DIA designers to attain novel insights in examining and improving consistency maintenance. The information metric of the synchronization messages takes into consideration both factors that affect consistency maintenance, namely the suitability of the extrapolation model and network conditions. If an extrapolation model gives poor remote consistency, the information measurements can help identify whether the problem is the extrapolation equation failing to utilize the predictability in the entity motion (a fast information decay) or a poor network connection that cannot support the necessary data transmission. In the former case, improving the underlying network connection alone would be of little help. One could either replace the derivatives in the messages with some parameters containing more information about the entity state without changing the extrapolation algorithm (such as in the case of Neuro-reckoning [McCoy *et al.* 2007]), or switch to a better algorithm that extrapolates the predictability into the remote state model.

The dynamic extrapolation framework proposed in this chapter focuses on DIAs where the avatar motion plays a central role and the periodical *state updates* constitutes the major part of the overall game traffic. The entity state that evolves continuously with the passage of time is predictable from contextual dynamics. The impact of a lower update rate (i.e. delaying an update) could be compensated for by including more information in the packet to, possibly, reduce prediction error. On the other hand, a higher update rate could also make up for a reduced amount of

information per packet. It should be noted that critical events such as a car crashing or explosions are, in real systems, updated by non-periodical *event updates*. These are relatively rare in terms of network traffic caused and are thus not considered in this chapter.

In the experiments presented, the second-order extrapolation exhibits advantages over the first-order extrapolation in information utilization, and the active extrapolation model changes with network conditions. For the first-order extrapolation, the disadvantage of less information per packet can be compensated for by a faster update under some network conditions. However, the different results in active model selection for the two game motions (Figure 5.7) indicate that the overall outcome of the trade-off between packet size and packet rate depends on the information characteristics of the particular user motion, and is application-dependent. The results presented here are representative of applications where more accurate physics are integrated for higher fidelity simulation (such as aircraft simulations and games that incorporates real world physics). These applications would thus have similar second-order dynamics on the interpolation scales of interest to human perception, and the effects of the player manipulations probably only manifest themselves on longer time scales. For these applications, the second-order extrapolation model would have an information advantage, as shown in the experiment.

However it should be mentioned that including more data in the update packets does not always guarantee better consistency [McCoy *et al.* 2007], because whether or not a higher-order extrapolation can utilize the additional information in the messages depends on how well the model fits the entity motion. It is then possible that for some application types a simple prediction method, such as the linear model, would utilize more information and therefore be always selected as the active model since it provides the higher information rate without causing heavier traffic load. In such cases employing the proposed dynamic framework, which would make a correct but static choice of the active model, seems unnecessary.

In the experimental results presented, the information model built from the training data exhibits a good generalization on the test data. Using the information model

built from the training motions, the dynamic extrapolation model can precisely pick the active extrapolation model that gives higher information rate on the test motion data. This provides a good indication that the information model successfully captures the dependence or predictability of the entity motion. Generally speaking, in order for the trained information model to be used in the dynamic extrapolation model, the application state dynamic is required to be stationary so that the training and test data may be considered statistically homogeneous. This is unlikely to be problematic for most DIAs, because the intensive user movement over long time periods means that the limited spatial environment is well explored and therefore sufficient data can be collected for accurate information calculation.

5.4 Concluding Remarks

The first part of this chapter presents the design of a novel mechanism for controlling PCM operation, referred to as the Information-Based Dynamic Extrapolation Model. The model is designed to minimize inconsistency, which is one of the key performance requirements of a DIA, through maximizing the efficiency of information communication of the data transmission. It operates by evaluating the suitability of a set of candidate extrapolation models using the information model, and adjusting the active extrapolation model and its data transmission rate to match the inferred network conditions. Such an approach guarantees the highest information rate about the true entity motion is delivered to the remote host for producing the entity state model.

The most critical component of the proposed framework is the information metric that measures the information utility of the candidate extrapolation models. The information model of PCMs quantifies the amount of information that is extrapolated by a predictive scheme from the synchronization messages transmitted. Such information provides predictability for the future states of the entity. The information model provides an analytical measurement of whether the data included in a message is worthy of the network traffic load it causes. Therefore it combines

characteristics of the user behavior and networked data transmission, and allows for a joint optimization of PCM operation.

In the second part of this chapter, the performance of the Information-Based Dynamic Extrapolation Model is analyzed through experimental studies on two DIA scenarios under simulated network conditions. Results show that by maximizing the information efficiency of the data transmission on the bottleneck link, the Information-Based Dynamic Extrapolation Model can minimize the level of inconsistency arising in the application. The active extrapolation model selected under a specific network condition also varies among entity motions that exhibit different information characteristics. Although only two polynomial extrapolation models are examined, the proposed dynamic extrapolation model is a general framework that can include any form of extrapolation algorithm.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, a novel information model that employs an information theoretic approach has been introduced for the analysis of Predictive Contract Mechanisms — a group of widely deployed techniques designed to reduce network traffic in a DIA based on the remote extrapolation of entity states. In this chapter the most important issues and the primary contributions that are presented in this thesis are reviewed. Some future directions for research based on the results are also discussed.

6.1.1 The Information Model for PCMs

This thesis has proposed the novel concept of using information theory to evaluate the operation of PCMs. A theoretic examination of such techniques is introduced with respect to their efficiency in communicating reduced state information in return for perceptually tolerable inconsistency across a DIA. In particular, the information model introduces the use of concepts derived from information theory to quantify the information generated by the user behavior and transmitted by PCMs to the remote host. This information measures the amount of predictability that can be used to estimate the remote entity state model from infrequent ESUs through the

use of predictive schemes. The proposed information model provides a theoretical foundation for a reinterpretation of PCMs as a form of lossy video compression and further optimization of PCMs in resolving the trade-off between consistency and throughput. Simulations were performed to validate the proposed approach in realistic DIAs.

In developing and subsequently implementing the information model, a number of observations have been made from which a number of conclusions can be drawn:

1. The entropy of an entity motion trajectory provides a quantified measure of the minimal amount of data required to eliminate the uncertainty of the entity state and fully determine its value, given no prior knowledge about the motion (such as historical trajectory or user behavior patterns).

2. The mutual information measurement $I(\boldsymbol{u}; x_\tau)$ between the instantaneous motion status (e.g. entity position, velocity, etc.) in an ESU and the entity state at varying future time instances enables a quantified evaluation of the predictability of the user behavior. This predictability, as defined in (4.1), indicates the extent to which the uncertainty of the entity state at a future time could be reduced through efficient exploration of the knowledge encapsulated in the current motion status. Generally, including more motion status in an ESU improves the predictability. However, the predictability decays with increasing prediction spans, which is why the remote host needs continuous ESUs to maintain consistency.

3. The mutual information $I(\tilde{x}_\tau; x_\tau)$ between the true entity state and the modeled state measures the amount of predictability that is actually utilized by an extrapolation model to build the remote entity model. This predictability provides a quantified evaluation of the reduction in data transmission due to the use of statistical temporal dependence within the user behavior. Unlike the encapsulated predictability, a larger ESU with more state derivative information does not guarantee higher extrapolated predictability. The extrapolation model under use must properly capture the user behavior patterns to utilize the delivered information efficiently.

4. The concept of the functioning period of an ESU enables a life-cycle analysis of the transmitted data and incorporates network latency, the key inhibiting factor that DIAs must combat, into the information model. Network latency imposes negative impact on consistency maintenance by making the information in an ESU outdated by the time the update arrives on the remote host, and thus causing information loss.

5. The information model is based on the statistical dependence between the content of the ESUs and the true or modeled entity state, and is independent of the details of operation of the extrapolation model. Therefore the information model is applicable to any form of PCMs including standard and novel techniques, i.e. it is model agnostic.

6. The mutual information measurement of motion trajectories of different entities quantifies the intensity of cross-entity interactions from the information theory perspective. This information reveals the predictability between closely interacting entities and provides a foundation for using such statistical relations to further reduce data transmission in a DIA, by predicting the state of an entity from the state of another object identified as closely coupled with the former.

The primary conclusion reached during the design and development of the information model can be summarized as an analogy between PCMs and video compression. Namely, from the point of view of exploiting predictability in the user behavior, the operation of PCMs can be seen as a form of information reduction and compression: by allowing for local modeling error within the threshold, the original information rate $H(x)$ of the entity dynamic is reduced to the local storing rate $R_{s,local}$; the network latency endured by the transmission adds further information loss and only an information rate of $R_{s,remote}$ arrives on the remote host, which is finally reconstructed at a lower rate of $R_{u,remote}$ by the imperfect extrapolation model. This analysis enables separate investigations of each factor that contributes to inconsistency arising from the execution of a PCM. Similar to the situation in video compression, the reduction in data transmission using PCMs comes from a mixture of

lossless compression (the utilized predictability) and lossy compression (irreversible information loss induced by the local error threshold, network latency and imperfect extrapolation). The information loss is reflected in inconsistency arising across the DIA. Such an information perspective facilitates DIA designers in attaining a deeper understanding of PCM operation in terms of an efficient exploitation of the predictability in the user behavior. It also aids them in developing future improvements of PCMs for better consistency maintenance in a DIA.

6.1.2 The Information-Based Dynamic Extrapolation Model

This thesis has also proposed the concept of a novel adaptive multiple-model approach for minimizing inconsistency across a DIA using rate-based PCMs. In particular, the Information-Based Dynamic Extrapolation Model introduces a general framework for selecting an appropriate remote extrapolation model and an optimal update rate based on performance evaluation using the information model and inferred network conditions. Simulations were performed to validate the proposed approach.

In developing and subsequently implementing the Information-Based Dynamic Extrapolation Model, a number of observations have been made from which a number of conclusions can be drawn:

1. It is demonstrated that adjusting PCM configurations solely based on factors at either the application layer (e.g. local prediction error) or the network layer (i.e. the available bandwidth) does not guarantee the optimization of consistency because the efficiency of the transmitted data in providing information to model the entity state is not considered.

2. In many cases, increased information utilization for an extrapolation model comes at the price of a larger volume of state derivatives in the transmitted ESUs. The suitability of employing such an extrapolation model depends on whether the data included in a message is worthy of the increased network

traffic load it causes. Based on the experimental studies on first and second-order DR, the lower information utilization of first-order DR could, in some cases, be compensated for by a faster update rate. This also depends on the user behavior and the underlying network conditions.

3. It was observed that by dynamically switching extrapolation formulas and the update rate in response to changing network conditions, the Information-Based Dynamic Extrapolation Model can minimize remote inconsistency when compared to static extrapolation schemes. Based on the information model trained to capture the information characteristics of the user behavior, the proposed dynamic extrapolation framework accurately selected an appropriate extrapolation model that produced the highest information rate about the true entity motion under specific network conditions.

The primary conclusion reached during the development of the Information-Based Dynamic Extrapolation Model can be summarized as follows: by measuring the predictability provided by the transmitted data, the information model enables the proposed dynamic extrapolation framework to combine both the application layer and the network layer factors to realize a joint optimization of PCMs to minimize remote inconsistency arising from rate-based PCM operation. This is achieved through the optimized usage of the available bandwidth on the bottleneck link of a DIA. Such an approach is in contrast to the existing adaptive schemes that either ignore the impact of network performance on inconsistency or only employ a single extrapolation model. Consequently, the Information-Based Dynamic Extrapolation Model achieves the best consistency that could be achieved by the set of candidate extrapolation models under constrained and varying network conditions.

6.2 Future Work

The information model and dynamic extrapolation framework presented in this thesis have been demonstrated to be effective tools for examining and improving consistency maintenance in DIAs. This section presents some suggestions for future work

to expand on the work presented in this thesis.

6.2.1 Extended Study of the Information Model

The information model proposed in this work provides a theoretical perspective towards operation of PCMs, in terms of measuring how much predictability in user behaviors is drawn, transmitted and then exploited. Such work could be extended to gain more insights about the information flow in consistency maintenance. In particular, further analysis and improvement can be considered in the following four respects:

- In Figure 5.7, the favorable choices of extrapolation model under varying network conditions are presented. It seems, especially in Figure 5.7(a), that there is a fine boundary curve that separates the two "choice zones". This boundary certainly depends on numerous factors such as user behavior, extrapolation model, and packet structure. If a closed form description of such a boundary could be drawn, it will bring profound understanding of the predictability within entity motions and also improvement of the dynamic extrapolation framework in that the extrapolation model selection could be based on a function describing the boundary, instead of a pre-trained look-up table as in Figure 5.7.

- By comparing Figure 4.18 and Figure 4.20, it is identified that large amount of information available in the NR packets is not properly used because of the simple linear extrapolation model currently used. It is then necessary to improve the extrapolation for an even better consistency maintenance. To improve the usage of the available information in the NR packets, an extrapolation function could be trained so that each NR velocity value is mapped to a segment of trajectory. In doing so, the remote entity state model is formed as a combination of "prototype motions" or "base functions" that more precisely reflect the original entity state than the simple linear extrapolation. Of course,

such an approach would compromise the transparency of NR to the remote host and increase implementation overhead.

- In Section 4.4, it has been demonstrated that information measurement can reflect the evolution of the interaction between entities. Only observations of such interaction measurement is provided in this work. However, such an approach could be employed to improve the regulation of the update packets. For example, based on the information measurement of the intensity of user interaction, interest management techniques could distinguish inter-user relations and only send frequent updates between those who are closely interacting. Such an approach improves the efficiency of the bandwidth usage.

- Currently, all information concerning entity state is treated with equal importance. However, it is well understood that human perception imposes different importance to features in entity motion. Therefore, the information model could be extended by considering such bias of human perception. Based on some human vision analysis and psychological study, different importance could be assigned to the raw information content and form a weighted information measurement.

6.2.2 Effect of Realistic Network Environments

For the convenience of illustrating the concept of using the information measurements to examine and improving PCM operation, simplified yet realistic network environments have been employed throughout this thesis. This work could be expanded by considering the effect of even more realistic network environments on the information-based approaches. In particular, realistic network conditions can be considered in the following two respects:

- In Chapters 3 and 4, the information measurement of the remote extrapolation is modeled under fixed latency, assuming that all the ESUs experience the same network delay and, thus, the same information loss caused by network latency. However, as discussed in Chapter 1, packet transmission in the

realistic Internet endures unpredictable variation in latency, also referred to as jitter. Jitter results in a variable functioning period for each transmitted ESU and consequently a variable amount of information carried by each ESU. Incorporation of jitter will expand the information model in two ways. First, the evaluation of PCMs can be more accurate compared to the average assessment in the work presented in this thesis. Second, the negative impact of jitter on consistency maintenance (Section 1.1.4) can be examined from the information theory perspective.

- In Chapter 5, the network condition estimator in the Information-Based Dynamic Extrapolation Model is implemented simply as two control parameters AB and L_a. An interesting extension to this work is to implement the dynamic extrapolation framework in a realistic network environment, which would facilitate further examination of its performance in the face of issues such as jitter and packet loss. Also, some network condition estimator algorithm must be implemented as a part of the framework. Such estimation algorithms generally introduce additional information to be transmitted over the network. A thorough study of the impact of such algorithms on the dynamic extrapolation framework would extend this work in implementing the framework in realistic network environments.

6.2.3 On-Line Information Estimation

Throughout this thesis, the information model for a specific user behavior is obtained through an off-line training process that is generally computationally intensive. The development of a fast information estimation algorithm that enables on-line information estimation would expand the work presented in this thesis in three respects:

- In the Information-Based Dynamic Extrapolation Model, the current off-line information model reflects the information characteristics of the user behavior as a whole. In other words, it measures the average predictability of the

entity motion. An adaptive information model that can be trained in real-time using historical motion data within a sliding-window can account for possible changes of user behavior during DIA deployment, and thus allows for a detailed evaluation of the trade-off between packet rate and packet size, and enables more accurate active model selections.

- An on-line information estimation would allow the usage of the information measurements to act as a local consistency metric that governs the ESU generation. In particular, an information-based PCM could be developed, in which the local host only generates and sends an ESU to the remote host when the estimated information quality of the local entity model decays below a pre-determined threshold. Such an approach could guarantee the level of information rate provided for the remote host to reconstruct its entity state model.

- An on-line information estimation would enable considerable extensions to the investigation of cross-entity predictability presented in Section 4.4. For example, it would be possible to evaluate inter-entity dependence using mutual information at run-time to detect a group of entities whose states are closely coupled with each other. If the interaction among the entities is identified as one of the typical types (Section 4.4.1), the action and state of one entity can then be modeled based on the behavior of the other entities in the group. Such an information measurement facilitates the investigation of user behavior and interaction from a theoretic perspective. It also allows the development of collaborative predictive approaches that would further reduce the requirement of data transmission to maintain consistency in a DIA.

The on-line information estimation algorithm must be able to operate on small datasets and be computationally efficient. However, neither of the two algorithms used in this thesis satisfies such requirements, since the simple algorithm of occurrence counting demands large amounts of data and the KDE algorithm is computationally complex. A plausible solution is to employ Fast Gaussian Transformation to improve the computational efficiency of the KDE algorithm [Elgammal *et al.* 2003;

Yang *et al.* 2003] so that a real-time information estimation with a short period of historical motion data can be achieved.

6.2.4 Improving the Dynamic Extrapolation Framework

In Chapter 5, only two extrapolation models are considered in the candidate set. Although first and second-order DR have gained popularity in most DIA systems, including more extrapolation models would strengthen the work presented in this thesis by allowing for a better performance of the dynamic extrapolation framework. The Information-Based Dynamic Extrapolation Model can be improved in three respects:

- As a straightforward extension, other extrapolation models could be included in the candidate set for the investigation of their performance from the information theory perspective. Given wider options, the Information-Based Dynamic Extrapolation Model could further reduce the inconsistency arising across a DIA. Examples of extrapolation models that could be included in the candidate set include, but are not limited to, Neuro-reckoning [McCoy *et al.* 2007], Dynamic Target-Interception Models [Stolzenburg *et al.* 2002; Belkhouche *et al.* 2007; McCoy 2007], and other predictive statistical user models [Zukerman and Albrecht 2001]. It should be noted that these advanced predictive schemes generally require a training process to capture patterns in a particular user behavior and are thus application-dependent. However, the statistical behavior models are usually trained in an off-line manner. Therefore including complicated extrapolation models does not affect the operation of the dynamic extrapolation model.

- In the proposed dynamic extrapolation framework, at any time instance (except for the conversion period T_C) the active model is executed independently (without any mutual interaction with the other models). In contrast, Interacting Multiple-Model (IMM) algorithms combine the outputs from multiple extrapolation models based on a probabilistic measure of their performance [Rago

and Mehra 2000; Cooperman 2002; Farmer *et al.* 2002; Burkert *et al.* 2004]. Such an approach has shown excellent potential for estimating an accurate motion model for the current user behavior and could provide a means for selecting extrapolation models in an adaptive extrapolation scheme. A thorough analysis of such techniques therefore appears warranted in order to investigate the possibility of incorporating IMM algorithms in the proposed dynamic extrapolation framework.

- The proposed dynamic extrapolation model is currently focused on rate-based PCMs, which have direct control over the data transmission rate through the constant update packet rate. It would be an interesting extension to this work to incorporate threshold-based PCMs in the proposed framework. In order to control the data transmission rate through the local error threshold, there would need to be a form of mapping to explicitly reflect the relationship between a local error threshold and the resultant data rate.

  In video compression, the encoder faces the same problem. By adjusting the quantization scale (Figure 2.10), it attempts to allocate the same number of bits to encode each frame (or a group of frames) so that the encoded video is transmitted at a constant data rate. This is achieved by a "Rate-Distortion" analysis of the raw video, where the relationship between the quantization scale and the resultant data rate is established either by formal analysis or experiment [Lin and Ortega 1998; Cook *et al.* 2006; Cover and Thomas 2006].

  In light of the analogy between video compression and PCMs (Section **??**), a similar approach can be introduced. A "Threshold-Rate" analysis can be drawn from experimental studies on a specific entity motion. Pairs of threshold and the resultant data transmission rate can be tabulated to describe the relationship between the two, so that a proper threshold value can be found to produce the desired data rate that matches the available bandwidth. A typical Threshold-Rate relationship is expected to take the form shown in Figure 6.1, where a lower threshold value generally causes heavier network traffic.

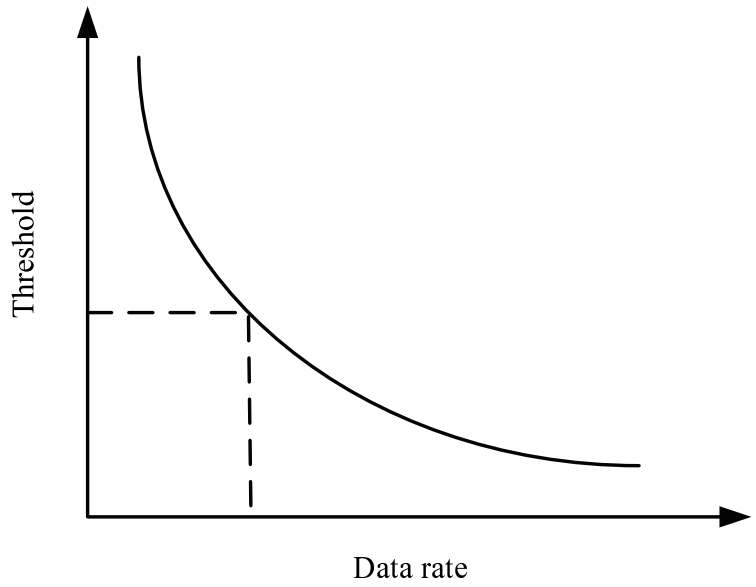


Figure 6.1 A simple illustration of the "Threshold-Rate" relationship in a PCM for a particular user motion.

6.3 Concluding Remarks

In this thesis the use of concepts and techniques derived from information theory have been used to examine and improve the efficacy of PCMs in DIAs. It has been shown that by employing information metrics as a quantified measure of the efficiency of PCMs in communicating reduced state updates in return for controlled inconsistency, the operation of PCMs can been regarded as a form of information reduction and compression.

It is hoped that such an information theory perspective may help designers to attain a deeper understanding of PCM operation, and the techniques proposed may be used as the basis for optimizing PCMs and developing improved techniques to support future generations of large-scale DIAs.

References

Aggarwal, S., H. Banavar, A. Khandelwal, S. Mukherjee, and S. Rangarajan (2004). Accuracy in Dead-Reckoning Based Distributed Multi-Player Games. In *Proc. 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '04)*, Portland, Oregon, USA, August 2004, pp. 161–165.

Aguilar, M., Y. Barniv, and A. Garrett (2003). Prediction of Pitch and Yaw Head Movements via Recurrent Neural Networks. In *Proc. of 2003 International Joint Conference on Neural Networks (IJCNN 2003)*, Volume 4, Portland, Oregon, USA, July 2003, pp. 2813–2818.

Akyildiz, I. F. and W. Yen (1996). Multivideo Group Synchronization Protocols for Integrated Services Networks. *IEEE Journal on Selected Areas in Communications 14*(1), 162–173, January 1996.

Armitage, G. (2003). An Experimental Estimation of Latency Sensitivity In Multiplayer Quake 3. In *Proc. 11th IEEE International Conference on Networks (ICON2003)*, Sydney, NSW, Australia, September/October 2003, pp. 137–141.

Atwood, N. K., K. A. Quinkert, M. R. Campbell, K. F. Lameier, B. C. Leibrecht, and W. J. Doherty (1991). Combat Vehicle Command and Control Systems: Training Implications Based on Company-Level Simulations. Technical Report A064642, U.S. Army Research Institute fot the Behavioral and Social Sciences, Alexandria, Virginia, USA, December 1991.

Babski, C., S. Raupp-Musse, D. Thalmann, S. Benford, A. Bullock, J. Tromp, S. Carion, N. Magnenat-Thalmann, Y. Chrysanthou, M. Slater, A. Steed,

M. Usoh, N. Farcet, E. Frécon, J. Harvey, N. Kuijpers, T. Rodden, G. Smith, G. Van Liempd, and N. Kladias (1999). The COVEN Project: Exploring Applicative, Technical, and Usage Dimensions of Collaborative Virtual Environments. *Presence: Teleoperators & Virtual Environments 8*(2), 218–236, April 1999.

Balch, T. (2000). Hierarchic Social Entropy: An Information Theoretic Measure of Robot Group Diversity. *Autonomous Robots 8*(3), 209–237, June 2000.

Barrus, J. W., R. C. Waters, and D. B. Anderson (1996). Locales: Supporting Large Multiuser Virtual Environments. *IEEE Computer Graphics and Applications 16*(6), 50–57, November 1996.

Bassiouni, M. A., M.-H. Chiu, M. Loper, M. Garnsey, and J. Williams (1997). Performance and Reliability Analysis of Relevance Filtering for Scalable Distributed Interactive Simulation. *ACM Trans. Model. Comput. Simul. 7*(3), 293–331, July 1997.

Behnke, S., A. Egorova, A. Gloye, R. Rojas, and M. Simon (2004). Predicting Away Robot Control Latency. In *RoboCup 2003*, Volume 3020 of *Lecture Notes in Computer Science*, pp. 712–719, August 2004.

Beigbeder, T., R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool (2004). The Effects of Loss and Latency on User Performance in Unreal Tournament 2003®. In *Proc. 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '04)*, Portland, Oregon, USA, August 2003, pp. 144–151.

Belkhouche, F., B. Belkhouche, and P. Rastgoufard (2007). Parallel Navigation for Reaching a Moving Goal by a Mobile Robot. *Robotica 25*(1), 63–74, September 2007.

Benford, S., J. Bowers, L. E. Fahlén, and C. Greenhalgh (1994). Managing Mutual Awareness in Collaborative Virtual Environments. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST '94)*, 1994, pp. 223–236.

Benford, S., C. Greenhalgh, T. Rodden, and J. Pycock (2001). Collaborative Virtual Environments. *Commun. ACM 44*(7), 79–85, July 2001.

Bhatti, N., A. Bouch, and A. Kuchinsky (2000). Integrating User-Perceived Quality into Web Server Design. *Computer Networks 33*(1-6), 1–16, June 2000.

Bhola, S., G. Banavar, and M. Ahamad (1998). Responsiveness and Consistency Tradeoffs in Interactive Groupware. In *Proc. 7th Annual ACM Symposium on Principles of Distributed Computing (PODC '98)*, Puerto Vallarta, Mexico, June/July 1998, pp. 324.

Blanchard, C., S. Burgess, Y. Harvill, J. Lanier, A. Lasko, M. Oberman, and M. Teitel (1990). Reality Built for Two: A Virtual Reality Tool. In *Proc. 1990 Symposium on Interactive 3D Graphics (SI3D '90)*, Snowbird, Utah, United States, 1990, pp. 35–36.

Blau, B., C. E. Hughes, M. J. Moshell, and C. Lisle (1992). Networked Virtual Environments. In *Proc. 1992 Symposium on Interactive 3D Graphics (SI3D '92)*, Cambridge, Massachusetts, USA, 1992, pp. 157–160.

Blow, J. (1998). A Look at Latency in Networked Games. *Game Developer 5*(7), 28–40, July 1998.

Bouillot, N. and E. Gressier-Soudan (2004). Consistency Models for Distributed Interactive Multivideo Applications. *SIGOPS Oper. Syst. Rev. 38*(4), 20–32, 2004.

Boulanger, J.-S., J. Kienzle, and C. Verbrugge (2006). Comparing Interest Management Algorithms for Massively Multiplayer Games. In *Proc. 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '06)*, Singapore, October 2006, pp. 6.

Burkert, T., J. Leupold, and G. Passig (2004). A Photorealistic Predictive Display. *Presence: Teleoperators & Virtual Environments 13*(1), 22–43, February 2004.

Cai, W., F. B. Lee, and L. Chen (1999). [an auto-adaptive dead reckoning algorithm for distributed interactive simulation]. In *Proc. 13th Workshop on Parallel and Distributed Simulation*, Atlanta, GA, USA, May 1999, pp. 82–89.

Calvin, J. O., C. J. Chiang, S. M. McGarry, S. J. Rak, and D. J. Van Hook (1999). Design, Implementation, and Performance of the STOW RTI Prototype (RTI-s). In *Spring SIW Workshop*, 1999.

Calvin, J. O., A. Dickens, B. Gaines, P. Metzger, D. Miller, and D. Owen (1993). The SIMNET Virtual World Architecture. In *Proc. 1993 EEE Virtual Reality Annual International Symposium*, Seattle, WA, USA, September 1993, pp. 450–455.

Calvin, J. O., J. Seeger, G. D. Troxel, and D. J. Van Hook (1995). STOW Realtime Information Transfer and Networking System Architecture. In *Proc. 12th Workshop on Standards for the Interoperability of Distributed Simulations*, Orlando, FL, USA, March 1995, pp. 343–353.

Capin, T., H. Noser, D. Thalmann, I. Sunday Pandzic, and N. Thalmann (1997). Virtual Human Representation and Communication in VLNet. *IEEE Computer Graphics and Application 17*(2), 42–53, March/April 1997.

Capps, M., D. McGregor, D. Brutzman, and M. Zyda (2000). NPSNET-V. A New Beginning for Dynamically Extensible Virtual Environments. *IEEE Computer Graphics and Applications 20*(5), 12–15, September/October 2000.

Capps, M. and D. Stotts (1997). Research Issues in Dveloping Networked Virtual Realities: Working Group Report on Distributed System Aspects of Sharing a Virtual Reality. In *Proc. 6th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Cambridge, MA, USA, June 1997, pp. 205–211.

Carlsson, C. and O. Hagsand (1993). DIVE — A Multi-User Virtual Reality System. In *Proc. 1993 IEEE Virtual Reality Annual International Symposium*, Seattle, WA, USA, September 1993, pp. 394–400.

Castronova, E. (2002). On Virtual Economies. *CESifo Working Paper Series* (752), 43, July 2002.

Chen, D. and C. Sun (1999). A Distributed Algorithm for Graphic Objects Replication in Real-Time Group Editors. In *Proc. International ACM SIGGROUP Conference on Supporting Group Work*, Phoenix, Arizona, USA, November 1999, pp. 121–130.

Chen, L. and G. Chen (2005). A Fuzzy Dead Reckoning Algorithm for Distributed

Interactive Applications. In *FSKD 2005*, Volume 3614 of *Lecture Notes in Computer Science*, pp. 961–971, July 2005.

Cheshire, S. (1996). Latency and the Quest for Interactivity. In *White paper commissioned by Volpe Welty Asset Management, L.L.C., for the Synchronous Person-to-Person Interactive Computing Environments Meeting*, San Francisco, USA, November 1996, pp. 8.

Chim, J., R. W. H. Lau, H. V. Leong, and A. Si (2003). CyberWalk: A Web-Based Distributed Virtual Walkthrough Environment. *IEEE Transactions on Multivideo 5*(4), 503–515, December 2003.

Choukair, Z. and D. Retailleau (2000a). Integrating QoS toCollaborative Distributed Virtual Reality Applications. In *Proc. 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. (ISORC 2000)*, Newport Beach, California, USA, March 2000, pp. 320–327.

Choukair, Z. and D. Retailleau (2000b). A QoS Model for Collaboration Through Distributed Virtual Environments. *Journal of Network and Computer Applications 23*(3), 311–334, July 2000.

Chris, G. (1999). *Large Scale Collaborative Virtual Environments*. Springer-Verlag, 1999.

Churchill, E. F., D. N. Snowdon, and A. J. Munro (Eds.) (2001). *Collaborative Virtual Environments: Digital Places and Spaces for Interaction* (1st ed.). Secaucus, NJ, USA: Springer-Verlag New York, Inc., April 2001

Claypool, M. (2005). The Effect of Latency on User Performance in Real-Time Strategy Games. *Computer Networks 49*(1), 52–70, September 2005.

Claypool, M. and K. Claypool (2006). Latency and Player Actions in Online Games. *Commun. ACM 49*(11), 40–45, November 2006.

Cohen, P., B. Heeringa, and N. Adams (2002). Unsupervised Segmentation of Categorical Time Series into Episodes. In *Proc. 2002 IEEE International Conference on Data Mining*, Maebashi City, Japan, March 2002, pp. 99–106.

Comerford, R. (1996). Interactive Video: An Internet Reality. *IEEE Spectrum 33*(4), 29–32, April 1996.

Cook, G. W., J. Prades-Nebot, Y. Liu, and E. J. Delp (2006). Rate-Distortion Analysis of Motion-Compensated Rate Scalable Video. *IEEE Transactions on Image Processing 15*(8), 2170–2190, 2006.

Cooperman, R. (2002). Tactical Ballistic Missile Tracking using the Interacting Multiple Model Algorithm. In *Proc. 5th International Conference on Information Fusion*, Volume 2, 2002, pp. 824–831.

Cosby, L. N. (1995). SIMNET — An Insider's Perspective. In *Proc. Conference of Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, Orlando, FL, USA, April 1995, pp. 59–72.

Cover, T. M. and J. A. Thomas (2006). *Elements of Information Theory* (2nd ed.). New Jersey, USA: John Wiley & Sons, Inc., Hoboken, July 2006.

Cronin, E., A. R. Kurc, B. Filstrup, and S. Jamin (2004). An Efficient Synchronization Mechanism for Mirrored Game Architectures. *Multivideo Tools and Applications 23*(1), 7–30, May 2004.

CSCW (1986). Proc. 1986 ACM Conference on Computer-Supported Cooperative Work. Austin, Texas, USA. ACM. Conference Chair-Krasner, Herb and Program Chair-Greif, Irene, December 1986.

Das, T. K., G. Singh, A. Mitchell, P. S. Kumar, and K. McGee (1997). NetEffect: A Network Architecture for Large-Scale Multi-User Virtual Worlds. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST '97)*, Lausanne, Switzerland, September 1997, pp. 157–163.

de Oliveira, J. C. and N. D. Georganas (2003). VELVET: An Adaptive Hybrid Architecture for Very Large Virtual Environments. *Presence: Teleoperators & Virtual Environments 12*(6), 555–580, December 2003.

Delaney, D. (2004). *Latency Reduction in Distributed Interactive Applications Using Hybrid Strategy-Based Models.* Ph. D. thesis, National University of Ireland Maynooth, Maynooth, Ireland, November 2004.

Delaney, D. and T. Ward (2004). A Java Tool for Exploring State Estimation using the Kalman Filter. In *Proc. 2004 IEEE Irish Signals and Systems Conference (ISSC)*, Belfast, June 2004, pp. 679–684.

Delaney, D., T. Ward, and S. McLoone (2003). On Reducing Entity State Update Packets in Distributed Interactive Simulations Using a Hybrid Model. In *Proc. 21st IASTED International Multi-Conference on Applied Informatics*, Innsbruck, Austria, 2003.

Delaney, D., T. Ward, and S. McLoone (2006a). On Consistency and Network Latency in Distributed Interactive Applications: A Survey — Part I. *Presence: Teleoperators & Virtual Environments 15*(2), 218–234, 2006.

Delaney, D., T. Ward, and S. McLoone (2006b). On Consistency and Network Latency in Distributed Interactive Applications: A Survey — Part II. *Presence: Teleoperators & Virtual Environments 15*(4), 465–482, 2006.

Diot, C. and L. Gautier (1999). A Distributed Architecture for Multiplayer Interactive Applications on the Internet. *IEEE Network 13*(4), 6–15, July/August 1999.

Dube, P., Z. Liu, S. Sahu, and J. Silber (2005). Last Mile Problem in Overlay Design. In *Proc. 2005 IEEE Global Telecommunications Conference (GLOBECOM '05)*, November/December 2005, Volume 2, pp. 920–925.

Duncan, T. P. and D. Gracanin (2003). Pre-Reckoning Algorithm for Distributed Virtual Environments. In *Proc. 2003 Winter Simulation Conference*, Volume 2, New Orleans, Louisiana, USA, December 2003, pp. 1086–1093.

El Saddik, A., D. Yang, and N. Georganas (2003). A Lightweight Multi-Session Synchronous Multivideo Collaborative Environment. In *Book of Abstracts. 2003 ACS/IEEE International Conference on Computer Systems and Applications*, Tunisia, Tunis, July 2003, pp. 30.

Elgammal, A., R. Duraiswami, and L. S. Davis (2003). Efficient Kernel Density

Estimation Using the Fast Gauss Transform with Applications to Color Modeling and Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*(11), 1499–1504, November 2003.

Ellis, C. A. and S. J. Gibbs (1989). Concurrency Control in Groupware Systems. In *Proc. 1989 ACM SIGMOD International Conference on Management of Data (SIGMOD '89)*, Portland, Oregon, USA, 1989, pp. 399–407.

Färber, J. (2002). Network Game Traffic Modelling. In *Proc. 1st Workshop on Network and System Support for Games (NetGames '02)*, Braunschweig, Germany, April 2002, pp. 53–57.

Farmer, F., C. Morningstar, and D. Crockford (1994). From Habitat to Global Cyberspace. In *Digest of Papers. Compcon Spring '94*, San Francisco, CA, USA, February/March 1994, pp. 186–192.

Farmer, M., R.-L. Hsu, and A. Jain (2002). Interacting Multiple Model (IMM) Kalman Filters for Robust High Speed Human Motion Tracking. In *Proc. 16th International Conference on Pattern Recognition*, 2002, Volume 2, pp. 20–23.

Feng, W.-c., F. Chang, W.-c. Feng, and J. Walpole (2005). A Traffic Characterization of Popular On-Line Games. *IEEE/ACM Trans. Netw. 13*(3), 488–500, June 2005.

Frécon, E. (2004). DIVE: Communication Architecture and Programming Model. *IEEE Communications Magazine 42*(4), 34–40, April 2004.

Frécon, E. and M. Stenius (1998). DIVE: A Scaleable Network Architecture for Distributed Virtual Environments. *Distrib. Syst. Engng 5*, 91–100, 1998.

Frías-Martínez, E., G. D. Magoulas, S. Chen, and R. Macredie (2005). Modeling Human Behavior in User-Adaptive Systems: Recent Advances using Soft Computing Techniques. *Expert Systems with Applications 29*(2), 320–329, 2005.

Fritsch, T., H. Ritter, and J. Schiller (2005). The Effect of Latency and Network Limitations on MMORPGs: A Field Study of Everquest2. In *Proc. 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '05)*, Hawthorne, NY, USA, October 2005, pp. 1–9.

Frohnmayer, M. and T. Gift (2000). The TRIBES Engine Networking Model. In *Proc. of Game Developers Conference (GDC 2000)*, San Jose, California, USA, March 2000, pp. 191–207.

Fujimoto, R. (2001). Parallel and Distributed Simulation Systems. In *Proc. 2001 Winter Simulation Conference*, Volume 1, Arlington, VA, USA, December 2001, pp. 147–157.

Funkhouser, T. A. (1995). RING: A Client-Server System for Multi-User Virtual Environments. In *Proc. 1995 Symposium on Interactive 3D Graphics (SI3D '95)*, Monterey, California, USA, April 1995, pp. 85–92.

Galli, R. and Y. Luo (2000). Mu3D: A Causal Consistency Protocol for a Collaborative VRML Editor. In *Proc. 5th Symposium on Virtual Reality Modeling Language (VRML '00)*, Monterey, California, USA, February 2000, pp. 53–62.

GameSpy (2003). Massively Multiplayer Online Games: The Past, The Present, and The Future. *GameSpy Industries, Inc.*, 2003.

Garrett, A., M. Aguilar, and Y. Barniv (2002). A Recurrent Neural Network Approach to Virtual Environment Latency Reduction. In *Proc. 2002 International Joint Conference on Neural Networks (IJCNN '02)*, Volume 3, Honolulu, HI, USA, May 2002, pp. 2288–2292.

Gautier, L. and C. Diot (1998). Design and Evaluation of MiMaze. A Multi-Player Game on the Internet. In *Proc. 1998 IEEE International Conference on Multivideo Computing and Systems*, Austin, TX, USA, June/July 1998, pp. 233–236.

Gautier, L., C. Diot, and J. Kurose (1999). End-to-End Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet. In *Proc. 8th Annual Joint Conference of the IEEE Computer and Communications Societies*, Volume 3, New York, NY, USA, March 1999, pp. 1470–1479.

Greenhalgh, C. (1998). Awareness-Based Communication Management in the MASSIVE Systems. *Distrib. Syst. Engng 5*, 129–137, September 1998.

Greenhalgh, C. and S. Benford (1995a). MASSIVE: A Collaborative Virtual Environment for Teleconferencing. *ACM Trans. Comput.-Hum. Interact. 2*(3), 239–261, September 1995.

Greenhalgh, C. and S. Benford (1995b). MASSIVE: A Distributed Virtual Reality System Incorporating Spatial Trading. In *Proc. 15th International Conference on Distributed Computing Systems*, Chapel Hill, North Carolina, USA, May 1995, pp. 27–34.

Greenhalgh, C., S. Benford, and G. Reynard (1999). A QoS Architecture for Collaborative Virtual Environments. In *Proc. 7th ACM International Conference on Multivideo (MULTIVIDEO '99)*, Orlando, Florida, USA, October/November 1999, pp. 121–130.

Greenhalgh, C., J. Purbrick, and D. Snowdon (2000). Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring. In *Proc. 3rd International Conference on Collaborative Virtual Environments (CVE '00)*, San Francisco, California, USA, September 2000, pp. 119–127.

Gutwin, C. (2001). The Effects of Network Delays on Group Work in Real-Time Groupware. In *Proc. 7th European Conference on Computer-Supported Cooperative Work (ECSCW '01)*, Bonn, Germany, September 2001, pp. 299–318.

Gutwin, C., C. Fedak, M. Watson, J. Dyck, and T. Bell (2006). Improving Network Efficiency in Real-Time Groupware with General Message Compression. In *Proc. 20th Anniversary Conference on Computer Supported Cooperative Work*, Banff, Alberta, Canada, November 2006, pp. 119–128.

Hanawa, D. and T. Yonekura (2005). On the Error Modeling of Dead Reckoned Data in a Distributed Virtual Environment. In *Proc. 2005 International Conference on Cyberworlds (CW '05)*, Singapore, November 2005, pp. 279–288.

Hanawa, D. and T. Yonekura (2006). A Proposal of Dead Reckoning Protocol in Distributed Virtual Environment Based on the Taylor Expansion. In *Proc. 2006 International Conference on Cyberworlds (CW '06)*, Lausanne, Switzerland, November 2006, pp. 107–114.

Hanawa, D., T. Yonekura, and Y. Kishi (2005). An Error Analysis of Polynomial Form Dead Reckoning Model Based on a Numerical Analysis. *Advanced Modeling and Optimization 7*(1), 85–98, 2005.

Harada, H., N. Kawaguchi, A. Iwakawa, K. Matsui, and T. Ohno (1998). Space-Sharing Architecture for a Three-Dimensional Virtual Community. *Distributed Systems Engineering 5*(3), 101–106, September 1998.

Harcsik, S., A. Petlund, C. Griwodz, and P. Halvorsen (2007). Latency Evaluation of Networking Mechanisms for Game Traffic. In *Proc. 6th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '07)*, Melbourne, Australia, September 2007, pp. 129–134.

Henderson, T. N. H. (2003). *The Effects of Relative Delay in Networked Games*. Ph. D. thesis, Dept. Computer Science, University College London, London, UK, February 2003.

Henninger, A. E., A. J. Gonzalez, M. Georgiopoulos, and R. F. DeMara (2000). Modeling Semi-Automated Forces with Neural Networks: Performance Improvement through a Modular Approach. In *Proc. 9th Conference on Computer Generated Forces and Behavioral Representation (CGF-BR '00)*, Orlando, Florida, USA, May 2000, pp. 1–8.

Henninger, A. E., A. J. Gonzalez, M. Georgiopoulos, and R. F. DeMara (2001). A Connectionist-Symbolic Approach to Modeling Agent Behavior: Neural Networks Grouped by Contexts. In *CONTEXT 2001*, Volume 2116 of *Lecture Notes in Computer Science*, pp. 198–209, January 2001.

id Soft (1993). Doom Official Website, http://www.idsoftware.com/games/doom/.

id Soft (1996). Quake Official Website, http://www.idsoftware.com/games/quake/.

Imagine (1993). Imagine Official Website, http://www.imagine.ie.

IEEE (1996). IEEE Standard for Distributed Interactive Simulation Communication Services and Profiles. *IEEE Std 1278.2-1995*, April 1996.

IEEE (1998). IEEE Standard for Distributed Interactive Simulation — Application Protocols. *IEEE Std 1278.1a-1998*, August 1998.

IEEE (2000). IEEE Standard for Modeling and Simulation M & S High Level Architecture (HLA) — Framework and Rules. *IEEE Std 1516-2000*, i–22, September 2000.

Jefferson, D. R. (1985). Virtual Time. *ACM Trans. Program. Lang. Syst. 7*(3), 404–425, July 1985.

Jefferson, D. R. (1990). Virtual Time II: Storage Management in Conservative and Optimistic Systems. In *Proc. 9th Annual ACM Symposium on Principles of Distributed Computing (PODC '90)*, Quebec City, Quebec, Canada, August 1990, pp. 75–89.

Jehaes, T., D. De Vleeschauwer, T. Coppens, B. Van Doorselaer, E. Deckers, W. Naudts, K. Spruyt, and R. Smets (2003). Access Network Delay in Networked Games. In *Proc. 2nd Workshop on Network and System Support for Games (NetGames '03)*, Redwood City, California, 2003, pp. 63–71.

Jeong, J., J. C. Gore, and B. S. Peterson (2001). Mutual Information Analysis of the EEG in Patients with Alzheimer's Disease. *Clinical Neurophysiology 112*(5), 827–835, May 2001.

Johnson, W. R., T. W. Mastaglio, and P. D. Peterson (1993). The Close Combat Tactical Trainer Program. In *Proc. 25th Conference on Winter Simulation (WSC '93)*, Los Angeles, California, USA, December 1993, pp. 1021–1029.

Joslin, C., T. Di Giacomo, and N. Magnenat-Thalmann (2004). Collaborative Virtual Environments: From Birth to Standardization. *IEEE Communications Magazine 42*(4), 28–33, April 2004.

Joslin, C., T. Molet, N. Thalmann, J. Esmerado, D. Thalmann, I. Palmer, N. Chilton, and R. Earnshaw (2001). Sharing Attractions on the Net with VPark. *IEEE Computer Graphics and Applications 21*(1), 61–71, January/February 2001.

Joslin, C., I. S. Pandzic, and N. M. Thalmann (2003). Trends in Networked Collaborative Virtual Environments. *Computer Communications 26*(5), 430–437, May 2003.

Kapolka, A. (2003). The Extensible Run-Time Infrastructure (XRTI): An Experimental Implementation of Proposed Improvements to the High Level Architecture. Master's thesis, Dept. Modeling, Virtual Environments and Simulation, Naval Postgraduate School, CA, USA, December 2003.

Kenny, A., S. Mcloone, and T. Ward (2009). Controlling Entity State Updates to Maintain Remote Consistency within a Distributed Interactive Application. *ACM Trans. Internet Technol. 9*(4), 1–25, September 2009.

Kenny, A., S. McLoone, T. Ward, and D. Delaney (2006). Using User Perception to Determine Suitable Error Thresholds for Dead Reckoning in Distributed Interactive Applications. In *Proc. 2006 IET Irish Signals and Systems Conference*, Dublin, Ireland, June 2006, pp. 49–54.

Kinsner, W. (2002). Compression and Its Metrics for Multivideo. In *Proc. 1st IEEE International Conference on Cognitive Informatics*, Calgary, Canada, August 2002, pp. 107–121.

Koenen, R. (2002). Overview of the MPEG-4 Standard. Technical Report ISO/IEC JTC1/SC29/WG11 N4668, International Organization for Standardization, March 2002.

Krüger, W., C.-A. Bohn, B. Fröhlich, H. Schüth, W. Strauss, and G. Wesche (1995). The Responsive Workbench: A Virtual Work Environment. *Computer 28*(7), 42–48, July 1995.

Kushner, D. (2002). The Wizardry of Id. *IEEE Spectrum 39*(8), 42–47, August 2002.

Laird, J. E. (2002). Research in Human-Level AI using Computer Games. *Commun. ACM 45*(1), 32–35, January 2002.

Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM 21*(7), 558–565, July 1978.

LaViola Jr., J. J., D. A. Feliz, D. F. Keefe, and R. C. Zeleznik (2001). Hands-Free Multi-Scale Navigation in Virtual Environments. In *Proc. 2001 Symposium on Interactive 3D Graphics (I3D '01)*, Research Triangle Park, NC, USA, March 2001, pp. 9–15.

Lea, R., Y. Honda, and K. Matsuda (1997). Virtual Society: Collaboration in 3D Spaces on the Internet. *Computer Supported Cooperative Work (CSCW) 6*(2–3), 227–250, June 1997.

Lea, R., Y. Honda, K. Matsuda, and S. Matsuda (1997). Community Place: Architecture and Performance. In *Proc. 2nd Symposium on Virtual Reality Modeling Language (VRML '97)*, Monterey, California, USA, February 1997, pp. 41–50.

Lee, B.-S., W. Cai, S. J. Turner, and L. Chen (2000). Adaptive Dead Reckoning Algorithms For Distributed Interactive Simulation. *International Journal of Simulation Systems, Science & Technology 1*(1-2), 21–34, December 2000.

Lee, D., M. Lim, S. Han, and K. Lee (2007). ATLAS: A Scalable Network Framework for Distributed Virtual Environments. *Presence: Teleoperators & Virtual Environments 16*(2), 125–156, April 2007.

Lee, D., J. Yang, H. Y. Youn, C. Yu, and S. Hyun (2000). Entity-Centric Scalable Concurrency Control for Distributed Interactive Applications. In *Proc. IEEE International Performance, Computing, and Communications Conference, (IPCCC '00)*, Phoenix, AZ, USA, February 2000, pp. 544–550.

Leigh, J. and A. E. Johnson (1996). CALVIN: An Immersivideo Design Environment Utilizing Heterogeneous Perspectives. In *Proc. 3rd IEEE International Conference on Multivideo Computing and Systems*, Hiroshima, Japan, June 1996, pp. 20–23.

Leigh, J., A. E. Johnson, and T. A. DeFanti (1997). CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments. *Virtual Reality: Research, Development and Applications 2*(2), 217–237, December 1997.

Li, D., L. Zhou, and R. Muntz (2000). A New Paradigm of User Intention Preservation in Realtime Collaborative Editing Systems. In *Proc. 7th International*

Conference on Parallel and Distributed Systems, Iwate, Japan, July 2000, pp. 401–408.

Li, F. W., L. W. Li, and R. W. Lau (2004). Supporting Continuous Consistency in Multiplayer Online Games. In *Proc. 12th Annual ACM International Conference on Multivideo*, New York, NY, USA, October 2004, pp. 388–391.

Li, W. (1990). Mutual Information Functions versus Correlation Functions. *Journal of Statistical Physics 60*(5), 823–837, 1990.

Liang, L. A. H., W. Cai, B.-S. Lee, and S. J. Turner (1999). Performance Analysis of Packet Bundling Techniques in DIS. In *Proc. 3rd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications*, Greenbelt, MD, USA, October 1999, pp. 75–82.

Lin, D. W. and D. W. Lin (1994). Fractal Image Coding as Generalized Predictive Coding. In *Proc. 1994 IEEE International Conference Image Processing (ICIP-94)*, Volume 3, pp. 117—121.

Lin, K.-C. and D. E. Schab (1994). The Performance Assessment of the Dead Reckoning Algorithms in DIS. *Simulation 63*(5), 318–325, November 1994.

Lin, K.-C. and D. E. Schab (1995). Network Load in Distributed Interactive Simulation. *Journal of Aircraft 32*(6), 1392–1394, November/December 1995.

Lin, L.-J. and A. Ortega (1998). Bit-Rate Control using Piecewise Aproximated Rate-Distortion Characteristics. *IEEE Transactions on Circuits and Systems for Video Technology 8*(4), 446–459, August 1998.

Lin, Q., H. K. Neo, L. Zhang, G. Huang, and R. Gay (2007). Grid-Based Large-Dcale Web3D Collaborative Virtual Environment. In *Proc. 12th International Conference on 3D Web Technology (Web3D '07)*, Perugia, Italy, April 2007, pp. 123–132.

Linden Laboratories (2003). Second-Life Official Website, http://secondlife.com/.

Lloyd, J. (2004). The Torque Game Engine. *Game Devel. Mag. 11*(8), 8–9, 2004.

Lui, J. (2001). Constructing Communication Subgraphs and Deriving an Optimal Synchronization Interval for Distributed Virtual Environment Systems. *IEEE Transactions on Knowledge and Data Engineering 13*(5), 778–792, September/October 2001.

Lungarella, M. and R. Pfeifer (2001). Robots as Cognitive Tools: Information-Theoretic Analysis of Sensory-Moter Data. In *Proc. 2nd IEEE-RAS International Conference on Humanoid Robotics (Humanoids 2001)*, Tokyo, Japan, November 2001, pp. 245–252.

Macedonia, M. R. (1995). *A Network Software Architecture For Large Scale Virtual Environments*. Ph. D. thesis, Naval Postgraduate School, Monterey, CA, USA, June 1995.

Macedonia, M. R., D. P. Brutzman, M. J. Zyda, D. R. Pratt, P. T. Barham, J. Falby, and J. Locke (1995). NPSNET: A Multi-Player 3D Virtual Environment over the Internet. In *Proc. 1995 Symposium on Interactive 3D Graphics (SI3D '95)*, Monterey, California, USA, April 1995, pp. 93–94.

Macedonia, M. R. and M. J. Zyda (1997). A Taxonomy for Networked Virtual Environments. *IEEE Multivideo 4*(1), 48–56, January/March 1997.

Macedonia, M. R., M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz (1994). NPSNET: A Network Software Architecture For Large Scale Virtual Environments. *Presence 3*(4), 265–287, 1994.

MacKenzie, I. S. and C. Ware (1993). Lag as a Determinant of Human Performance in Interactive Systems. In *Proc. INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, Netherlands, April 1993, pp. 488–493.

Marsh, J., M. Glencross, S. Pettifer, R. J. Hubbold, J. Cook, and S. Daubrenet (2004). Minimising Latency and Maintaining Consistency in Distributed Virtual Prototyping. In *Proc. 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI '04)*, Singapore, June 2004, pp. 386–389.

Marshall, D. (2008). *Improving Consistency in Distributed Interactive Applications.* Ph. D. thesis, National University of Ireland Maynooth, Maynooth, Ireland, December 2008.

Marshall, D., S. McLoone, D. Delaney, and T. Ward (2006). Statistical Determination of Hybrid Threshold Parameters for Entity State Update Mechanisms in Distributed Interactive Applications. In *Proc. 10th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '06)*, Torremolinos, Malaga, Spain, October 2006, pp. 85–94.

Marshall, D., S. McLoone, and T. Ward (2008). Optimising Consistency by Maximising Bandwidth Usage in Distributed Interactive Applications. *ACM Transactions on Multivideo Computing, Communications and Applications (ACM TOMCCAP)*, To appear.

Marshall, D., S. McLoone, T. Ward, and D. Delaney (2006). Does Reducing Packet Transmission Rates Help to Improve Consistency within Distributed Interactive Applications? In *Proc. of 9th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games*, Dublin, Ireland, November 2006, pp. 88–92.

Mauve, M. (2000a). Consistency in Replicated Continuous Interactive Video. In *Proc. 2000 ACM Conference on Computer Supported Cooperative Work (CSCW '00)*, Philadelphia, Pennsylvania, USA, 2000, pp. 181–190.

Mauve, M. (2000b). *Distributed Interactive Video.* Ph. D. thesis, University of Mannheim, Schloss, Mannheim, Germany, November 2000.

Mauve, M., V. Hilt, C. Kuhmunch, and W. Effelsberg (2001). RTP/I — Toward a Common Application Level Protocol for Distributed Interactive Video. *IEEE Transactions on Multivideo 3*(1), 152–161, March 2001.

Mauve, M., J. Vogel, V. Hilt, and W. Effelsberg (2004). Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications. *IEEE Transactions on Multivideo 6*(1), 47–57, February 2004.

McCoy, A. (2007). *Data-Driven Modelling Approaches for Network Traffic Reduction in Distributed Interactive Applications.* Ph. D. thesis, National University of Ireland Maynooth, Maynooth, Ireland, February 2007.

McCoy, A., D. Delaney, S. McLoone, and T. Ward (2005). 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.

McCoy, A., S. McLoone, D. Delaney, and T. Ward (2006). Formalizing a Framework for Dynamic Hybrid Strategy Models in Distributed Interactive Applications. In *Proc. 2006 IET Irish Signals and Systems Conference*, Dublin, Ireland, June 2006, pp. 357–362.

McCoy, A., S. McLoone, and T. Ward (2004). 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.

McCoy, A., T. Ward, S. McLoone, and D. Delaney (2006). Using Neural-Networks to Reduce Entity State Updates in Distributed Interactive Applications. In T. Ward (Ed.), *Proc. 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, Maynooth, Ireland, September 2006, pp. 295–300.

McCoy, A., T. Ward, S. McLoone, and D. Delaney (2007). Multistep-Ahead Neural-Network Predictors for Network Traffic Reduction in Distributed Interactive Applications. *ACM Trans. Model. Comput. Simul. 17*(4), 16, 2007

Meehan, M., S. Razzaque, M. Whitton, and J. Brooks, F.P. (2003). Effect of Latency on Presence in Stressful Virtual Environments. In *Proc. 2003 IEEE Virtual Reality*, Stanford University., CA, USA, March 2003, pp. 141–148.

Mellon, L. and D. West (1995). Architectural Optimizations to Advanced Distributed Simulation. In *Proc. 27th Conference on Winter Simulation (WSC '95)*, Arlington, Virginia, USA, December 1995, pp. 634–641.

Microsoft (2005a). Xbox Official Website, http://www.xbox.com.

Microsoft (2005b). Xbox Live Official Website, http://www.xbox.com/live.

Miller, D. C. and J. A. Thorpe (1995). SIMNET: The Advent of Simulator Networking. *IEEE Proceedings 83*(8), 1114–1123, August 1995.

Mills, D. (1991). Internet Time Synchronization: The Network Time Protocol. *IEEE Transactions on Communications 39*(10), 1482–1493, October 1991.

Moon, Y.-I., B. Rajagopalan, and U. Lall (1995). Estimation of Mutual Information using Kernel Density Estimators. *Phys. Rev. E 52*(3), 2318–2321, September 1995.

Morse, K. L., L. Bic, and M. Dillencourt (2000). Interest Management in Large-Scale Virtual Environments. *Presence: Teleoperators & Virtual Environments 9*(1), 52–68, February 2000.

Natrajan, A. and J. Reynolds, P.F. (1999). Resolving Concurrent Interactions. In *Proc. 3rd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications*, October 1999, pp. 85–92.

NCsoft (1998). Lineage Official Website, http://www.lineage.com/.

Ng, B., A. Si, R. W. Lau, and F. W. Li (2002). A Multi-Server Architecture for Distributed Virtual Walkthrough. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST '02)*, Hong Kong, China, November 2002, pp. 163–170.

Nichols, J. and M. Claypool (2004). The Effects of Latency on Online Madden NFL Football. In *Proc. 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '04)*, Cork, Ireland, June 2004, pp. 146–151.

Origin Systems (1997). Ultima-Online Official Website, http://www.uoherald.com-/news/.

Paninski, L. (2003). Estimation of Entropy and Mutual Information. *Neural Computation 15*(6), 1191–1253, June 2003.

Pantel, L. and L. C. Wolf (2002a). On the Impact of Delay on Real-Time Multiplayer Games. In *Proc. 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '02)*, Miami, Florida, USA, May 2002, pp. 23–29.

Pantel, L. and L. C. Wolf (2002b). On the Suitability of Dead Reckoning Schemes for Games. In *Proc. 1st Workshop on Network and System Support for Games (NetGames '02)*, Braunschweig, Germany, April 2002, pp. 79–84.

Park, K. S., Y. J. Cho, N. K. Krishnaprasad, C. Scharver, M. J. Lewis, J. Leigh, and A. E. Johnson (2000). CAVERNsoft G2: A Toolkit for High Performance Tele-Immersive Collaboration. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST '00)*, Seoul, Korea, October 2000, pp. 8–15.

Park, K. S. and R. Kenyon (1999). Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment. In *Proc. 1999 IEEE Virtual Reality*, Chicago, IL, USA, March 1999, pp. 104–111.

Pettifer, S., J. Cook, J. Marsh, and A. West (2000). DEVA3: Architecture for a Large-Scale Distributed Virtual Reality System. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, Seoul, Korea, October 2000, pp. 33–40.

Qin, X. (2002). A Delayed Consistency Model for Distributed Interactive Systems with Real-Time Continuous Video. *Journal of Software 13*(6), 1029–1039, 2002.

Quax, P., P. Monsieurs, W. Lamotte, D. De Vleeschauwer, and N. Degrande (2004). Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game. In *Proc. 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '04)*, Portland, Oregon, USA, August 2004, pp. 152–156.

Rago, C. and R. Mehra (2000). Robust Adaptive Target State Estimation for Missile Guidance using the Interacting Multiple Model Kalman Filter. In *Proc. 2000 IEEE Position Location and Navigation Symposium*, San Diego, CA, USA, March 2000, pp. 355–362.

Rahn, S. (1998). WorldToolKit Release 8 Technical Overview. Technical report, SENSE8 Corporation, Mill Valley, CA, USA, February 1998.

Roberts, D., R. Aspin, D. Marshall, S. McLoone, D. Delaney, and T. Ward (2008). Bounding Inconsistency using a Novel Threshold Metric for Dead Reckoning Update Packet Generation. *Simulation 84*(5), 239–256, May 2008.

Roberts, D., D. Marshall, S. McLoone, D. Delaney, T. Ward, and R. Aspin (2005). Exploring the Use of Local Consistency Measures as Thresholds for Dead Reckoning Update Packet Generation. In *Proc. 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '05)*, Montreal, Quebec, Canada, October 2005, pp. 195–202.

Roberts, D. and P. Sharkey (1997). Maximising Concurrency and Scalability in a Consistent, Causal, Distributed Virtual Reality System, Whilst Minimising the Effect of Network Delays. In *Proc. 6th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Cambridge, MA, USA, June 1997, pp. 161–166.

Roberts, D., P. M. Sharkey, and P. D. Sandoz (1995). A Real-Time Predictive Architecture for Distributed Virtual Reality. In *Proc. 1st Workshop on Simulation and Interaction in Virtual Environments*, Des Monies, Iowa, USA, July 1995, pp. 279–288.

Roehl, B. (1995). Distributed Virtual Reality: An Overview. In *Proc. 1st Symposium on Virtual Reality Modeling Language*, San Diego, California, USA, December 1995, pp. 39–43.

Roehle, B. (1997). Channeling the Data Flood. *IEEE Spectrum 34*(3), 32–38, March 1997.

Roulston, M. S. (1999). Estimating the Errors on Measured Entropy and Mutual Information. *Physica D: Nonlinear Phenomena 125*(3-4), 285–294, January 1999.

Roussos, M., A. E. Johnson, J. Leigh, C. A. Vasilakis, C. R. Barnes, and T. G. Moher (1997). NICE: Cmbining Constructionism, Carrative and collaboration

in a virtual learning environment. *SIGGRAPH Comput. Graph. 31*(3), 62–63, August 1997.

Saddik, A., A. Rahman, S. Abdala, and B. Solomon (2008). PECOLE: P2P Multivideo Collaborative Environment. *Multivideo Tools Appl. 39*(3), 353–377, September 2008.

Sas, C., R. Reilly, and G. OŠHare (2003). A Connectionist Model of Spatial Knowledge Acquisition in a Virtual Environment. In *Proc. 2nd Workshop on Machine Learning, Information Retrieval and User Modelling (MLIRUM)*, Johnstown, PA, June 2003, pp. 40–48.

Schaefer, C., T. Enderes, H. Ritter, and M. Zitterbart (2002). Subjective Quality Assessment for Multiplayer Real-Time Games. In *Proc. 1st Workshop on Network and System Support for Games*, Braunschweig, Germany, April 2002, pp. 74–78.

Schwartz, P., L. Bricker, B. Campbell, T. Furness, K. Inkpen, L. Matheson, N. Nakamura, L.-S. Shen, S. Tanney, and S. Yen (1998r). Virtual Playground: Architectures for a Shared Virtual World. In *Proc. ACM Symposium on Virtual Reality Software and technology*, Taipei, Taiwan, Novembe 1998, pp. 43–50.

Seay, A. F., W. J. Jerome, K. S. Lee, and R. E. Kraut (2004). Project Massive: A Study of Online Gaming Communities. In *Extended Abstracts. 2004 Human Factors in Computing Systems (CHI '04)*, Vienna, Austria, April 2004, pp. 1421–1424.

Sharkey, P., M. Ryan, and D. Roberts (1998). A Local Perception Filter for Distributed Virtual Environments. In *Proc. 1998 IEEE Virtual Reality Annual International Symposium*, Atlanta, GA, USA, March 1998, pp. 242–249.

Shaw, C., M. Green, J. Liang, and Y. Sun (1993). Decoupled Simulation in Virtual Reality with the MR Toolkit. *ACM Trans. Inf. Syst. 11*(3), 287–317, July 1993.

Sheldon, N., E. Girard, S. Borg, M. Claypool, and E. Agu (2003). The Effect of Latency on User Performance in Warcraft III. In *Proc. 2nd Workshop on Network and System Support for Games (NetGames '03)*, Redwood City, California, May 2003, pp. 3–14.

Shi, X.-B., X. Wang, J. Bi, F. Liu, D. Yang, and X.-Y. Liu (2009). A DR Algorithm Based on Artificial Potential Field Method. *Multivideo Tools and Applications 45*(1-3), 247–261, October 2009.

Shim, K.-H. and J.-S. Kim (2001). A Dead Reckoning Algorithm with Variable Threshold Scheme in Networked Virtual Environment. In *Proc. 2001 IEEE International Conference on Systems, Man, and Cybernetics*, Volume 2, Tucson, AZ, USA, October 2001, pp. 1113–1118.

Singh, G., L. Serra, W. Png, A. Wong, and H. Ng (1995). BrickNet: Sharing Object Behaviors on the Net. In *Proc. 1995 Virtual Reality Annual International Symposium*, Research Triangle Park, NC, USA, March 1995, pp. 19–25.

Singhal, S. K. (1996). *Effective Remote Modelling in Large-scale Distributed Simulation*. Ph. D. thesis, Dept. Computer Science, Stanford University, Stanford, CA, USA, August 1996.

Singhal, S. K. and D. R. Cheriton (1995). Exploring Position History for Efficient Remote Rendering in Networked Virtual Reality. *Presence: Teleoperators & Virtual Environments 4*(2), 169–193, Spring 1995.

Singhal, S. K. and D. R. Cheriton (1996). Using Projection Aggregations to Support Scalability in Distributed Simulation. In *Proc. 16th International Conference on Distributed Computing Systems*, Stanford, CA, USA, May 1996, pp. 196–206.

Singhal, S. K. and M. Zyda (1999). *Networked Virtual Environments: Design and Implementation* (1st ed.). New York: Addison-Wesley, 1999.

Smed, J., T. Kaukoranta, and H. Hakonen (2002a). A Review on Networking and Multiplayer Computer Games. Technical Report TUCS Technical Report No 454, Turku Centre for Computer Science, University of Turku, Turku, Finland, April 2002.

Smed, J., T. Kaukoranta, and H. Hakonen (2002b). Aspects of Networking in Multiplayer Computer Games. *The Electronic Library 20*(2), 87–97.

Snowdon, D. N. and A. J. West (1994). AVIARY - Design Issues for Future Large-Scale Virtual Environments. *Presence: Teleoperators & Virtual Environments 3*(4), 288–308, Fall 1994.

Srinivasan, S. (1996). Efficient Data Consistency in HLA/DIS++. In *Proc. 28th Conference on Winter Simulation (WSC '96)*, Coronado, California, USA, December 1996, pp. 946–951.

Stefik, M., D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar (1987). WYSIWIS Revised: Early Experiences with Multiuser Interfaces. *ACM Trans. Inf. Syst. 5*(2), 147–167, April 1987.

Steuer, R., J. Kurths, C. O. Daub, J. Weise, and J. Selbig (2002). The Mutual Information: Detecting and Evaluating Dependencies Between Variables. *Bioinformatics 18*(supplement 2), S231–S240.

Steuer, R., L. Molgedey, W. Ebeling, and M. J.-M. no (2001). Entropy and optimal partition for data analysis. *Eur. Phys. J. B 19*(2), 265–269.

Stolzenburg, F., O. Obst, and J. Murray (2002). Qualitative Velocity and Ball Interception. In M. Jarke, G. Lakemeyer, and J. Koehler (Eds.), *Proc. Advances in Artificial Intelligence (KI 2002)*, Volume 2479 of *Lecture Notes in Computer Science*, pp. 95–99. Springer Berlin / Heidelberg.

Strom, R., G. Banavar, K. Miller, A. Prakash, and M. Ward (1997). Concurrency Control and View Notification Algorithms for Collaborative Replicated Objects. In *Proc. 17th International Conference on Distributed Computing Systems*, Baltimore, MD, USA, May 1997, pp. 194–203.

Stytz, M. (1996). Distributed Virtual Environments. *IEEE Computer Graphics and Applications 16*(3), 19–31, May 1996.

Sugano, H., K. Otani, H. Ueda, S. Hiraiwa, S. Endo, and Y. Kohda (1997). SpaceFusion: A Multi-Server Architecture for Shared Virtual Environments. In *Proc. 2nd Symposium on Virtual Reality Modeling Language (VRML '97)*, Monterey, California, USA, February 1997, pp. 51–58.

Sun, C. and D. Chen (2000). A Multi-Version Approach to Conflict Resolution in Distributed Groupware Systems. In *Proc. 20th International Conference on Distributed Computing Systems*, Taipei, April 2000, pp. 316–325.

Sun, C., X. Jia, Y. Zhang, Y. Yang, and D. Chen (1998). Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Trans. Comput.-Hum. Interact. 5*(1), 63–108, March 1998.

Sunday Pandzic, I., T. K. Capin, E. Lee, N. Magnenat Thalmann, and D. Thalmann (1997). A Flexible Architecture for Virtual Humans in Networked Collaborative Virtual Environments. In *Comput. Graph. Forum 1997*, Volume 16, Budapest, Hungary, October 1997, pp. C177–C188.

Sung, U.-J., J.-H. Yang, and K.-Y. Wohn (1999). Concurrency Control in CIAO. In *Proc. 1999 IEEE Virtual Reality*, Houston, Texas, USA, March 1999, pp. 22–28.

Sutherland, I. E. (1968). A Head-Mounted Three Dimensional Display. In *Proc. 1968 Fall Joint Computer Conference (AFIPS '68)*, San Francisco, California, USA, December 1968, pp. 757–764.

Tanenbaum, A. S. (1998). *Computer Networks* (3rd ed.). Beijing, China: Tsinghua University Press, July 1998.

Thurau, C., C. Bauckhage, and G. Sagerer (2003). Combining Self Organizing Maps and Multilayer Perceptrons to Learn Bot-Behavior for a Commercial Game. In *Proc. 4th International Conference on Intelligent Games and Simulation (GAME-ON '03)*, London, UK, November 2003, pp. 119–123.

Thurau, C., G. Sagerer, and C. Bauckhage (2004). Imitation Learning at All Levels of Game-AI. In *Proc. 5th International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*, Reading, UK, November 2004, pp. 402–408.

Tudor, P. (1995). MPEG-2 Video Compression. *Electronics Communication Engineering Journal 7*(6), 257–264, December 1995.

Vaghi, I., C. Greenhalgh, and S. Benford (1999). Coping with Inconsistency Due to Network Delays in Collaborative Virtual Environments. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST '99)*, London, UK, December 1999, pp. 42–49.

Van Hook, D. J., S. J. Rak, and J. O. Calvin (1994). Approaches to Relevance Filtering. In *Proc. 11th Workshop on Standards for the Interoperability of Distributed Simulations*, 2004, pp. 26–30.

Verna, D., Y. Fabre, and G. Pitel (2000). Urbi et Orbi: Unusual Design and Implementation Choices for Distributed Virtual Environments. In *Proc. 6th International Conference on Virtual Systems and Multivideo (VSMM2000)*, Softopia, Gifu, Japan, October 2000, pp. 714–724.

Waters, R. and J. Barrus (1997). The Rise of Shared Virtual Environments. *IEEE Spectrum 34*(3), 20–25, March 1997.

Waters, R. C. and D. B. Anderson (1997). Diamond Park and Spline: Social Virtual Reality with 3D Animation, Spoken Interactiion, and Runtme Extendability. *Presence: Teleoperators & Virtual Environments 6*(4), 461–481, August 1997.

Wolfson, O., L. Jiang, A. P. Sistla, S. Chamberlain, N. Rishe, and M. Deng (1998). Databases for Tracking Mobile Units in Real Time. In *ICDT '99*, Volume 1540 of *Lecture Notes in Computer Science*, January 1998, pp. 169–186.

Wray, M. and R. Hawkes (1998). Distributed Virtual Environments and VRML: An Event-Based Architecture. *Computer Networks and ISDN Systems 30*(1–7), 43–51, April 1998.

Yang, C., R. Duraiswami, N. Gumerov, and L. Davis (2003). Improved Fast Gauss Transform and Efficient Kernel Density Estimation. In *Proc. 2003 9th IEEE International Conference on Computer Vision*, Volume 1, Nice, France, October 2003, pp. 664–671.

Yasui, T., Y. Ishibashi, and T. Ikedo (2005). Influences of Network Latency and Packet Loss on Consistency in Networked Racing Games. In *Proc. 4th ACM*

SIGCOMM Workshop on Network and System Support for Games (NetGames '05), Hawthorne, NY, USA, October 2005, pp. 1–8.

Yu, S.-J. and Y.-C. Choy (2001). A Dynamic Message Filtering Technique for 3D Cyberspaces. *Computer Communications 24*(18), 1745–1758, December 2001.

Yu, Y., Z. Li, L. Shi, Y.-C. Chen, and H. Xu (2007). Network-Aware State Update For Large Scale Mobile Games. In Z. Li (Ed.), *Proc. 16th International Conference on Computer Communications and Networks (ICCCN 2007)*, Honolulu, Hawaii, USA, August 2007, pp. 563–568.

Yura, S., T. Usaka, and K. Sakamura (1999). Video Avatar: Embedded Video for Collaborative Virtual environment. In *Proc. 1999 IEEE International Conference on Multivideo Computing and Systems*, Volume 2, Florence, Italy, July 1999, pp. 433–438.

Zander, S. and G. Armitage (2005). A Traffic Model for the Xbox Game Halo 2. In *Proc. 2005 International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '05)*, Stevenson, Washington, USA, June 2005, pp. 13–18.

Zhang, L. and Q. Lin (2007). MACVE: A Mobile Agent Based Framework for Large-Scale Collaborative Virtual Environments. *Presence: Teleoperators & Virtual Environments 16*(3), 279–292, June 2007.

Zhang, X., T. Ward, and S. McLoone (2008). Towards an Information Model of Consistency Maintenance in Distributed Interactive Applications. *International Journal of Computer Games Technology 2008*(4), 1–10, 2008.

Zhang, X., T. Ward, and S. McLoone (2009). Exploring an Information Framework for Consistency Maintenance in Distributed Interactive Applications. In *Proc. 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT '09)*, Singapore, October 2009, pp. 121–128.

Zhou, S., W. Cai, F. B.-S. Lee, and S. J. Turner (2001). Consistency In Distributed Interactive Applications. In *Proc. 2001 European Simulation Interoperability Workshop (Euro-SIW 2001)*, London, UK, June 2001, pp. 01E–SIW–003.

Zhou, S., W. Cai, F. B.-S. Lee, and S. J. Turner (2004). Time-Space Consistency in Large-Scale Distributed Virtual Environments. *ACM Trans. Model. Comput. Simul. 14*(1), 31–47.

Zhou, S., W. Cai, S. J. Turner, F. B.-S. Lee, Junhu, and Wei (2007). Critical Causal Order of Events in Distributed Virtual Environments. *ACM Trans. Multivideo Comput. Commun. Appl. 3*(3), 15. 1236474.

Zimmermann, R., E. Chew, S. A. Ay, and M. Pawar (2008). Distributed Musical Performances: Architecture and Stream Management. *ACM Trans. Multivideo Comput. Commun. Appl. 4*(2), 1–23, May 2008.

Zukerman, I. and D. W. Albrecht (2001). Predictive Statistical Models for User Modeling. *User Modeling and User-Adapted Interaction 11*(1), 5–18, March 2001.

Appendices

A. Matlab Code for Dead Reckoning

```
%%%%%%%%%%%%%%%%%%%%%%%%
% Extrapolation models %
%%%%%%%%%%%%%%%%%%%%%%%%

function p_state = extrapol(cur_time,ESU,range,mode)
% Extrapolate current state values to predict the next state.
% Inputs:
% cur_time - the current simulation time instance.
% ESU - The ESU to be extrapolated. ESU is a 1 by 10 vector which
%       contains the releasing time of the ESU, and positions,
%       velocities and accelerations in three dimensions.
% range - the range of the state space. Set to empty for an infinite
%         space.
% mode - indication of extrapolation method order.
% Outputs:
% p_state --- predicted state value.

switch mode
    case 0
        p_state=ESU(2:4);
```

```
    case 1
        p_state=ESU(2:4)+(cur_time-ESU(1))*ESU(5:7);
    case 2
        p_state=ESU(2:4)+(cur_time-ESU(1))*ESU(5:7)+0.5*ESU(8:10)
        *((cur_time-ESU(1)))^2;
    otherwise
        disp('No such extrapolation mode!')
end if isempty(range)
    return;
else
    p_state(p_state>range(2,:))=range(2,p_state>range(2,:));
    p_state(p_state<range(1,:))=range(1,p_state<range(1,:));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dead reckoning on local hosts %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [LocalApprox,ESU,Errors]=DR_Local(d,h,range,mode)
% Simulation of local executions of Dead Reckoning.
% Inputs:
% d - true entity trajectory, N by 10 vector in which each row
%     description of the entity state at a single moment.
% h - local threshold value.
% range - the range of the state space. Set to empty for an infinite
%         space.
% mode - indication of extrapolation method order.
% Outputs:
% LocalApprox - the local entity state model.
% ESU - vectors of ESUs generated.
```

```
% Errors - local prediction errors.

N=size(d,1);
LocalApprox=zeros(size(d));
ESU=d(1,:);
Errors=zeros(N,1);
LocalApprox(1,:)=d(1,:); LocalApprox(:,1)=d(:,1);
for i=2:N
            % Estimating a future state
            tempstate=extrapol(d(i,1),ESU(end,:),range,mode);
            e=d(i,2:4)-tempstate;
            % Prediction error in Euclid Distance
            Errors(i)=sqrt(sum(e.^2));
            if Errors(i)-h>eps
                % An threshold violation
                LocalApprox(i,:)=d(i,:);
                ESU=[ESU;d(i,:)];
            else
                LocalApprox(i,2:4)=tempstate; %accept prediction
                LocalApprox(i,5:end)=ESU(end,5:end);
            end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%
% Add latency to each ESU %
%%%%%%%%%%%%%%%%%%%%%%%%%%

ESUrecv=[ESUsend(:,1)+[0;Latency*ones(size(ESUsend,1)-1,1)],ESUsend];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dead reckoning on remote hosts %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [RemoteApprox]=DR_Remote(T,ESU,range,mode)
% Simulation of remote executions of Dead Reckoning.
% Inputs:
% T - simulation time instances.
% ESU - received ESUs.
% range - the range of the state space. Set to empty for an infinite
%         space.
% mode - indication of extrapolation method order.
% Outputs:
% RemoteApprox - remote entity state model.

RemoteApprox=zeros(length(T),size(ESU,2)-1);

for i=1:length(T)
    RemoteApprox(i,1)=T(i);
    % Finding the most recent ESU
    ESUindex=find(ESU(:,1)<=T(i), 1, 'last' );
    RemoteApprox(i,2:4)=extrapol(T(i),ESU(ESUindex,2:end),
    range,mode);
    RemoteApprox(i,5:end)=ESU(ESUindex,6:end);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B. Matlab Code for Information Estimation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Joint probability by the simple algorithm %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [P,xrange,yrange]=JointPro(x,y)
% Joint probability matrix of two time series.
% The probability is calculated as the occurrence frequency.
% Inputs:
% x,y - two time series.
% Outputs:
% P - probability matrix, P(i,j) is the probability that
%     x is ith value in xrange and y is the jth value in yrange.
% xrange,yrange - unique values of x and y, respectively.

N=length(x);
xrange=unique(x);
yrange=unique(y);
P=zeros(length(xrange),length(yrange));
% Counting occurrence
for i=1:N
    I=find(xrange==x(i));
    J=find(yrange==y(i));
    P(I,J)=P(I,J)+1;
end

P=P./N;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mutual information by the simple algorithm %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [MI,err]=MuInfo(x,y,mode)
% Cross mutual information of two time series.
% Inputs:
% x,y - time series.
% mode - correction for finite size effect. 0 for no correction,
%         1 for simple correction.
% Outputs:
% MI - mutual information.
% err - estimated bias.

N=length(x);
[Pxy,xrange,yrange]=JointPro(x,y);
% Marginal probabilities
Px=sum(Pxy,2);
Py=sum(Pxy,1);
MI=0;
for i=1:length(xrange)
    for j=1:length(yrange)
        if Pxy(i,j)==0
            continue;
        else
            MI=MI+Pxy(i,j)*log2(Pxy(i,j)/(Px(i)*Py(j)));
        end
    end
end
if mode
    r=find(Pxy);
    mxy=length(r);
```

```
    mx=length(xrange);
    my=length(yrange);
    err=(mxy-mx-my+1)/(2*N);
    MI=MI-err;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Joint probability by KDE algorithm %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function P=KDE2(x)
% KDE estimation of joint probability.
% Inputs:
% x - combined vector [x1,x2] for two variables x1 and x2.
% Outputs:
% P - joint probability matrix p(x1,x2).

[N,M]=size(x);
dimcell=cell(1,M);
dims=zeros(1,M);
S=cov(x);
ddet=det(S^(1/2));
% An singular covariance
if ddet==0
    P=NaN; % Full information
    return;
end
h=(4/(M+2))^(1/(M+4))*(N^(-1/(M+4)));
for i=1:M
    dimcell{i}=unique(x(:,i));
```

```
    dims(i)=length(dimcell{i});
end
P=zeros(dims);
if M==1
    P=zeros(dims,1);
end
% For all possible state values
for i=1:prod(dims)
    sub=index2sub(i,dims);
    xx=x(1,:);
    for j=1:M
        xx(j)=dimcell{j}(sub(j));
    end
    % Distance between the state to be estimated and all sampled data
    temp=ones(N,1)*xx-x;
    u=zeros(1,N);
    % Summation of the kernel functions
    for j=1:N
        u(j)=(temp(j,:)*inv(S)*temp(j,:)')/(h^2);
    end
    K=1/((2*pi)^(M/2)*h^M*ddet)*exp(u*(-1/2));
    P(i)=sum(K)/N;
end
P=P/sum(P(1:end));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mutual information by KDE algorithm %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function MI=KDEMuInfo(P,xdim,ydim)
```

```
% Estimation of mutual information by KDE algorithm.
% Inputs:
% P - Joint probability marix of x and y.
% xdim - columns that describe the first variable x.
% ydim - columns that describe the second variable y.
% Outputs:
% MI - Mutual information.

if isnan(P)
    MI=NaN;
    return;
end
N=size(P);
p=P;
% Marginal probability
for i=ydim
    px=sum(p,i);
    p=px;
end
px=reshape(px,[prod(N(xdim)),1]);
% Individual entropy
Ex=px.*log2(px);
Ex(isnan(Ex))=0;
Ex=-sum(Ex);
p=P;
for i=xdim
    py=sum(p,i);
    p=py;
end
py=reshape(py,[prod(N(ydim)),1]);
Ey=py.*log2(py);
Ey(isnan(Ey))=0;
```

```
Ey=-sum(Ey);
P=reshape(P,[prod(N),1]);
% Joint entropy
Exy=P.*log2(P);
Exy(isnan(Exy))=0;
Exy=-sum(Exy);
MI=Ex+Ey-Exy;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C. Table of Mathematical Symbols

Ω	time-space inconsistency
$\theta(k)$	entity orientation
δ	simulation time-step
τ	prediction span
$a(k)$	entity acceleration
d	maximum prediction time-delay in Neuro-reckoning
h	local threshold
k	simulation step index
q	maximum prediction horizon Neuro-reckoning
r_i, r_{ij}	sample occurrence
$\boldsymbol{u}(k)$	ESU
$x(k), y(k)$	entity state
$x(k+\tau) = x_\tau(k)$	entity state at τ steps after k
$\hat{x}(k)$	remote entity state model
$\tilde{x}(k+\tau) = \tilde{x}_\tau(k)$	state speculation sequence at τ steps after k
$v(k)$	entity velocity
AB	available bandwidth
$H(X)$	entropy
$H(x)$	information generation rate
$H(X,Y)$	joint entropy
$H(X\|Y)$	conditional entropy
$I(X;Y)$	mutual information
$I(\boldsymbol{u}; x_\tau)$	available information
$I(\tilde{x}_\tau; x_\tau)$	extrapolated information
L	network latency
L_a	overall latency
L_T	transmission delay through bottleneck link
L_f	other factors in latency except L_T

MI_s	locally stored mutual information
N	sample size
OCC 95%	sample occurrence statistics
$R_{s,local}$	locally stored information rate
$R_{s,remote}$	remotely stored information rate
$R_{u,remote}$	utilized information rate
$S = \{s_i\}$	entity state space
T_f	functioning period
T_{Lf}/T_{Rf}	local/remote functioning period

Index