INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Stability of Haptic Displays in Distributed Virtual Environment

Md. Wahidul Islam

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

November 2001

ABSTRACT

Stability of Haptic Displays in Distributed Virtual Environment

Md. Wahidul Islam

The term "Haptic" is associated with the sense of 3D touch from Virtual Environment. Stability is a critical issue in haptic simulation. Unlike other conventional robotic manipulators, haptic devices inherently function in close proximity to humans. The unpredictable nature of human operator and the virtual environment are the main sources of instability in a haptic Virtual Reality system. To meet the growing demand of the development of Distributed Virtual Environment (DVE), stability of haptic displays in DVE is now an interesting and challenging research problem. In the case of network-based haptic display the criteria for stability are far more complicated and requires the force and kinematic data to be transferred over the communication links. The delays over network add phase lag to the signal and this lag limits the effective bandwidth which in turn may hamper the stability of tele-robot or haptic display system. The current approaches toward stability in presence of expected time delays are to reduce the total loop gain sacrificing the performance requirements. To guarantee stability in the presence of random and unpredictable time delays new control algorithms and software are required.

In this thesis, Dead Reckoning Algorithm in haptic simulation is introduced to guarantee stability for any kind of time delays or loss of information over network. Haptic dead-reckoning allows a comparatively simpler, local environment model to serve the haptic rendering while being periodically updated by a more complex and accurate model. Two force calculation models - "Complex" and "Simple" are simulated with an order reduction technique in a two user DVE with haptic devices. Force outputs for different types of interaction from both the force calculation models along with the errors between them are investigated. Stability and discrete-time passivity of haptic displays are guaranteed for any kind of delays without sacrificing the total loop gain. This technique enhances the real-time performance in force feedback realization and reduces the human risk involved.

Dedicated to my parents...

ACKNOWLEDGMENTS

I could not have completed this thesis without the assistance of many people who helped me either directly or indirectly. First and foremost, I would like to express my thanks and indebtedness to my supervisors Dr. Asim J. Al-Khalili and Dr. Khashayar Khorasani for their constructive technical advice, financial support, constant guidance, and encouragement throughout this work. The discussion they had with me relating the thesis played the most significant role in writing this thesis. I appreciate the time they devoted to help me understand the concepts involved in this thesis.

Without the continuous support of my family and friends, it would not be possible for me to successfully complete all the complex works and simulations. I thank my parents for their life time support that came from the far end of the East (Bangladesh). I also thank my brothers and sisters for their endless inspiration towards my graduate studies. I express my gratitude to all my friends who helped me reading the proof of my initial report.

Special thanks to our system administrators, Claude and Guy for installing and fixing problems of the Phantom haptic device and the Ghost API. I also thank Daniel and Wojciech in helping me with UNIX and Framemaker. The Inter Library Loan people deserve my thanks for helping me in locating and even managing copies of the papers not available in the local libraries.

Finally thanks to all those people who have not been mentioned here but who have helped me during my masters. And of course without almighty Allah nothing would have accomplished by me.

Md. Wahidul Islam

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

VE	Virtual Environment
DVE	Distributed Virtual Environment
Net-VE	Networked Virtual Environment
HMD	Head Mounted Display
DIS	Distributed Interactive Simulation
PDU	Protocol Data Unit
AOIM	Area of Interest Manager
LAN	Local Area Network
WAN	Wide Area Network
UDP	User Datagram Protocol
TCP	Transfer Control Protocol
IP	Internet Protocol
PICA	Protocol Independent Compression Algorithm
MUD	Multi-User Dungeon
NSA	Network Software Architecture
VRML	Virtual Reality Modeling Language
CS	Central Server
DoD	Department of Defense
HTTP	Hypertext Transfer Protocol
MTP	Multicast Transport Protocol
RMP	Reliable Multicast Protocol
ALF	Application Level Framing
SRM	Scalable Reliable Multicast
NAK	Negative Acknowledgement
ISTP	Interactive Sharing Transfer Protocol
URL	Uniform Resource Locator
ESPDU	Entity State Update Protocol Data Unit
RTP	Real-Time Protocol
VRTP	Virtual Reality Transfer Protocol

HTML	Hypertext Markup Language
SNMP	Simple Network Monitoring Protocol
SNTP	Simple Network Transfer Protocol
DWTP	Distributed Worlds Transfer and Communication Protocol
MBone	Multicast Backbone
CBone	Cyberspace Backbone
HLA	High Level Architecture
LOD	Level of Detail
FSR	Force Sensitive Resistors
WYSIWYF	What You See is What you Feel
LTI	Linear Time-Invariant
CAD/CAM	Computer Aided Design and Machining
CSG	Constructive Solid Geometry
NURBS	Non-Uniform Rational B-Spline
BSP	Binary Space Partitioning
Obb	Oriented Bounding Boxes
NOD	Null Object Detector
AABB	Axis-aligned Bounding Box

Chapter 1

Introduction

The domain and scope of Virtual Reality (VR) or Virtual Environment (VE) is very broad and may be seen as inclusively encompassing many other information technology related applications. Virtual environments and virtual reality applications are principally characterized by human operators explicitly interacting with dynamic and truly realistic 3D world models with sophistication and significant complexity. Virtual reality today is not merely confined to the entertainment industry. Indeed, VR has found a strategic place in the fields of engineering and design, manufacturing, medicine, training, scientific visualization and multimedia. The concurrent engineering, product prototyping, tele-medicine and tele-surgery, tele-robotics and space research, and tele-immersion, among others, are all based on this emerging and enabling technology having significant and wide ranging social and economical implications. It is our view that the future of VR is toward Distributed/Networked Virtual Environments (DVEs). The Web is emerging as a new supercomputer with all the countless machines that are interconnected together by the internet. The distributed computing along with high speed personal computers and networks facilitate the development of innovative tools and technologies that may be utilized in distributed VEs where a simulated world runs not on a single computer but rather on several machines that are networked.

A DVE or Net-VE is a software system in which multiple users located at distant locations around the world interact with each other in real-time. It can be characterized by some common features as follows: A shared sense of space where all participants sharing the same space should get the same sensational feeling such as temperature, tactile acoustics etc. A shared sense of presence where each user is represented as an entity in the VE and should be able to interact with each other as if any user can see or feel the activities of others with realistic realization. A shared sense of time where real-time interaction should be feasible for each and every user. A streamline communication for performing cooperative tasks among all users such as collaborative design or training systems where there must exists feasible ways for communicating among the participants by gesture, text or voice. Immersion and sharing where the user must interact with the VE itself realistically to accomplish any individual or collaborative work [1].

Various components of the DVEs can be characterized as distribution which must contend with managing network resources, data loss, network failure and concurrency; Graphics that must maintain smooth, real-time display frame rates and allocation of CPU among rendering and other tasks; Interaction where the users should see the VE locally with distributed participants and be able to input user data through certain interaction devices in the VE. These components interact with each other in a complex way and work on top of a number of existing application services. The development of DVE is a difficult balancing act where optimization of one element can severely effect the other components. The problem areas involved in developing a successful and a "balanced" DVE are numerous ranging from network and end user interaction to hardware and software architectures.

1.1 Problem Areas

Despite trade-off in optimizing each component of the DVE there are numerous problem areas involved in the development of DVE that must be solved and requirements satisfied. The basic requirements in the development of any kind of successful and practical DVE may be specified as follows:

i. Network Bandwidth: The backbone of any DVE is the capability to exchange information over the data network. The information may contain the current state of the entity or avatar or it might be the initial massive download of the VE. As the number of users increases more information must be transmitted. In the era of limited bandwidth the communication architecture must be designed in an efficient manner to meet the restricted available bandwidth. Appropriate and efficient protocols must be developed to overcome this issue. In this regard an appropriate selection of the users in the DVE must be satisfied either through physical (geographical) consideration or other parameters that affect the communication bandwidth [2].

ii. Scalability: Scalability defines and determines the number of entities that may simultaneously participate in the system. The entities may represent human-controlled and computer-controlled avatars, a terrain or even logical objects such as weather information or group of objects. Factors that affect the scalability are network capacity, processor capabilities, rendering speeds and throughput of shared servers. It is the most "expensive" aspect of the development of DVE as it requires enhancement of almost all features of the DVE system. The complexity increases exponentially with the number of participating entities because of the possible interactions among those entities. There can be a maximum of 2n possible entity-entity interactions where n is the number of entities in the VE. But in reality an entity does not interact with every other entity and all entities do not interact at the same time. Therefore the concept of localization is introduced and applied by which the entities can be grouped in an "intelligent" and efficient way to minimize the communication between them. This process of grouping is rather dynamic with the VE as their existence inherently reduces the number of interactions [3].

iii. Heterogeneity: Heterogeneity deals with dissimilar network capabilities, end users' graphical display, computational and audio capabilities and even the interface attributes of the VE at the user's end. The design of DVE must support different network speeds to which users are connected to. The designer can in principle choose the lowest possible connection speed for the VE and user requirement. The problem with this choice is that it invariably reduces the realistic immersion sensations of the VE by the users. An optimal

choice can be made either dynamically in accordance with the available resources or may be obtained as a trade off with the level of immersion. On the other hand users may be equipped with a wide varieties of terminals ranging from PCs to high speed workstations. The response speed of one user may significantly impact the immersion quality of the other users. The interaction among user interfaces and VE also affect the quality of DVE. The interface devices may be Head Mounted Displays (HMD), crystal eyes or simple monitors for graphical displays as well as force feedback haptic displays. In designing a DVE the above issues must be considered and optimized carefully.

iv. Distributed Interaction: This is the most important quality for consideration in developing DVE. Each user must have a "realistic" illusion that the entire virtual environment is accessible and his/her actions are having a direct and an immediate impact on the environment. The design must mask any artifacts and anomalies that might arise because of the application's distributed nature. Different messages may incur different delays depending on the characteristics of the network and on the location of sources and destinations. Therefore the main issue here is that each host system must attempt to present a consistent real-time immersion despite the fact that all of the incoming information about the remote hosts are already out-of-date after arrival. This problem is severe in the case of collision, agreement and resolution among users. Accurate collision detection is very difficult as at any given particular time. No user has the correct information about the position of other users unless the object or the user is static and stationary. The problem becomes even more critical when in addition to the collision other kinds of information such as heat, sound, forces etc. are involved [1].

v. Real-Time Issues: The process and thread architecture of the DVE must be well defined because of the numerous and diverse contending tasks. Most of them have hard real-time constraints. The quick detection and processing of user action require real-time interaction with the DVE. Graphical image generation must occur at a fixed rate. Network packets arrive asynchronously and need to be processed as soon as they arrive. Modeling and collision detection must be performed at a reasonable rate with several samples per second. Usually all the available CPU cycles are devoted toward generating high quality

graphics populated with modeling, texture, rendering etc. requirements. One naive approach is to cycle through all of the tasks in a round-robin fashion sufficiently fast so as to meet their real-time constraints. Alternatively, the application may be partitioned and segmented into multiple threads that are optimally tuned and scheduled to balance their CPU use. Efficient scheduling algorithms are yet to be developed for resource allocation and sharing of the CPU in balancing different tasks in DVE.

vi. Security on Failure: While designing the DVE system the extent of failure that may affect the execution of the application must be considered and resolved. System Stop can be defined as the termination of the entire system if the missing resource is critical to the execution of the DVE. If the central server used for receiving and distributing the data has failed then the DVE is likely to cease operating. System closure is a kind of a failure that may prevent the arrival of new users. If the authentication server fails, new users cannot log into the system. System hindrance can be termed as the degradation of the quality of immersion but not significantly affecting the operation of DVE. For the disruption of a single user, other users may only experience sudden departure of that user from the environment. System continuance is a case when a critical service is supported by a backup server that shadows the primary server's state and can therefore be quickly activated to replace the primary server when that one fails [1]. Handling failure is rather a complicated task. When a single host that provides multiple services suddenly fails, all of those services become immediately unavailable. The allocation of resources and services among the hosts and networks must be achieved through a well defined and systematic procedure and methodology.

vii. Software Architecture: The software must be designed around a core library where components may be downloaded dynamically depending on the changing needs of the executing DVE. The software must ensure that the environment is easily downloaded, must meet the security requirements of the browser in case of Net-VE and must be compatible across different platforms and operating systems. The standard 3D graphics markup language VRML (Virtual Reality Modeling Language) has not yet achieved true portability across different platforms.

1.2 Objective

Haptic display is an emerging technology with the implication of wide spread application in human operator training design prototyping. However, very few of these applications have been realized so far. One of the reasons for this is that haptic displays can not generate convincing realistic illusion even with simple environment. The limitations mostly emerge from mechanical design, electromechanical interface, computing hardware, software development tools, control architecture and simulation algorithms. One of the major shortcomings of haptic displays is instability both in stand-alone and in a distributed VE system. The main objective of this thesis is in development of conditions for the guaranteed stability of haptic displays in DVE.

1.3 Contribution

A distributed 3D graphics environment with two users interacting simultaneously with force feedback devices are simulated. The feasibility of dead-reckoning algorithm to ensure stability of the haptic devices and the quality of immersion with real-time force feedback in the presence of unexpected delays over network is presented for different scenarios of interaction.

The outline of this thesis is as follows: In Chapter 2 current major issues and possible solutions in the development of DVE are presented. Chapter 3 deals with the stability criteria of stand alone haptic displays where Chapter 4 introduces the algorithms for stability of haptic displays in the distributed environment. Chapter 5 describes the implementation and results of dead-reckoning algorithm utilized to guarantee the stability of haptic displays in DVE. Finally, Chapter 6 concludes the thesis with suggestions for future work and possible improvements.

Chapter 2

Background Information

Successful development of a DVE relies on implementation of diversive technologies. The components of DVE should seemlessly interact and cooperate with each other in order to achieve the realistic immersion and quality of interaction. Some major technological issues that mostly affect the design and performance quality of a DVE are briefly discussed below.

2.1 Communication Systems Architectures

The communication architectures for DVE are based on Distributed Interactive Simulation (DIS) protocol and its Protocol Data Unit (PDU) [4]. The PDU contains necessary information to manage the state of different entities. The improvement over the standard DIS is the NPSNET-IV DIS [5]. The improvement facilitates smaller packet configuration with higher rate of information transmission and higher scalability. The physical carrier constraints which are not possible to overcome immediately, impose obstacles for high speed packet transmission. One way to address this issue is to have a trade off by developing appropriate communication architectures that make the best use of the existing network capacities. The basic client-server with multiple-server architecture requires high speed to have proper illusion of the virtual environment [6, 7]. Even servers with significant computational powers can't support the desired scalability requirements of the

DVE.

One of the most important design goals of DVE is to take full advantage of peer-to-peer architecture and to scale within the available limitations of the computing resources. In this architecture no server is involved in communication between any two hosts. Either broadcast or multicasting approaches are used to transfer packets. The problem with broadcasting approach is that it keeps the network and computers busy to discard useless packets and possibly waste bandwidth. In multicasting approach packets can be directed to machines that have subscribed to particular and agreed upon multicast groups.

In multicasting approach when the packets subscribed to a specified multicast group arrive at network interfaces, they are brought through the operating system kernel where they can be processed or discarded. Like broadcast schemes, multicast approaches flood the Local Area Network (LAN) but reduce requirements for processing at each user side in DVE. For a scalable DVE, multicasting seems to be the only practical approach. To efficiently utilize the multicasting scheme, distributed software must be developed for assigning packets to proper multicast groups. This software is known as Area of Interest Manager (AOIM). This layer of software assigns outgoing packets to the appropriate multicast group, receives incoming packets and propagates them to the appropriate state table in the local user machine, keeps track of available groups and takes care of incoming and outgoing stream information [8].

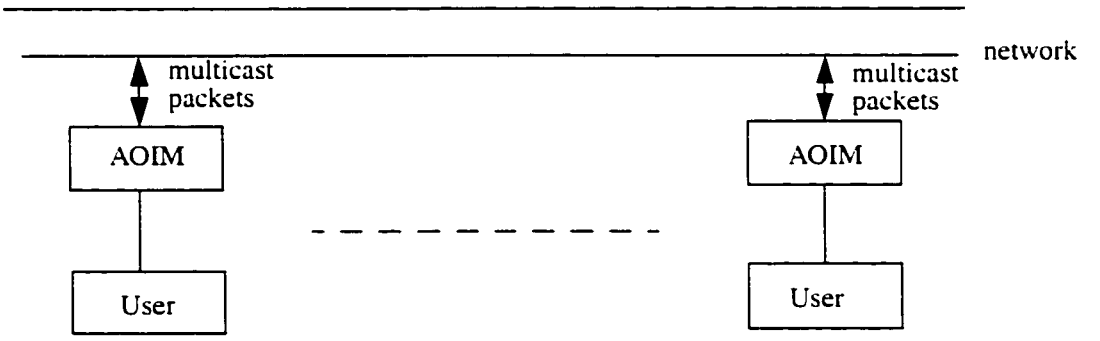

Figure 2.1. AOIM software layer.

The main current problems and limitations are that multicast capable routers in WAN (Wide Area Network) are not yet universally implemented. For the purpose of propagating packets, multicast packets are wrapped inside a unicast UDP (User Datagram Protocol) packet using a machine designated as a multicast router. The region of network where multicasting is known and the UDP packet is destined to, a server is available to remove the unicast UDP wrapper and put the packet in the local segment as a multicast one. Still the success of DVE over net depends on the development of internet capable multicasting approaches.

Management of available resources for DVE or Net-VE is an active area of research. To have a well populated DVE the requirements for resource utilization are critical. A realistic DVE requires to represent thousands to millions of entities as either users or computer simulated entities. Therefore the crucial problem is how to efficiently manage these resources and simultaneously cope with the limited network bandwidth and processor speed. Savings in processor load can be used to generate higher quality graphics (after ensuring that realistic interactions at the user end are achieved), whereas savings in the network bandwidth can be used to manage shared state and to exchange the updates more frequently. The problem here is that different Net-VE architectures use different techniques to manage the resources and a commonly accepted standard and protocol is yet to be developed.

The addition of any new user imposes significant constraints and requirements on the bandwidth. Each new participant must receive the initial VE state and updates of the other existing participants. This requires new updates on the existing shared state of the VE and new interactions. Moreover, each new user introduces additional shared state to the VE relating to its own position, orientation and graphical representation vis-a-vis other objects or entities in its own environment. To maintain these additional information, stringent constraints are imposed on the bandwidth and the processors. The resources can be approximated using an empirical expression using the variables like resources, number of messages, hosts-bandwidth requirement, speed of network and processor cycle [1]. The modification of any variable requires an increase or a decrease in other variables or even in

the quality of realism with the VE. The choice of which variable to compensate is application dependent. The resource management problem can thus be considered as an optimization problem subject to particular specifications and requirements. Towards this end, several approaches have been proposed in the literature. Some of the most common methods are described below.

(i) Packet Compression and Aggregation: This approach reduces the size of transmitted packets over the network. Either lossless or lossy compression techniques may be used to reduce the packet size. It can also be based on either "Internal" or "External" compression procedures. Internal compression does not consider the content in the previous packet, and compression is achieved by detecting redundancy within the data being transmitted. In contrast, External compression takes into account the content of the previously transmitted packet and the next packet is formed by taking into account only the changes from the previous packet. Though the external compression yields better bandwidth reduction over internal compression, it introduces and requires coupling and dependency between packets with explicit demands on successful transmission of all previous packets. The choice of the selected compression algorithm depends on many factors such as the frequency of packet updates, content of packets, communication architecture and protocol being used. One possible approach is that of Protocol Independent Compression Algorithm (PICA) [9], where redundant information from successive update packets are eliminated. Localized compression is another approach where compression or decompression tasks are performed only when packets are entering or leaving high speed LAN in order to meet the high traffic constraints over WAN [10]. Packet aggregation reduces the number of transmitted packets by merging information from multiple packets into a single packet. The merging is performed by eliminating repetition both within an individual packet and between successive packets. Updates for multiple entities are generally not generated simultaneously. It introduces a trade off in packet aggregation. In time-out-based transmission policy [11], individual updates are collected and transmitted as one packet after a certain period of time. In quorum-based transmission policy [11], individual updates are collected and transmitted as one packet until the aggregated packet contains a certain number of updates. Currently the best

approach seems to be a hybrid strategy composed of these two techniques. The aggregation server collects packets from different hosts of similar type managed by AOIM and merge them into a single packet to be transmitted. This approach further improves over aggregation at each host. The reason is that most of the hosts manage only one or few entities on their side. Therefore, the collection of packets from multiple hosts that are members of the same multicast group further reduces the number of packets to be transmitted. One DVE might have multiple aggregation servers. The distribution of aggregation servers depends on the nature of the grouped users and on the type of entities for which it is dedicated to.

(ii) Area of Interest Manager (AOIM): This approach reduces bandwidth by optimizing the number of hosts that receive each message. It partitions the VE into a set of workable small scale environments or classes for reducing the computational load on hosts and at the same time for minimizing the communications in the networks and for localizing the reliability problems. The AOIM exists in every simulator that distributes and partitions the processing among hosts as shown in Figure 2.1. The data flow optimization can be defined by using the aura-nimbus model to quantify a client's region of interaction [12]. A "focus" refers to the area of space in which a client is interested on and "nimbus" is the area in which that client can be detected. A client's "aura" is a bounding region containing both the focus and nimbus for determining the interaction. When two client's aura overlap there is a possibility of interaction between those two. This model was first used in DIVE [13] and MASSIVE [14] VE systems. The problem with this model is that it can not scale effectively. The AOIM residing in between each host and LAN receive subscription details from each participating hosts to determine which of the hosts requests are satisfied by that data to only send information to the corresponding host.

(iii) Culling Techniques: Different culling techniques are employed for defining multicasting group. Namely, Spatial, Functional and Temporal cullings are commonly used in the literature [15]. The *spatial culling* technique is used to determine communication between entities based on visual occlusion. Multi-User Dungeons (MUDs) [16] have initially used this idea. If two users are in the visual range of each other

then they can be in the same group otherwise, if their mutual visualization is occluded by some obstacle then they no longer should be in the same group. *Functional culling* is based on the functional groupings of clients. Entities having similar working functions can be grouped together to communicate with each other. *Temporal culling* depends on the resolution of the update rate. Certain entities may not actually require real time updates of all of its entities due to their particular functionality. Entities of this kind can be grouped together to have a low-resolution update. Consequently, the grouping of the clients depends on the software architecture and the algorithms used in the generation of the AOIM software. In a realistic DVE the simulation of complex behavior and functionalities of different entities poses a great challenge in culling techniques. Each specific application requires a particular culling technique that is suitable and efficient for it as it is hard to implement a universal culling technique in grouping all the hosts and entities. The dynamic behavior of entities introduces another challenge for assigning a particular group to each entity or host dynamically. NPSNET culls communications by separating the VE into grid cells. Hosts receive information only from those cells in which they have interest. Clients with low-bandwidth and those that have a small field of awareness occupy only the grid they are in and clients with higher-bandwidth and with greater awareness require a greater number of grids [8]. There is an inherent delay associated with the information transfer when an entity changes its multicast group to another one as in addition to the update information transfer now there is also the need to have the initial information about the new part of the VE. To avoid this boundary or temporal aliasing it is conceivable that the same entity can belong to several groups at the same time and only the information about the region or group in which it is in are transferred to. The association groups to any particular entity is managed dynamically as the entity moves or change its group in the VE.

Another possible concept in chunking the VE is the use of Locale and Beacon that is commonly employed in the development of Spline [17, 18]. Locales divide a large VE into many chunks that can be processed separately which is not apparent to the user. Each user sees several locales at the same time including its point of view and its neighbors. The user does not experience any temporal aliasing between locales or any abrupt changes while its

point of view change from one locale to another. Each locale is addressed with a separate set of multicast addresses having its own coordinate system. On the other hand beacon approach supports content-addressable communication. It facilitates in deciding what locales to attend to based on the content of that locale. To implement this process each beacon object contains two key fields: a tag and the multicast address of the locale that contains the beacon. The creation of tools to support the design of locales is still an area of research work.

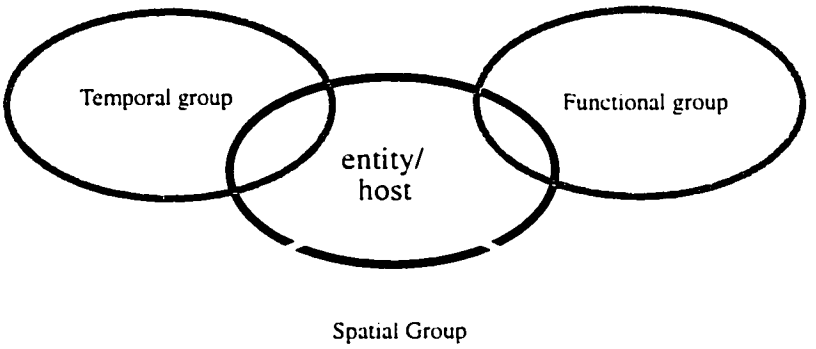

Figure 2.2. Entity and multicast group.

(iv) Multicasting is a solution to the problem of partitioning the available data among a set of groups. One naive approach to the group mapping is to assign a different multicast address to each entity in the VE [19]. Each host executes its AOIM based on the available information about the entities that are present in the VE, and then subscribes to the multicast groups for entities of local interest. The problem is that entities cannot specify their aura and have no control over which hosts will receive that information. Another possible approach is that each entity subscribes to the information for entities that are located nearby in the VE [20]. The allocation technique can be extended as multiple group address to each entity. Subsequently each entity can transmit specific type of packet to a particular address. The current state of participating entities may be managed through some direc-

tory service. Beacon Server [17] for each region (locale) of the VE has a designated multicast address for receiving information about entities of the associated region. It also shares a beacon multicast group which is used by the clients to query and locate the appropriate beacon server. The problem with allocation of one multicast group per entity is that in general due to the large number of addresses required for a VE, interferences with other applications using those addresses may occur. In addition to creating an overhead on the network routers for processing joining and leaving requests, this approach may potentially overflow network capacity at each host. This limitation leads to the "promiscuous mode" [1] which forces processing of the packets to be performed within the software protocol stack. One of the possible solutions to this dilemma is to recycle the available addresses [21]. A large number of addresses introduces overhead on the router state information. The optimization of the number of state information to be transmitted and multicast packet processing in the router are major challenging research problem in this area.

Another approach for assigning multicast addresses is through one group per region basis. The VE is partitioned, as discussed earlier, into cells or locales. In a hexagonal cell system, where the partitioned cells are hexagons, a dynamic directory service is implemented [8]. The entity that has been active within a cell for the longest time acts as a group leader. When an entity enters a region it multicasts a "join" packet and while leaving it multicasts a "leave" packet. The group leader maintains a list of all the entities currently in the region and is responsible for providing that information to any new entity in that cell. The problem with this approach is that at the boundary of the two cells all the entities crossing it must subscribe to multicast groups for both the adjacent cells. This problem is alleviated to some extent by the concept of locale. It is generally impossible to implement and achieve the full advantage of the AOIM and multicasting within a single system. Rather a hybrid approach should be attempted to integrate "best" elements from both techniques [19].

(v) Exploiting Perceptual Limitation is another approach for managing the available network resources and hence for increasing the scalability of DVE. Human perceptual limitations are exploited in two ways to reduce resource requirements. In the exploitation

of Level of Detail Perception, information about entities in the VE can be provided at multiple levels of detail and at different update rates. The users located in close proximity of the entity need to receive higher detail resolution and update frequency information whereas other users can have lower resolution and update frequency information as long as a certain level of realism is maintained. This approach introduces multiple channel architectures which leads to different levels of data frequency and reliability at different channels [22]. Rigid-body channel requires the least network bandwidth and processor computation resources. It supports far range viewers and provides three types of updates-position, orientation and structure. Approximate-body channel provides more structural information in addition to position and orientation such as radial length, articulation vector and local coordinate system points. A full-body channel provides the highest level of resolution about the entity's dynamic position, orientation, structure and clearly requires the highest bandwidth. When compared with the single channel approach, the multiple channel approach introduces more computation at the source host, more traffic on the network close to the source and some additional bandwidth at destination if multiple hosts subscribe to different channels. In exploiting Temporal Perception entities are grouped in two categories- active and passive. Active entities being human participants or computer controlled entities take actions on their own, whereas passive entities only react to events from the environment or active entities and do not generate their own actions. More update information is required for active entities compared to the passive ones [23]. The challenges for DVE designers are to allocate an appropriate number of channels for each entity without overflowing the multicasting addresses when the VE is populated with a large number of entities.

(vi) Network Software Architecture is the last optimization technique used for scalability of DVE. This architecture is based on the client-server and peer-to-peer communication paradigms, and optimization is achieved through combining these two basic architectures. A basic client-server architecture suffers from scalability problems due to the fact that all traffic must pass through the central server acting as a simple broadcast reflector or a filtering reflector or a packet aggregator. Partitioning clients across multiple servers can help to improve the scalability dilemma. Figure 2.3 shows a possible architec-

ture where clients send and receive updates through one of the servers and the servers themselves communicate using peer-to-peer protocol [7]. The server to server communication is controlled through some control messages which contain a list of entity locations, the grid or cell information [24] or they may contain entities' functional interests [15]. Multiple servers also introduce problems of their own. The update packets need to go through multiple servers which introduce greater latency compared to a single server architecture. The total processing and bandwidth required by the servers are naturally greater than the resources required in a single server system. To take full advantage of the high speed WAN in distributing the workload among servers, bi-level multicasting groups ([10] and [25]) can also be employed.

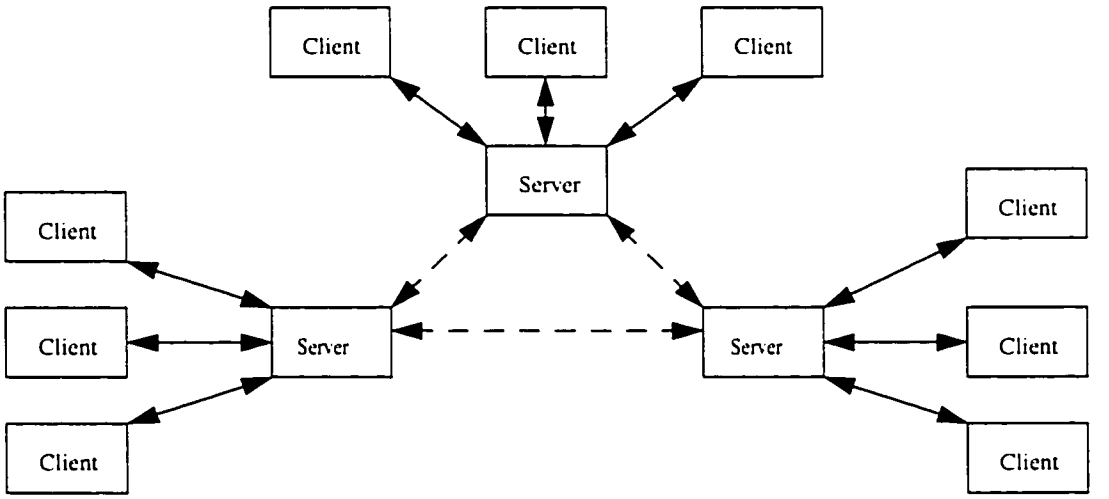

Figure 2.3. Partitioned clients across multiple servers.

Another approach in Network Software Architecture is to partition the VE among multiple servers. Each server is responsible for the communication of entities of particular region of the VE as shown in Figure 2.3. This architecture illustrates the challenges in exchanging information. Given that the designer must statically define the visibility relationships among the regions of the VE, when a server receives a packet it checks for the

region in which the active entities can see or realize the updates and forward the updates to the servers that manage that region with all those specific entities. An advanced approach that may be used is to partition server environments into different levels of hierarchy for reducing the overhead of the peer-to-peer communications among the servers in lower level hierarchies. The problem here is that there must be some local information among the server partitions. Otherwise, the server hierarchy degenerates into a server-to-server broadcast system. Therefore, the servers' regions of responsibility defined should correspond to the local regions in the environment. A hybrid peer-server architecture that embody the best characteristics of client-server and peer-peer architectures has then emerged [26]. Over short distance and high-bandwidth links the hybrid approach employs peer-peer communication and for long distance and low-bandwidth links it invokes client-server communication. The technique is therefore rather adaptive and dynamically adjusts the system architecture for network topology, host location and network dynamics.

Development of a universal approach for network software architecture is still not feasible at present for complex, dynamic and heterogeneous network environments. Considerable amount of work remains to be done in this area.

2.2 Real-Time Protocol Paradigms

To place the VE on a network requires that the environment and the supporting software be downloaded to a browser running on any machine supported by any platform so that various users can see and interact with the DVE. With the development of Virtual Reality Modeling Language (VRML) the possibility of a large scale DVE over a network is now promising and research is now aiming towards the development of a proposed specific sets of protocols that could enhance the communication needs of DVE. Several approaches are proposed in the literature to meet the needs of real-time protocols for DVE. It is an ongoing research activity to develop a protocol like HTTP to meet the distributed requirements of DVE over the net. Among the earliest protocols attempted were Central Server (CS) system and the Distributed Interactive Simulation (DIS) standard [4] by the Department of Defense (DoD, USA). Both CS and DIS provide the best possible real time performance and state quality. DIS uses the peer-to-peer communication and have better speed over the

CS. But being centralized in nature CS provides better state quality than DIS. The main problem with CS is that it inhibits scalability. Though DIS is fully distributed in nature and supports scalability, every node in the DIS simulation is informed of every change that occurs anywhere in the simulated world. Therefore, each node is informed with torrent of information so that the bandwidth of the network is wasted which would then limit the scalability capabilities. Furthermore, DIS requires dedicated special-purpose network to have a realistic DVE.

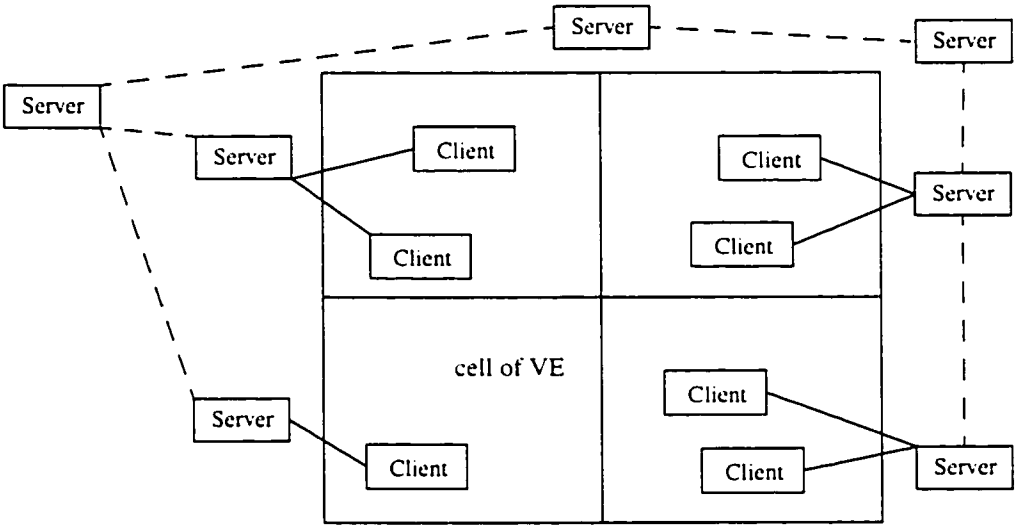

Figure 2.4. Client-server architecture with hierarchy of servers.

Multicast Transport Protocol (*MTP-2*) is a multicast protocol developed to achieve reliability. A central server called master manages the ordering and numbering of packets and guarantees some sort of reliability through sending tokens to all the members. Clients are divided into two groups namely producer-clients that tries to send packets and consumer-clients that attempts to receive packets. The flow control is managed by time-out value and re-sending of all the lost packets [26]. RMP (Reliable Multicast Protocol) is another protocol that uses token passing to increase reliability. The lost packets are managed to revive through re-sending via a point-to-point connection (UDP). The problem

with this approach is the complexity and delay involved in joining and leaving a multicast group. When joining all members must be informed of the new member through token site and while leaving all pending messages have to stable [27]. Application Level Framing (ALF) / Scalable Reliable Multicast (SRM) is another reliable multicast protocol in which the NAK (Negative Acknowledgement) explosion effect in response of missing packets from clients is prevented by requesting a recovery state after a random period [28]. The problem with this protocol is that the network latencies have to be very short to ensure the retransmission of the packet is detected before an additional retransmission is sent.

Interactive Sharing Transfer Protocol (ISTP) developed at the Mitsubishi Electric Research Lab is a transport layer protocol and is meant to meet the basic requirements such as real-time quality-state updates are transferred to meet hard deadlines, state quality-processes closely agree on the state of the data they share, scalability-supports as much users as possible like internet, malleability-communication of all kinds of data so that application can be extended at run time, WWW like-DVEs can be rapidly expanded by the addition of content from many sources, and Low bandwidth-users with low and limited bandwidth must be supported and served as long as they are connected [26]. The basic architecture of ISTP with object sharing is shown in Figure 2.5. The world model contains the local copy of the data that processes share via the shared data object model of distributed shared memory. Each object in the world model is owned by a process and only the owning process which is transferable is responsible for any modifications in the model. The model is broken into chunks as locales [18]. Each process maintains the partial copy of only the locale of the world model they are interested in. Peer-to-peer communication is used to meet the real-time needs and scalability is achieved by assigning each locale with separate multicast communication group. Malleability is achieved via a small object called links whose main component is Uniform Resource Locator (URL). WWW-like operation is achieved through locales, links and beacons [17]. Locales allow content to be encapsulated and integrated with pre-existing content, beacons allow it to be located and links make it possible to be retrieved. Lower bandwidth is achieved through reliable communication of entity state update protocol data unit (ESPDU) removing the need for keep-alive messages.

The architecture of ISTP is developed based on sub-protocols such as HTTP, UDP, TCP and Real-Time Protocol (RTP). Specifically the 1-1 connection sub-protocol is used to establish and maintain TCP connection, and the object state transmission sub-protocol is used to communicate the state of objects either by TCP or UDP between two ISTP processes. The streaming audio sub-protocol is used to communicate audio data over RTP. The region-based communication and content-based communication are higher level protocols on top of the three sub-protocols discussed above. Region-based communication supports sharing of information about objects in the world model using UDP multicast backed by TCP unicast when UDP messages are lost, whereas content-based communication supports content addressable connections using reliable communication (TCP) to find requested beacons [26]. The problem with ISTP is that it is still not optimized for video streaming and security is not supported. Moreover, it is more prone to Spline (Scalable Platform for Large Interactive Networked Environments, developed at Mitsubishi Electric Research Lab) platform and is not generalized for different kinds of DVE.

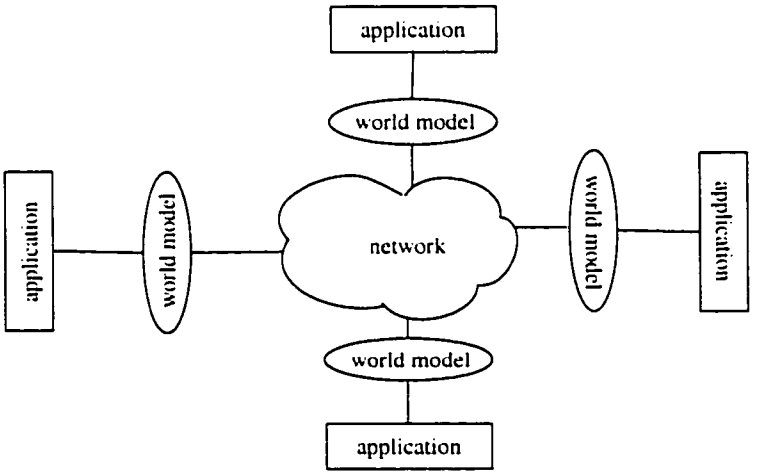

Figure 2.5. Sharing of world in ISTP.

In order to support VRML world over web a more generalized protocol known as Virtual Reality Transfer Protocol (VRTP) is under development [27]. This protocol is based

on multicast architecture and four types of data objects to be communicated. The data objects are: (i) Light Weight Interaction messages are composed of state, event and control information as used in ESPDU in the DIS protocol [4] and are implemented using unicast or multicast architectures. The packet is an encapsulated one without fragmentation. (ii) Network Pointers are references to objects that can be multicast to receiving groups. (iii) Heavy-weight Objects are large data objects that require reliable connection oriented transmission and (iv) Real-time Streams are live audio, video, DIS behaviors, sequential graphics images or other continuos stream traffic that require real-time delivery, sequencing and synchronization which is implemented with a multicasting architecture. The network software architecture is positioned somewhere between pure client-server and pure peer-peer architectures. Users and entities contribute both static content and dynamic streams. The entities participating in DVE simultaneously need to act as clients, servers and peers. All these actions must be provided at once by the VRTP architecture. Entity interactions will occur in multiple ways across the spectrum (Figure 2.6).

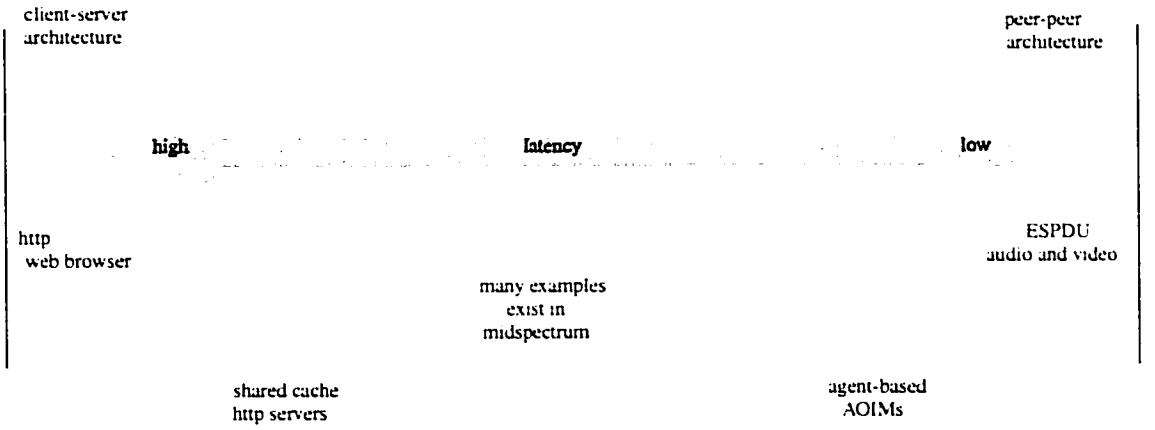

Figure 2.6. Spectrum of the communication architecture in VRTP.

Essentially VRTP is an optimized combination of the available components such as HTTP, DIS, multicast streaming, Java agents, network monitoring etc. VRML is now

extending the flat 2D functionality of HTML into four dimensions: 3D geometric space and temporal behaviors. VRTP is needed to support multiple simultaneous users of shared large-scale web-based interactive 3D graphics just as HTTP was needed to support large-scale use of HTML. VRML graphic primitives provide platform independent capability of putting together 3D scenes. Moreover VRML 2.0 script connections via Java, Java script and links to behavior protocol (DIS) enables animations of scenes both locally and globally [28]. The internet application layer is insufficient to meet the demand of VRML in making a realistic distributed 3D graphics world. Consequently, there is a need to develop a new protocol dedicated to DVE over the net. The four components of VRTP are defined as: (i) Client - when looking at others worlds, (ii) Server - showing others one's own world, (iii) Peer-to-peer - send and receive multiple real-time streams of audio, video and behavior interactions and (iv) Monitoring - diagnose and correct problems across the whole network. Numerous web browsers can act as client component in addition to public domain browsers such as Arena and Amaya from World Wide Web Consortium [29] or Mosaic by NCSA [30]. Researchers are also attempting to incorporate Java Virtual Machine for the Java API for VRML, on-the-fly behavior dial-a-protocol updates, enabling distributed network monitoring programmability and providing automatic VRTP upgrade capabilities. Apache is a natural choice for server component. The research implications regarding this is to extend server capabilities by using perl and other languages. Including up-to-date MBone applications [31] for audio, video, whiteboard, DIS and other behavior protocols have to be embedded to get a highly efficient cross-platform solutions for streaming data. The success of ongoing research on "reliable multicast" protocols and resource reservation will present other peer-to-peer transport protocol in addition to connectionless UDP. The meaningful use of time stamps among participants of DVE using Simple Network Transfer Protocol (SNTP) [32] and query capabilities using Simple Network Monitoring Protocol (SNMP) [33] make them a possible choice for component monitoring. For MBone connectivity evaluation and diagnosis mtrace and mrinfo can also be integrated as discussed in [34] and [35].

Another approach for generalizing protocols for DVE is the Distributed Worlds Transfer and Communication Protocol (DWTP) [36]. It's goal is to support scalability, heteroge-

neous network, high throughput, error correction and adaptive flow control. The data types are (i) event - information about entity state change, (ii) messages - pre-defined set of special events, (iii) file - contains scene description, texture, audio files, avatars etc. and client-server approach is used for the transmission of streams- video and audio data. The protocol uses IP multicasting for many-to-many communication and tasks are distributed among different daemons. The daemons could be one of the following: (i) world daemons - to join a shared virtual world, (ii) reliability daemons - to detect transmission failure, (iii) recovery daemons - to recover from transmission failure and temporary disconnection, and (iv) unicast daemons - to support non-multicast capable participants. As explained earlier large VE is chunked into several multicast groups. All daemons communicate via multicast address only. Additional daemons can be added on new hosts depending on the existing load but it does not put any additional load on the existing daemons. The non-multicast capable users communicate with the unicast daemon using TCP/IP or UDP/IP. As this protocol depends on unreliable UDP/IP, the reliability is achieved by reliability and recovery daemons. Wide area testing with large number of participants on top of this protocol has not yet been demonstrated. Further research is required to support heterogeneous set of participants with different connection bandwidths.

A network with moderately high bandwidth and low latency is essential among universities, academia and government agencies for studying design issues on virtual environment protocols, the development of tools for rapid development of VE protocols and the development of networked software architecture. This leads to the development of Cyberspace Backbone (CBone) [37]. This backbone should include dedicated multicast capabilities, dedicated bandwidth and dedicated latency.

2.3 Real-Time Issues

In the simulation of DVE generally different kinds of processes and threads are involved. The main issues here are resource management and sharing. In case of incorporating force feedback devices efficient algorithms have to be developed for real time interaction between the 3D graphics world and the force-feedback interface assuring the stability of the feedback device along with the realistic immersion of end users. The modelling of the

graphics world also affect the development and performance of the force-feedback devices. The process and thread architecture of the DVE must be well defined because of the numerous and diverse contending tasks. Most of the tasks have hard real-time constraints. The rapid detection and processing of user's action require real-time interaction with the DVE. Graphical image generation must occur at a fixed rate. Network packets arrive asynchronously and need to be processed as soon as they arrive. Modeling and collision detection must be performed at a reasonable rate with several samples per second. Usually all the available CPU cycles are devoted toward generating high quality graphics populated with modeling, texture, rendering etc. requirements. One naive and simple approach would be to cycle through all of the tasks in a round-robin fashion sufficiently fast so as to meet their real-time constraints. Alternatively, the application may be partitioned and segmented into multiple threads that are optimally tuned and scheduled to balance their CPU use. Efficient scheduling algorithms are highly needed, but yet to be developed for resource allocation and sharing of the CPU in balancing different tasks in DVE. The requirement of real time interaction between processes can be presented with the discussion of single threaded and multi-threaded task-driven structures.

2.3.1 Single-Threaded Architecture

In this simplest form of the process interaction different tasks are cycled through a round-robin fashion with reasonably high speed processors and systems. Different tasks may be categorized with their sequence of operations as shown in Figure 2.7. Most single-threaded VEs enter the cycle after the *initialization* phase. This phase includes tasks such as database reading, input/output device initialization with ports set for further communication, network sockets opened, graphics window initialization with the required menus and environment condition, initial downloading of the world model etc., and tends to be as one of the most important parts as other phases often reuse the functionalities from this phase. After initialization, the main event loop is started and proceeds until the execution of the DVE program. The event loop contains different contending events which are described below [1]. Developing this system is rather simple and its performance is good for simple VE if the total traversing time for the complete cycle is less than 100 ms [38]. Otherwise, even in this architecture the smoothness of graphics dynamics and the immer-

sive force feedback realization would be lost. The specific descriptions regarding the system are outlined below (refer to Figure 2.7):

(i) The event *read input devices* may introduce a considerable amount of lag in the loop. Usually the devices connected with the workstations through the serial port may run from "low" baud rates to around 56,000 baud rate [1]. The conventional steps involved in this event are shown in Figure 2.8. A command is sent from the application program down the serial port to the controller of the input device. The controller is a specially designed low cost CPU which queries the input device and in response receives a set of values of the current states of the device. The controller converts the device's state into a meaningful format and sends back the data through the serial port to the workstation. The application program in the workstation receives the state information and accordingly proceeds to the next event.

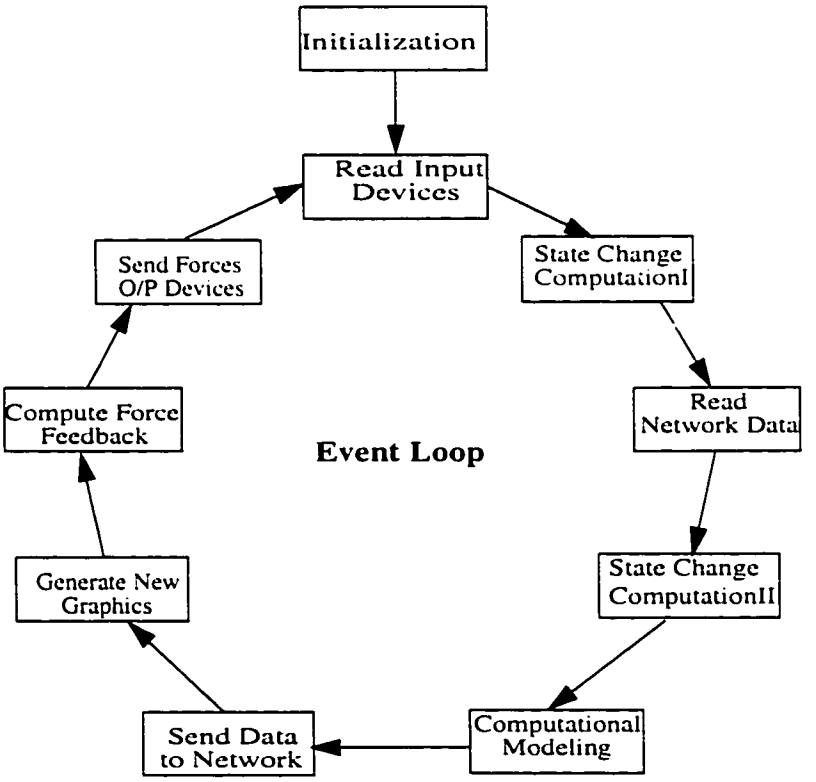

Figure 2.7. A single threaded DVE.

(ii) In the event *state change computation I* the states of different participants and entities are updated according to the input received from the previous event. The time required for this event depends upon the sophistication required in processing new states. The state changes may include updating the local user position received from the input device position changes, updating the change of viewpoint due to input received from HMD (Head Mounted Display) like devices or updating the position of other entities locally due to interaction with the local user entity. Generally, the computational complexity in this event is not "high" and it does not include issues such as hard deadline consideration.

(iii) The *read network data* queries the network to retrieve any information arrived from other users in DVE. The query should be performed only if any data has arrived, otherwise the query should return right away. Generally, reading data from the network through the operating system kernel is rather expensive and a slow operation because it involves numerous data copy operations among memory buffers as the packet is processed by the protocol stack.

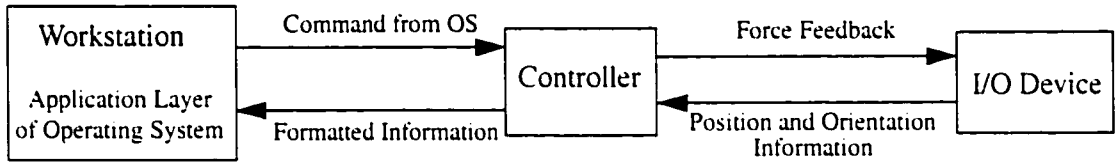

Figure 2.8. I/O device connection and information flow.

(iv) The packets received, if any, in the previous event are processed in the event *state change computation II*. The data is extracted from the PDU (Protocol Data Units) and modifications are done in the states table. The latest information about the states of different entities and force feedback due to their interactions, according to the information contained in the header and data fields of the packets are stored. The changes due to this event

appear as the position and orientation changes of the entities representing other users in the DVE.

(v) The *computational modeling* event computes the physical modeling, collision detection, force feedback due to collision and any other computationally expensive operations. If the modelled VE is simple and without any collision complications, feedback from VE, or complex interaction among entities, then the event loop may ignore this particular event. However, for a complex VE this event introduces a considerable amount of lag due to its processing time. The amount of lag generated in this event depends highly on the level of complexity of the VE required.

(vi) The event *send data to network* involves reading data from buffer, packet formation and sending the packets to the appropriate users (by multicasting) over the network. The packets are sent through the kernel of the operating system. The information contained in the data usually involve the position and configuration of the local user, force feedback information for other users resulting from collision with others and the corresponding state update information. This event introduces considerable lag due to the traversing of kernel of the operating system.

(vii) All the updated information is visualized through the event *generate new graphics*. The time required for this event depends on the complexity of the VE, the number of polygons, type of texture, quality of illumination, light, shadow, level of detail (LOD) of the VE etc. to be rendered, the graphics hardware available on the local workstation and the frame rate required. This event does present many research problems dealing with real time issues as discussed later in this thesis.

(viii) The event *compute force feedback* involves the computation of the feedback forces to be sent to the output device after a collision has occurred either with a local object on the local user side or with any other entities or objects on the other users' side. The data required for the computation are retrieved from the state table updated in the event state change computation II. The computation of the force feedback is performed

locally after receiving the required parameters for the calculation of the force. The time lag involved in this event depends on the resolution and degree of freedom required for the output device. The controller shown in Figure 2.8 is partly responsible for these computations thus freeing some of the workstation main processor load. The realism depends mostly on the performance and computational power of the controller.

(ix) Finally, in the event *send forces to output devices*, the calculated forces are sent to the output device. This event does not introduce much lag into the system as most of the computations are performed by the controller shown in Figure 2.8. We will discuss more about the real-time force feedback issues later in the thesis.

For a fairly complex VE the above technique for managing the threads is not practical. Complicated VE systems demand more computational resources and hence time in every event of the event loop shown in Figure 2.8. Clearly the 100 ms limit is easily exceeded for a very complex VE, which in turn would make the graphics appear as a jerky one, and the quality of immersion either in the display or in the force feedback interaction is greatly hampered. To support a highly populated and realistic VE, multi-threaded architecture is the only feasible solution.

2.3.2 Multi-Threaded Architecture

This architecture partitions the events in the event loop of a single-thread architecture into subsystems. The subsystems work in parallel with a shared memory among them as shown in the Figure 2.9. The results from the computation of each subsystem are stored temporarily in the shared memory and are protected from retrieving for a certain time. The retrieval and write operation is controlled through a protocol which alerts the subsystems when to read and write what kind of data. An efficient protocol of this kind is yet to be developed for interaction of multiple force feedback devices with DVE. Efficient parallel programming in the shared memory environment is a possible solution as discussed in [39] and [40].

As shown in Figure 2.9 the *input/output subsystem* contains read input devices, state change computation I which computes feedback force and send it to O/P devices events. This subsystem may have several threads for several input/output devices. One of the advantages of partitioning the I/O devices is that the speed of the cycle through the thread does not depend on threads in other subsystems. As a result the speed of the I/O devices does not effect the speed of any other block [41].

The *network read subsystem* contains the read network data and state change computation II events. It reads the packets at the network interface either by reading all the packets until no packets are present or one by one packets at a time with finally writing them to the shared memory. For a highly populated VE the first approach is preferable in the sense that duplicate and obsolete packets can be discarded from further computation in the state change computation II event. But this approach may result in discontinuities in the position of entities. The second approach happens to be computationally costly and better suited for small scale DVE. One of the major advantages of this subsystem is that it frees the processor by blocking read of the network without a no wait, non-blocking I/O call which in turn reduces calls through the operating system's kernel and allows the thread to be put to sleep when there are no packets to be read [5].

The computational modeling event in the *modeling subsystem* performs the predictive modeling of objects controlled by other remote hosts, computes the physics and properties of objects residing on local machine as well as any other calculations necessary for the states of the VE. This subsystem can be parallelized if the computations are complex and parallelism is possible. The parallelism requires the development of efficient private protocols to share the data among threads [41]. To meet the real time updates of the VE the modeling subsystem could be comprised of two threads, the dead-reckoning thread and the actual computation thread. If the actual calculations require more than 100 ms, then the results from the dead-reckoning thread are used until the calculations in the computation thread are completed. The utilization of a proper convergence algorithm can minimize the error between the dead-reckoned and the actual calculated values [42].

The generate new graphics is located in the *display subsystem*. This subsystem reads the state data from the shared memory and generates proper display by sending the complete set of transformed 3D models down the graphics pipeline. In its usual implementation this subsystem utilizes four threads namely "app", "cull", "draw" and "timer" [43]. The *App* thread is responsible for computing the view volume, the view point, view direction for the VE and any other application dependent modeling. The *Cull* thread eliminates most of the 3D model thus minimizing the total number of polygons to be sent through the graphics pipeline [44, 45]. The culling technique uses the bounding volume associated with each node of the hierarchical data structure of the scene graph [46]. The *Draw* thread finally draws the visible polygons and sends them to the graphics pipeline. Synchronization is achieved among several frames through the *timer* thread. Each of these threads operate in parallel operating on each frame in the sequence of frames. This multi-threaded subsystem requires sufficient processor resources to support the overhead of managing multiple threads.

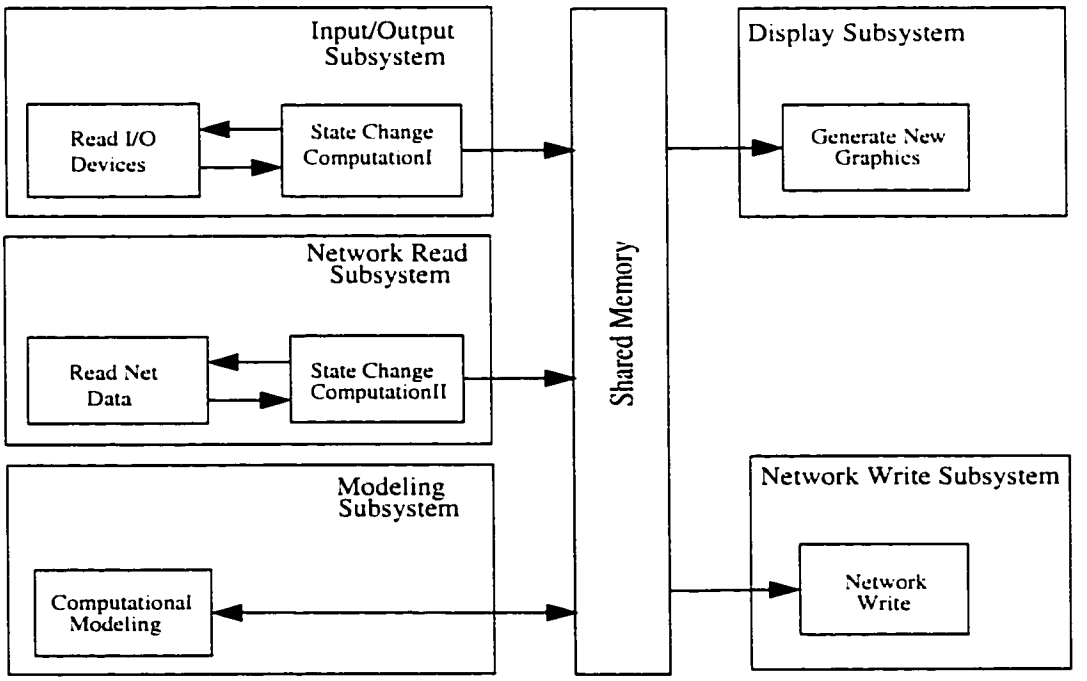

Figure 2.9. Multi-threaded Architecture.

The subsystem *network write* involves send data to network event of the event loop. For a simple VE this thread just reads state data from the shared memory, makes packets (PDUs) and then sends them to the network. For a complex VE, in addition to the above functions, it also reduces the number of packets that are transmitted to the network. This feature is implemented through predictive modeling, i.e. if other hosts which can observe the local host entity can predict the state of the local entity using the dead-reckoning algorithm [2], then there is no need to send the state information packets to those hosts frequently. In another approach the multicasting and AOIM (Area of Interest Manager) [3] are used by which a state packet is sent only to that address which is shared by the users of the same multicast group.

The real-time system design for the DVE is a complex problem requiring balancing of the computational requirements against the performance requirements and the actual available hardware resources. For a multi-threaded architecture VE on a multiprocessor machine that has a processor for each thread, each thread can run in parallel and there is generally no major complication on the computational resource management issue. The problem arises when the number of threads are more than the number of processors. If the threads are comprised of system calls where a thread will be blocked until the hardware in other users return data, the blocked thread yields the processor to the other waiting threads. In this kind of operating system, the input/output and network read subsystem operate well as long as the I/O bus has enough speed and has no net-read operation until there is a packet on the network interface. But in the display subsystem the app, cull and draw threads require extensive computational power, and hence may block the system to perform effectively. For high end graphics workstations this resource management is provided as part of the scene graph processing software and comes with the graphics hardware. However, an efficient and a general computational resource management for complex culling is not developed yet and does provide a great potential research problem. The modeling subsystem thread has significant problems in resource management as parallel computation is implemented for complex VE. Much of the current research involves on the management of the resources in this kind of parallel computation. In its simplest form network write subsystem does not require much computational resource manage-

ment. But for a scalable DVE the computation of AOIM requires significant amount of resource management. One approach is to implement a *thread scheduler* which will schedule the threads according to the priority assigned to the threads. This priority assignment of the threads is VE application dependent.

The performance and benefits of the DVE in multiple threads can only be achieved if the system of threads is properly balanced by means of partitioning of the threads by taking full advantage of the processor cycles and memory. However, the actual partitioning and construction of the DVE is application dependent and can not be generalized in a single and or a comprehensive guide.

2.4 Force-Feedback Issues

The information requirements for many tasks from VE needs dextrous manipulation and the sense of touch. In a VE for collaborative design a participant actively uses tactile feedback to manipulate or move an object to perform a task. This type of novel feedback information is useful in identification of object's positions and orientations. The development of multi-user virtual environment has become a major area of research in the fields of computer engineering, tele-robotics and control. However, there has been a few efforts to date for developing force feedback strategies in a distributed or shared virtual environments. In many distributed robotic systems applications such as master/slave systems and haptic displays connected to a distributed VR simulator, the network-based force feedback issues have become an important problem to deal with.

2.4.1 Hardware Design Issues

There is a fine distinction between touch and force feedback. Touch sensors provide information on contact-surface geometry, the smoothness of the contact surface and even the grasped object's slippage due to gravity. Whereas the force feedback gives information on the total contact force, contact surface compliance and grasped object weight. The forces applied by the fingers and palm depend mainly on the way objects are grasped, and may be classified as (i) *power grasp*- where whole hand and palm are used, and has high stability and force but lacks dexterity and (ii) *precision grasp*- where only fingertips are used and it

exerts less force with higher dexterity [47]. For touch feedback total force feedback is of less interest. The fingertip spatial resolution is about 2.5 mm. and the temporal resolution. which refers to the perception of a sequence of contacts in time, is around 300-400 Hz [48]. Virtual touch and force feedback need to replicate real-time computed contact forces, surface geometry, smoothness, slippage etc. In addition to the real-time requirement, virtual force feedback applies real forces on the user's hand. These forces need to be large enough to stop the user's hand and at the same time need not to be too large to harm the user. In the early stages of the development of force feedback actuators five main approaches were identified [49], namely visual, pneumatic, vibro-tactile, electro-tactile and neuro-muscular simulations. Electro-tactile feedback provides electric pulses to the skin with varying width and frequency, and neuro-muscular simulation provides the signal directly to the user's primary cortex. Both are risky to the user's safety concern, and are not considered any more for designing haptic devices [50]. In pneumatic touch feedback devices micro air pockets are placed in a glove along with force sensitive resistors (FSR) on twenty different places which are most sensitive on the hand [51]. In vibro-tactile actuators, voice coils were driven at 250 Hz with variable amplitude modulation using a real- time control circuit [52].

Different force feedback devices have been developed in this industry where they differ in the type of actuators used, resolution, range, bandwidth and in functionalities like force sensing and feedback, portability, range of motion etc. [49]. The simulation of the weight of an object, its inertia and contact with stiff walls require force feedback at the user's palm and wrist. In the following some of the most commonly reported devices are described briefly. *Master Arm* is a four degrees of freedom force feedback device with direct-drive electric actuators [53]. The problems with this device were lack of portability, difficulty in use, complexity and high cost. For easy and widespread use as a desktop feedback system *joystick* was designed with three degrees of freedom [54]. It was integrated with large actuators and high bandwidth and allowed simulation of object's inertia as well as contact surface texture. The problem with joystick was its limited work envelope and restricted degrees of freedom. Enhanced joystick was developed with six degrees of freedom and larger envelope [55]. In more dextrous tasks it is necessary to control simulated

forces on independent fingers rather than at the wrist. *Rutgers Portable Master* was developed with a feedback structure that uses four pneumatic micro-cylinders placed in the palm of a glove on a small 'L' shaped platform [56]. In its stand alone configuration it does not allow the simulation of virtual object weight as no wrist feedback is provided with limited degree of freedom. The *LRP Hand Master* provides more degrees of freedom and force feedback to 14 hand locations as discussed in [57] and [58]. The *SAFIRE Master* has eight degrees of freedom, three for the thumb, three for the index and two for the middle finger. The actuators are DC motors which have their own torque sensors and control. The problem with this device is its high weight. The Force Arm Master uses DC actuators to produce force feedback to shoulder, elbow and forearm with five degrees of freedom [50]. It also suffers from the heavy weight problem, and makes it difficult to integrate in VR training simulation systems.

The sensation of touch and force feedback are not mutually exclusive but complementary and are termed as *haptic feedback*. While manipulating real objects their weight and hardness as well as their surface smoothness and geometry are sensed. The sensing glove *CyberGlove* incorporate these two kinds of feedback in a single device [59]. Desktop haptic devices are already finding widespread applications as force "display" for surgical simulators, nanoscale manipulation and for interactive computer graphics. At the core of the development of haptic devices is creating a "realistic" force-position interface with the VE. The quality of this interface can be measured in terms of *impedance accuracy* which implies how close the impedance matches that of the VE, and *impedance resolution* or *fidelity* which represents the level of impedance discrimination that can be detected at the haptic interface [60]. The dynamics of the haptic devices are the principal barriers to achieving high impedance accuracy and precision. Improvements in the design of efficient drive train (cables, harmonic drives), higher strength-to-weight ratio materials over the last few years have facilitated the reduction of the inherent undesirable dynamics of the haptic devices. However, further reduction and simplification of the dynamics are becoming increasingly difficult due to larger actuators, drive mechanism and linkages, as these contribute to more inertia and friction. Another problem is that even if the force output of the haptic devices is adequate, the natural dynamics of the devices might

prohibit the better force discrimination required for simulating dextrous tasks as in the case of abrupt change of impedance in the VE [61, 62, 63]. Further reduction of the dynamics is feasible through active control by feedforward or force feedback from the force-torque sensor mounted at the haptic interface. Though gravity and friction compensations are often used to reduce dynamics, modification of the inertial dynamics requires force or acceleration feedback. However, as the physical properties of the haptic device change with time, feedback control design allows a more robust design strategy and solution [60].

Two classes of haptic displays are currently available, (i) *Impedance Displays* which measure motion and display force with low inertia and highly back-drivable mechanism and (ii) *Admittance Displays* which measure force and display motion with high inertia and non back-drivable manipulators fitted with force sensors and driven by a position or velocity control loop [64]. Examples of impedance displays are Phantom [65], Pantograph [66], Pen-Based Force Display [67] etc. Among admittance displays WYSIWYF display [68], Iowa State/Boeing virtual aircraft control column [69] are a few to mention. Generally, three criteria are followed in designing a haptic interface, (i) free space must feel free, (ii) solid virtual objects must feel stiff and (iii) virtual constraints must not be easily saturated [65]. The first criterion requires the natural dynamics of the haptic device not to distract the user from the environment being simulated (for the PHANToM^{TM} haptic display it is less than 0.2N). The apparent mass and friction of the apparatus should be as low as possible. This fidelity can be satisfied either through passive design or by active control. The second criterion requires the haptic device to produce enough stiffness to make the user believe that contact with a static object has taken place. For improving fidelity this stiffness is usually at least greater than 20 N/cm. Using very high bandwidth controllers can make the mechanism quite stiff. The third criterion implies that the device must be capable of producing enough force to make user feel the virtual object as solid. The control law must be computed at a fast rate usually at 1000 Hz. The saturation requirement is a function of the peak torque outputs of the motors. A block diagram of a typical PHANToM^{TM} haptic display is shown in Figure 2.10.

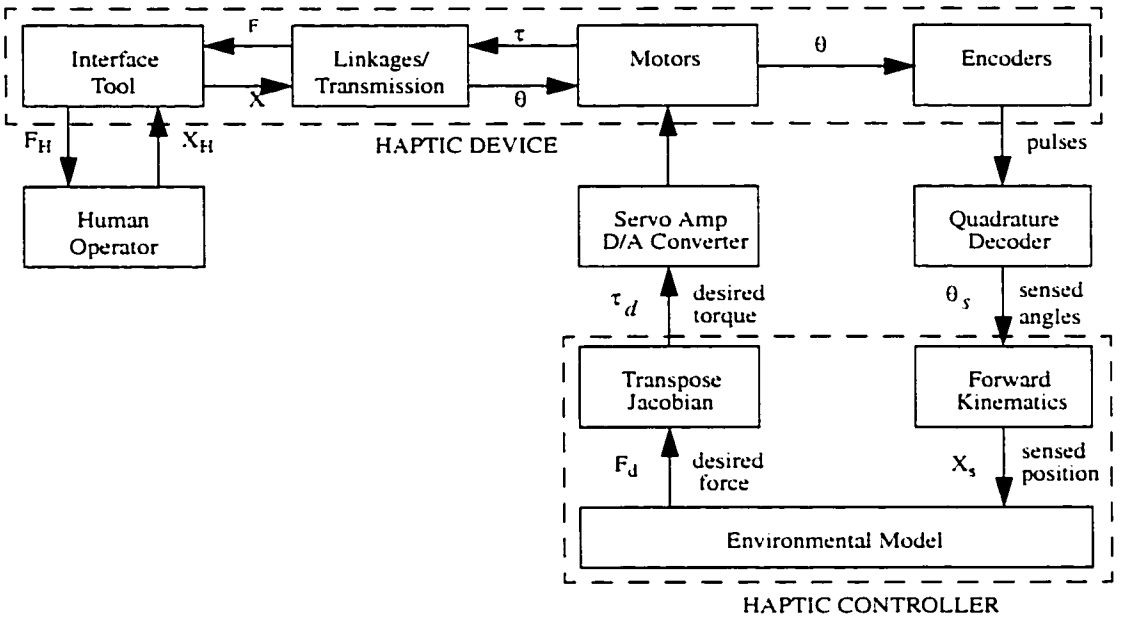

Figure 2.10. PHANToM™ haptic interface controller.

The encoders mounted on the motor shaft read the angles of the motors which are then converted into joint angles. The angles are then sent to a forward kinematics module inside the controller to determine the position of the tool tip, x_s. An environmental model uses the position and desired physical properties to generate a desired force response, F_d. This force is then mapped using a Jacobian to a set of desired torques, τ_d to be produced by the motors in the haptic device. This kind of control architecture is termed as "impedance control with force feedback" and is quite popular in fabrication of haptic devices. Another control architecture, "admittance control with position feedback" is also developed in robotic applications [70]. However, any one control approach can not be generalized and termed as superior. The control design has significant effect on the quality of the haptic feedback. The first approach is promising, if the stability problems associated with explicit force feedback can be resolved. Researches are aiming to meet the demand of higher force output and fidelity requirements, and it seems that force feedback controllers will play an important role in the successful implementation of 3D touch from a virtual environment.

The stability criteria for this kind of control architecture are discussed in chapter 3.

2.4.2 Modeling Issues

In almost all of our hand's activities either for extracting information from or for altering the environment, we use both the sensory and motor parts of our haptic system. Correspondingly, a haptic interface needs to sense our motor actions and display appropriate haptic displays. Whenever we touch an object, it imposes forces on our skin. These net forces, plus the posture and motion of hands and arms, are transmitted to the brain as kinesthetic information which gives the sense of large shapes and spring-like compliances. In contrast receptors embedded in the skin convey tactile information such as spatial and temporal variations of force distribution on the skin within the contact region with an object, which gives the sense of slipping of surfaces, fine textures, small shapes and softness.

To evoke sensations the objects in VE must be modelled according to their geometry, material properties, kinematic and dynamic properties. The interacting forces must be modelled by appropriate computational methods known as "haptic rendering" [71]. The algorithms rely on the model used, and must be considered carefully to meet the real-time needs. Usually a *force server* is dedicated in the force servo loop for real-time force feedback computations. This server tracks the probe of the haptic device and executes the force-feedback servo loop. In doing so the low-level force servo loops get decoupled from higher-level control. The force output is realized by the *intermediate representation* for a force model instead of sending a single force vector to the force-feedback controller [72]. The use of this kind of intermediate representation is application dependent. An immersive system could send a representation of nearby surfaces, a molecular modeling system might use spheres of contact whereas a simulation of physical force fields might send the equation of force field. Some of the most commonly used intermediate representations are discussed below.

(i) *Plane and Probe*: The force server keeps the models of plane which the probe can contact. After the penetration of the probe through the plane, a restorative spring force that

depends on the depth of the penetration is applied. The application approximates a local plane to the surface at the user's hand location each time through the main loop. The forces are updated at around 1 kHz by the force server whereas the plane's position is updated at around 20 Hz [73]. The problem with this approach is that the forces produced are always perpendicular to the surface and the surface feels "oily" with the probe tending to slip off convex areas and into concave areas.

(ii) *Texture and Surface friction*: To have realistic friction of a virtual object both static and kinematic friction have to be modelled. One approach is to represent the variations in the surface height by using a 2D lateral force field proportional to the gradient of the surface height function [74]. To simulate the texture, a surface populated by snags was proposed in [75]. When the probe tip is not stuck in a snag, it moves across the surface opposed by the friction. When it encounters a snag, a force tangent to the surface pulls the tip towards the center of the snag and it sticks there until it moves out of the snag in any tangent direction. Basically the snag distribution over the surface controls the transition from kinematic to static friction.

(iii) *Multiple Probes*: To feel multiple contacts between the VE and the hand, the modeling of multiple probes requires that each probe has its own local surface with which they can collide. This is an active area of research for simulating the sense of touch for both fingers as well as for the palm with proper resolution and level of immersion. One approach presented in [75] gives the sense similar to sticking a single finger in a very stiff glove.

(iv) *Multiple Planes*: To model an object with sharp inner edge requires simulation of multiple planes to constrain the probe tip in several directions at once. More research is required to sense interaction with multiple planes.

(v) *Point-to-Point Springs*: The objects in a multi body simulation are subject to multiple forces and the calculation of forces is very complex and time consuming. To overcome this problem a simulated spring is used to connect the probe endpoint from force server to the appropriate body in the simulation of the application loop [76]. The application con-

trols the motion of one end point of the spring at its slower rate, while the other endpoint follows the probe motion at the force update rate. Typically the spring applies force both to the user's hand at the point of contact and to the environment. To adjust the rapid movement of the probe viscosity can be adjusted [77].

(vi) *Multiple Springs*: To simulate torque multiple springs are attached to multiple application points and to multiple probes [75]. Further research is required to sense immersive torque realizations.

Modeling of haptic objects in VE is still at its early stages of development. Various artifacts have to be considered and simulated in real-time. Advanced algorithms have to be developed to compute force feedback for any kind of interaction in VE. The problem becomes more complex when interacting in a DVE. The unpredictable delays over network has a significant effect over the performance of the haptic display. A stable haptic system in a stand alone VE can be unstable in a DVE. More robust control algorithms and techniques are necessary to simulate the realistic force sensation by all the end-users.

2.5 Collision Detection

The purpose of collision detection or contact determination is to automatically report of a geometric contact when it is at the verge of occurring or when it has already occurred. The object models may be polygonal, splines, or other geometric surfaces. Collision detection concerns and issues are encountered in force feedback realization from VE, computer aided design and machining (CAD/CAM), robotics and automation, manufacturing, computer graphics, animation and simulated environments. In many of the tasks involved in the above areas, contact analysis and spatial reasoning among static and dynamic objects such as tolerance verification, engineering analysis, concurrent assembly, motion planning, animated figure articulation, walkthrough, etc., collision detection is considered as a major computational bottleneck. A wide range of methodologies and techniques such as hierarchical representation, geometric reasoning, algebraic formulations, spatial partitioning, analytical methods and optimization methods, have been proposed in the

literature [78]. Application of a specific algorithm depends on the geometric model representation, the desired query types and on the nature of the VE.

2.5.1 Representation of the Model

Among different types of model representations used in the simulation of realistic objects, the most popular one in the literature is shown in Figure 2.11 [78]. The simplest and most commonly used models are *polygonal objects*. Among these the most general one is the *polygon soup* which is a collection of polygons that are not geometrically connected and has no topology information available. If the polygons form a closed surface or volume then the model is defined as *structured* solid with *convex* (closed) and *nonconvex* models. The *constructive solid geometry (CSG)* forms objects from primitives such as blocks, spheres, cones, cylinders, tori, etc. The combination of the primitives is done with a set of theoretic operations such as union, intersection, and set difference [79, 80]. This technique enables one in the design process of building shapes by means of cutting with intersection, set difference techniques and joining simple shapes with union technique to form more complex objects. This kind of models make the collision detection algorithm easier to develop and implement, and the detection algorithm can be optimized for certain kind of predefined primitives [81]. The problems with this approach are that an accurate boundary and surface representation and rounding of edges are difficult to formally describe as well as computationally expensive [82]. Alternatively, *implicit surfaces* are defined with mappings from space to real numbers. They are the loci of points where $f(x, y, z) = 0$. The inside of the model is represented as $f(x, y, z) < 0$, and the outside is defined as $f(x, y, z) > 0$. The implicit surface described with the polynomial in x, y and z is called algebraic, which includes algebraic surfaces, higher order functions and convolution surfaces [83, 84]. This kind of objects are also used as primitives in CSG [85, 86]. Parametric surfaces are mappings from some subset of the plane to the whole space. They represent the surface boundary of the objects. Non-Uniform Rational B-Spline (NURBS) is one example of parametric objects, which are easy to render and polygonalize [87, 88].

2.5.2 Types of Queries

Different information is required to calculate the occurrence of a collision and the result-

ing response. Intersection of objects is required to find if the objects have touched each other or not, and is important for physically based modeling and animation [89, 90]. Calculation of separation or distance of objects is needed to find the minimum distance between objects, and if the objects are penetrated or not, and if penetrated what is the information required to determine the minimum translational distance to separate them. The distance calculation is also useful in calculating reaction forces [91]. The information on position and motion of objects are necessary for prediction and real-time calculation of any future collision either in the VE or the DVE.

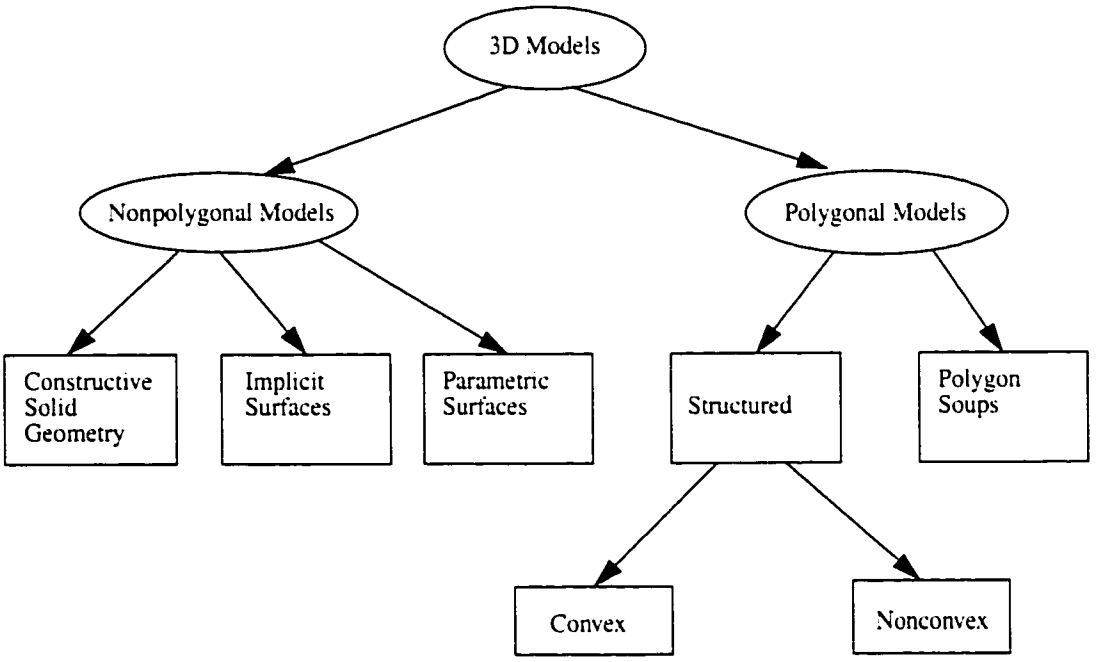

Figure 2.11. Classification of 3D models.

2.5.3 Types of Simulation

Different simulation environments affect the design and application of corresponding collision detection algorithms. The most common types of simulation environments are (i) *pair processing vs. nbody processing*- where in pair processing only two objects are considered for detecting the collision where as in nbody processing there are several objects

[78]. (ii) *static vs. dynamic*- where slight differences between successive dynamic positions of the objects are exploited. For instance some algorithms are based on the position and motion information of the objects at successive time steps [89], and (iii) *rigid vs. deformable*- where the rate of deformation over time is exploited in developing the algorithm. In a highly complex and realistic VE the differences in dynamics and object properties affect the application and development of the collision detection algorithms for different region and different kinds of interactions.

2.5.4 Algorithms for Polygonal Models

A number of algorithms have been proposed in the collision detection literature achieving relatively good results. In applications involving rigid motion, spatial and temporal coherence, as pointed out in static vs. dynamic simulation, are exploited to design algorithms for convex polyhedra. When two objects come into contact local properties such as distance between them, motion in discrete time steps and object properties are considered [92]. Different hierarchical approaches have been widely used in the development of collision detection algorithms. The most popular one is the bounding volume which includes axis-aligned boxes and spheres. These bounding volumes are chosen for the overlap tests in the collision detection [89]. Among different hierarchical structures cone trees, k-d trees and octrees [93], sphere trees [94], trees based on S-bounds [80] are widely used in the development of collision algorithm. Binary space partitioning (BSP) [95] and spatial partitioning based on space-time bounds or four-dimensional testing [96] have also been used. All of these methods perform "rejection test" or "sweep and prune" whenever two objects under consideration are far apart. When objects are in close proximity and can have multiple contacts, subdivision of the bounding volume into more boxes are considered to determine good potential contacts at the cost of computational power. This approach could potentially have adverse effects on the real-time collision realization but yield near accurate results. More recent work have been attempted in developing a fast algorithm based on oriented bounding boxes (OBB), which compute the actual area of collision after determining a collision between two objects [97].

2.5.5 Algorithms for Non-Polygonal Models

A great deal of research has been conducted in geometric and solid modeling to compute the intersection of surfaces. The surfaces may be represented as splines or algebraic surfaces. Most of the early algorithms are based on accurate computation of the intersection for static models. The collision detection in a dynamic environment is a major research issue in this area. Among different types of non-polygonal models, the intersection test for CSG models is rather simple as the CSG models exploit the set operations as explained earlier. The set intersection testing is sufficiently appropriate for these kinds of models. If the intersection test results in a null set then the objects are not collided, otherwise they collide. This technique is often referred to as null object detector (NOD) [80]. The boundary representation of the solids are used in the development of the algorithm. One useful and well known approach is known as "S-Bounds" which defines points in space that can be applied to the expression of the fundamental objects in CSG tree to yield the "in", "out" and "on" classification of those points. To speed up the algorithm a good number of sample points are classified which can be evoked later to determine the intersection [81]. Another approximate and easy method to implement employs interval arithmetic to evaluate implicit functions over box-like regions of space to determine the "in", "out" or "on" classifications. The point classification technique explained above has been extended to regions obtained from adaptive subdivision of space. The precision of the result is dependant on the fineness of the subdivision of the space. The setback of this approach is that it may not be able to detect the contact status of disjoint models which are almost touching and it does not perform well in real-time collision detection [98].

For parametric surfaces four basic techniques known as, *subdivision method*, *lattice method*, *tracing method* and *analytic method*, are employed to find the object intersection. The subdivision methods work by simply dividing the domain of the two surface patches in tandem and examining the spatial relationship between patch subsections. The subsections are further divided and examined or terminated depending on various criteria. The accuracy of this method depends on the fineness of the subdivisions. Interval arithmetic is used to adaptively subdivide the domain of the surfaces to refine an approximation of the intersection curve until a finite precision is achieved. Instead of confirming a collision

directly, this method report upper and lower bounds on the patches' closest approach in the space. A zero value of the lower bound indicates the occurrence of collision with the patch of that parametric surface [99]. To subdivide the domain with time varying parametric surfaces, the domain is adaptively subdivided with a priority queue which is used to direct the subdivision so as to locate the earliest pair of subregions which overlap in space [100]. The problem with this approach is that it is computationally expensive and the motion of the objects need to be expressed as closed-form functions of time. For deformable surfaces the improved algorithm is based on the tangency condition which describes the point of contact at the moment of contact. Two curved surfaces must have a common point of contact with opposite normals and the converging points must be moving toward each other. Based on this condition any pair of domain patches can be eliminated from further consideration when they don't contain opposing normals [101]. The lattice method exploits the fact that the intersection curve of two surfaces in space has a pre-image on the domains of both patches. It then attempts to locate specific points of these pre-images by selecting many isoparametric curves which criss-cross the surface like lattice. The intersection pre-image meeting the isoparametric curve can be found by analyzing the degree of the polynomials and the derivatives. The problem is that this technique fails when the intersection curve form a very small closed loop [102]. The tracing method trace the intersection curve in sufficiently small steps until the edge of the patch is found or until the curve returns to itself to close a loop starting from a given or known point from the boundary. At the intersection point on the surfaces, the intersection curve must be mutually orthogonal to the normals of the surfaces and the tracing path is given by the cross product of the normals. Finally, analytic method represents the model itself as an implicit function. The locus of roots of this scalar function map out curves in the plane which are the preimages of the intersection curve [103].

The shape and the property of the "inside-outside" functions for implicit surfaces are computed using the point samples described earlier. An efficient algorithm is presented in [104] for curved models composed of either spline surfaces or algebraic surfaces in constrained rigid motion. The limitation of this algorithm is that it does not perform well for objects with higher degree primitives.

2.5.6 Multi-Body Collision

The collision detection in an environment populated with entities or objects becomes time consuming due to pairwise interference checks. The efficiency of a given algorithm in this environment depends on the speed to which it can eliminate unnecessary pairwise checks. Some recent algorithms in the multi-body environment exploit the spatial arrangement or the acceleration and velocity information of the objects to reduce the number of pairwise interference checks. The *scheduling* algorithms maintain a queue of all object pairs that might collide. The sorting is based on the lower bounds of the time of collision. The lower bounds of time to collision are calculated adaptively and updated during a possible collision [91]. Another popular algorithm, *I-COLLIDE* is based on the spatial and temporal coherence which implies that objects are expected to have not moved far between computational time frames. The number of pairwise intersection checks are reduced by sorting axis-aligned bounding boxes (AABBs) surrounding the objects of interest. Its run time is linearly dependent on the number of objects in the environment. It takes the advantage of the fact that for a collision to occur between two AABBs, their projection onto the x, y and z axes must overlap [105]. The sweep and prune algorithm maintains a list of intervals for each bounding box for each dimension. The overlap check is then reduced to looking for interval overlap for each of the three axes to determine the 3D bounding box overlap. By sorting the interval lists, the algorithm can determine which time steps overlap and if there is overlap in all three dimensions then the bounding boxes overlap. In *uniform spatial subdivision* algorithm the space is divided into units or cells and each object is placed in some cell. If the cells occupied by each box are shared somehow by another object then a collision is possible. For an environment with equal size of objects, this algorithm is quite efficient. The problem with this algorithm is that it is difficult to set an optimal size of cell and more cells require more memory allocation [106].

2.6. Conclusion

A substantial amount of research is being conducted on developing suitable prototypes, toolkits, protocols, network and communication architectures to support the demands of large scale DVE. Toolkits developed in different universities are widely available and easily understandable. The Department of Defense has been pursuing its own architecture,

High Level Architecture (HLA), for interoperable VE [107]. As fewer and fewer development of DIS-based simulations are being conducted in 1999, HLA represents the migration path for existing DIS-based DVEs. One of the major impediments in adopting multicasting over internet in its full extent is the lack of multicast router. Currently personal computers and workstations are not properly supported for multicasting. Network cards are yet to be developed to support large multicast subscription capability at the user end. Development of DVE is a multidisciplinary problem. The advancements in high speed network, graphics card, communication architecture, protocol, graphics toolkits, and interface devices all impact and affect the development of successful large scale realistic DVE. A lot of research has been conducted in the literature on collision detection. However, the results are mostly confined to rigid objects. Much remains to be investigated for detecting collision between deformable objects and measuring penetration and post collision effects with accuracy and efficiency. The calculation of contact points and penetration distances between objects are extremely difficult to perform for models with smooth surfaces having plenty of concavities. For an "extremely large" DVE the computational burden of collision detection introduces additional complexities due to the large memory requirements of local VEs and shortage of efficient real-time and parallel computing algorithms.

In the next chapter we will discuss more about the stability criteria of stand-alone haptic display and the source of unstable behavior and its impact on the collision detection and real-time performance.

Chapter 3

Stability Issues of Haptic Display

3.1 Introduction

The term 'haptic' is associated with the sense of 3D touch. As haptics form an essential part of most of our interactions with the real world, any VR application that involves simulating the real world benefits from haptic interactions when combined with stereo visual and auditory displays. Distributed haptic feedback technology finds a wide range of applications in training - mostly for military and medical surgery applications, in science and business to enable users to experience, represent and manipulate complex multi-dimensional data sets, in commerce to allow customers to feel and interact with products, in entertainment to permit users to feel and manipulate different environmental behaviors, interact tools and avatars, in education to give students a feel of realistic and non realistic phenomena at a variety of spatial and temporal scale and even in arts to create virtual works of art either by individuals or by many artists and users.

An haptic simulation is composed of several elements required to provide a human operator/user with an artificial sense of kinesthetic presence in a virtual world. Haptic in a shared environment includes, as a minimum, multi users at possibly geographically different locations, haptic display at each user, a computer model of the virtual environment residing at each host and the network to connect multi users. The level and quality of inter-

action depends on the physical constraint of the communication media and on the available computational power. A typical architecture for force feedback in a shared environment is introduced by the active operator model in [108].

Stability is a critical issue in haptic simulation. Unlike other conventional robotic manipulators, haptic devices inherently function in close proximity to humans. An unpredicted instability in a haptic system can severely harm the operator. The unpredictable nature of human operator and the virtual environment are the main sources of instability in a stand alone haptic VR system. Researchers have investigated the use of a two port network model to characterize stability and performance issues in tele-operation. A framework for the design of tele-operators based on the two port hybrid matrix was introduced by Hannaford [109]. To guarantee stability of a bilateral tele-operation system a two port network theory was used by Anderson and Spong [110]. Colgate [111] presented criteria for coupled stability in bilateral systems and introduced an impedance shaping for bilateral control. He also proposed in [112] the use of a virtual coupling network between the haptic display and the virtual environment. A similar "god-object" approach that couples a haptic device to a virtual environment through a virtual spring-damper was suggested by Zilles and Salisbury [113]. The concept of stable haptic simulation into a two port framework and a virtual coupling network to include both impedance and admittance type haptic displays was introduced by Adams and Hannaford [114]. They also identified the duality between the impedance and admittance models of the haptic simulations.

3.2 Background Information and Terminologies

Several concepts that are commonly used in the field of force feedback in VE are briefly described below.

Virtual Environment: Computer generated 3D model, not essentially stereographic, of some physically motivated scene. It can be either impedance or admittance type in terms of interaction with haptic interface. The impedance environment is one which accepts velocities and generates forces according to some physical model. The penalty based approaches belong to this class and up to now are considered to be the most popular in the

literature [65, 67, 75, 115, 116]. On the other hand the admittance type VE accepts forces and returns velocities (or positions). The constraint-based techniques belong to this class [113].

Haptic Display: Belongs to a class of tele-robotic manipulators configured to convey kinesthetic information to a human operator. As in the case of VE it can be either Impedance or Admittance type display. In impedance type displays forces are generated in response to measured displacements. These systems generally have low inertia and are highly back-drivable. The most popular PHANToM haptic display falls into this category. In admittance type displays displacements are generated in response to measured forces.These systems are generally characterized by high inertia and non back-drivable. Carnegie Mellon University's WYSIWYF Display based upon a PUMA 560 industrial robot falls into this category.

Haptic Interface: Is a link between a human operator and the virtual environment and includes both the haptic display and the necessary software for stable interaction operation.

Haptic Simulation: It includes the synthesis of human operator, haptic interface and virtual environment that all together create a kinesthetically immersive environment.

Haptic Architecture: Four different types of architectures are possible with the combination of impedance and/or admittance type virtual environments and impedance and/or admittance type haptic displays. The scalability and performance issues in bilateral teleoperation systems are described extensively using two port models and circuit theory in [109] and [110].

Instability and Passivity: The haptic display system is composed of three main components- human operator, display and the simulation environment. Due to the unpredictable dynamic nature of the operator and the complexity of VE, the operation and the simulation are highly uncertain and difficult to model. To make the analyses tractable a standard

approach has been established that require the display and the simulation subsystem appear to be passive to the user, i.e. the operator should not extract energy from the haptic display on a continuing basis. The implementation of passive VE is essential to guarantee stability of haptic display. The basic difficulty may be described with the example of virtual spring. An ideal physical spring is a lossless system where the energy stored by squeezing will be freed by releasing it. As the virtual spring is implemented in discrete time the force provided by the spring will not increase smoothly with deflection. The force will be repeatedly hold at each sampled time (zero-order hold). The average force during squeezing will be slightly less than for physical spring of identical stiffness and the average force during release will be slightly greater. Figure 3.1 shows that the delicate balance of stored and released energy is lost in this case. Thus, the spring acts to store or generate energy without dissipating it.

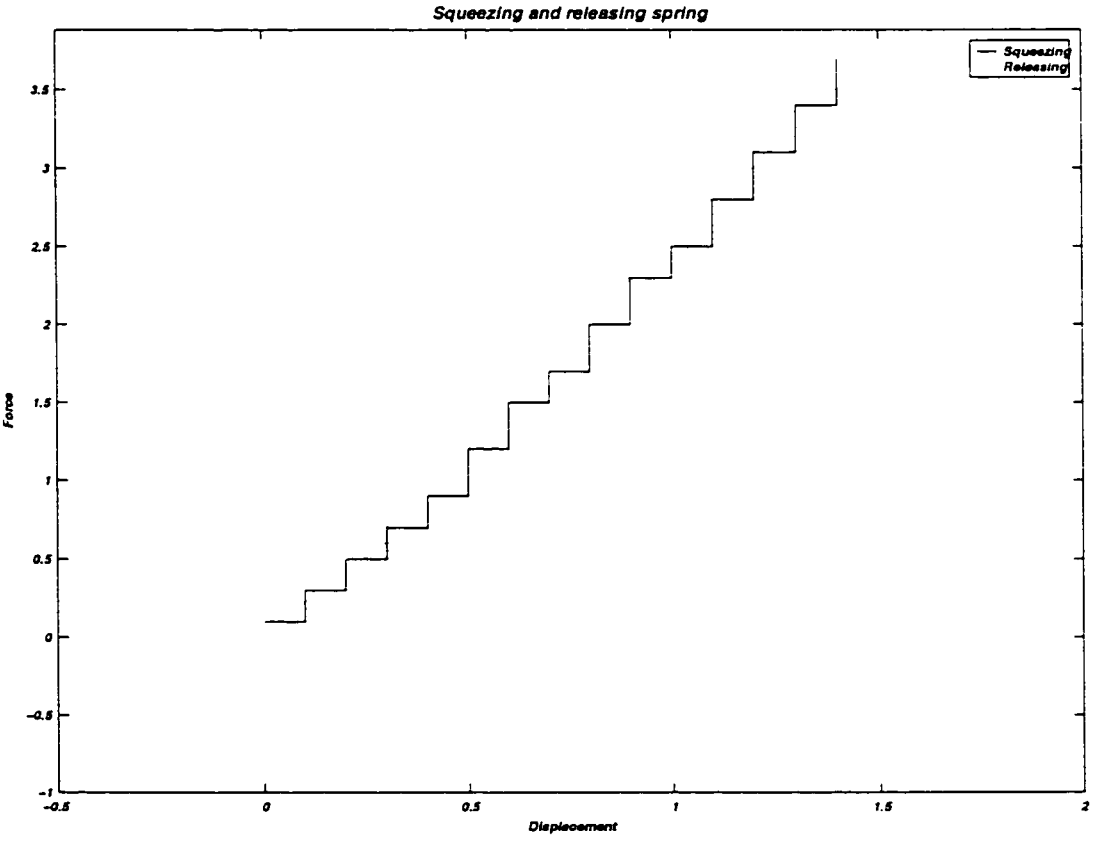

Figure 3.1. Loss of passivity in the virtual spring.

Any realistic implementation of VE includes some dissipation along with the capability of producing energy via a haptic display due to the inherent delay and information loss in the real-time sample and hold process and data transmission. The simplest example of passive physical behavior which can cause energy generation is unilateral constraint which occur whenever two rigid bodies collide. Two physical systems which are stable in isolation and also stable when rigidly coupled together, are stable when allowed to collide. The fact is that collisions do not generate energy in rigid body contact. For discrete time systems, it is possible that even simplest VE with unilateral constraint will be directly responsible for instability.

Stability Criteria: Considering the human user and the virtual environment as passive operators, a haptic interface can be represented as a linear two port network with terminals for the human operator (F_h, V_h) and the virtual environment (F_e', V_e') as shown in Figure 3.2. The necessary and sufficient conditions for unconditional stability are established in [71]. Several necessary criteria for a stable haptic system are as follows:

Criterion-1: A continuos (discrete) linear time-invariant system is stable if and only if its corresponding characteristic equation has no roots in the right half s-plane (outside the unit circle in the z-plane) and only simple roots on the imaginary axis (unit circle).

The haptic interface must be designed for any unpredictable and time varying behavior corresponding to the human operator as well as the virtual environment. The system must be able to handle a variety of uncertainties and nonlinearities. The stability criteria that will be developed is based upon the assumption that the human operator and the virtual environment must be passive operators. The human interaction with a robotic manipulator can be considered passive but passivity for the environment is, in general, not predictable as numerical integration routines could not necessarily and strictly adhere to the physical laws. However, haptic interfaces that are designed under the conservation laws of physics are found to be very robust when coupled to almost a passive VE [72]. The basic criterion is that any Linear Time-Invariant (LTI) passive system will be stable when coupled to a system which is itself passive.

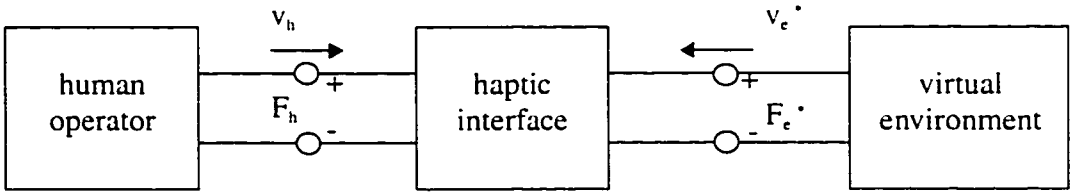

Figure 3.2. A haptic simulation model with two-port network

Criterion-2: A general statement of "passivity" of a haptic interface is that the kinetic energy of the mass must be less than the total energy input by the source $f(t)$

$$\frac{1}{2}mv^2(t) < \int_0^t f(\tau)v(\tau)d\tau, \quad \forall t > 0, \quad \text{for any admissible } f(t) \tag{3.1}$$

Criterion-3: The continuos (discrete) time linear immittance matrix P (a mapping between two vectors $y = Pu$ is an immittance mapping if $y^T u = F_1 v_1 + F_2(-v_2)$) will be "positive real" if and only if P has no poles in the right half of s-plane (outside the unit circle in the z-plane), has only simple poles on the imaginary axis (unit circle), and

$$P(j\omega) + P^T(j\omega) \geq 0, \qquad \forall \omega \geq 0 \tag{3.2}$$

The last condition can be further simplified to

$$Re(p_{11}) \geq 0 \ , \qquad Re(p_{22}) \geq 0 \tag{3.3}$$

$$Re(p_{11})Re(p_{22}) - \left|\frac{p_{21} + p_{12}}{2}\right|^2 \geq 0, \qquad \forall \omega \geq 0 \tag{3.4}$$

Performance Criteria: The Z-width, defined as the achievable range of impedances that the haptic interface can stably present to the operator was proposed by Colgate and Brown [73], as a measure of the performance of the haptic system. For unilateral constraint the range of impedances to be achieved can be expressed from almost no resistance in the case

of a contact parallel to or outside of the wall to almost complete resistance in the case of contact inside or normal to the wall.

3.3 Development of a Stand Alone Stable Haptic System

Preservation of the passivity in the closed loop system has been widely used in the design of haptic interfaces in VE. Haptic interfaces are assumed to operate in an almost ideal environment with collocated sensors and actuators, and where they are interacting with a VE whose physical counterpart is passive. These factors may contribute to unstable oscillations when the haptic interface is grasped by a human operator. Sometimes the higher frequency oscillations require energy to sustain them, which is then supplied by the interface and that tends to destabilize the overall system. As a consequence the system is no longer passive, but active as a direct consequence of the time delay and loss of information inherent in sampling.

To remedy this problem a simple model of the system was considered. The model intends to simulate a unilateral constraint that is present in the VE. The unilateral constraint (piece-wise linear operator) is needed to account for collision, contact and ubiquitous in the physical world e.g. a bouncing ball on the floor. Therefore, the resulting overall system is nonlinear. However, up to a certain range of operations the necessary and sufficient conditions established for the linear system may still be quite adequate and the performance may be actually acceptable. Figure 3.2 is a model of one degree-of-freedom haptic interface where m is the inertia and b is the damping of haptic device, y is the position, v is the velocity of display hardware, f is the total force generated by the operator and T is the sampling period.

The model considered here is a sampled data-system (continuos-time plant and discrete-time controller). Amplifier and sensor dynamics, nonlinearity and noise are ignored. The virtual environment in the feedback controller is represented by a stable linear, shift invariant transfer function, $H(z)$. Practical virtual environments cannot be composed strictly of linear operators. For a model close to reality it is necessary to include the nonlinear element, unilateral constraint, to account for collisions and contact as shown

in the feedback path of the Figure 3.3. A simple example of unilateral constraint is a ball bouncing on the wall or floor where completely different equations of motion apply when the ball is and is not in contact with the floor.

A general statement of passivity has been obtained in equation (3.1). A rigorous analysis of passivity for this kind of a system (Figure 3.2) has been performed in [117]. The necessary and sufficient condition for stability are obtained as follows:

$$b > \frac{T}{2(1-\cos\omega T)} Re\left\{(1-e^{-j\omega T})H(e^{j\omega T})\right\}, \qquad 0 \le \omega \le \omega_N \tag{3.5}$$

where $\omega_N = \frac{\pi}{T}$ is the nyquist frequency.

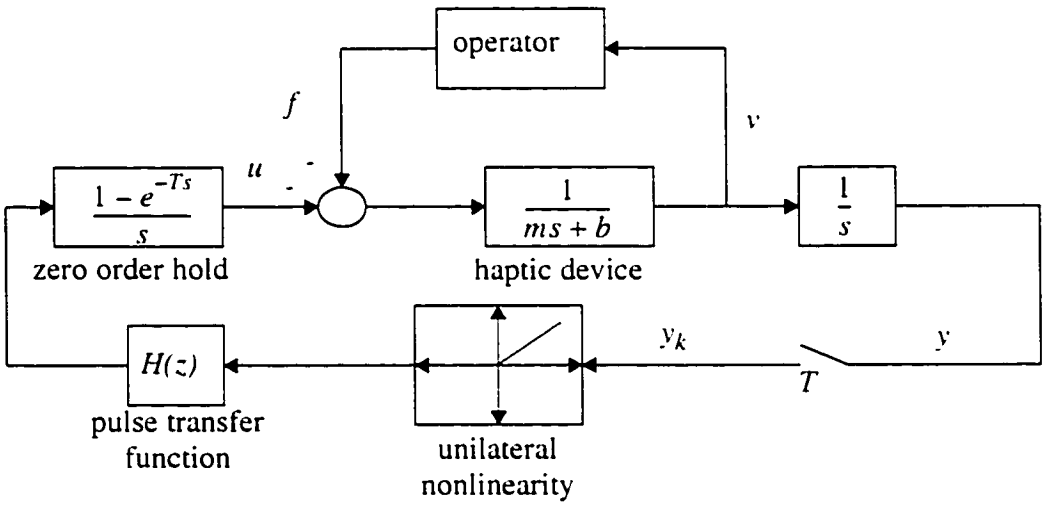

Figure 3.3. Simplified model of the haptic display system

In this system we have considered the common implementation of the "Virtual Wall" consisting of a virtual spring (K_e) and a virtual damper (b_e) in a mechanical parallel [112] configuration as described below:

$$VE(z) = b_e + \frac{Ke}{s}\bigg|_{s \to \frac{z-1}{Tz}} \tag{3.6}$$

where $VE(j\omega) = \frac{1}{j\omega}H(j\omega)$. Using (3.5), we get the conditions of passivity are as follows:

$$b > \frac{T}{2(1-\cos\omega T)} Re\left\{(1-e^{-j\omega T})\left(Ke + \frac{be(e^{j\omega T}-1)}{Te^{j\omega T}}\right)\right\},\ 0 \le \omega \le \omega_N$$

By using simple algebraic manipulations the above expression leads to

$$b > \frac{T}{2} - b_e \cos\omega T,\quad 0 \le \omega \le \omega_N \tag{3.7}$$

and for $\omega = \omega_N$ we obtain

$$b > \frac{KT}{2} + b_e \tag{3.8}$$

and for the case of negative virtual damping that is applicable to simulating insertion through the wall, the condition is modified to,

$$b > \frac{KT}{2} + |b_e| \tag{3.9}$$

The above design condition does provide us with some guidelines for implementing a stand alone haptic interface system. In addition to the above design guidelines when the "unilateral nonlinearity" that is associated with any unilateral constraint is taken into consideration (refer to Figure 3.3), one needs to still determine the required stability conditions.

It turns out that it is quite difficult to find necessary stability conditions for nonlinear systems. Lyapunov functions are developed for nonlinear time-invariant systems in feedback with static or time-varying nonlinearities. Circle criterion gives a sufficient condition for the stability of a system with a single nonlinear gain in the feedback path [118]. The Popov criterion is less conservative than circle criterion and it defines the absolute stability if the Popov-plot touches or lies to the right side of an arbitrarily chosen straight line. The Tsypkin criterion performs better than circle and Popov criteria and is specially designed for sampled data system [119]. These criteria are too conservative to be useful in the analyses of stability of haptic displays due to their limited nature of their analysis of nonlinear characteristics. The unilateral nonlinearity involved in haptic display is well defined and it must be exploited using different approaches. Mitra [120] derived sufficient stability conditions for this kind of systems considering saturation nonlinearity. Finally, the most recent and the least conservative stability conditions for feedback systems incorporating a linear, shift invariant operator and a unilateral nonlinearity was reported in [121]. In addition, they also consider the non-zero reference input situation. The limitation of this approach is that it is strictly a criterion for determining the absence of an oscillation rather than verifying the stability condition.

3.4 Simulation of Stand Alone Haptic Display System

For a system level simulation, we have basically ignored the computational cost of our simulation procedures, and for the sake of simplicity, have ignored any nonlinearity involved (refer to Figure 3.4) by selecting sufficiently large inherent damping (b) for the haptic device (to ensure that a large amount of energy is lost due to friction). Thus, while moving, the haptic device will consume a large amount of energy during every sample period. The haptic device is passive without a feedback loop as the nonlinearity is present only in the device. This leads to the situation where the reference force input to the comparator (after zero-order hold) in Figure 3.4 is zero during any given sample period. Thus the kinetic energy of the mass of the haptic device will at no point be greater than the total energy input by $f(t)$. This condition guarantees the passivity of the haptic device with a linear controller even in the presence of unilateral constraint in the feedback loop.

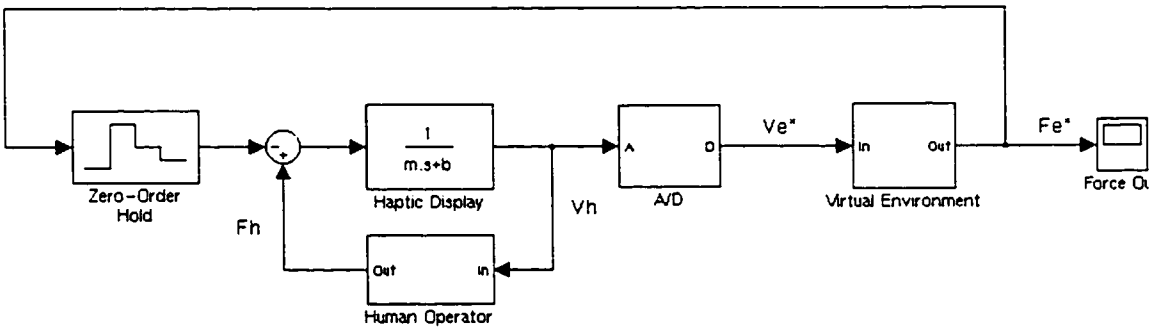

Figure 3.4. Block diagram of the stand alone haptic system

To design the stand alone haptic system, the design guideline developed in (3.8) is used. For the first simulation of this scenario the parameters chosen were m=1, b= 0.5, b_e= 0.1 and K_e= 0.2. The sampling rate was selected at a high rate of T= 0.01 sec. The main reason behind this choice is to keep the damping coefficient of the haptic device within an acceptable range suitable for a physical development, and to further dissipate some energy in order to ensure passivity. Another obvious reason is that for a fixed device damping coefficient, use of high sampling rates allows one to simulate very stiff and dissipative virtual constraints. With a step input the simulation results, the force output F_e, from first order and second order VE is shown in Figure 3.5. The force remains sufficiently stable without any oscillations and it matches the stability criteria given in (3.9) for this type of a system. The VE was modelled as simple as possible with one zero and one pole and the passivity was preserved for the first order system. The force output is quite acceptable during the transient region (1 to 4 sec.) which is also depicting some steady state error. The force output from a second order system depicts more transient oscillation than the first order system which finally settles down and merges with the first order output. The velocity output V_e from the haptic device for a unit step input are shown in Figure 3.6. The velocity output from the first order system has reached a steady state of 0.2, representing a motion of the haptic display terminus which is realized by the human operator. The initial transient oscillations (1 to 4 sec.) are inherent to the system and must be kept low for safe interaction with the device. The velocity output from the second order system is rather

oscillatory and the transient interval is more than the first order system. The force and velocity outputs from the third and the fourth order VEs (after adding poles at the imaginary axis) are shown in Figure 3.7 and 3.8, respectively. Both of these show oscillatory behavior and for higher order they tend to be positive exponential envelope with oscillation.

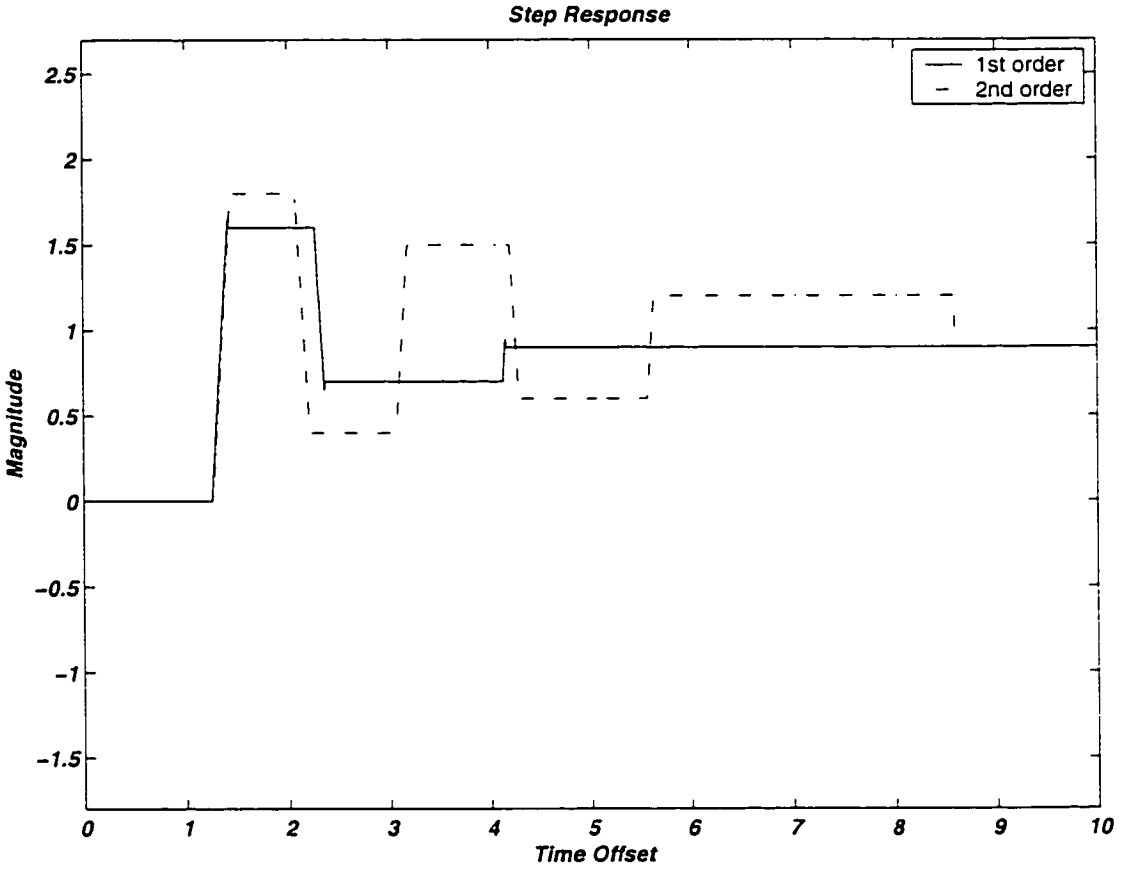

Figure 3.5. Force output from VE with first and second order VEs

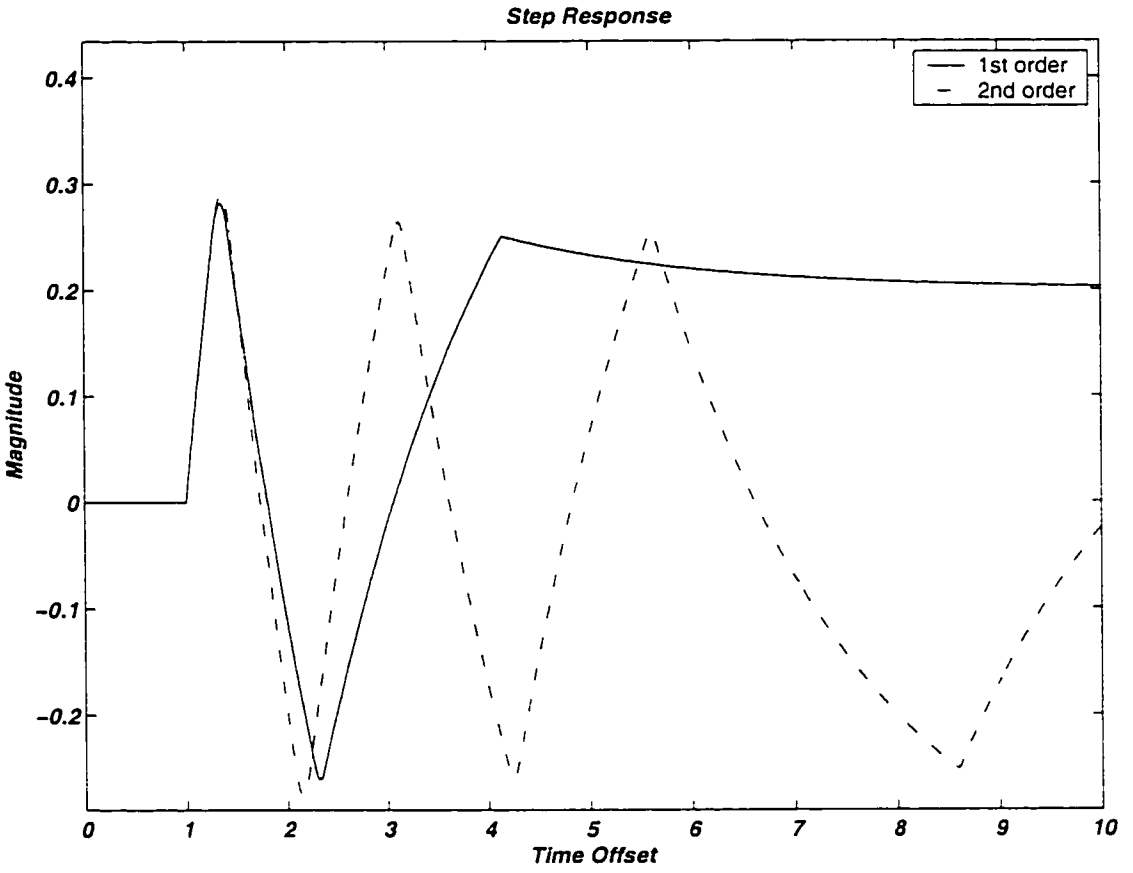

Figure 3.6. Velocity output from VE with first and second order VEs

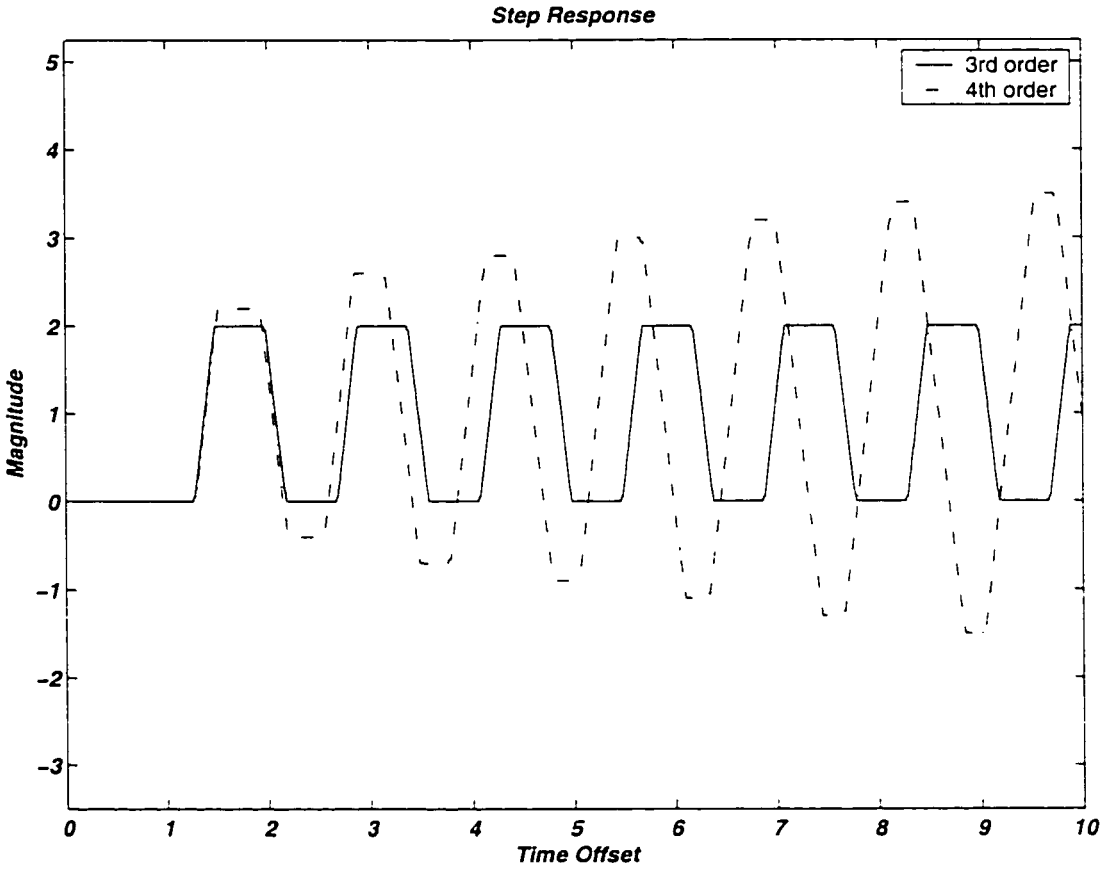

Figure 3.7. Force output from VE with third and fourth order VEs

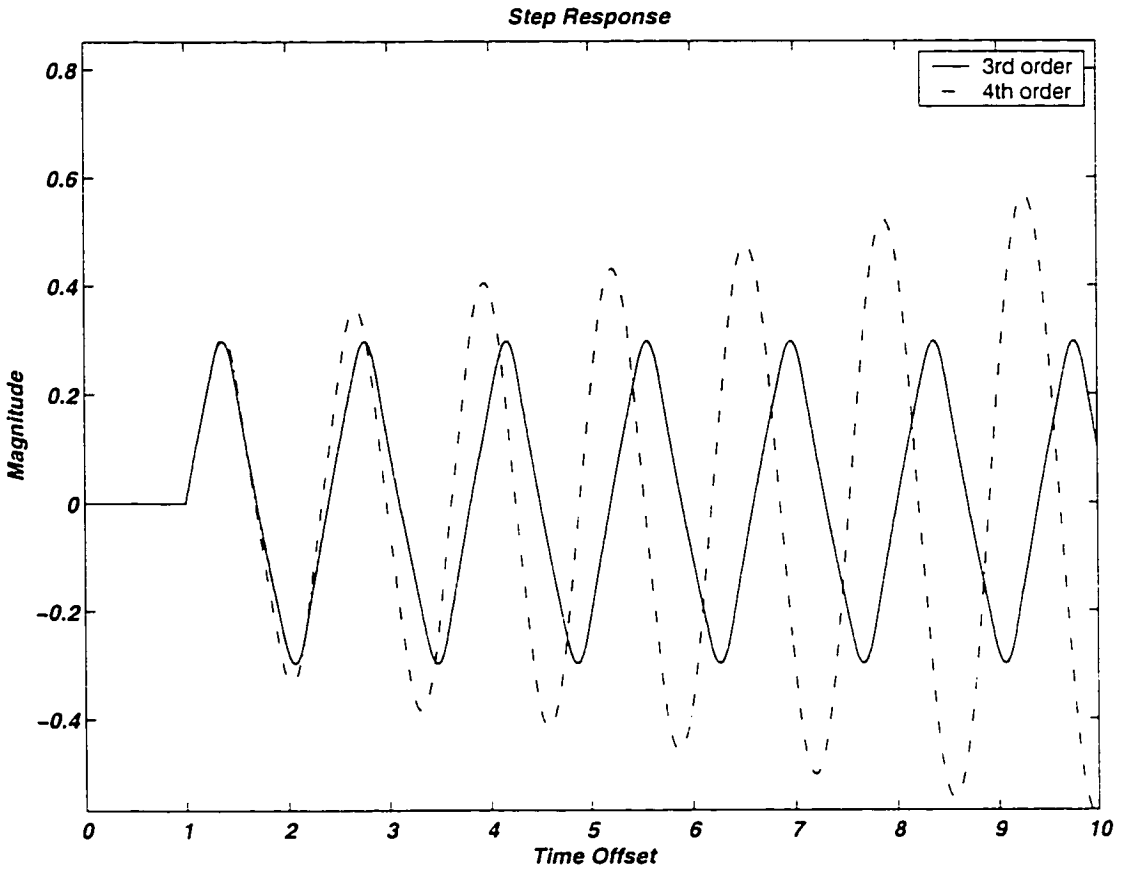

Figure 3.8. Velocity output from VE with third and fourth order VEs

3.5 Conclusion

A stable, high performance haptic display of virtual environments is a combination of electromechanical design, parameter tuning, software design and psychophysics. So far, not much successful work has been done to extend the stability criteria (3.9) of the haptic device to a fairly complex environment populated by different types of object properties such as soft, spongy, deformable etc. Recently, Colgate et al. [122], have shown that passivity of a virtual wall could be extended to the passivity of a broad class of linear virtual environments by considering the VE to be discrete-time passive and connecting the

environment to the haptic display via a "virtual coupling" consisting of a virtual spring and a damper. The advantage of this approach is that the stability is assured by ensuring the simulation as discrete-time passive rather than tuning many parameters in a complex simulation.

In the next section a rigorous analysis will be performed for stability properties in a distributed environment in presence of network delay. Dead reckoning algorithm will be introduced and implemented towards the guaranteed stability of the haptic display.

Chapter 4

Stability of Haptic Display in DVE

4.1 Introduction

In the case of a network-based haptic display the criteria for stability of the overall system is considerably more complicated and involved when compared to those that have been studied so far in the literature for single user haptic displays. An application with distributed implementation requires that force and kinematic data be transferred over the communication links. The time required for transmission and processing of data introduces delays which add phase lag to the signal and this lag limits the effective bandwidth of the system which in turn may limit and hamper the stability of the tele-robotic or haptic display system. One of the current approaches available to study stability of systems in the presence of expected time delays is to simply reduce the overall loop gain but sacrificing the performance requirements of the resulting system. To guarantee stability in the presence of random and unpredictable time delays new control algorithms and software schemes are required. The dead reckoning algorithm [1] can play an important role in this case by addressing stability issues for arbitrary time delays over a network configuration. Haptic dead reckoning allows a comparatively 'simpler', local environment model to serve the haptic rendering while being periodically updated by a more 'complex' and accurate network communication model.

4.2 Development of a Networked Haptic Feedback System

Toward the above goal, two identical haptic systems were networked and a unit delay was incorporated in the transmission line. The resulting architecture is shown figure 4.1. The velocity information is transferred over the network from one user side to the other and is combined (negatively) with the velocity information at the local user. The resultant information is then fed to the VE residing over the local user end. The motivation behind this idea is to support the collaborative or shared operation of two users in the VE. For simplicity identical haptic devices and virtual environments have been considered. Using two identical haptic displays may not appear to be a reasonable assumption as users may have different kinds of haptic devices e.g. either impedance type or admittance type displays or they may also have different degrees of freedom. We have chosen this scenario for the sake of simplicity of the ensuing computations. However, the use of identical VEs is quite reasonable as multi-user VE requires initial downloading of the environment and further communication with other users either through unicast or multicast modalities for the transfer of information and periodic updating of the local environment. Networked multi-user VEs are extensively discussed in [1].

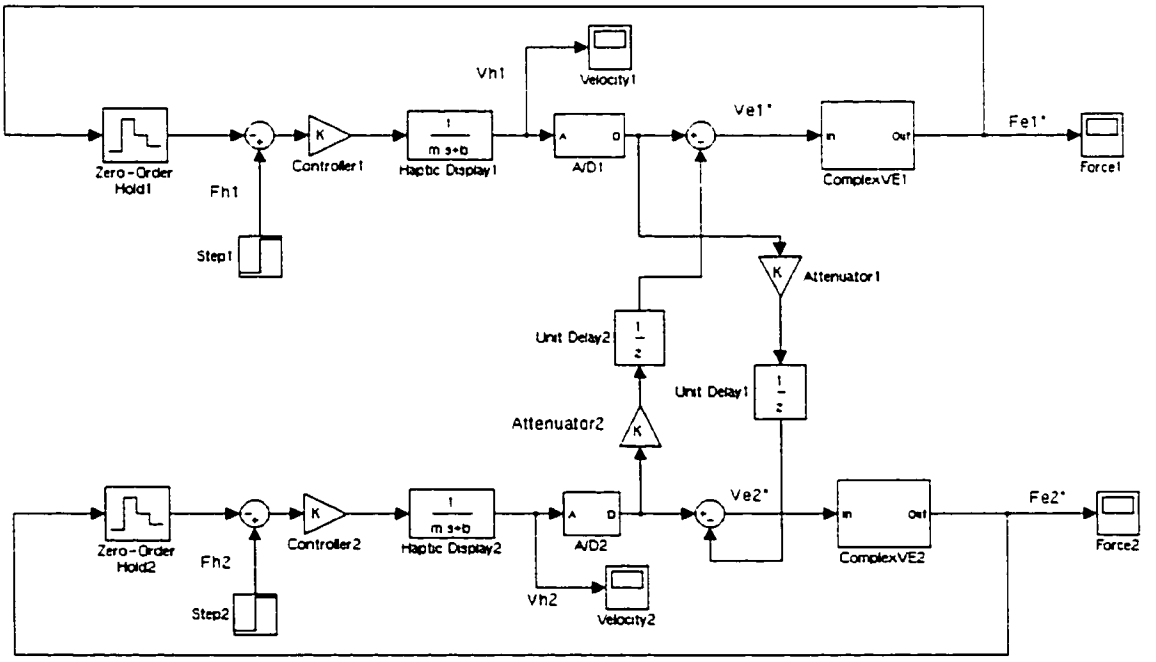

Figure 4.1. Networked haptic feedback system with unit delay

The signal flow diagram of the networked haptic system is shown in figure 4.2. The input-output relationship is derived and is shown as an immitance matrix given by

$$\begin{bmatrix} V_{h_1} \\ V_{h_2} \end{bmatrix} = \begin{bmatrix} HD_1 & -z^{-1}(HD_2)(VE_1)(ZOH)(HD_1) \\ -z^{-1}(HD_1)(VE_2)(ZOH)(HD_2) & HD_2 \end{bmatrix} \times \begin{bmatrix} F_{h_1} \\ F_{h_2} \end{bmatrix} \quad (4.1)$$

The immitance matrix is of the form Y= GI, where Y = output vector, G = admittance matrix and I = input vector, and

$G = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}$, where g_{ij}'s are frequency dependent terms and are defined according to (4.1).

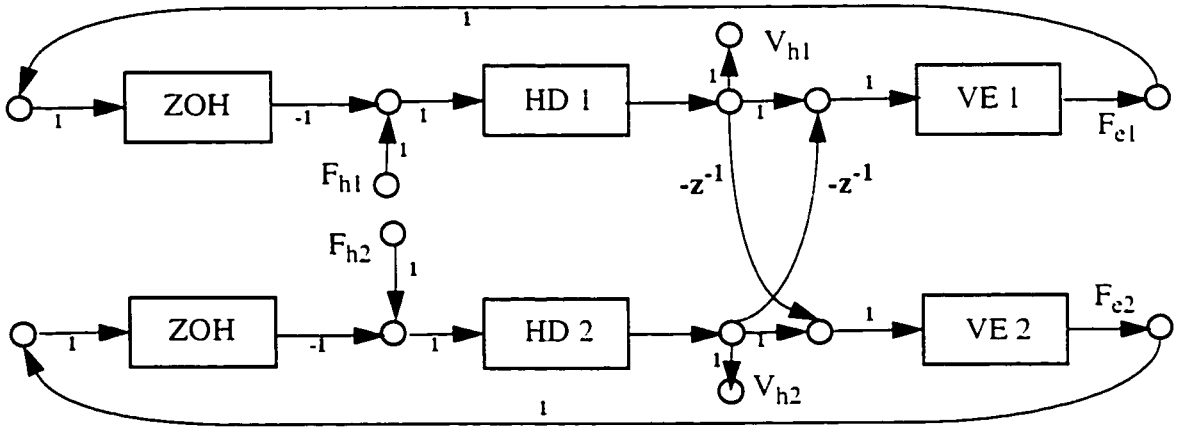

Figure 4.2. Signal flow graph of networked VE

To establish the design guidelines for such a system we can use criteria-3 presented earlier in Chapter 3, section 3.2 for stability consideration. Using equations (3.3) and (3.4)

the stability conditions become

$$Re(g_{11}) \geq 0, \qquad Re(g_{22}) \geq 0 \tag{4.2}$$

and

$$Re(g_{11})Re(g_{22}) - \left|\frac{g_{21} + g_{12}}{2}\right|^2 \geq 0, \qquad \forall \omega \geq 0 \tag{4.3}$$

The haptic display system 1 (i.e. HD_1) can be expressed in the z-domain using a first order difference approximation technique

$$HD_1(s) = \left.\frac{K}{m_1 s + b_1}\right|_{s \rightarrow \frac{z-1}{Tz}}$$

or equivalently

$$HD_1(j\omega) = \frac{TKe^{j\omega}}{(m_1 + b_1 T)e^{j\omega} - m_1} \tag{4.4}$$

and similarly the haptic display system 2 (i.e. HD_2) can also be expressed as

$$HD_2(j\omega) = \frac{TKe^{j\omega}}{(m_2 + b_2 T)e^{j\omega} - m_2} \tag{4.5}$$

By applying equation (4.2) to the above system results in the following inequality $m_1 + b_1 T \geq m_1 \cos\omega$ and $m_2 + b_2 T \geq m_2 \cos\omega$, which leads to one of the design criteria for the haptic display system in order to obtain a stable system.

In order to analyze (4.3) we have considered two identical haptic systems with the same sampling rate at both the user end and other parts of the system. The analysis of this

condition now involves z-transformation of the VE. As before the first order difference approximation is used for the transformation and we obtain

$$VE(s) = b_e + \frac{K_e}{s}\bigg|_{s \to \frac{z-1}{Tz}} \Rightarrow VE(j\omega) = b_e + \frac{K_e T(e^{j\omega})}{e^{j\omega} - 1} \tag{4.6}$$

Also note that the combined sampler and the zero order hold can be expressed as

$$ZOH(z) = \frac{z+1}{2z} \Rightarrow ZOH(j\omega) = \frac{e^{j\omega}+1}{2e^{j\omega}} \tag{4.7}$$

After some algebraic manipulations the real and imaginary parts of g_{12} and g_{21} ($g_{12} = g_{21}$) were obtained. By setting, without loss of generality, $m_1 = m_2 = m$, $b_1 = b_2 = b$ and $b_{e_1} = b_{e_2} = b_e$, $K_{e_1} = K_{e_2} = K_e$, the following expressions are now obtained by applying (4.3) to our haptic VE system:

$$\begin{aligned} Re(g_{12})/X &= 4m(m+bT)\sin^2\omega(b_e+KT) - 2m^2 b_e \sin^2\omega \\ &\quad -(m+bT)^2\{2b_e(1-\cos\omega)(\cos\omega-\cos 2\omega) + K_e T(\cos\omega - \cos 3\omega)\} \end{aligned} \tag{4.8}$$

and

$$\begin{aligned} Im(g_{12})/X &= 2mK_eT(m+bT)\sin 2\omega - m^2 2b_e \sin\omega(1-\cos\omega) - 2m^2 K_e T\sin\omega \\ &\quad -4mb_e(m+bT)(1-\cos\omega)\sin\omega + (m+bT)^2\{2b_e(\sin\omega+\sin 2\omega)(1-\cos\omega) + K_eT(\sin\omega - \sin 3\omega)\} \end{aligned} \tag{4.9}$$

where

$$X = \frac{T^2K^2}{4(1-\cos\omega)\{(m+bT)^2 - 2m(m+bT)\cos\omega + m^2\}^2}.$$

It can be shown that the condition (4.3) may be re-arranged into the following inequality, where expressions from (4.8) and (4.9) are to be applied

$$Re(g_{11})Re(g_{22}) \geq \{Re(g_{12})\}^2 + \{Im(g_{12})\}^2 \tag{4.10}$$

To summarize the design guidelines for the networked haptic display are given by the condition given in (4.9).

From the above discussion and for the system under consideration we can conclude that unconditional stability requires that the haptic interface to be stable for any set of passive human operators and the VE in real world. The haptic interface will remain stable whether the operator holds it with a steel grip or breaks contact completely. The environment may simulate free or rigidly constrained motion without loss of generality.

To test and verify the system performance requirements the design guidelines provided in equations (4.8), (4.9) and (4.10) were invoked. For the sake of numerical simulations the selected parameters were considered as follows $m_1 = m_2 = 1$, $b_1 = b_2 = 0.3$, $b_{e_1} = b_{e_2} = 0.1$, $K_{e_1} = K_{e_2} = 0.2$ and the sampling rate $T = 0.01$. The attenuation factor for the delay was kept at 1 implying that there was no attenuation over the network. The simulated force output from VE_1 is shown in Figure 4.3. The steady state error with respect to the step input is greater than that of the stand alone haptic system shown in Chapter 3, Figure 3.4(a). Furthermore, the oscillatory region increases from that in Figure 3.4(a). The ideal force output should be null throughout the operation. But due to the introduction of delay element between the systems the ideal behavior is lost and the realistic force realization is hampered. The velocity output is shown in Figure 4.4. The velocity is rather sluggish and has an exponential growth (i.e. unstable) which is not acceptable as the end user is concerned. The user will feel the object as "spongy" instead of free motion. The steady state velocity is reached to the value of 2 whereas in Figure 3.4(b) it is reached to 0.2. These outputs are affected by the additional couplings resulting from the other haptic system having a delay of one unit between them. Similar case holds for the other end of the networked system. Both the force and velocity behave in the same

manner with a similar haptic and a virtual environment model.

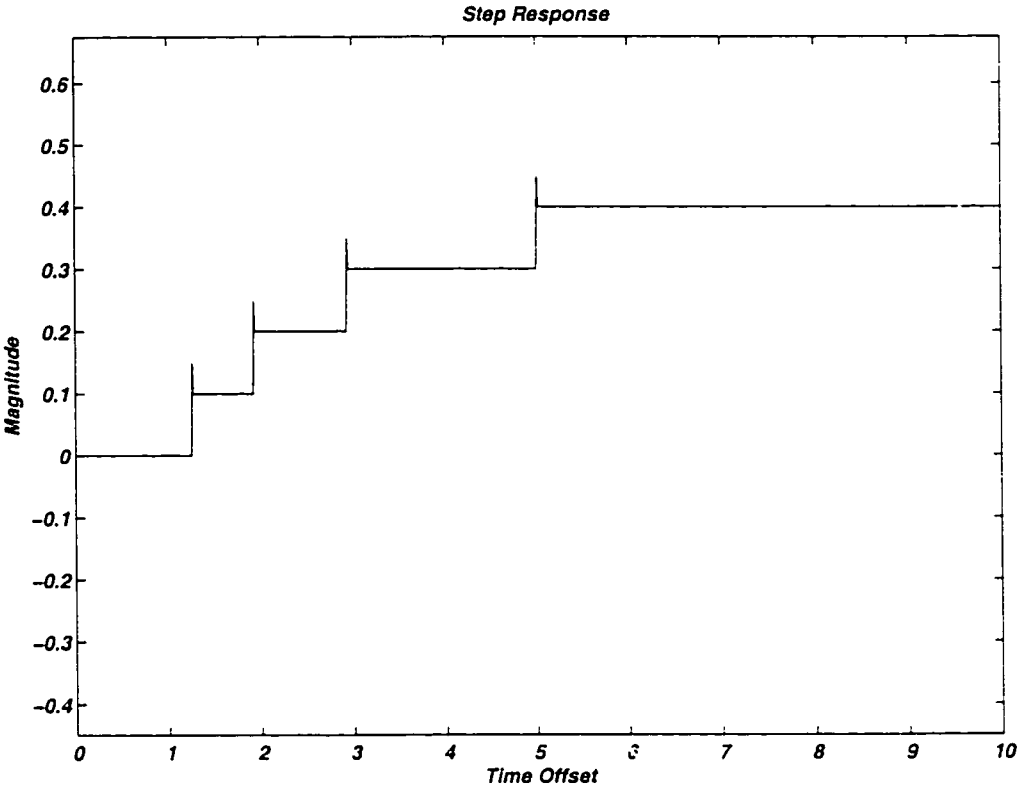

Figure 4.3. Force output from networked haptic system

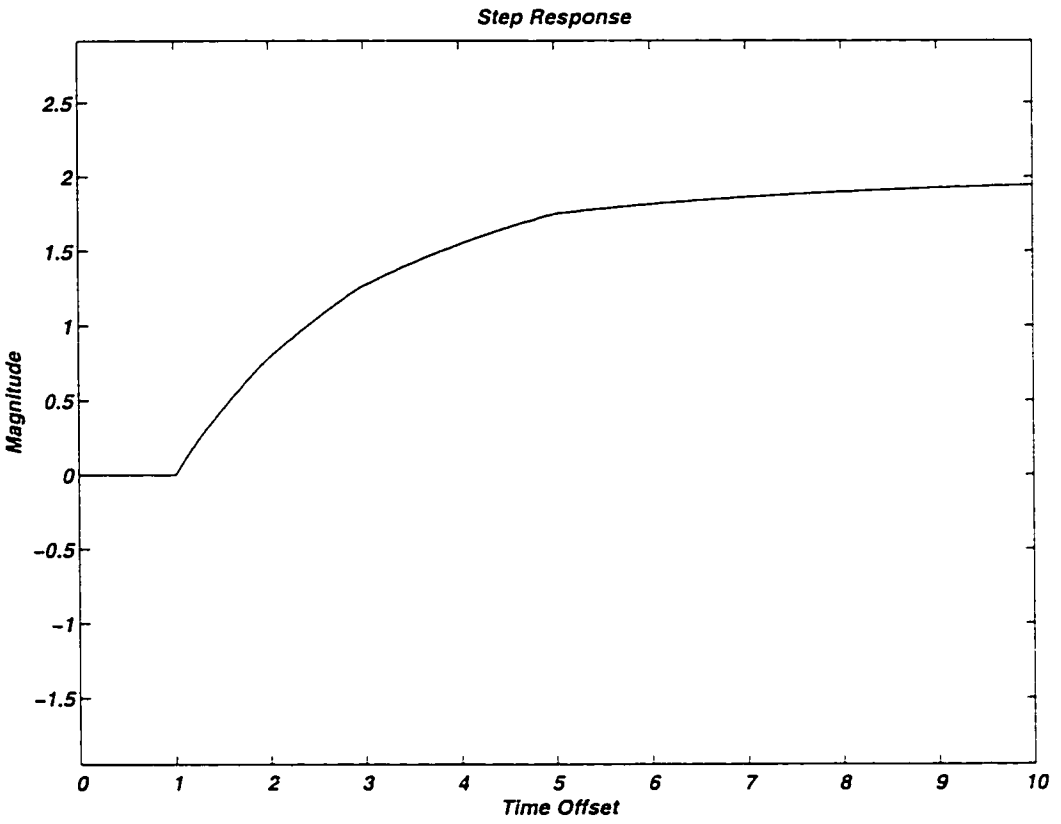

Figure 4.4. Velocity output from networked haptic system

4.3 Network Approach in Distributed Simulation

Multi-port networks fundamentally consist of one-port with generalized impedances, two-ports with generalized transformers and ideal energy junctions with common-effort and common-flow junctions. A physical system can be thought of as a collection of one-port subsystems which communicate via effort and flow variables. The effort and flow variables pass over network consisting of two ports, junctions and interconnecting power bonds [123]. Tellegen's theorem [124] consider a lumped finite network with b branches and n nodes. For any kth branch of the network, if i_k is the branch current and v_k is the branch voltage for different set of operating conditions then with the sign convention

$$\sum_{k=1}^{b} v_k i_k = 0 \tag{4.11}$$

which leads to the consequence that a network composed of only passive components is itself passive [124].

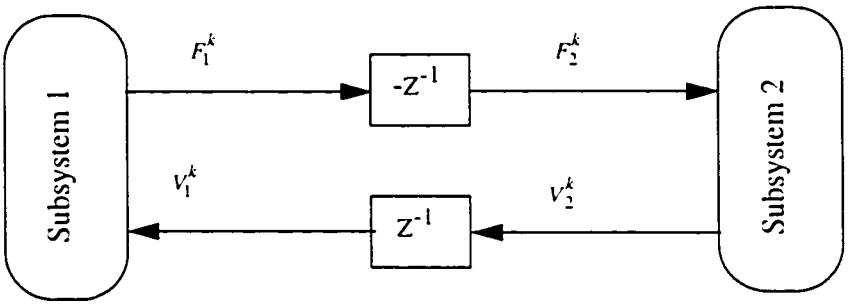

Figure 4.5. A Distributed two-port simulation model

The simplest possible architecture of a distributed simulation can be shown in Figure 4.5, where two subsystems interact over a network with unit delays. Treating the junctions as a two-port, the relation between the subsystems can be expressed with *immittance matrix* [114]. Possible immittance matrices are the "impedance matrix" Z where y is force and u is the velocity vector, the "admittance matrix" Y where y is the velocity vector and u is the force vector and the "hybrid matrix" H where y is composed of v_1 and F_2 and u is composed of v_2 and F_1. The effort and flow relation between the two subsystems in Figure 4.5 can be expressed with "hybrid matrix" as:

$$\begin{bmatrix} V_1^k \\ F_2^k \end{bmatrix} = \begin{bmatrix} 0 & z^{-1} \\ -z^{-1} & 0 \end{bmatrix} \begin{bmatrix} F_1^k \\ V_2^k \end{bmatrix} \tag{4.12}$$

where the two-port "hybrid matrix" is defined as $H(z) = \begin{bmatrix} 0 & z^{-1} \\ -z^{-1} & 0 \end{bmatrix}$.

For this two-port to be passive, it is necessary that $H(e^{j\omega T}) + H^*(e^{j\omega T})$ be passive for any ω and T, where T is the time delay corresponding to z^{-1}. If the two-port is composed of only passive elements, $H(e^{j\omega T}) + H^*(e^{j\omega T})$ would have all zero eigenvalues [3]. For this particular case, we have:

$$H(e^{j\omega T}) + H^*(e^{j\omega T}) = \begin{bmatrix} 0 & e^{-j\omega T} \\ -e^{-j\omega T} & 0 \end{bmatrix} + \begin{bmatrix} 0 & -e^{j\omega T} \\ e^{j\omega T} & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -2j\sin\omega T \\ 2j\sin\omega T & 0 \end{bmatrix} \tag{4.13}$$

The eigenvalues are $\lambda = \pm 2\sin\omega T$. Therefore, the only value of T for which the junction is passive (lossless) is $T = 0$ (no delay).

The above straightforward approach to exchanging effort and flow information between simulation subsystems would not therefore be passive due to the computational and communication delays. An improved approach that incorporates the inherent delays over transmission lines is introduced in [114] to guarantee discrete-time passivity. This type of junction model is based on a lossless, dispersionless (dielectric lossless) transmission line (t-line) as shown in Figure 4.6.

If we consider that the t-line has mass per unit length m and stiffness per unit length k and the length is L, then the t-line has a total mass M = mL and total stiffness of K = kL. The "surge impedance" ("characteristic impedance" - well known property of the t-line for wave reflection) of the t-line is $Z_o = \sqrt{MK}$ and the transmission delay is $T = \sqrt{M/K}$. If $e_a(s)$ and $f_a(s)$ are the Laplace domain effort and flow variables at one end, and $e_b(s)$ and

$f_b(s)$ are at the other end, the input-output relation can be represented in terms of the "hybrid matrix" as [121]:

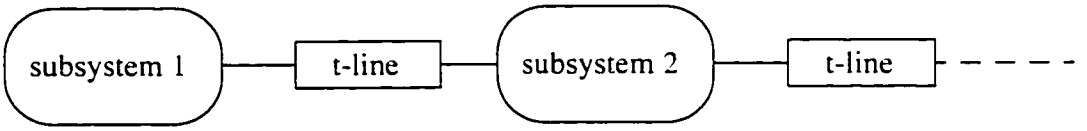

Figure 4.6. Transmission line model of distributed simulation

$$\begin{bmatrix} e_a(s) \\ e_b(s) \end{bmatrix} = Z_o \begin{bmatrix} \dfrac{e^{-Ts}+e^{Ts}}{e^{-Ts}-e^{Ts}} & \dfrac{2}{e^{Ts}-e^{-Ts}} \\ \dfrac{2}{e^{Ts}-e^{-Ts}} & \dfrac{e^{-Ts}+e^{Ts}}{e^{-Ts}-e^{Ts}} \end{bmatrix} \begin{bmatrix} f_a(s) \\ f_b(s) \end{bmatrix} \tag{4.14}$$

or alternatively,

$$\begin{bmatrix} e_a(s) \\ e_b(s) \end{bmatrix} = Z_o \begin{bmatrix} \dfrac{\lambda^2+1}{\lambda^2-1} & \dfrac{2\lambda}{1-\lambda^2} \\ \dfrac{2\lambda}{1-\lambda^2} & \dfrac{\lambda^2+1}{\lambda^2-1} \end{bmatrix} \begin{bmatrix} f_a(s) \\ f_b(s) \end{bmatrix} \tag{4.15}$$

where $\lambda = e^{-Ts}$ is the delay operator. For this system to be passive, the t-line must itself be passive provided that the subsystems are already passive. It is necessary that $H(e^{j\omega T}) + H^*(e^{j\omega T})$ be positive for any ω and T for the t-line to be passive. Therefore, in this case we have

$$(H + H^*)\big|_{s=j\omega} = Z_o \begin{bmatrix} \dfrac{e^{-j\omega T} + e^{j\omega T}}{e^{-j\omega T} - e^{j\omega T}} & \dfrac{2}{e^{j\omega T} - e^{-j\omega T}} \\ \dfrac{2}{e^{j\omega T} - e^{-j\omega T}} & \dfrac{e^{-j\omega T} + e^{j\omega T}}{e^{-j\omega T} - e^{j\omega T}} \end{bmatrix} + Z_o \begin{bmatrix} \dfrac{e^{j\omega T} + e^{-j\omega T}}{e^{j\omega T} - e^{-j\omega T}} & \dfrac{2}{e^{-j\omega T} - e^{j\omega T}} \\ \dfrac{2}{e^{-j\omega T} - e^{j\omega T}} & \dfrac{e^{j\omega T} + e^{-j\omega T}}{e^{j\omega T} - e^{-j\omega T}} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.16}$$

From (4.16) it follows that when $\lambda^2 \neq 1$, $H(e^{j\omega T}) + H^*(e^{j\omega T})$ has all zero eigenvalues and therefore, t-line is passive and lossless. But when $\lambda^2 = 1$, $H(e^{j\omega T}) + H^*(e^{j\omega T})$ becomes indefinite and the t-line is no more guaranteed to be passive and lossless.

A rigorous analysis of the t-line approach is conducted for unilateral and bi-lateral constraints in [125]. The problem with this approach is that it considers only the definite time delays over the transmission line and does not consider the situation of unexpected delays due to congestion and/or loss of packets over the network which is quite common in today's communication networks. Handling this type of delay is quite vital for the performance and stability of the force feedback devices in a DVE.

4.4 Dead Reckoning in Shared State

Dead Reckoning is defined as the transmission of regular or non-regular updated information to and from remote hosts where prediction and convergence concepts (to be defined formally later) are used to manage states with these updates. This technique provides weakly consistent state maintenance (with bounded error) in order to fully utilize the available bandwidth and maximize the number of participants (or hosts).

Dead reckoning algorithm is quite a popular technique in network-based virtual environments in order to maintain the shared states of different entities. It is quite efficient given the limited bandwidth of communication channels and processing powers of the host computers. The idea behind it is to transmit state update packets less frequently and

instead use information contained in those updates to approximate the true shared state. This state prediction technique is known as *dead reckoning protocol*. In the latest standard in IEEE std. 1278.1, a dead reckoning model (DRM) notation is introduced. A DRM is defined by three elements. The first element indicates whether the model specifies a rotation as either fixed (F) or rotating (R), the second element specify if the DRM rated are to be held constant as either rate of position (P) or rate of velocity (V), and the third one specifies the coordinate system to be used with the dead reckoning algorithm as either world coordinate (W) or body axis coordinates (B). These elements are grouped as DRM (F or R, P or V, W or B). The dead reckoning protocol consists of two elements - Prediction and Convergence defined more precisely below.

Prediction: The most common technique used here is derivative polynomials to predict an entity's future position. In the simplest zero-order polynomial the instantaneous position is used to predict the updated position, that is

$$\text{predicted position after t seconds} = \text{position}.$$

The zero-order polynomial does not fully support the requirements for dead reckoning and gives, in general, very poor quality displays due to a sudden movement of an entity. First order polynomial may be considered to be the next obvious choice. In this case, the instantaneous velocity information is used to predict the next position, that is

$$\text{predicted position after t seconds} = \text{position} + \text{velocity} \times t$$

Similarly, incorporating a second order derivative into the update packets will yield more accurate position prediction, in other words

$$\text{predicted position after t seconds} = \text{position} + \text{velocity} \times t + 0.5 \times \text{acceleration} \times t^2$$

and so on.

The anticipated position of an entity is usually calculated based on the last (or past) accurate state information of the entity using some extrapolation equation [126]. The extrapolation equations can be divided into two groups: one-step formulae and multi-step formulae [127]. One-step formulae only use the last state update packet to extrapolate an entity's position, whereas, multi-step formulae use the last two or more state update packets in the extrapolation. For a position update dead reckoning the extrapolation equations are as follows:

1st order (one-step): $$x_t = x_{t'} + v_{t'}\tau \tag{4.17}$$

1st order (two-step): $$x_t = x_{t'} + \frac{x_{t'} - x_{t''}}{t' - t''}\tau \tag{4.18}$$

2nd order (one-step): $$x_t = x_{t'} + v_{t'}\tau + 0.5a_{t'}\tau^2 \tag{4.19}$$

2nd order (two-step): $$x_t = x_{t'} + v_{t'}\tau + 0.5\frac{v_{t'} - v_{t''}}{t' - t''}\tau^2 \tag{4.20}$$

where $x_{t'}$, $v_{t'}$ and $a_{t'}$ represent the position, velocity and acceleration of the entity in the last state update packet, respectively. Similarly, $x_{t''}$, $v_{t''}$ and $a_{t''}$ are the position, velocity and acceleration of the second last packet, respectively, and τ is the elapsed time from the last update, that is $\tau = t - t'$.

The choice of the derivative polynomial order has to be decided adaptively depending on the nature of the data. In other words, the remote host can dynamically assign the order to update the predicted position. The criteria for selecting the order of the polynomial depends on the History-based dead reckoning [1] where the predictive polynomial depends on the past behavior of the entity for certain duration. In this respect, the Position History-Based Dead Reckoning (PHBDR) was developed by Singhal and Cheriton for PARADISE net VE in Stanford [1]. The protocol evaluates the last three position updates and checks for minimal or substantial acceleration. It then selects a first-order derivative polynomial to predict next position. Threshold cutoff values are used to define the bounds for both a minimal and substantial acceleration categories.

From the above discussion, one may conclude that although high precision updated positions may be obtained by using higher-order polynomials, however, this comes with a cost to the system. The reason behind this is that transmission of higher-order derivatives is an overhead to the system given the strict bandwidth constraints and computational resource limitations. The extra overhead may actually reduce and diminish the advantages of using the more complicated dead reckoning algorithms. Moreover, it is harder to get accurate instantaneous information about higher-order derivatives.

Convergence: This algorithm is used to correct the inexact prediction whenever an updated information is received. A piece-wise constant approximation may quickly represent the prediction without visual distortion. Though this type of convergence is faster and simpler to understand and implement, it does not provide the best visual display. A slower and a smoother convergence technique is known as the linear convergence. The entity travels in a straight line (convergence path) from the current position to the convergence point along the new predicted path. A more slower, smoother and sophisticated convergence techniques can also be achieved by curve-fitting approaches. Cubic Splines can in principle achieve the best approximation at the cost of extra computational overhead.

4.5 Haptic Dead Reckoning

Dead reckoning for haptic systems can be employed using two models of an environmental system:

i. A local and a simpler environment model to meet the high rate local needs of the haptic display, and

ii. A full system representing the high fidelity display and realization.

The local model computes the updated forces (or displacements) to be applied to the haptic device. Kinematic information, stiffness and damping, are passed from the local system to the environment server. The environment server is responsible for passing the updated model parameters back to the local system. The updating procedure in the environment server relies on linearizing the full system model and then simplifying or approximating this linearized model. The simplification procedure consists of the

following steps [42]:

i. *Linearization:* The system model is linearized around the current operating point. The result of this linearization is a system of N first-order ordinary differential equations with an $N \times N$ Jacobian matrix of the rates-of-change of the states for the full system.
ii. *Eigenvalues and Eigenvectors*: These are calculated from the Jacobian matrix.
iii. *Diagonalization:* Using the eigenvectors, the Jacobian matrix is then diagonalized.

iv. *Order Reduction:* The model-order is reduced using the diagonalized system of ordinary differential equations.
v. *Update:* A set of coefficients are calculated from the preceding steps and the coefficients are used to define a simplified, linear set of ordinary differential equations in state space representation. These coefficients are transmitted to the local system for use over the updating interval.

Compared to the dead reckoning for maintaining the position of entities as described earlier, the above procedure is very similar to the prediction algorithm. The linearization and simplification procedures described above are more or less the same for both impedance and admittance type displays. The only difference is in the way the coefficients are calculated. This difference arises due to the type of the calculations needed for the output that could be either force or displacement. The order of the differential equations have a significant affect on the performance of the algorithm. The higher the order of the model the more accurate information will be available to the local models, albeit at the cost of extra computational burden. The sudden changes in the interacting environment may cause instability in the haptic display. Collision, the abrupt change of the interacting surface have significant role on the performance of the display device. One way to get rid of this extraneous behavior of the interacting device is to provide the behavioral information of the interacting surface or the environment and the post collision data to the environment server. After detecting the abrupt changes in the environment the new operating point is selected and the above calculations are performed again on the new operating point. While calculating the coefficients for the full system server care must be

taken to meet the Z-width to ensure the stability of the overall system.

The instability due to random and unexpected time delays are minimized through the updated information from the environment server. Whenever new data (packets) arrives the convergence algorithm is used to offset non regularities in the output calculation.

4.6 Networked Haptic System with Dead Reckoning

As described earlier for dead reckoning a simpler and an approximate model is used instead of the complex model. The simplification procedure was discussed in section 4.5. As an example of the simplification process and results let us assume that the transfer function of the VE model $VE(z)_{complex} = \frac{Kz}{(z-a)(z-\alpha)}$ and the approximated model can be given as

$$VE(z)_{simple} = \frac{(Kz)/(1-\alpha)}{(z-a)} \tag{4.21}$$

In the simulation results our strategy was to observe different force outputs for different values of the complex environment and then compare them with the force output from the simpler version of the VE. The velocity output is obtained from the two systems and the resulting difference between them was computed. The specifics of the simple and the complex VE are shown in Figure 4.7.

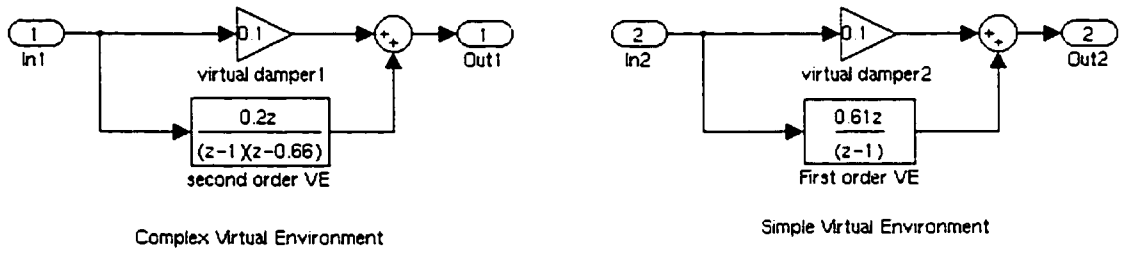

Figure 4.7. Complex and simple virtual environments

Two different systems similar to Figure 4.1 are modelled with the two different environments as shown in Figure 4.7. The force output from the system with complex environment is shown in Figure 4.8. After an initial step rise for a duration from 2.5 to 7 sec. where the output force remains constant, it becomes oscillatory at around 7.5 sec. The force output from the system with simpler environment is shown in Figure 4.9. The difference between these two force outputs is quite interesting. Using the simple environment the oscillatory behavior of the output force due to complex environment is removed. The error between these two forces is shown in Figure 4.10. The maximum error magnitude obtained is 0.3. For the reduction of the model from a second order to a first order given a unit step input, the maximum error is quite significant (30%).

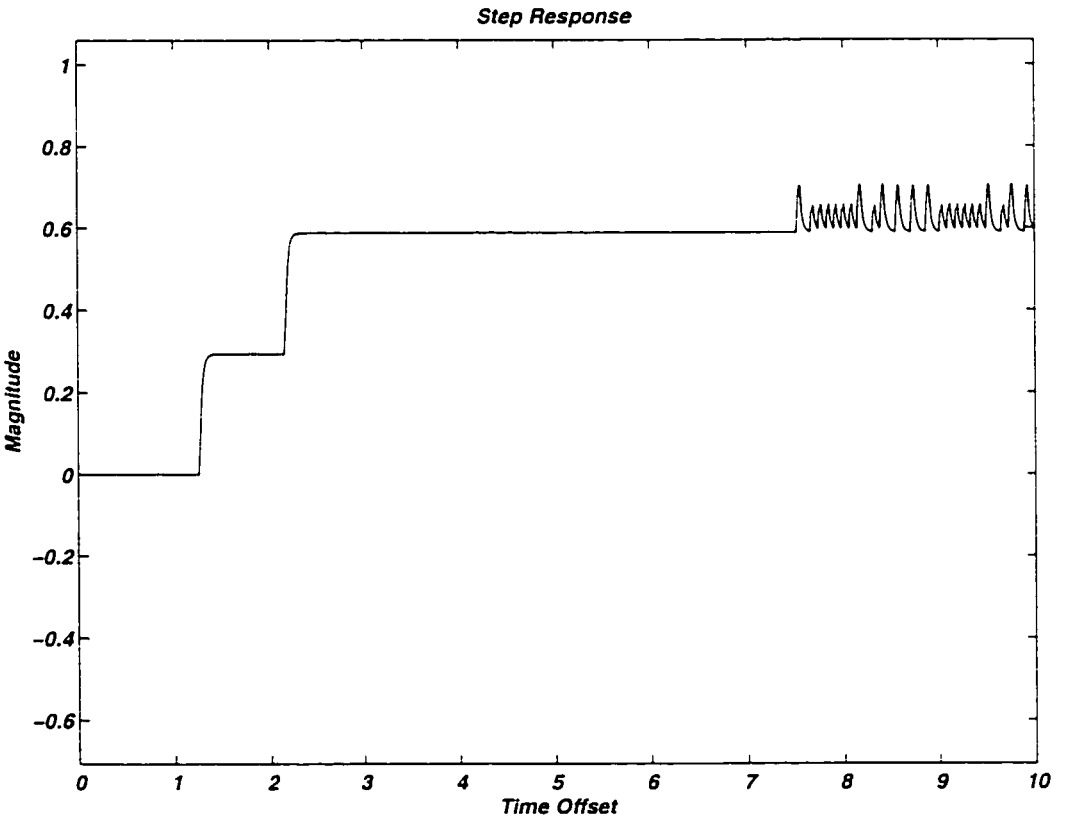

Figure 4.8. Force output from the complex VE

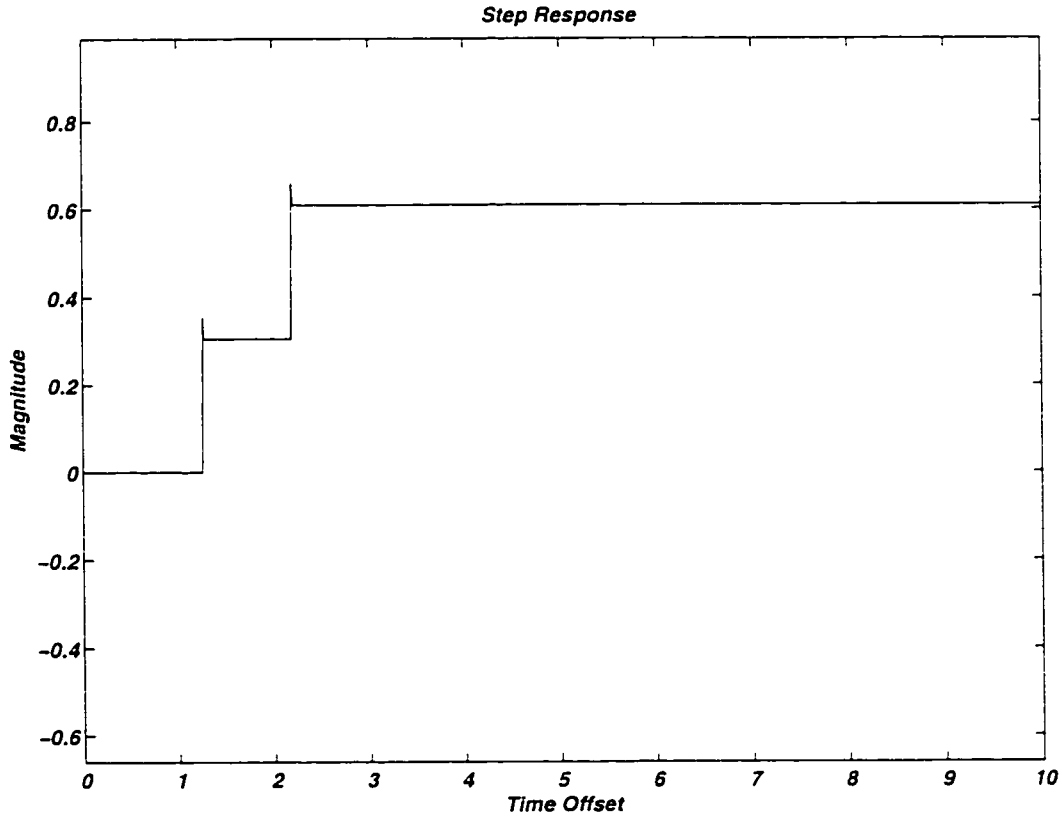

Figure 4.9. Force output from the simple VE

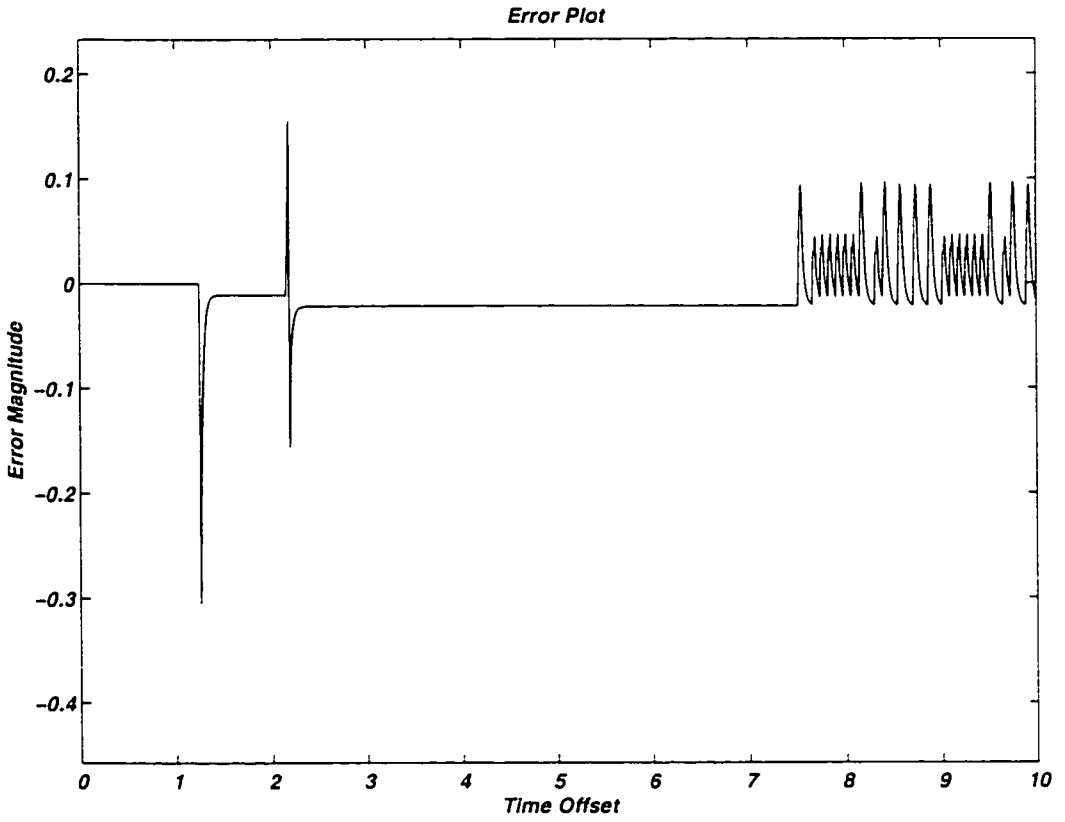

Figure 4.10. Error between force outputs from the simple and complex VE

The velocity output from the system with complex VE is shown in Figure 4.11. The exponential growth of the velocity gives a realization to the user as a "spongy" object before it reaches a constant value. After reaching the constant value it appears as a unilateral constraint object to the user. In Figure 4.11 the velocity reaches a constant value (1.25) at around 7.5 sec.

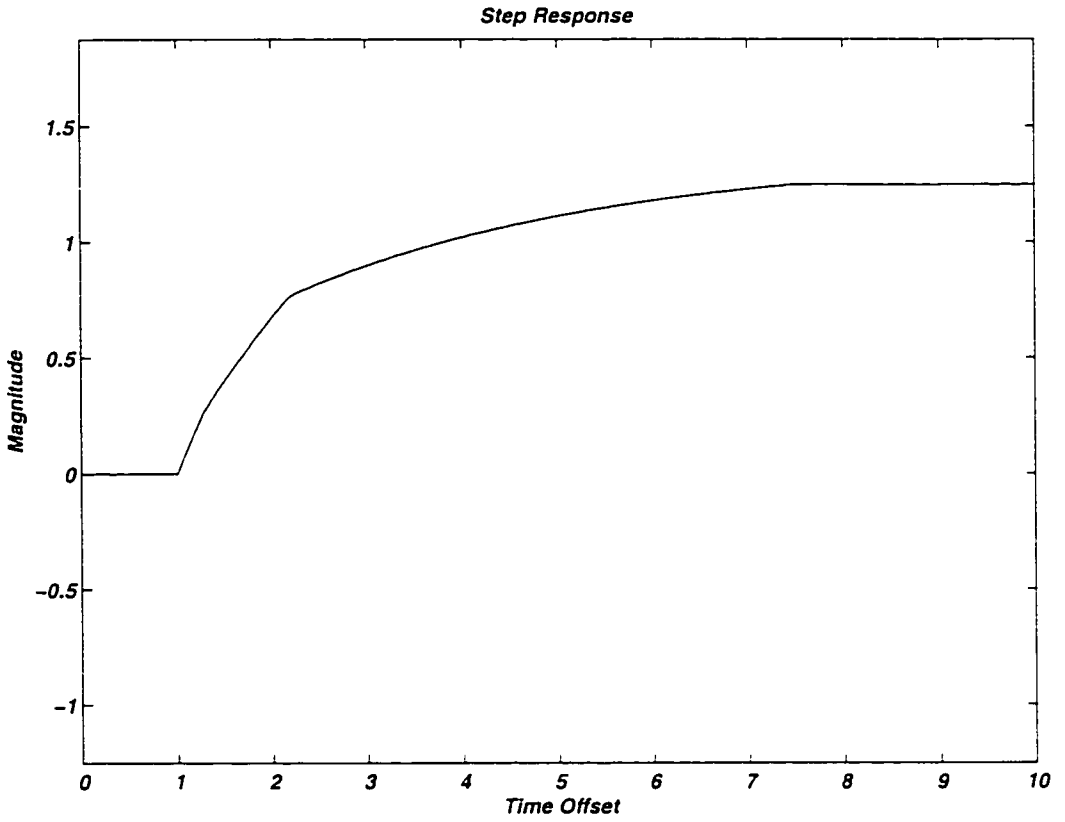

Figure 4.11. Velocity output from haptic system with the complex VE

Similarly, the Figure 4.12 shows the velocity output from the system with simple environment. The output has almost similar behavior as of Figure 4.11. The constant value reaches after 9.5 sec. The error between these two velocity outputs are plotted in Figure 4.13. The maximum error obtained is about 0.06.

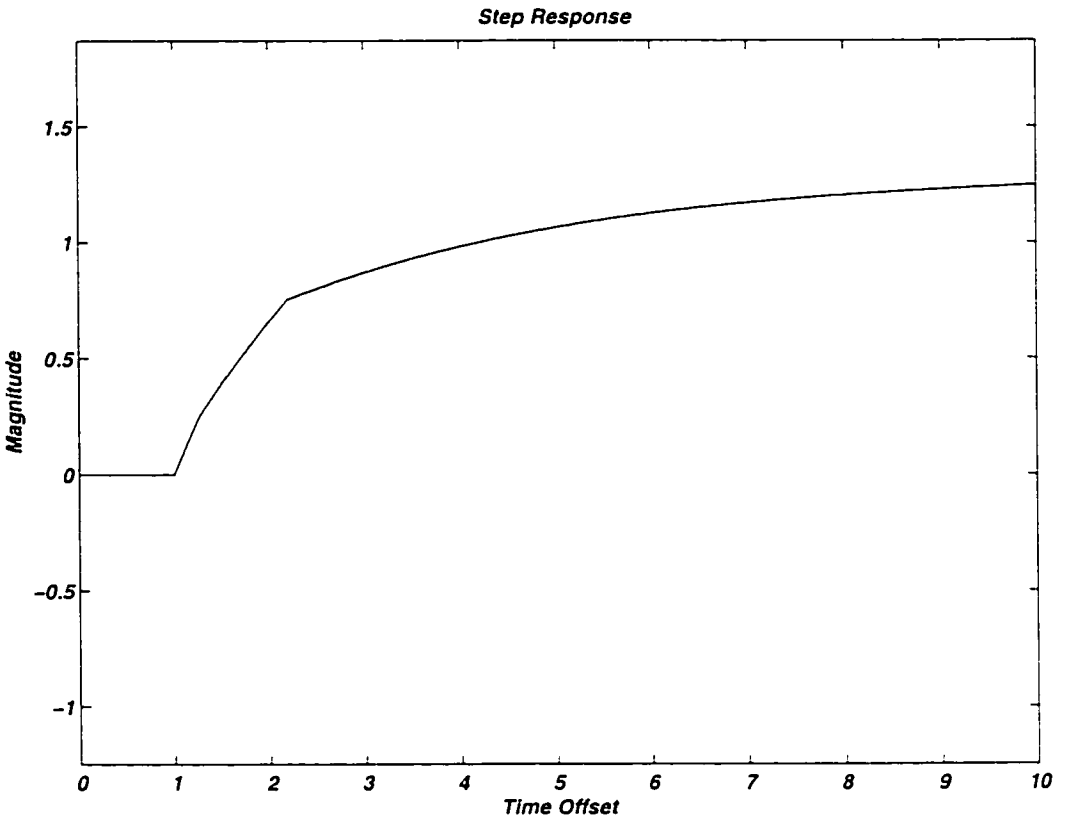

Figure 4.12. Velocity output from haptic system with the simple VE

From the above discussion and figures it is quite evident that the model reduction technique is fairly dependent on the resolution and precision requirements of both the haptic device and human finger tips. For a fine precision and resolution of the haptic device it would not be possible to reduce the order of the system drastically. On the other hand, to meet the hard real-time constraints, the force calculation model must be simpler. Thus in order to assure a real-time interaction and realistic immersion of the VE, there is a trade-off between allowable error and computational time required. For improved real-time immersion, the accuracy of the force computation has to be sacrificed, and similarly for better accuracy the quality of the immersion may have to be degraded.

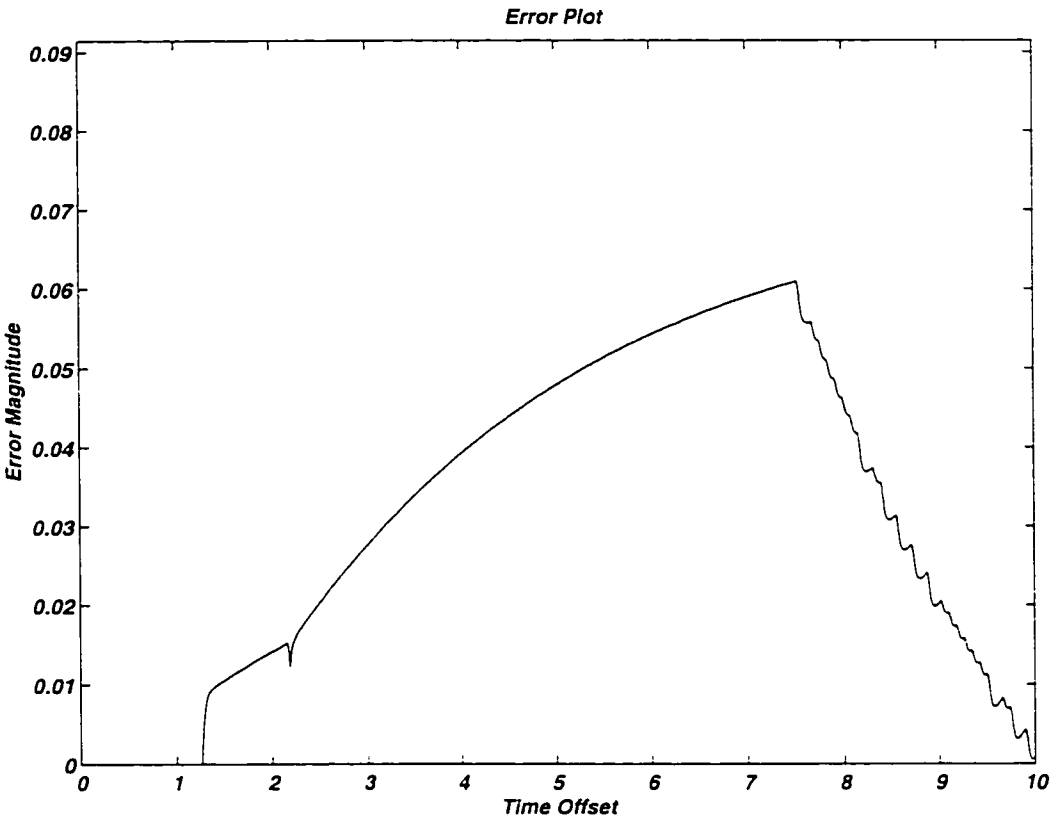

Figure 4.13. Error between the velocity output from the complex and simple VEs

4.7 Conclusion

The implementation of "simple" and "complex" world models were rather simple compared to the high end real world graphic applications where the interactive VE changes dynamically and simultaneously with the kinesthetic information. The robustness of our proposed algorithm subject to a rapidly changing haptic VE and the subsequent stability and the end user implication and requirements are yet to be investigated. The impact of just before and after the collision in a networked haptic interface when implementing our algorithm may have some significant effects on the overall performance of the system. The design guideline developed in (4.10) considering the delay parameter over the network is

not practically feasible due to the unpredictable nature of the delay in today's communication network.

Our next goal is to implement the proposed algorithm in a two user VE using two PHANTOM™ haptic displays in a collaborative environment. The delay will be simulated locally to make it unpredictable in nature.

Chapter 5

Simulation and Results

5.1 Introduction

A collaborative 3D-graphics environment with two simultaneous users is simulated on a single computer with dual processor- one for graphical simulation and the other one is for force calculation (haptic simulation). Two processes "graphics" and "haptics" run independent of each other and a real-time interaction between these two processes is maintained throughout the simulation. "Tree" data-structure is used for both the "graphics" and "haptics" environment simulation. Two force calculation models - "complex" and "simple" are simulated. Force outputs for different types of interaction from both the force calculation models along with the errors between them are investigated.

5.2 Graphical Modelling

The 3D-graphical environment is simulated using "tree" data structure and termed as "scene graph". The "scene graph" is the structure that holds all of the current elements of the scene, such as geometries, lights and positional information. The "scene graph" is an ordered collection of nodes in the form of a directed acyclic graph which holds hierarchical scene data. It provides a very powerful scene structure for real-time 3D simulation. Its hierarchical framework provides the capacity of grouping objects together

spatially. This spatial hierarchy is very important in maintaining efficient simulation containing numerous distinct moving parts. In addition to animation the scene graph also provides an efficient way of culling, picking objects from scene, collision detection and level-of-detail (LOD) perception.

The basic element, node, holds either geometry, light, fog and positional data, or is a structural element that is used to maintain the hierarchy of the graph. Nodes can be simply an element of content or can be either grouping or procedural element. "Content" nodes contain geometry, transformation (position and orientation), light and other rendering information. "Grouping" nodes are rather organizational nodes used to group certain geometric objects and "procedural" nodes contain certain additional information like LOD. Nodes are ordered in a directed hierarchical fashion. In other words, nodes are attached from the top to bottom, in a tree like structure. A node that has other nodes attached to it at the bottom is a "parent" to those nodes beneath it. The nodes attached immediately underneath another node are the "children" of that node. If two nodes share the same parent, then they are considered to be "siblings". Figure 5.1 illustrates the parent, child and sibling structure.

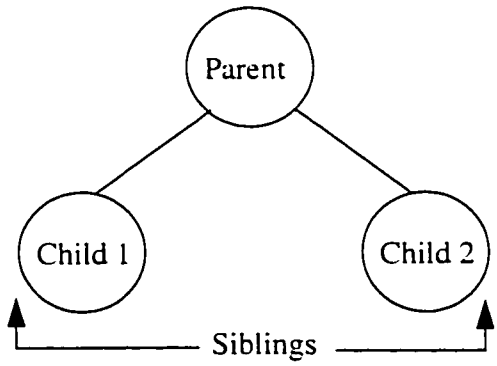

Figure 5.1. Parent, child and sibling relationship

The starting point of the "tree" is called a root node. Because of the hierarchical

structural in a top to bottom manner of the scene graph, the root node represents the top point on the scene graph. Each scene graph has a single root node, and this can not be shared with other scene graphs. The root node is the entry point into the scene graph while traversing it. This "traversing" process is always the same with visiting each node of the tree in a top to bottom, left to right order. In other words, when "traversing" encounters a node with more than one child, it walks down the first child's branch, completely traversing this portion of the tree before returning back up and processing the second child's branch as shown in Figure 5.2 where *geom node* is the geometry node and *postn node* is the positional node and the numbers show the sequence of traversing the scene graph.

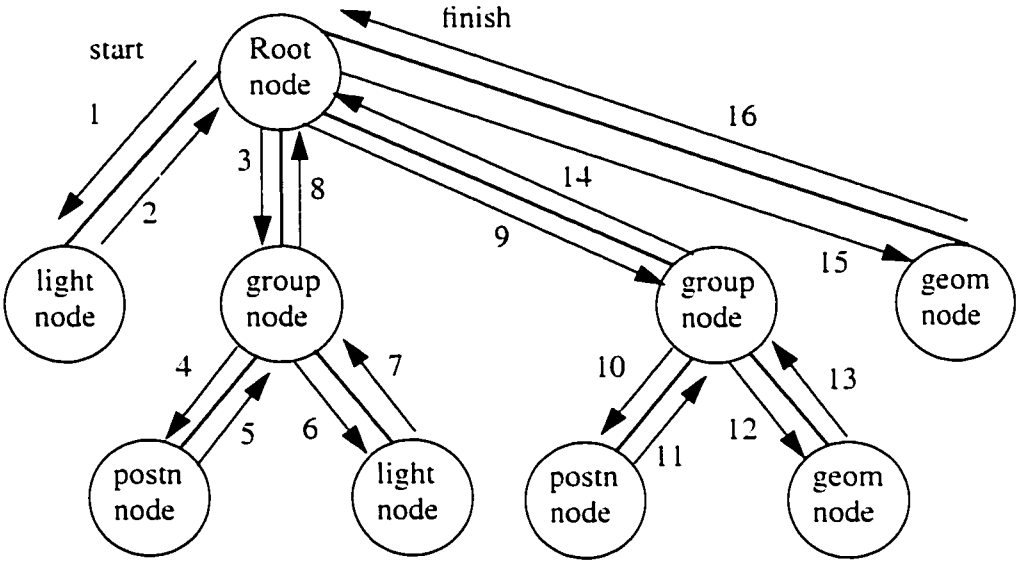

Figure 5.2. Traversing the scene graph

During the process of traversing the scene is drawn. When a node is encountered, it is evaluated and processed depending on its content. More explicitly, when a geometry node is encountered, it is drawn with current position and orientation with the current lighting. When a light node is encountered, the light is added to the currently active set of lights. When a transform node is encountered, the current orientation and position information

are modified. The entire scene graph tree is traversed once per frame.

5.3 Haptic Modelling

Haptic modelling is similar to graphical modelling and based on the hierarchical collection of nodes called "scene graph". The internal nodes of the tree provide a means for grouping objects, orienting and scaling the sub-tree relative to the parent node, and adding dynamic properties to their sub-trees. The terminal nodes of the tree, called leaves, represent actual geometries or interfaces. Leaves also contain an orientation and scale relative to their parent nodes.

The terminus of the haptic interaction device is represented as a point within the scene graph. The interaction forces between this point and objects or effects within the scene are computed according to the force calculation model which depends on the properties of the objects and simulation environment. The resulting parameters of the force calculation model are then sent to the haptic interaction devices (Phantom) for realization. Applications can treat the haptic interaction point in the graphical world as either the physical location of the haptic interaction device terminus within its physical work-space or the computed location of the interaction point constrained to the surface of geometric objects. The latter point is known as the Surface Contact Point (SCP). This SCP is used to generate all surface interaction forces.

The collision detection in the haptic environment is modelled using the concept of "bounding volume". The bounding volumes are approximate boxed or sphere shaped containers that encapsulate whole or in parts of the objects in the environment. These are used to minimize the number of interactions between objects in the work-space and the haptic interaction device in each servo-loop pass. The servo-loop is the haptic simulation process and will be discussed in a later section. "Sweep and prune" technique is used to search and detect the interaction precisely. In each servo-loop pass the collision detection is tested for each bounding volume in a certain area near the SCP. If a bounding volume is in close proximity with the SCP, then that bounding volume is further broken into its underneath bounding volumes and the detection of collision process goes on to select a

more suitable sized bounding volume. The "sweep and prune" technique depends on the precision requirement. For extreme precision the final bounding volume may end up in a point shaped solid sphere/cube. For potential and practical application the final bounding volume usually end up in a moderate size of cube or sphere. Thus in servo-loop pass the bounding volume is used to minimize the number of SCP-object interactions and enhance the real-time performance.

Geometric objects are simulated with geometry consisting of rigid surfaces. When the physical position of the haptic interface end-point passes through the surface of an object, the object does not deform. Instead, the SCP is maintained for each haptic device. The SCP is forced to a point on the surface of any object intersected by that haptic device. The SCP never penetrates the surface of an object, even if the position of the associated node representing the haptic device does penetrate the logical boundaries of the surface as shown in Figure 5.3. The force is calculated using the spring-damper model. An additional node is created as the same size, position and orientation of the object representing haptic device object in the haptic scene graph. The graphical update depends on this SCP node instead of the real object node representing the haptic device object.

When computing and sending a force representative of haptic device object touching or intersecting a geometry node, the force normal to the surface is calculated using a spring-dashpot (spring-damper) model for the surface. The force is proportional to the difference in the position of the SCP from the actual position maintained in the node representing the haptic device. In addition, the tangential forces are computed using a stick-slip friction model where the coefficient of friction is considered in force calculation when the relative speed between the colliding objects is not zero.

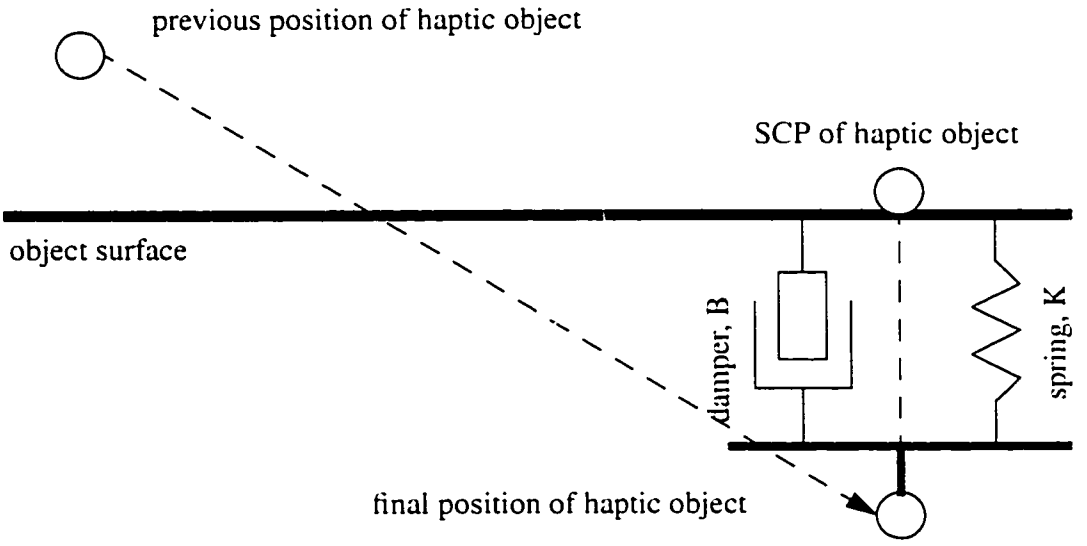

Figure 5.3. Surface Contact Point representation of haptic device object

5.4 Real-time Interaction

The two processes graphical and haptic simulation run in parallel as a single-threaded architecture as explained in Chapter 2, Section 2.3.1. Different tasks are cycled through a round-robin fashion on high speed processors. The reason of doing this is due to the simple graphics and properties of the virtual environment used here. The real-time interaction does not involve any network data to be processed. The simulated environment is rather a collaborative virtual environment on a single system and represents a dummy distributed virtual environment. Two users interact simultaneously with the same graphical environment. The real-time programming involve only in reading the users' input, updating the graphics and then sending back the feedback force to the users. The graphical process runs at a frame rate of 30 Hz whereas the haptic process maintains an update rate of 1000 Hz.

The overall process interaction is shown in Figure 5.4, where the two processes are named as "application process" and "haptic process". The haptic simulation loop defines all the dynamics and force calculation of the simulated model. The application process

starts with the *scene creation*. The scene is created with the specification of haptic and graphical scene graph as explained in Section 5.2 and 5.3. After creating the scene the application process goes to a "servo-control loop" which basically initiates the haptic simulation process. Subsequently the *application-specific (core) function* gets initiated which includes the generation and maintenance of all the graphical scene graphs. After updating the graphical simulation the *haptic state update* starts with a bi-directional information transfer with the haptic simulation process. Depending on the haptic state update data, the graphical simulation as well as the force feedback are updated. When simulation ends, the *clean-up* process frees-up all the memory allocated to the simulation [128].

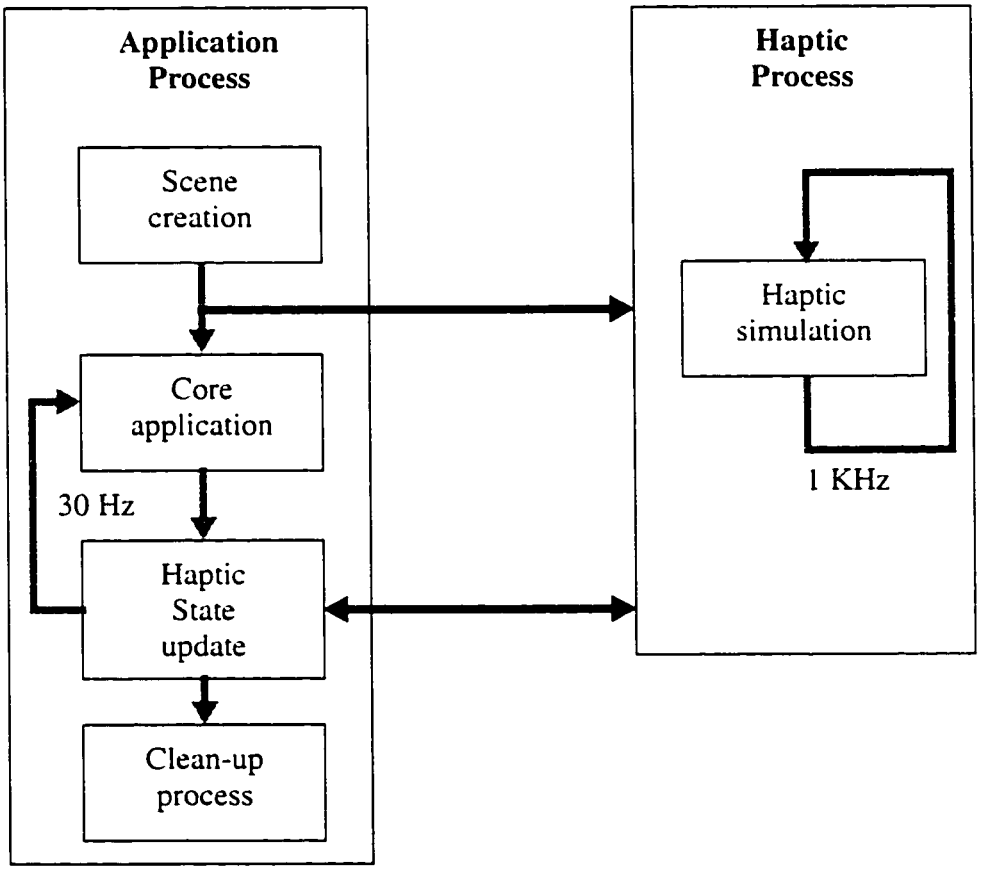

Figure 5.4. Real-time interaction between application and haptic processes

The *core application* process is maintained by a simulation manager that manages a container called "universe" which contains all the objects like geometries, sensors, lights, viewpoints, texture etc. It can have more than one scene graph to describe the environment. Every aspect of the simulation takes place in the universe. The simulation loop is entered and exited by particular functions calls. Figure 5.5 shows the normal sequence of execution of different actions in the graphical simulation loop.

At the beginning of the core application the sensor inputs (force inputs) are read. Depending on these inputs the universe's action functions are called. These functions define the required modifications needs in the graphical environment. The changes in the objects may be the change of positions, rotations and even collisions with other objects or no changes at all if the sensor inputs are not enough to yield any modification in the environment. The objects then perform the tasks that are implied on them. At the new situation the universe is rendered again with new data and rendering information. Finally, the simulation back to the step where it again waits for the sensor input to occur if the simulation is not stopped or exited through function calls [129].

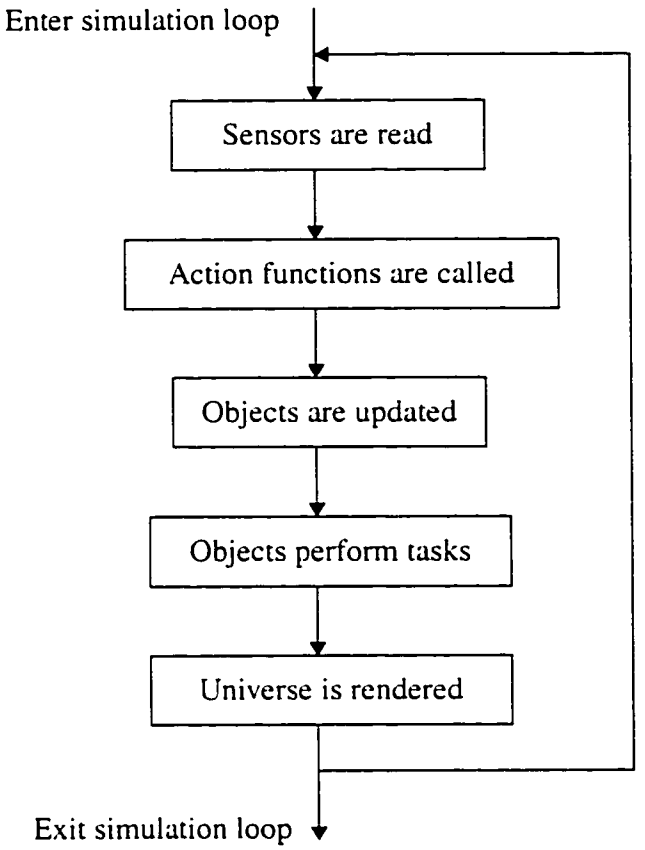

Figure 5.5. Simulation loop in the core application function

5.5 Simulation of Dynamics

This process runs in the simulation loop of the haptic process as shown in Figure 5.4. and deals with the motion of the objects. force calculation. object velocities. acceleration. collision detection etc. It is called at a rate of 1000 Hz in the simulation loop. This process maintains an active dynamic list of the objects that are inside the active environment. An object is added to the dynamic list whenever an external force is applied to the object. or if the application purposely add it to the dynamic list. An object remains in a dynamic list until it stops moving for a length of time of one second. This process interprets forces and

velocities on the objects and performs reflections off of the work-space boundaries (boundary of the simulated environment) defined by the object. It also performs collision detection between the object and its targets.

The feedback force (F_{out}) is calculated using the damping coefficient and spring constant as given in (5.1) below:

$$F_{out} = F_{in} - b \times v_{old} - K \times \nabla S \tag{5.1}$$

where F_{in} is the input force by the user, b is the damping coefficient, v_{old} is the velocity of the object just before collision, K is the spring constant and ∇S is the distance penetrated in the object. The new acceleration (a) and velocity (v_{new}) are calculated using the force (5.1) after considering the gravitational force as follows:

$$a = \frac{F_{out}}{m} + g \tag{5.2}$$

$$v_{new} = v_{old} + \nabla T \times a \tag{5.3}$$

where m is the mass of the object and ∇T is the elapsed time.

The collision is detected whenever there is a target object to collide. When objects collide, the algorithm performs a perfectly elastic collision, i.e., objects exchange their velocities. If the objects are moving roughly at the same speed, the algorithm treats them as a single object. The collision algorithm treats each as a cubic bounding box in its simplest form. Collision is detected whenever the distance in all three directions (X, Y and Z) violates the pre-set thresholds. After collision the object moves in the direction which is violated the least and it moves on until it no longer penetrates in that direction. If the objects' velocities are very similar and within a difference of 0.03 cm/sec. they are considered to be stucked together and then they simply exchange velocities. Otherwise, the velocities are dampened by the elasticity of the objects by being multiplied by the ratio of masses

between the objects according to Newton's law of momentum. Subsequently the objects performing collision are added to the active dynamic list. If one object was formerly at rest, it must be added manually to the active dynamic list. Otherwise, the scene will consider the object as being stationary. The scene automatically maintains an active dynamic list which is composed of those objects that are known to be moving. It adds objects to the active dynamic list when an external force is applied, and it removes those objects when they return to rest.

The objects are confined in a six-sided boundary box. If the object's position is against the boundary walls they rebounce and their velocity is computed after dampening it. Figure 5.6 shows the simulated environment.

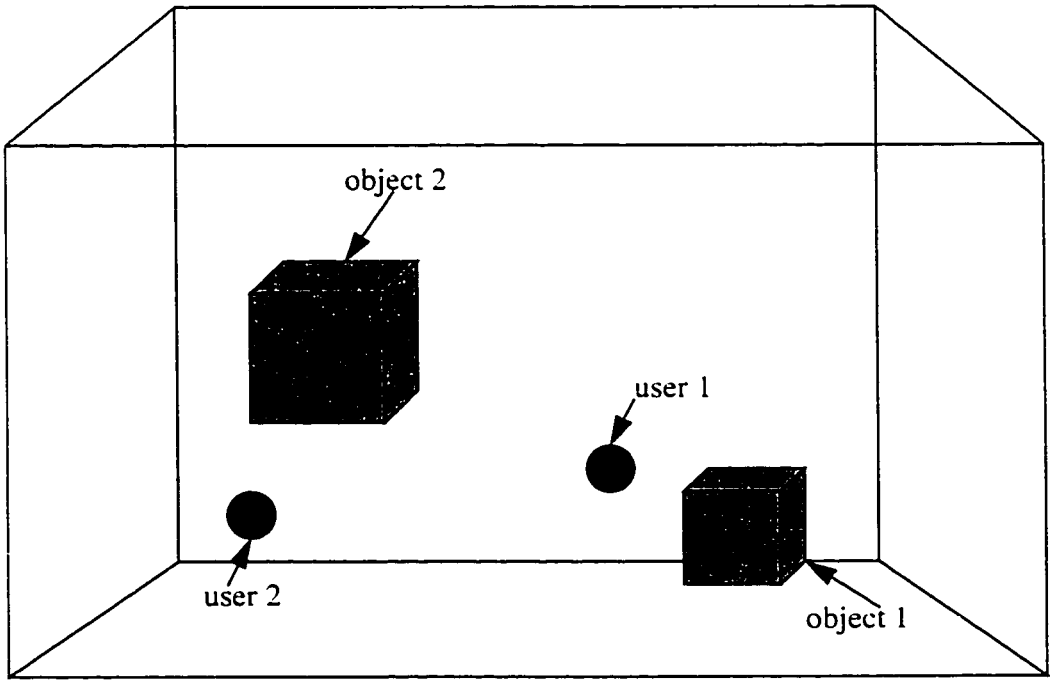

Figure 5.6. Work-space of the simulated environment

5.6 Simulation of Simple and Complex Environments

As explained in Section 4.6, two haptic environments with different haptic force calculation models were simulated. The complex model was simulated as a second order system as given below:

$$VE_{complex} = b + \frac{K_c z}{z(z-1)} \tag{5.4}$$

where b is the damping coefficient and K_c is the spring constant. The simple force calculation model was simplified as given in (4.21). The simplified model is given below:

$$VE_{simple} = b + \frac{K_s z}{z-1} \tag{5.5}$$

The constants used for the force calculation are $b = 0.4$, $K_c = 0.1$ and $K_s = 0.9$. Different types of interaction like unilateral (collision with stiff wall), bilateral (collision with movable object) and slip (moving on the rough surface) were performed using both the simulated environments. The force outputs from both the force calculation models for different types of interactions and error between them are plotted in Figures 5.7, 5.8 and 5.9. For unilateral collision as shown in Figure 5.7, it is quite evident that the force output behaves in a similar fashion. The error at the beginning of the interaction is quite noticeable while at the end of the interaction the error is reduced. The zero error corresponds to the situation when there is no interaction. The bilateral collision as shown in Figure 5.8 also has similar behavior. The maximum for simple model reaches one sample later than the complex model. The slip action performs fairly well for both models with higher initial error and lower final error.

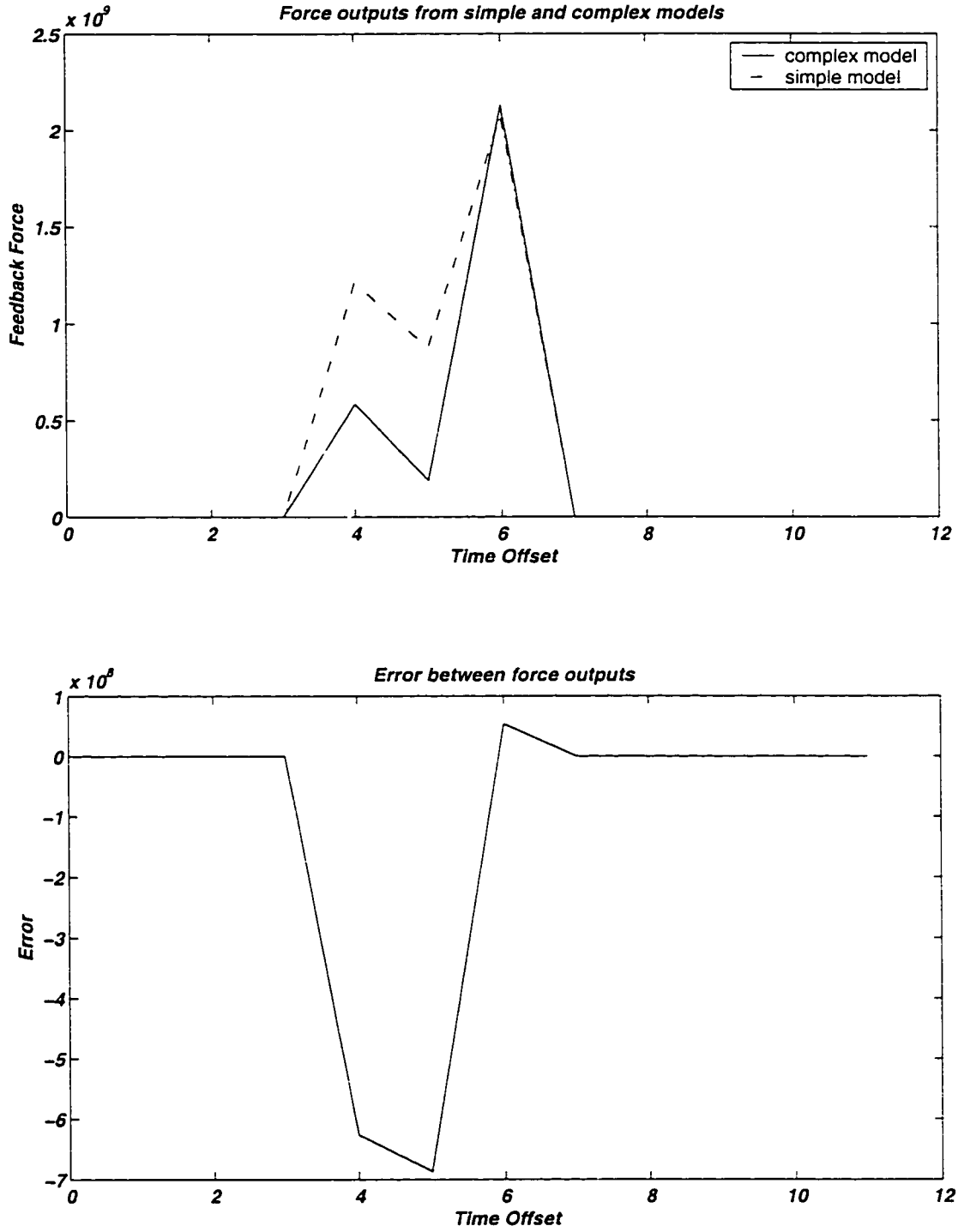

Figure 5.7. Uni-lateral collision with simple and complex models

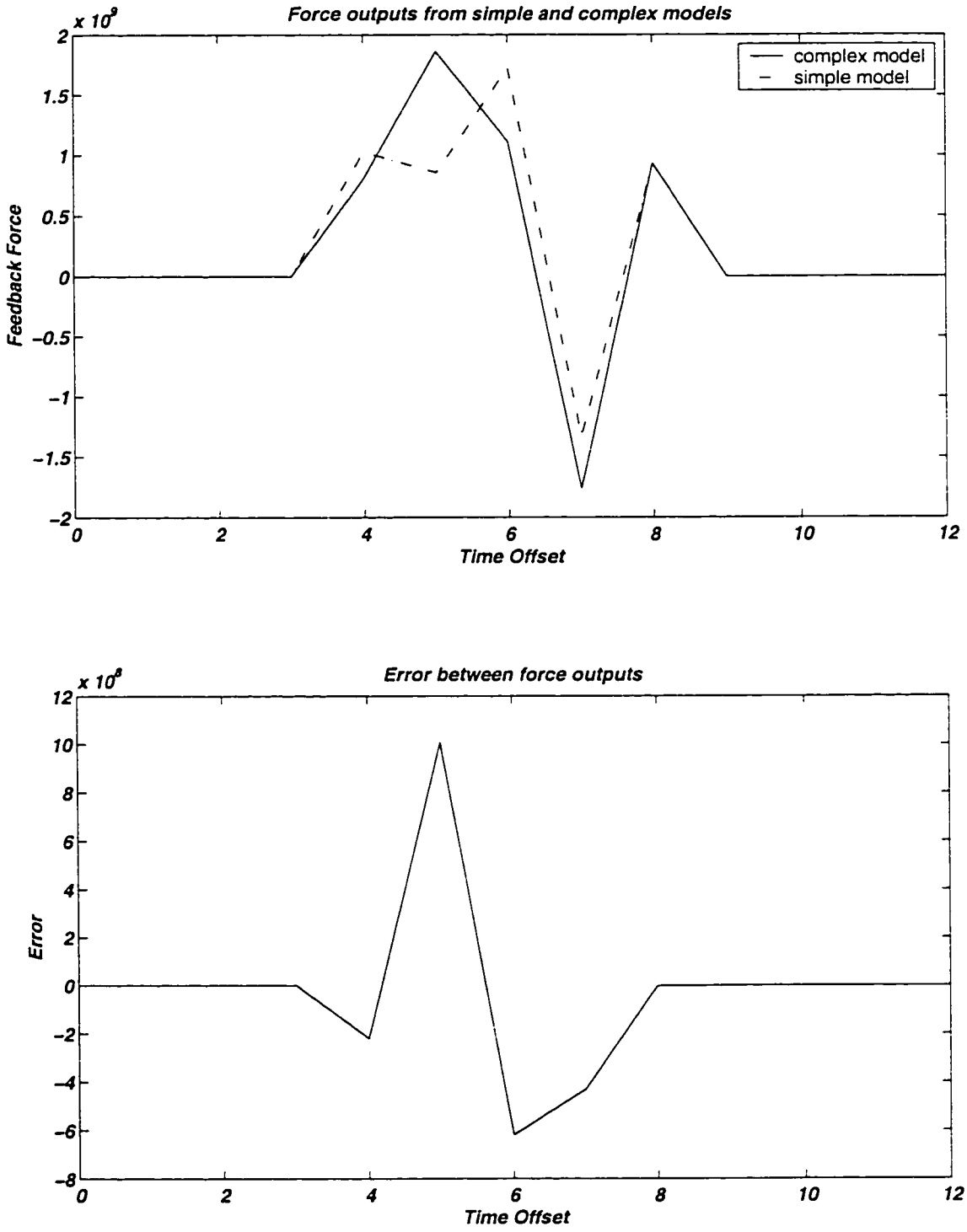

Figure 5.8. Bi-lateral collision with simple and complex models

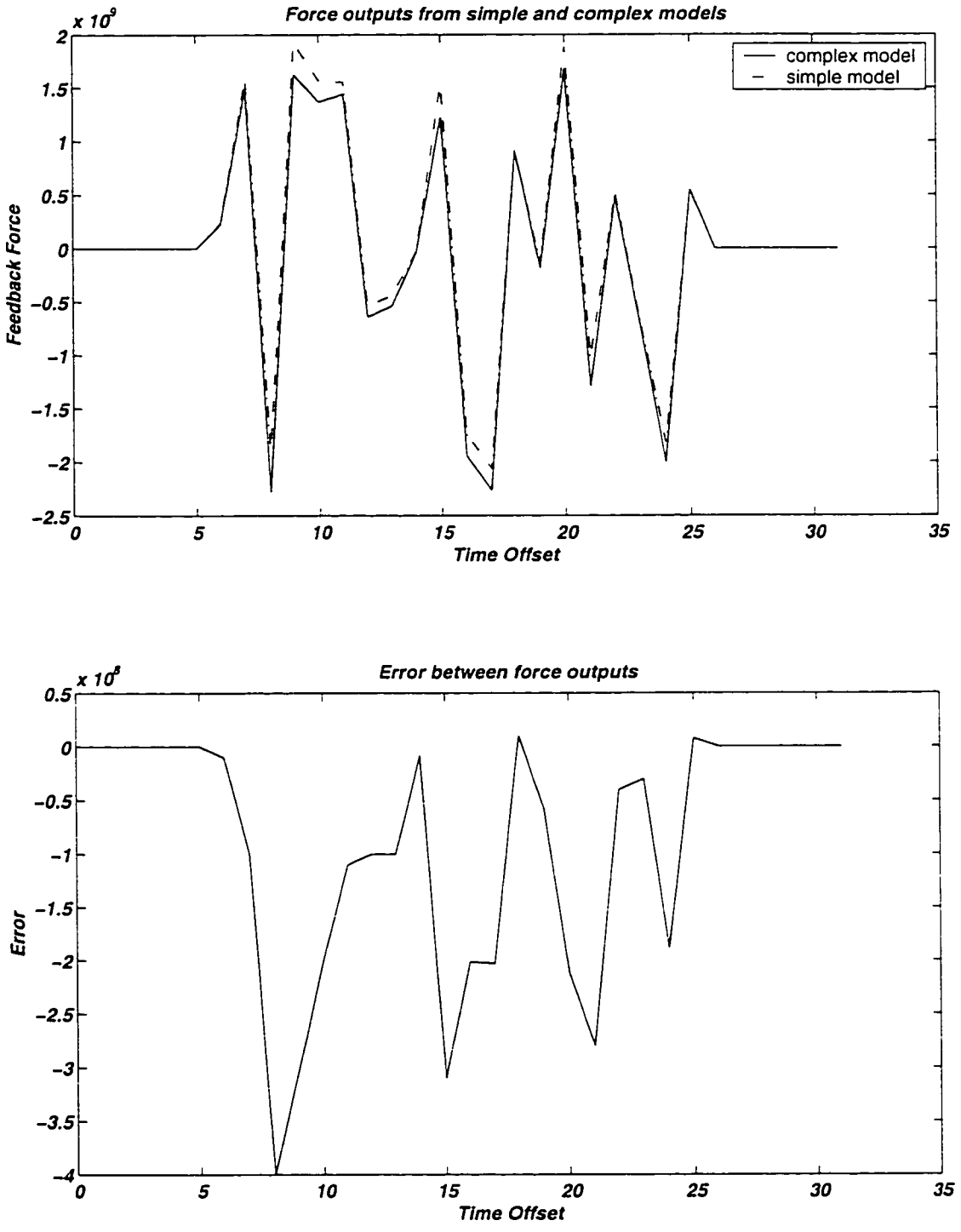

Figure 5.9. Slip with simple and complex models

5.7 Conclusion

From expressions (5.4) and (5.5) it is quite evident that the quality of the simplified model and hence the force output from the simple model depends on how the model is simplified. The complex model considered here is only second order force calculation model and the reduced model is therefore first order system. The error between the force outputs should vary widely if the complex model is of much higher order than the simplified model. Consequently, the simplification process depends on the quality of error requirement. A better simplified model always can be achieved at the expense of higher computation burden and degraded real-time performance. Another approach would be to manipulate the error by tuning the coefficients b, K_c, K_s in (5.4) and (5.5). If a threshold error is maintained in the simplification process, then for a certain period of time the coefficients can be tuned to set the error below that threshold value. However this approach is only feasible for that specific certain duration or for a certain sampled output.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

As already explained a stable haptic display become unstable in a DVE in the presence of delays over network due to loss of passivity. For expected delays a transmission line modelling approach can help to guarantee discrete time passivity. But it did not consider the unexpected delays or loss of information over network. To address this issue Dead Reckoning Algorithm in force calculation model is proposed to meet the stability and performance requirements of haptic displays in the presence of any kind of delays. For a realistic simulation a 3D graphics world is developed where two users can interact and play around with cubical objects in a confined workspace. The force-feedback is sensed through "Phantom" haptic devices at the two users' ends. The graphical and haptic simulations are maintained on two different processors and the real time interaction between them is maintained through "Worldtoolkit" (for graphical simulation) and "Ghost" (for haptic simulation) software toolkits.

Using "simple" and "complex" force calculation models, force outputs from haptic displays with different types of interaction such as uni-lateral, bi-lateral and slippage are investigated. The nature of force outputs from two models for each type of interaction matches each other leaving error between them. The similar nature of these two force

outputs justifies the initial modelling and results from the two force calculation models. During delays or loss of information the "simple" model is used for feedback force calculation and thus the haptic devices are prevented from instability or unwanted behaviors. The stability and passivity are guaranteed as long as the gain of the "simple" force calculation model is not increased significantly by violating the passivity criteria explained earlier. The order reduction technique used here does not consider the modification of gain in the VE model. So a passive haptic display in a "complex" VE should also be passive in a "simple" VE. If any kind of tuning of the coefficients of the simplified model is obvious then the post tuning coefficients must be tested for passivity criteria.

The choice of model order reduction algorithm should limit the magnitude of error. The amount of threshold error, the error at which switching between the force calculation models happens, should define the number of model order to be reduced. For a higher number of model order reduction the real-time performance of force computation may significantly be improved after sacrificing the quality of force-feedback realization. In contrast, a lower number of model order reduction yields better force-feedback realization but degrades the real-time performance. Thus the choice of threshold error is a critical issue in selecting the model order reduction algorithm. For a realistic immersion the maximum error should not get such a value that the feedback-force yields different sense of touch than the object really is. Thus the choice of threshold error should depend on how much the quality of force realization can be sacrificed without hampering objects' haptic properties.

The force computation is completely local to the system and it does not depend on the network data except when the objects' properties are updated. Any kind of communication model should not effect the behavior of this algorithm as long as the user's VE are updated periodically. Another interesting application of this algorithm is that it is suitable for any kind of VE and haptic display combinations.

6.2 Future Work

The VR technology is not a stand alone field of research interest. The development of higher quality graphics accelerator, processor, high speed network and data communication, specific protocol for 3D graphics world, specific real time algorithms and even the development of 3D modelling language effect this next generation technology. The core problem lies in the need of quality of realistic sense of immersion in the 3D graphics world.

A distributed environment with multiple users over network can be simulated. When the number of users grows the traffic of information transfer can be reduced by applying dead reckoning algorithm both for positional updates and for force calculations. There should be information slots in packets regarding the properties of the objects like coefficient of friction, coefficient of restitution, damping factor and the order of the model for force calculation. Deformable objects are of much higher order and complex in force calculation and the dead reckoning algorithm can play a significant role in transferring the information for this kind of objects. Further investigation is required on how dead reckoning algorithm behaves in objects with different haptic properties along their surfaces. Another research issue is to study the impact of the model reduction on the real-time performance of the VE application. Dead reckoning algorithm can play a significant role in manipulating hazardous materials and performing time critical robotic manipulation. Due to loss of information unstable manipulation can be protected at the expense of precision.

Bibliography

[1] S. Singhal, M. Zyda, *"Networked Virtual Environments, Design and Implementation"*, ACM Press, SIGGRAPH Series, 1999.

[2] M.R. Macedonia, M.J. Zyda. *"A Taxonomy of Networked Virtual Environments"*, In Workshop on Networked Realities, Boston, MA, October 1995.

[3] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, S. Zeswitz, *"NPSNET: A Network Software Architecture For Large Scale Virtual Environments"*, *Presence*, vol. 3, No. 4, Fall 1994.

[4] Institute of Electrical and Electronics Engineers. *"IEEE Standard for Distributed Interactive Simulation-Application Protocols"*, IEEE Std. 1278.1-1995, Piscataway, NJ: IEEE Standards Press, September 1995.

[5] M.R. Macedonia, *"A Network Software Architecture for Large Scale Virtual Environments"*, Ph.D dissertation, Naval Postgraduate School, Monterey, CA. June 1995.

[6] D. Cohen, *"NG-DIS-PDU: The Next Generation of DIS-PDU (IEEE-1278) (94-87)"*, In Proceedings of the 10th Workshop on Standards for Distributed Interactive Simulations, 735-742, Orlando, FL, March 1994.

[7] T.A. Frankhouser, *"RING: A Client-server System for Multi-user Virtual Environments"*, In Proceedings of the 1995 SIGGRAPH Symposium on Interactive 3D Graphics, 85-92, ACM SIGGRAPH, Monterey, CA, April 1995.

[8] M. R. Macedonia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, P.T. Barham, *"Exploiting Reality with Multicast Groups"*, IEEE Computer Graphics and Applications, 38-45, September 1995.

[9] D.J. Van Hook, J. O. Calvin, D.C. Miller, *"A Protocol Independent Compression Algorithm (PICA). Advanced Distributed Simulation Memorandum 20PM-ADS-005"*, MIT Lincoln Laboratories, Lexington, MA, April 1994.

[10] J.O. Calvin, D.C. Miller, J. Sagger, et al., *"Application Control Techniques System Architecture"*, Technical Report RITN-1001-00, MIT Lincoln Laboratories, Lexington, MA, February 1995.

[11] S.K. Singhal, *"Effective Remote Modeling in Large-scale Distributed Simulation and Visualization Environments"*, Ph.D dissertation, Dept. of Computer Science,

Stanford University, Stanford, CA, August 1996.

[12] C. Greenhalgh, S. Benford, *"Virtual Reality Teleconferencing: Implementation and Experience"*, In Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'95), Stockholm, September 1995.

[13] C. Carlsson, O. Hagsand, *"DIVE- A Platform for Multi-User Virtual Environments"*, Computer and Graphics, vol. 17, No. 6 (1993), pp. 663-9.

[14] C. Greenhalgh, S. Benford, *"MASSIVE: A Collaborative Virtual Environment for Teleconferencing"*, ACM TOCHI, vol. 2, No. 3, September 1995.

[15] M. Capps, S. Teller, *"Communication Visibility in Shared Virtual Worlds"*, In Proceedings of the sixth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 187-192, IEEE Computer Society, Cambridge, MA, June 1997.

[16] P. Curtis, D.A. Nicholas, *"MUDs Grow Up: Social Virtual Reality in the Real World"*, 1994, *ftp://ftp.parc.xerox.com/pub/MOO/papers/MUDsGrowUp.ps*.

[17] J. W. Barrus, R. C. Waters, D. B. Anderson, *"Locales and Beacons: Efficient and Precise Support for Large Multi-user Virtual Environments"*, In Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS), 204-213, IEEE Neural Network Council, Santa Clara, CA, March 1996.

[18] J. W. Barrus, R. C. Waters, D. B. Anderson, *"Locales: Supporting Large Multi-user Virtual Environments"*, IEEE Computer Graphics and Applications, 16(6):50-57, November 1996.

[19] H. Abrams, K. Watsen, M. Zyda, *"Three tiered interest management for large-scale virtual environments"*, In Proceedings of Virtual Reality Systems and Technology (VRST) 1998, ACM, Taipei, Taiwan, November 1998.

[20] S.K. Singhal, D.R. Cheriton, *"Using Projection Aggregations to Support Scalability in Distributed Simulation"*, In Proceedings of the 16th International Conference on Distributed Computing System (ICDCS), 196-206, IEEE Computer Society, Hong Kong, May 1996.

[21] S. Milner, *"STOW real-time communications architecture: Requirements, Approach and Rationale"*, Presentation to STOW Technical Evaluation Team, May 1995.

[22] G.D. Kessler, L.F. Hodges, *"A Network Communication Protocol for Distributed Virtual Environment Systems"*, In Proceedings of the 1996 IEEE Virtual Reality Annual International Symposium (VRAIS), 214-221, IEEE Neural Network Council, San Jose, CA, April 1996.

[23] M.D. Ryan, P.M. Sharkley, *"Casual Volumes in Distributed Virtual Reality"*, In Proceedings of the 1997 IEEE International Conference on Systems, Man and Cybernatics, 1067-1072, Orlando, October 1997.

[24] N. Farcet, P. Torguet, *"Space-scale Structure for Information Rejection in Large-scale Distributed Virtual Environments"*, In Proceedings of the 1998 IEEE Virtual Reality Annual International Symposium (VRAIS), 276-283, IEEE Neural Networks Council, Atlanta, GA, March 1998.

[25] D.J. Van Hook, D.P. Cebula, S.J. Rak, C.J. Chiang, P.N. DiCaprio, J.O. Calvin, *"Performance of STOW RITN Application Control techniques"*, In Proceedings of the 14th Workshop on Standards for the Interoperability of Distributed Simulations, Orlando, March 1996.

[26] C. Borrman, J.Ott, H.C. Gehrcke, T. Kerschat, N. Seifert: *" MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport"*, International Conference on Computer Communications and Networks (ICCCN 94), 1994.

[27] B. Whetten, T. Montogomery, S. Kaplan, *"A High Performance Totally Ordered Multicast Protocol, Theory and Practice in Distributed Systems"*, Springer Verlag LCNS 938.

[28] S. Jacobsen, V. Liu, C.S. McCanne, L. Zhang, *"A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing"*, Scalable Reliable Multicast (SRM), ACM SIGCOMM'95.

[29] R. C. Waters, D. B. Anderson, D. L. Schwenke, *"Design of the Interactive Sharing Transfer Protocol"*, WET ICE'97 -- IEEE Sixth Workshops on Enabling Technologies for Collaborative Enterprises: Infrastructure for Collaborative Enterprises, IEEE Computer Society Press, Los Alamitos, CA, June 1997, pp. 140-147.

[30] D. Brutzman, M. Zyda, W. Kent, M. Macedonia, *"Virtual Reality Transfer Protocol (vrtp) Design Rationale"*, Workshops on Enabling Technology: Infrastructure for

Collaborative Enterprises (WET ICE): Sharing a Distributed Virtual Reality, Massachusetts Institute of Technology, Cambridge Massachusetts, June 18-20, 1997.

[31] D. Brutzman, *"The Virtual Reality Modeling Language and Java"*, Communications of the ACM, vol. 41, no.6, June 1998, pp. 57-64.

[32] World Wide Web Consortium (W3C), overview Web page, October 1996, http://www.w3.org/.

[33] National Center for Supercomputer Applications (NCSA), *"What is Mosaic?"* Web page, October 1996, http://www.ncsa.uiuc.edu/Indices/Discover/WhatIsMosaic.html

[34] M.R. Macedonia, D. Brutzman, P. Donald, *"MBone Provides Audio and Video Across the Internet"*, IEEE Computer, vol.27, no.4, April 1994, pp. 30-36.

[35] D. Mills, *"Simple Network Time Protocol (SNTP)"*, Request for Comments (RFC) 1769, March 1995, ftp://ds.internic.net/rfc/rfc1769.txt

[36] W. Broll, D. Schick, *"DWTP- a basis for networked VR on the Internet"*, In the Proceedings of the Electronic Imaging'98, San Jose, January 24-30, 1998.

[37] D. Brutzman, M. Zyda, M. Macedonia, *"Cyberspace Backbone (CBone) Design Rationale"*, Fifteenth Distributed Interactive Simulation (DIS) Standards Workshop, Orlando, Florida, September 16-20, 1996, paper 96-15-099, pp.657-665.

[38] M. Wloka, *"Lag in multiprocessor VR"*, PRESENCE: Teleoperators and Virtual Environments 4(1): 50-63, Winter 1995.

[39] H. El-Rewini, T.G. Lewis, *"Distributed and Parallel Computing"*, Greenwich, CT: Manning Press, 1998.

[40] F. B. Schneider, *"On Concurrent Programming"*, New York: Springer-Verlag, 1997.

[41] G. Robertson, S.K. Card, J. Mackinlay, *"The Cognitive Coprocessor Architecture for Interactive User Interfaces"*, In Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology, 10-18, ACM SIGGRAPH, Williamsburgh, VA, November 1989.

[42] J. P. Wilson, R.J. Kline-Schoder, M. A. Kenton, N. Hogan, *"Algorithms for Network-Based Force Feedback"*, Fourth PHANTOM Users Group Workshop (PUG99).

[43] J. Rohlf, J. Helman, *"IRIS Performer: A high-performance multiprocessor toolkit for real-time 3D"*, In SIGGRAPH 1994 Conference Proceedings, 381-394, ACM SIG-

GRAPH, Orlando, FL, August 1994.

[44] T. Funkhouser, S. Teller, C. Sequin, D. Khorramabadi, *"The UC Berkeley system for interactive visualization of large architectural models"*, PRESENCE: Teleoperators and Virtual Environments 5(1):13-44, Winter 1996.

[45] M. Mine, H. Weber, *"Large Models for Virtual Environments: A review of work by the architectural walkthrough project at UNC"*, PRESENCE: Teleoperators and Virtual Environments 5(1):136-145, Winter 1996.

[46] J. Clark, *"Hierarchical Geometric Models for Visible Surface Algorithms"*, Communication of the ACM 19(10):547-554, October 1976.

[47] M. Cutkosky, R. Howe, *"Human Grasp Choice and Robotic Grasp Analysis"*, Dextrous Robot Hands, S. Venkataraman and T. Iberall Eds., pp. 5-31, Springer Verlag, 1990.

[48] C. Sherrick, J. Craig, *"The Psychophysics of touch"*, in Tactual Perception- A Source Book, edited by W. Schiff and E. Foulke, Cambridge University Press, pp. 55-81, 1982.

[49] K. Shimoga, *"Finger Force and Touch Feedback Issues in Dextrous Telemanipulation"*, Proceedings of NASA-CIRSSE International Conference on Intelligent Robotics Systems for Space Exploration, Troy, NY, September 1992.

[50] G. Burdea, P. Coiffet, *"Virtual Reality Technology"*, John Wiley & Sons, Inc., 1994.

[51] R. Stone, *"Virtual Reality Tutorial"*, MICAD Conference, Paris, France, 200 pp., 1992.

[52] N. Patrick, *"Design, Construction, and Testing of a Fingertip Tactile Display for Interaction with Virtual and Remote Environments"*, Masters Thesis, Department of Mechanical Engineering, MIT, August 1990.

[53] K. Tanie, T. Kotoku, *"Force Display Algorithms for Virtual Environments"*, IEEE Workshop on Force Display in Virtual Environments and Application to Robotic Teleoperation, pp. 60-78, Atlanta, GA, May 1993.

[54] B. Schmult, R. Jebens, *"A High Performance Force Feedback Joystick"*, Virtual Reality Systems '93 Conference, pp.123-129, New York City, March 1993.

[55] H. Iwata, *"Pen-based Haptic Virtual Environment"*, Proceedings of IEEE Virtual Reality Annual International Symposium, Seattle, WA, pp. 287-292, September

1993.

[56] G. Burdea, J. Zhuang, E. Roskos, D. Silver, N. Langrana, *"A Portable Dextrous Master with Force Feedback"*, PRESENCE: Teleoperators and Virtual Environments, Vol. 1, No.1, pp. 18-27, March 1992.

[57] M. Bouzit, P. Richard, P. Coiffet, *"LRP Dextrous Hand Master Control System"*, Technical Report, Laboratiore de Robotique de Paris, 21 pp., January 1993.

[58] P. Coiffet, M. Bouzit, G. Burdea, *"The LRP Dextrous Hand Master"*, Proceedings of VR Systems Fall 93 Conference, New York City, October 1993.

[59] J. Kramer, *"Force Feedback and Texture Simulating Interface Device"*, US Patent 5,184,319, February 1993.

[60] C. Carignan, K. Cleary, *"Closed-Loop Force Control for Haptic Simulation of Virtual Environments"*, Haptics-e, The Electronic Journal of Haptics Research (http://www.haptics-e.org), Vol. 2, No. 2, February 2000.

[61] R. Baumann, R. Clavel, *"Haptic Interface for Virtual Reality Based Minimally Invasive Surgery Simulation"*, In Proceedings of IEEE International Conference on robotics and Automation, pp. 381-386, 1998.

[62] D. O. Popa, S.K. Singh, *"Creating Realistic Force Sensations in a Virtual Environment: Experimental System, Fundamental Issues and Results"*, In Proceedings of IEEE International Conference on Robotics and Automation, pp. 59-64, 1998.

[63] K. Cleary, C. Lathan, C. Carignan, *"Simulator/planner for CT Directed Needle Biopsy of the Spine"*, In Surgical-Assist Systems, pp. 218-224, SPIE, 1998.

[64] T. Yoshikawa, Y. Yokokohji, T. Matsumoto, X. Z. Zheng, *"Display of Feel for the Manipulation of Dynamic Virtual Objects"*, Transaction ASME Journal on Dynamic System Measurement and Control, vol. 117, no. 4, pp. 554-558, 1995.

[65] T. H. Massie, J. K. Salisbury, *"The Phantom Haptic Interface: A Device for Probing Virtual Objects"*, In Proc. of ASME International Mechanical Engineering Congress Exhibition, Chicago, IL, 1994, pp. 295-302.

[66] V. Hayward, J. Choksi, G. Lanvin, C. Ramstein, *"Design and Multi-objective Optimization of a Linkage for a Haptic Interface"*, Advances in Robot Kinematics and Computational Geometry, Boston, MA:Kluwer, 1994, pp. 352-359.

[67] P. Buttolo and B. Hannaford, *"Pen Based Force Display for Precision Manipulation of Virtual Environments"*, Proc. IEEE Virtual Reality Annu. Int. Symp., Raleigh, NC, 1995, pp. 217-225.

[68] Y. Yokokohji, R. L. Hollis, T. Kanade, *"What You See is What You can Feel - Development of a Visual/Haptic Interface to Virtual Environment"*, In Proceedings of IEEE Virtual Reality Annual International Symposium, Los Alamitos, CA, 1996, pp. 46-53.

[69] C. L. Clover, G. R. Luecke, J. J. Troy, W. A. McNeely, *"Dynamic Simulations of Virtual Mechanisms with Haptic Feedback Using Industrial Robotic Equipment"*, Proceedings of IEEE International Conference on Robotics and Automation, Albuquerque, NM, 1997, pp. 3205-3210.

[70] G. D. Glosser, W. S. Newman, *"The Implementation of a Natural Admittance Controller on an Industrial Robot"*, In Proceedings of IEEE Conference on Robotics and Automation, pp. 1209-1215, 1994.

[71] R. J. Adams and B. Hannaford, *"Stable Haptic Interaction with Virtual Environments"*, IEEE Trans. on Rob. and Automat., vol. 15, no. 3, pp. 465-474, June 1999.

[72] R. J. Adams, M.R. Moreyra and B. Hannaford, *"Stability and Performance of Haptic Displays: Theory and Experiments"*, in Proc. ASME Int. Mech. Eng. Congr. Exh., Anaheim, CA, 1998, pp. 227-234.

[73] J. E. Colgate, J. M . Brown, *"Factors Affecting the Z-width of a Haptic Display"*, Proc. IEEE Int. Conf. Robot. and Automat., Los Alamitos, CA, 1994, pp. 3205-3210.

[74] Y. Adachi, T. Kumano, K. Ogino, *"Intermediate Representation for Stiff Virtual Objects"*, In Proceedings of IEEE Virtual Reality Annual International Symposium, pp. 203-210, March 1995.

[75] M. Minsky, M. Ouh-Young, O. Steele, F.P. Brooks and M. Behensky, *"Feeling and Seeing Issues in Force Displays"*, Computer Graphics, vol. 24, no. 2, pp. 235- 243, 1990.

[76] W.R. Mark, S.C. Randolph, M. Flinch, J. M. Van Verth, R. M. Taylor, *"Adding Force Feedback to Graphics Systems: Issues and Solutions"*, In Computer Graphics Proceedings, Annual Conference, ACM SIGGRAPH, pp. 447-452, 1996.

[77] M. C. Surles, *"An Algorithm with Linear Complexity for Interactive, Physically-based Modeling of Large Proteins"*, Proc. SIGGRAPH 92 ,Chicago, Illinois, July 26-31, 1992.

[78] M. Lin, S. Gottschalk, *"Collision Detection Between Geometric Models: A Survey"*, In Proceedings of the IMA Conference on Mathematics of Surfaces, 37-56, 1998.

[79] A. A. G. Requicha, J. R. Rossignac, *"Solid Modeling and Beyond"*, IEEE Computer Graphics and Applications, pp. 31-44, September 1992.

[80] C. M. Hoffman, *"Geometric and Solid Modeling"*, Morgan Kaufmann, San Mateo, California, 1989.

[81] S. Cameron, *"Approximation Hierarchies and S-bounds"*, Symposium on Solid Modeling Foundations and CAD/CAM Applications, pp. 129-137, Austin, TX, 1991.

[82] J. Keyser, S. Krishnan, D. Manosha, *"Efficient and Accurate B-rep Generation of Low Degree Sculptured Solids Using Exact Arithmetic"*, Symposium on Solid MOdeling, ACM/SIGGRAPH, pp. 42-55, 1997.

[83] T. W. Sederberg, *"Techniques for Cubic Algebric Surfaces"*, IEEE Computer Graphics and Applications, pp. 14-25, July 1990.

[84] J. Bloomenthal, B. Wyvill, *"Interactive Techniques for Implicit Modeling"*, In Computer Graphics (1990 Symposium on Interactive 3D Graphics), vol. 24, pp. 109-116, March 1990.

[85] R. T. Farouki, C. A. Neff, M. O'Connor, *"Automatic Parsing of Degenerate Quadric-surface Intersections"*, ACM Transaction on Graphics, 8:174-203, 1989.

[86] C. Shene, J. Johnstone, *"On the Planar Intersection of Natural Quadrics"*, Proceedings of ACM Solid Modeling, pp. 234-244, 1991.

[87] J. M. Lane, R. F. Riesenfeld, *"A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces"*, IEEE Transaction on Pattern Analysis and Machine Intelligence, 2(1):150-159, 1980.

[88] G. Farin, *"Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide"*, Academic Press Inc., 1993.

[89] M. C. Lin, *"Efficient Collision Detection for Animation and Robotics"*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of Califor-

nia, Berkeley, December 1993.

[90] M. C. Lin, D. Manocha, *"Fast Interference Detection Between Geometric Models"*, The Visual Computer, 11(10):542-561, 1995.

[91] B. Mirtich, J. F. Canny, *"Impulse-based Simulation of Rigid Bodies"*, In Proc. of ACM Interactive 3D Graphics, Monterey, CA, 1995.

[92] M. C. Lin, J. F. Canny, *"Efficient Algorithms for Incremental Distance Computation"*, In IEEE Conference on Robotics and Automation, pp. 1008-1014, 1991.

[93] H. Samet, *"Spatial Data Structures: Quadtree, Octrees and Other Hierarchical Methods"*, Addison Wesley, 1989.

[94] S. Quinlan, *"Efficient Distance Computation Between Non-convex Objects"*, In Proceedings of International Conference on Robotics and Automation, pp. 3324-3329, 1994.

[95] B. Naylor, J. Amanatides, W. Thibault, *"Merging bsp trees yield polyhedral modeling results"*, In Proc. of ACM SIGGRAPH, pp. 115-124, 1990.

[96] S. Cameron, *"Collision Detection by Four-dimensional intersection testing"*, Proceedings of International Conference on Robotics and Automation, pp. 291-302, 1990.

[97] S. Gottschalk, M. Lin, D. Manocha, *"Obb-tree: A Hierarchical Structure for Rapid Interference Detection"*, In Proc. of ACM SIGGRAPH'96, pp. 171-180, 1996.

[98] T. Duff, *"Interval Arithmetic and Recursive Subdivision for Implicit Functions and Constructive Solid Geometry"*, ACM Computer Graphics, 26(2):131-139, 1992.

[99] J. Snyder, *"Interval Arithmetic for Computer Graphics"*, In Proceedings of ACM SIGGRAPH, pp. 121-130, 1992.

[100] B. V. Herzen, A. H. Barr, H. R. Zatz, *"Geometric Collision for Time-dependent Parametric Surfaces"*, Computer Graphics, 24(4):39-48, 1990.

[101] J. Snyder and et. al., *"Interval Methods for Multi-point Collisions Between Time Dependent Curved Surfaces"*, In Proceedings of ACM SIGGRAPH, pp. 321-334, 1993.

[102] M. J. Pratt, *"Surface/surface Intersection Problems"*, In J. A. Gregory, editor, The Mathematics of Surfaces II, pp. 117-142, Oxford, 1986, Claredon Press.

[103] S. Krishnan, D. Manocha, *"An Efficient Surface Intersection Algorithm Based on the*

Lower Dimensional Formulation", ACM Transaction on Graphics, 16(1):74-106, 1997.

[104]M. C. Lin, D. Manocha, "*Efficient Contact Determination Between Geometric Models*", International Journal of Computational Geometry and Applications, 7(1):123-151, 1997.

[105]M. C. Lin, D. Manocha, J. Cohen, S. Gottschalk, "*Collision Detection: Algorithms and Applications*", In Proceedings of the 1998 Conference on Algorithms for Robotics Motion and Manipulation, 129-142, 1998.

[106]M. H. Overmars, "*Point Location in Fast Subdivisions*", Inform. Proc. Lett., 44:261-265, 1992.

[107]J. Dahman, R. Weatherly, F. Kuhl, "*Creating Computer Simulation Systems: An Introduction to the High Level Architecture*", Upper Saddle River, NJ: Prentice Hall, 1999.

[108]P. Buttolo, R. Oboe, B. Hannaford, B. McNeely: "*Force Feedback in Shared Virtual Simul tions*", Proceedings of MICAD, France, 1996.

[109]B. Hannaford, "*A Design Framework for Teleoprators with Kinesthetic Feedback*", IEEE Trans. Robotics and Automation, vol. 5, no. 4, 1989, pp. 426-434.

[110]R.J. Anderson, M.W. Spong, "*Asymptotic Stability for Force Reflecting Teleoperators with Time Delay*", Int. Journal of Robotics Research, vol. 11, no. 2, 1992, pp. 135-49.

[111]J.E. Colgate, "*Robust Impedence Shaping Telemanipulation*", IEEE Trans. Robotics and Automation, vol. 9, no. 4, 1993, pp. 374-384.

[112]J.E. Colgate, et. al., "*Implementation of Stiff Virtual Walls in Force-Reflecting Interfaces*", IEEE Virtual Reality Annual Int. Symposium, Seattle, WA, 1993, pp. 202-8.

[113]C.B. Zilles, J.K. Salisbury, "*A Constraint-based God-object Method for Haptic Display*", Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Pittsburgh, PA, 1995,pp. 146-151.

[114]R.J Adams, B. Hannaford, "*A Two-port Framework for the Design of Unconditionally Stable Haptic Interfaces*", Proc. IROS, Anaheim, CA,1998.

[115]W. Islam, K. Khorasani, A. J. Al-Khalili, S. Tafazoli, "*Development of Algorithms for Distributed and Collaborative Force Feedback using Haptic Interfaces in Virtual*

Environments" in conference DASIA' 2000 organized by EuroSpace, May 2000, Montreal.

[116] S. E. Salcudean and T.D. Vlaar, "*On the Emulation of Stiff Walls and Static Friction with a Magnetically Levited Input/ Output Device*", Trans. ASME J. Dyn. Syst., Meas., Contr., vol. 119, no. 1, pp. 127-132, 1997.

[117] J. E. Colgate, G. G. Schenkel, "*Passivity of a Class of Sampled-Data System: Application to Haptic Interfaces*", American Control Conference, Baltimore, 1994.

[118] K.S. Narendra, J.H. Taylor, "*Frequency Domain Criteria for Absolute Stability*", Academic Press, NY, 1973.

[119] Y.Z. Tsypkin, "*On the Stability in the Large of Nonlinear Sampled Data Systems*", Dokl. Akad. Nauk USSR, vol. 145, pp. 52-55, 1962.

[120] D. Mitra, "*The Absolute Stability of High-Order Discrete-Time Systems Utilizing the Saturation Nonlinearity*", IEEE Transaction on Circuits and Systems, vol. CAS-25, no. 6, pp. 365-371, 1978.

[121] J.C. Tsai, "*Toward Guaranteed Stability in the Haptic Display of Virtual Environments*", Phd. Dissertation, Dept. of Mechanical Engg., Northwestern University, Dec., 1996.

[122] J.E. Colgate, M.C. Stanley and J.M. Brown, "*Issues in the Haptic Display of Tool Use*", Dept. of Mechanical Engg., Northwestern University, 1995.

[123] D. Karnopp, R. Rosenberg, "*System Dynamics: A Unified Approach*", John Wiley and Sons, New York, 1975.

[124] P. Penfield Jr., R. Spencer, S. Duinker, "*Tellegen's Theorem and Electrical Networks*", M.I.T. Press, Cambridge, Massachussetts, 1970.

[125] H. M. Paynter, "*Analysis and Design of Engineering Systems*", The M.I.T. Press, Cambridge, Massachusetts, 1960.

[126] W. Cai, F. B. S. Lee, L. Chen, "*An Auto-Adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation*", Proceedings of the Thirteenth Workshop on Parallel and Distributed Simulation, May 1-4, 1999, Atlanta, ACM, pp: 82-89.

[127] K. C. Lin, "*Dead Reckoning and Distributed Interactive Simulation*", In Proc. of SPIE Conference (AeroSense' 95), Orlando, Florida, April 1995.

[128] "*GHOST SDK, Programmer's Guide Version 3.0*", SensAble Technologies, 1996-2000.

[129] "*WorldToolkit, Reference Manual Release 9.0*", Engineering Animation Inc., 1991-1999.