# Virtual Reality

## Annual International Symposium '95



*March 11–15, 1995*
*Research Triangle Park, North Carolina*



**Sponsored by**
**IEEE Computer Society Technical Committee on Computer Graphics**
**IEEE Neural Networks Council Technical Committee on Virtual Reality**

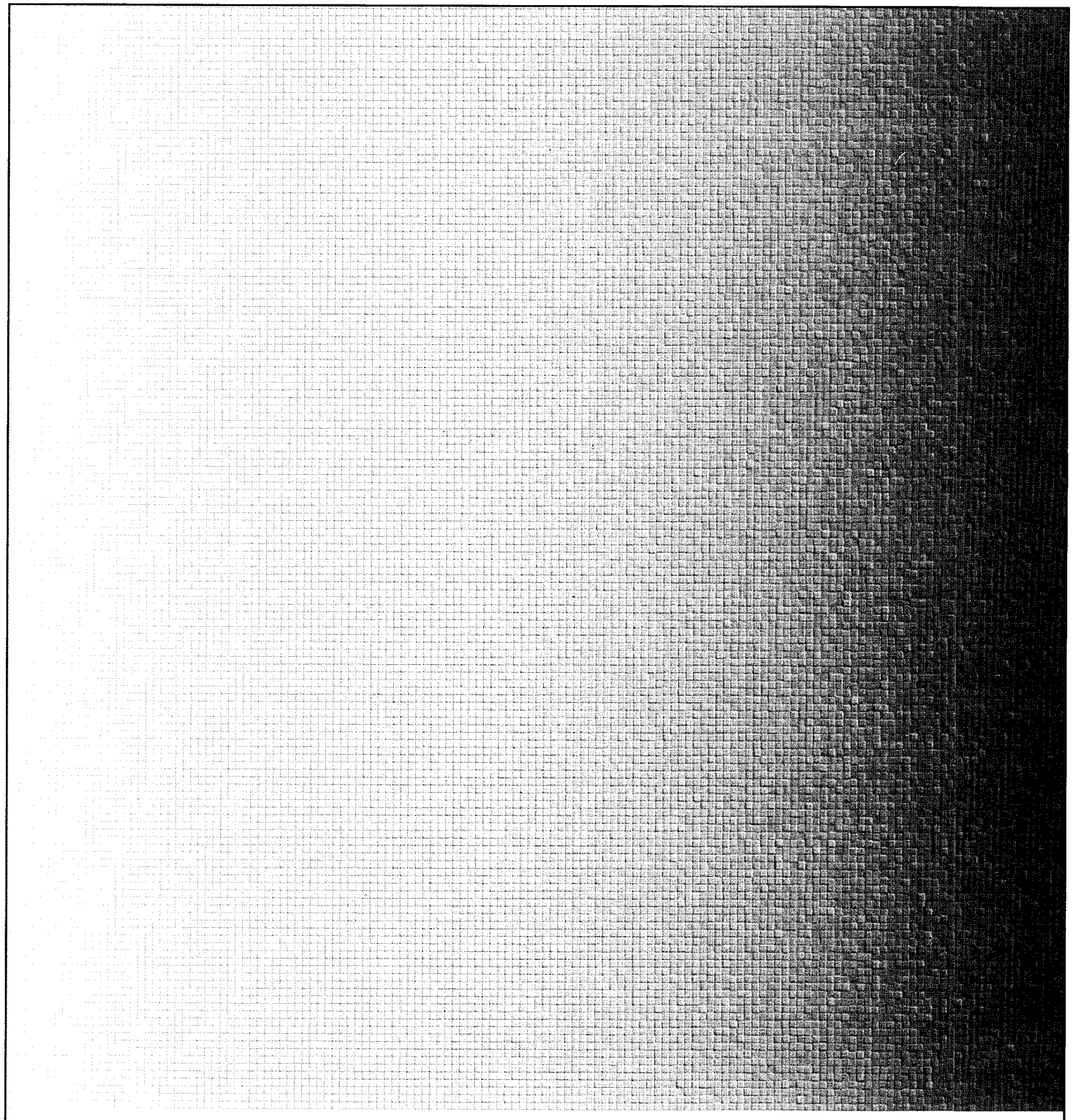
IEEE Computer Society Press

The Institute of Electrical and Electronics Engineers, Inc.

Proceedings

# Virtual Reality Annual International Symposium'95

N00014-95-1-0825



Proceedings

# Virtual Reality Annual International Symposium'95

March 11 - 15, 1995

Research Triangle Park, North Carolina

*Sponsored by*

**IEEE Computer Society Technical Committee on Computer Graphics**

**IEEE Neural Networks Council Technical Committee on Virtual Reality**

IEEE Computer Society Press
Los Alamitos, California

Washington • Brussels • Tokyo

IEEE Computer Society Press
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1264



IEEE Computer Society Press Order Number PR07084
Library of Congress Number 95-75103
IEEE Catalog Number 95CH35761
ISBN 0-8186-7084-3 (paper)

*Additional copies may be ordered from:*

IEEE Computer Society Press
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1264
Tel: +1-714-821-8380
Fax: +1-714-821-4641
Email: cs.books@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: +1-908-981-1393
Fax: +1-908-981-9667

IEEE Computer Society
13, Avenue de l'Aquilon
B-1200 Brussels
BELGIUM
Tel: +32-2-770-2198
Fax: +32-2-770-8505

IEEE Computer Society
Ooshima Building
2-19-1 Minami-Aoyama
Minato-ku, Tokyo 107
JAPAN
Tel: +81-3-3408-3118
Fax: +81-3-3408-3553

Editorial production by Regina Spencer Sipple
Cover design by Joseph Daigle/Schenk-Daigle Studios
Printed in the United States of America by Braun-Brumfield, Inc.

The Institute of Electrical and Electronics Engineers, Inc.

# Table of Contents

# Foreword

## "Timothy Leary Wasn't Invited"

It seemed to me that "we" and "they" had both shown up for the first VRAIS conference, VRAIS '93, in Seattle. "We" were the "techies," the scientists and engineers who are exploring and developing virtual environments technology. "They" were the off-the-wall fringe element that VR seems to attract. "Our" manifesto is that VR is an important new human-computer interface technology with broad potential for educating and training people; helping designers design, even helping to protect people's lives. It is a fascinating area of scientific investigation, one that is inherently multidisciplinary.

"Their" fundamental belief seems to be that the key applications of VR are enhancing cosmic consciousness, replacing sexuality, and substituting for psychotropic drugs. "We" think of people such as Ivan Sutherland, Fred Brooks, Henry Fuchs, and Scott Fisher as our technical leaders and inspirations. "Their" candidates for technical leadership in VR appear to be Timothy Leary and whoever wrote the screenplay for *Lawnmower Man.*

On the first day of VRAIS '93, "they" sometimes dominated the questioning of speakers and panelists, vigorously competing with each other to see who could use the word "cyberspace" the most times in a sentence, and raising critically important issues such as whether or not scientists are sufficiently familiar with their own bodies. "Their" ranks thinned out considerably by the second day of the conference, and by the third day, the "Looney Tunes" were gone. There was no one left but us "techies." The carefully refereed, solidly technical content of VRAIS '93 had evidently driven them away in states of boredom and lack of comprehension. We were off to a good start.

One month later, there was another victory for our side: Steve Bryson and Steve Feiner organized the Research Frontiers in Virtual Reality workshop in conjunction with IEEE Visualization '93 in San Jose, CA. Once again, there was a consistent emphasis on scientific content and quality.

It was clear that, as Andy van Dam remarked at the time, VR needed an annual technical conference, but it didn't need two. Accordingly, the IEEE Neural Networks Council's Virtual Reality Technology Committee, which sponsored VRAIS '93, and the IEEE Computer Society's Technical Committee on Computer Graphics, which sponsored Frontiers, agreed to combine their efforts and produce a single, annual, VR conference. VRAIS '95 is the first VR conference co-sponsored by these two organizations. We skipped 1994 because we wanted to move the conference date to minimize potential conflicts with other meetings on human-computer interface topics. This change of conference dates apparently caused some confusion among attendees of the earlier meetings, which I regret and apologize for. Henceforth, I think you can plan on VRAIS meetings always being held around the "Ides of March."

I want to thank two sponsoring organizations, and especially their leaders, Larry Rosenblum of the TCCG and Pat Simpson of the Neural Networks Council, for their farsightedness and flexibility in agreeing to the merge, and their generous financial and moral support over the past year. A special note of thanks is also due to the Office of Naval Research and ONR scientific officer Marc Lipman, for granting substantial financial support toward the preparation and printing of this proceedings.

I asked Steve and Steve to be co-program chairs for VRAIS '95 because I was so impressed with the technical strength and quality of the program they put together for Frontiers'93. I consider this proceedings to be indisputable evidence that they have lived up to my expectations. They, and the program committee they organized, did an exemplary job of reviewing the submitted papers and selecting the best for the conference. Their work has further established VRAIS as the place for scientists and engineers working in virtual environments R&D to come to present, evaluate, and discuss the most important new technical ideas and approaches in the field. We are well on our way.

I also want to thank Judi Qualy-White, my indispensable Finance Chair for VRAIS '95; Bob Marks, the Organizations Chair; Brenda Thein, the Publications Chair who oversaw the production of these proceedings; Mary Lou Padgett, our tireless Publicity Chair; Jannick Rolland, the Local Arrangements Chair who conceived and organized the UNC and RTI lab tours; Blake Hannaford, who put together the video proceedings; Beth Wenzel, the Tutorials Chair; Karen Haines, the Exhibits Chair; Toshio Fukuda and Mel Slater, the international liaisons; and Nadine Miner, who assisted with press relations and many other tasks. You have been a great group to work with.

I hope that the conference attendees find this proceedings, and VRAIS '95 itself, to be both professionally beneficial and personally enjoyable. Welcome to VRAIS '95!

**David Mizell**
*General Chair*

# Message from the Program Co-Chairs

What a difference a year-and-a-half makes! It seems like only yesterday that we were participating in VRAIS'93, and a month later running the 1993 IEEE Symposium on Research Frontiers in Virtual Reality. These events symbolized the "coming of age" of virtual reality as an academic research discipline. As in all coming-of-age rituals, there was difficulty, uncertainty, excitement, and anticipation as to who we were and where we wanted to go. VRAIS '95 is different, firmly establishing an annual conference where state-of-the-art, high-quality research results in virtual reality are reported.

While VRAIS has matured, we feel that virtual reality as a field still has a long way to go, which, of course, continues to keep it exciting. In the foreword to the 1993 IEEE Symposium on Research Frontiers in Virtual Reality, we wrote:

> While virtual reality's vision of immersive, interactive three-dimensional environments is compelling and has attracted many adherents, few applications have left the research laboratory. There are several reasons for this. Virtual reality is difficult, requiring interdisciplinary techniques and exceptional technological performance. Beyond the obvious problems with our current technology, there are many unanswered questions about how to build useful, effective virtual reality systems and applications.

We feel that this situation has remained substantially unchanged and there is still a great deal more to be done. We are heartened, however, by the recent announcements of low-end 3D rendering, display, and tracking devices, which will make the technology accessible to many more researchers and users. This year, the conference program concentrates on several key research areas including human factors, haptic interfaces, and distributed environments, reflecting the broad range of activities in our interdisciplinary field.

The VRAIS '95 program begins with invited speaker Henry Fuchs of the University of North Carolina, Chapel Hill, who will discuss "Research Challenges in Virtual Environments: The Race Between Achievements and Expectations." We then launch into a pair of sessions on human factors, perception, and the concept of presence, highlighting the importance of studying how people experience the virtual worlds that we are building. The first of these sessions is on human factors in general. After lunch, we examine perception and presence, covering the ways in which users adapt to the visual and auditory displays found in today's virtual reality systems. We end the first day with a session on calibration and registration for displays and trackers that addresses technical solutions to some of the problems of distortion and misalignment between the real and virtual worlds caused by current hardware.

We start the second day of the program with an invited talk by Bowen Loftin of the University of Houston and NASA Johnson Space Center on "The Use of Virtual Environments for Training the Hubble Space Telescope Flight Team." Next, we examine haptic interfaces, first in a technical session with papers describing various systems and techniques for force feedback, followed after lunch by a panel that surveys issues in haptic displays. We end the day with a session on techniques for creating virtual worlds, including the use of computer vision, animation of virtual humans, and collision detection.

Our third day begins with a session on tools for virtual reality, including novel approaches to designing displays, trackers, and software simulators. The rest of the day is devoted to an in-depth treatment of distributed virtual worlds. The last morning session examines distributed infrastructure, presenting the approaches implemented by three different research groups. After lunch, we hear about applications of distributed virtual reality for collaborative work, training, and concurrent engineering. VRAIS '95 concludes with a panel discussion on networked virtual environments.

This high-quality program would not be possible without the contributions of our program committee and additional reviewers, who have our deepest thanks. We are grateful for the guidance and support provided by Larry Rosenblum and Andy van Dam. We thank Regina Sipple of IEEE Computer Society Press for producing these proceedings, and Carol Nichols of Meeting Management for logistical support. We would also like to thank the rest of the conference committee — especially our general chair, David Mizell — for the exceptional effort they have put into making VRAIS '95 a success. We hope that you find this program stimulating and enjoyable, and we look forward to seeing you at future VRAIS conferences.

**Steve Bryson** **Steve Feiner**

# VRAIS '95 Organizing Committee

**General Chair**
David W. Mizell, *Boeing Computer Services*
mizell@boeing.com

**Program Co-Chairs**
Steve Bryson, *CSC/NASA Ames Research Center*
bryson@nas.nasa.gov
Steve Feiner, *Columbia University*
feiner@cs.columbia.edu

**Organizing Chair**
Bob Marks, *University of Washington*
marks@milton.u.washington.edu

**Publications Chair**
Brenda Thein, *Army Research Laboratory*
bthein@arl.army.mil

**Finance Chair**
Judi Qualy-White, *Boeing Military Airplanes*
jqw@csmac8.cslab.ds.boeing.com

**Publicity Chair**
Mary Lou Padgett, *Auburn University*
mpadgett@eng.auburn.edu

**Press Relations**
Nadine Miner, *Sandia Labs*
neminer@isrc.sandia.gov

**Local Arrangements Chair**
Jannick Rolland, *University of North Carolina*
rolland@cs.unc.edu

**Video Proceedings**
Blake Hannaford, *University of Washington*
blake@uw-isdl.ee.washington.edu

**Tutorials Chair**
Beth Wenzel, *NASA Ames Research Center*
beth@eos.arc.nasa.gov

**Exhibits Chair**
Karen Haines, *University of New Mexico*
karen@sorcerer.eece.unm.edu

**Far East International Liaison**
Toshio Fukuda, *Nagoya University*
fukuda@mein.nagoya-u.ac.jp

**European International Liaison**
Mel Slater, *QMW University of London*
Mel.Slater@dcs.qmw.ac.uk

# VRAIS '95 Program Committee

Bernard D. Adelstein, Sterling Software/NASA Ames Research Center
Joanna Alexander, Zombie Inc.
Ron Azuma, Univ. of North Carolina, Chapel Hill
Norman Badler, University of Pennsylvania
Woodrow Barfield, Univ. of Washington
Stephen Benton, MIT
Chuck Blanchard, Talisman Dynamics, Inc.
Mark Bolas, Fake Space Labs
Kellogg Booth, Univ. of British Columbia
Pere Brunet, Polytechnical Univ. of Catalonia
Grigore Burdea, Univ. of North Carolina, Chapel Hill
Thomas P. Caudell, Univ. of New Mexico
Chris Codella, IBM T.J. Watson Research Center
Carolina Cruz-Neira, Univ. of Illinois at Chicago
Michael Deering, Sun Microsystems Computer Corporation
Rae Earnshaw, Univ. of Leeds
Steve Ellis, NASA Ames Research Center
José Encarnação, Fraunhofer Institute for Computer Graphics
Lennart Fahlen, Swedish Institute of Computer Science
Kim Fairchild, National Univ. of Singapore
Wolfgang Felger, Fraunhofer Institute for Computer Graphics
Scott Foster, Crystal River Engineering
Henry Fuchs, Univ. of North Carolina, Chapel Hill
Thomas Funkhouser, AT&T Bell Laboratories
Michael Gigante, Royal Melbourne Institute of Technology
Martin Göbel, Fraunhofer Institute for Computer Graphics
Mark Green, Univ. of Alberta
Hideki Hashimoto, Univ. of Tokyo
Richard Held, MIT
Michitaka Hirose, Univ. of Tokyo
Larry Hodges, Georgia Tech
John Hollerbach, Univ. of Utah
Philip Hubbard, Cornell Univ.
Hiroo Iwata, Univ. of Tsukuba
Rob Jacob, Tufts Univ.
Adam Janin, Boeing Computer Services
Mary Kaiser, NASA Ames Research Center
Ken-ichi Kameyama, Toshiba R&D Center
Arie Kaufman, SUNY, Stony Brook
Myron Krueger, Artificial Reality
Wolfgang Krüger, German National Research Center for Computer Science
James Lackner, Ashton Graybiel Spatial Orientation Laboratory
Andy Liu, Nissan Cambridge Basic Research
Nadia Magnenat-Thalmann, Univ. of Geneva
Beth Marcus, EXOS, Inc.
Michael W. McGreevy, NASA Ames Research Center
Margaret Minsky, Interval
Junji Nomura, Matsushita Electric Works, Ltd.
Shojiro Nagata, Stereo Researcher
Randy Pausch, Univ. of Virginia
Tom Piantanida, SRI International
Steve Pieper, Medical Media Systems
Ronald Pose, Monash Univ.
Timothy Poston, National Univ. of Singapore
Dennis Proffitt, Univ. of Virginia
Warren Robinett, Virtual Reality Games, Inc.
Jannick Rolland, Univ. of North Carolina, Chapel Hill
Joseph M. Rosen, Dartmouth-Hitchcock Medical Center
Larry Rosenblum, Naval Research Laboratory
Rick Satava, Advanced Research Projects Agency
Luis Serra, National Univ. of Singapore
Gurminder Singh, National Univ. of Singapore
Mel Slater, Univ. of QMW London
Henry Sowizral, Boeing Computer Services
Mandayam A. Srinivasan, MIT
Sharon Stansfield, Sandia National Labs
Lawrence W. Stark, Univ. of California at Berkeley
Dave Sturman, Medialab
Susumu Tachi, Univ. of Tokyo
Daniel Thalmann, Swiss Federal Institute of Technology
James Thomas, Battelle Pacific Northwest Laboratory
Andries van Dam, Brown Univ.
Elizabeth Wenzel, NASA Ames Research Center
Alan Wexelblat, MIT
David Zeltzer, MIT
Michael J. Zyda, Naval Postgraduate School

# Additional Reviewers

Kevin Arthur
D'Nardo Colucci
Rick Jacoby
Ian McDowall
Leonard McMillan
Kathryn Tesh
Anthony Webster

# Distributed Virtual Reality Infrastructure

# Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments

*Michael R. Macedonia, *Michael J. Zyda, David R. Pratt, Donald P. Brutzman, Paul T. Barham*
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5118 USA
+1-408-656-2305
{macedonia, zyda, pratt, brutzman, barham}@cs.nps.navy.mil

## ABSTRACT

We describe a network software architecture for solving the problem of scaling very large distributed simulations. The fundamental idea is to logically partition virtual environments by associating spatial, temporal, and functionally related entity classes with network multicast groups. We exploit the actual characteristics of the real-world large- scale environments that are simulated by focusing or restricting an entity's processing and network resources to its area of interest via a local Area of Interest Manager (AOIM). Finally, we present an example of how we would implement this concept for ground vehicles. We have begun design and construction of the AOIM for use with the NPSNET 3D vehicle simulator. NPSNET is currently the only Distributed Interactive Simulation (DIS) protocol compliant simulator using IP Multicast communications and is suitable for operation over the Internet.

**KEYWORDS:** Virtual Reality, Distributed Interactive Simulation, Internet Protocol Multicast, Distributed Interactive Entertainment, Large-scale Virtual Environments.

## INTRODUCTION

This paper outlines the problems and a proposed solution to the design and construction of large-scale distributed simulations. In particular this paper addresses the networking software architecture for large-scale virtual environments (VEs). We suggest a method that exploits the *spatial*, *functional*, and *temporal* relationships of real-world entities for partitioning VEs by associating network multicast groups with entity areas of interest.

The motivation for our effort is to expand the capability of virtual environments to serve large numbers (more than 1,000) of simultaneous users. Interest by the government, academic researchers, military, and telecommunications industry in large distributed virtual environments has been rapidly growing. In particular, distributed interactive entertainment applications such as multiplayer games, whether in-home or location-based, will require scalable network architectures in order to provide both rich environments and profitable returns.

Advances in computer architectures and graphics, as well as standards such as the IEEE 1278 Distributed Interactive Simulation (DIS) and BBN SIMNET protocols have made small scale (less than 300 players) realistic man-in-the-loop simulations possible [5,6,11]. These standards have been used by the military for several years. Unfortunately, SIMNET, which was developed for small unit training, and its descendant, DIS, are currently not suitable for large-scale multiplayer VEs.

## PRACTICAL PROBLEMS WITH THE DIS PROTOCOL

We list several major problems associated with scaling the current suite of DIS protocols in order to illustrate the difficulties of building large-scale VEs:

**Enormous bandwidth and computational requirements for large-scale simulation.** In schemes such as SIMNET and DIS, a simulation with 100,000 players would require 375 Mbit per second (Mbps) of network bandwidth to each computer participating in the simulation, an unrealistic requirement for an affordable system in this decade[7]. Maintaining the state of all other entities, particularly with dead-reckoning algorithms (which use second-order kinematics equations), will be a major bottleneck for large-scale simulation. Recent experiences with the U.S. Army's Simulated Theater of War (STOW) have shown this to be the case.

Faster computers and networks will not necessarily satisfy these needs. First, faster networks require faster processors merely to copy packets from the network into user space even before the application touches the protocol data unit (PDU). Second, the creeping demand for more realism (i.e. collision detection and constraint satisfaction) will introduce a rapid rise in computational and space complexity with

even modest size VEs [21].

We conjecture that 1000 entities are the limit to which a single host can realistically manage despite future advances in computer and graphics architectures.

**Multiplexing of different media at the application layer.** The current DIS protocol requires the application to multiplex and demultiplex different types of real-time data (e.g. simulation packets, audio, and video) at the application layer rather than at the network or transport layers. Therefore, the virtual environment must treat continuous video streams identically to bursty simulation traffic, i.e. allocation of buffers and timing at the application layer[20].

**Lack of an efficient method of handling static objects.** Large numbers of static entities such as bridges and buildings may change with respect to an event (e.g. an explosion). These and other stationary objects must send update messages at regular intervals to inform the participants of their current state. For example, a tank that has been destroyed must constantly inform the world that it is dead to inform new entrants or other entities that may have missed the original state change message.

**Models and world databases must be replicated at each simulator.** No mechanism in DIS exists to distribute objects on demand. For large-scale simulation, this is a necessity, particularly when the simulators are heterogenous, controlled by different organizations, and little coordination is expected prior to an exercise. Furthermore, it is not feasible nor efficient for each simulator to store every model and database for a 100,000 entity simulation. For example, a human simulation (e.g. a dismounted infantryman) on land normally does not need to concern itself with naval vessels, unless some unique scenario has the human near enough to the ocean so that it is visible.

## REASONS FOR PROBLEMS

**Event and State message paradigm.** A basic requirement for DIS has been that the simulation of the VE must be, as a whole, stateless - data is fully distributed among the participating hosts and entities are semi-persistent. Therefore, every entity must be made aware of every event (e.g. a missile detonation communicated by a Detonation Protocol Data Unit or DPDU) just on the chance it may need to know it. According to the protocol, an entity must, on a regular basis, communicate all of its state information (an Entity State Protocol Data Unit or ESPDU) to every member of the group - even though the data contained in the ESPDU is often redundant and unnecessary (e.g. aircraft markings). More importantly, these "keep alive" messages can comprise 70% of the traffic for large-scale simulations [13].

This paradigm as applied in DIS does not take into consideration that different simulated systems have different real-world sensing capabilities that translate into each entity's VE data requirements. In a large VE, it is unlikely that two entities representing ground vehicles separated by 200 Km need to be aware of each other. Yet, under the current architecture they must inform each other of state changes and updates.

The rationale for this is to avoid the reliability problems of a central server, to simplify communication protocols, and minimize latency while guaranteeing that hosts entering a simulation would eventually build their entity database through entity state and event messages. Furthermore, the use of broadcast ESPDU updates is part of the effort to maintain consistent view among the simulators within a particular tolerance.

**Real-time system trade-off's.** Reliability (guarantees that data sent is received) normally is compromised for real-time performance in large distributed groups. This is because in order to be truly reliable the system requires the use of acknowledgment schemes such as the one used in Transport Control Protocol (TCP) which defeats the notion of real-time, particularly if a player host must establish a virtual connection with every other entity host to ensure that each received data correctly. Therefore, large-scale environments must rely on connectionless (and therefore unreliable) network protocols such as the User Datagram Protocol (UDP) for wide-area communications.

The corollary is that a real-time environment should avoid transactions between entities since this requires reliable communications. Furthermore, schemes that use a central database do not work well in a large VE due to I/O contention. For example, AT&T's Imagination network limits the number of concurrent players in a game to four because they are centrally served and bandwidth is limited to the speed of modems (less than 28 Kbps).

**No "middleware" layer.** There does not exist a DIS protocol component that mediates between distributed VE applications and the network. The current DIS paradigm implies the use of a bridged network because every message is broadcast to every entity. However, internetworking (routing over the network layer) is necessary for large-scale simulations because it provides the capability to use commercial services as opposed to private networks to bring together diverse, geographically dispersed sites; use different local network topologies and technologies (e.g. Ethernet and FDDI); and take advantage of "rich" topologies for partitioning bandwidth, providing robustness and optimization of routes for minimizing latency. Confining DIS to the data link layer requires the use of

bridges which are on order of magnitude slower to reconfigure after a topological change than routers while the number of stations are limited to the tens of thousands. A network with routers is limited to the numbers accommodated by the address space [10].

**Origins as small unit training systems for Local Area Networks (LANs).** Many of these problems devolve from the fact that until recently DIS and SIMNET were used exclusively for small scale training simulations. In this mode it has been relatively easy to insure that the VE components have homogenous sets of models and terrain databases by replicating them at each host. The lack of middleware stems from the monolithic nature of these small scale environments which could be distributed using a single LAN. Hence, *broadcast* communication was sufficient for these limited environments.

These origins have also influenced the current assumptions about the density and rates of activity of entities in large-scale simulations that do not necessarily match the real world. Players in SIMNET participated for short periods (several hours) and were highly active because the purpose of the simulation was to train crews in coordinated drills. Furthermore, the density of entities with respect to the simulated area of play was high because that best represented a small unit engaged in close combat and because of the difficulty in using large terrain data bases.

## EXPLOITING REALITY

Increasing the number of entities by more than two orders of magnitude requires us to think beyond these artificial situations. We believe that it is incorrect to strictly extrapolate the SIMNET and DIS experience (or any of the small-scale research VEs) to large-scale VEs. Moreover, large VEs are likely to be domain specific in their requirements. We can exploit aspects of the real-world such as areas of interest and movement rates to efficiently use multicast groups, eliminate ESPDU keep-alive updates, enhance the reliability of large-scale VEs, and reduce overall bandwidth requirements.

In the real world, which virtual environments emulate, entities have a limited area of interest. For example, a tank on a battlefield can effect and observe other entities out to a range of less than 10 Km. On the other hand, a person on foot typically has an area of interest of only several hundred meters. This would be the case for a dismounted infantryman or a human simulated for a typical role-playing adventure game. The entities whose areas of interest overlap are members of a *spatial class* or group in the VE.

With respect to the military domain, group membership within these classes would change relatively slowly. Helmbold in his study on the rates of advance rates for land operation found that land combat operations stand still 90-99% of the time [16]. The world's record for aggregate movement in modern warfare was 92 Km/day for 4 days (or about 6 Km/hour) by the 24th Mechanized Infantry Division in Desert Storm [15,17]. Individual vehicles may move much faster, but they would not continue at high rates very long because they fight as part of units in which movement must be coordinated.

## RELATED WORK

The partitioning of virtual worlds into spaces is a common metaphor for VEs. Multi-User Dungeons (MUDs) have used this idea and projects like *Jupiter* from Xerox PARC have extended this to associating "rooms" with multicast video and audio teleconferences [24]. Lockheed has developed a similar concept for spatial partitioning that assumes the use of ATM multicast "channels", i.e. mapping relevant groups to ATM's Virtual Channel Identifiers. Though ATM multicast technology is not yet mature, (few vendors support it), these ideas present exciting possibilities.

Benford has described a concept for the spatial interaction of objects in a large-scale VE [22]. The spatial model uses different levels of awareness between objects based on their relative distance and mediated through a negotiation mechanism. An implementation using DIVE (Distributed Interactive Virtual Environment) uses "standard VR collision detection" to determine when the transitions between awareness levels should occur [26]. The MASSIVE project also uses this approach. However, the need for collision detection, reliable communication, and strong data consistency have made it difficult for DIVE and MASSIVE to scale beyond a handful of users [25]. This may be changing as their developers pursue the use of multicast communications and weaker data consistency.

## APPROACH

Our approach is computationaly efficient--constant time versus $O(\log n)$ for simple collision detection using octrees or bounding volumes--and takes advantage of multicast networks for partitioning the environment [19]. Additionally, we consider two other criteria for establishing relevance among entities and their communication in the VE.

Entities also may belong to a *functional class* in which an entity may communicate with a subset of entities. Therefore, simulated radio traffic should be restricted only to the interested parties of the group. Other types of functional classes could be related to system management or services such as time.

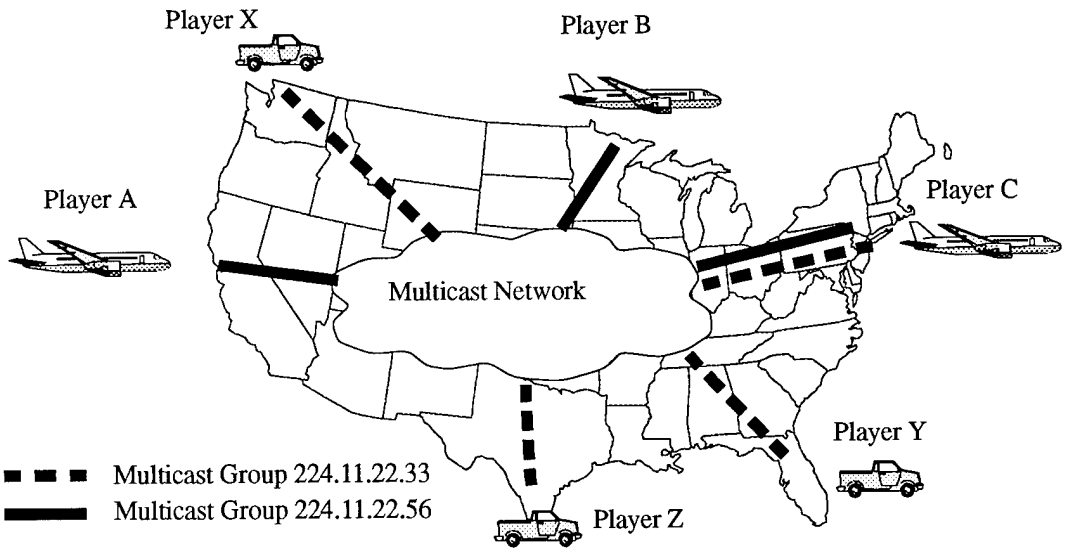



Figure 1. Simple illustration of multicast communications. Groups are expressed as IP Multicast Addresses. Note that Player C is a member of both multicast groups.

Another example of a functional class in the military domain would be a VE "air control" group. The group would include entities that are primarily concerned with entities or events occurring in the air. Therefore, air defense and aircraft entities would comprise the majority of the group. Aircraft and air defense systems are relatively sparse in the whole as compared to other combat systems such as tanks. Air defense systems would also belong to a small subset of the spatial class. Aircraft which are interested in a particular area of ground can "focus" and join a spatial group associated with its area of interest.

Finally, entities can belong to a *temporal class*. For example, some entities do not require real-time updates of all state changes. A system management entity might only need updates every several minutes. Similarly, a simulator of a space-borne sensor only needs a general awareness of ground vehicle entities and therefore can accept low-resolution updates. When there is a need for more resolution, the simulator, like aircraft entities, can focus and become part of a spatial group.

## DIS AREA OF INTEREST MANAGER

We propose the use of a software "glue" between the DIS event and state PDU paradigm and the network layers that is wedded to reality. The area of interest manager (AOIM) partitions the VE into a set of workable, small scale environments or classes to reduce computational load on hosts, minimize communications on network tail links, and localize reliability problems. Furthermore, the AOIM exists with every simulator to distribute partitioning processing among hosts.

## MULTICAST

The AOIM uses spatial, temporal, and functional classes for establishing membership in multicast network groups. Multicast services allow arbitrarily sized groups to communicate on a network via a single transmission by the source [10]. Multicast provides one-to-many and many-to-many delivery services for applications such as teleconferencing and distributed simulation in which there is a need to communicate with several other hosts simultaneously. For example, a multicast teleconference allows a host to send voice and video simultaneously to a set of (but not necessarily all) locations. With broadcast, data is sent to all hosts while unicast or point-to-point routes communication only between two hosts.

The Internet Protocol (IP) Multicast protocol provides an addressing scheme that permits unreliable, connectionless, multicast service that is routable over the Internet [2,19]. From the perspective of the AOIM, IP Multicast allows the creation of transient multicast groups that can be associated with an entity's area of interest (AOI).

In this context, IP Multicast addresses can essentially be used as context labels instead of physical destinations. Figure 1 shows this. Players X, Y, and Z send data to the IP Multicast group address 224.11.22.56 rather than explicitly forwarding packets to each and every player. The network takes over this requirement. Players A and B send and receive traffic relevant only to their group, 224.11.22.33,

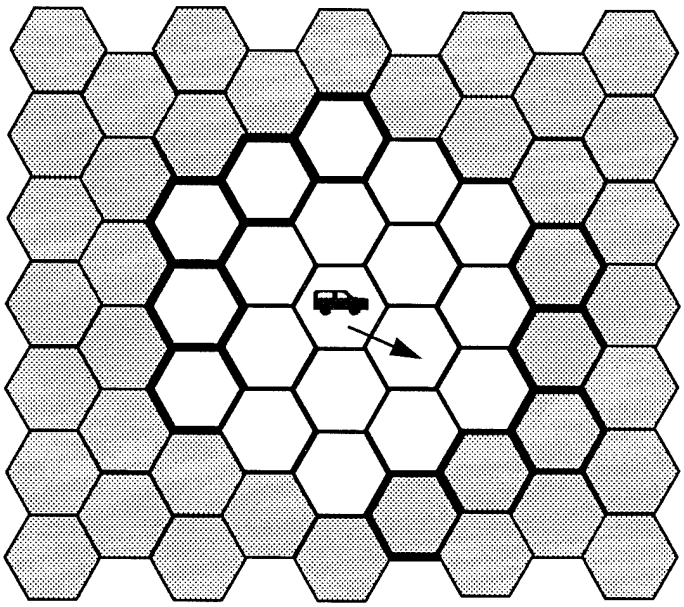


Figure 2. Area of Interest for vehicle mapped to a subset of multicast groups.

while C is a member of both and participates in each session.

Therefore, multiplexing and demultiplexing is done at the network level. This naturally provides a way of separating classes of traffic such as audio, video and simulation data. For example, the radio communications functional class would be mapped to a particular multicast group address or "channel".

As stated before, this partitioning is necessary to reduce the enormous computational requirements of large-scale (100,000 player) simulations. For a 1000 object exercise conducted in 1990 with SIMNET, the limiting factor was not network bandwidth, with loads running at 50%, but the local host processor performance [1]. Network simulations done by Van Hook have shown that a 90% reduction in traffic to a particular node is achievable for a 10,000 player exercise using multicast services [13].

## ASSOCIATIONS

To illustrate our ideas, we examine using the AOIM to associate spatial classes with multicast addresses. We suggest for this example partitioning the VE with appropriately sized hexagonal cells. Each cell is associated with a multicast group. In Figure 2 we associate a vehicle with 19 hexagons that represent its AOI. Hence, it is also a member of 19 network multicast groups. The entity's host listens to all 19 groups but, with two exceptions, it sends PDUs only to the one associated with the cell in which it is located.

There are several reasons we use hexagons. First, they are regular, have a uniform orientation, and have uniform adjacency [18]. As the vehicle moves through the VE, it uniformly adds and deletes the same number of cells/ multicast groups.

Secondly, a vehicle's AOI is typically defined by a radius - much like signal of transmitter in a cellular telephone system. If squares were used, we would either need to include more area than was necessary (and thus include more entities in our AOI) or use smaller grids - requiring more multicast groups - and compute which grids the vehicle should be associated with. Using hexagons with a 2.5 km radius, the AOI above ranges from 12.5 to 8.6 km and the area is 411 $km^2$. If the average density of vehicles was 2 per $km^2$, then the entity host communicates with approximately 800 other entities. As mentioned previously, the AOI varies with respect to the capabilities of the system simulated.

## GROUP CHANGES

Entities can belong to several groups at a time to avoid boundary or temporal aliasing. There will likely be few group transitions by a ground-based entity within an hour because, on average, groups of vehicles will move slowly relative to the entire VE. If a vehicle was moving at the Desert Storm record advance rate, it would transition on average a cell once an hour. The vehicle portrayed in Figure 2 must join and leave 5 multicast groups which are associated with cells at the periphery of its AOI where change is less critical - ameliorating the effects of latency

caused by joining and leaving new groups. The outlined clear cells are removed and the outlined grey cells are added as the entity transitions to a new cell.

We use group changes as an opportunity for database updates -- similar to a paged memory scheme -- in order to eliminate regular ESPDU updates. We do this in a logical, distributed manner using knowledge about the age of entities with respect to their particular group.

An entity joins a group as a passive or active member. Active members send as well as receive PDUs within the group, are located in the cell associated with the group, and can become the group leader. Passive members normally do not send PDUs to the group except when they join or leave. They are associated with the group because the cell is within their AOI, yet they are not located within the cell.

When an entity joins a new group it notes the time it entered and issues a *Join Request* PDU to the cell group. The PDU has a flag indicating whether it is active or passive. The group leader replies with a *Pointer* PDU that references the request and in turn multicasts a PDU containing a pointer to itself or another active entity. The new member sends a *Data Request* PDU to the referenced source which issues a *Data* PDU containing the aggregate set of active entity PDUs. A passive entity becomes an active member of a group by reissuing the *Join Request* PDU with a flag set to active when entering a cell. Departures from the group are announced with a *Leave Request* PDU.

We use the oldest member of the group as the election method for group leader. We make use of timestamps to determine the oldest member. The first active member of a group will issue several Join Request PDUs before concluding that its is the sole member of the group and therefore the oldest. When a passive entity determines that there is no leader, it merely listens for active members. A new active member of an established group issues a Join Request PDU, receives the Data PDU, notes the join timestamps of the members, and keeps track of those who enter and leave.

### RATIONALE

The Data PDU may be sent reliably to the issuer of the Join Request PDU via a unicast protocol as a heavy-weight object. With a large member distributed simulation, reliability, as provided in the Transmission Control Protocol (TCP), would normally penalize real-time performance merely by having to maintain timers for each host's acknowledgment. Moreover, flow control is also not appropriate for DIS since systems with humans in the loop can recover from a lost state message more gracefully than from late arrivals. Fortunately, within the context of DIS, a certain amount of unreliability is tolerable and is mediated through the use of the dead-reckoning and smoothing algorithms [4,8]. Other applications such as packet voice and video can use adaptive techniques to handle lost packets and delays [9]. However, we can reliably send the Data PDU because the entity will normally be joining a group that is at the periphery of its AOI where latency is not as critical.

**Communications model.** We conjecture that a large-scale real-time VE cannot guarantee strong data consistency and reliable communication among all its participants simultaneously. Instead, four types of communication can be established which, used together, allow stronger consistency than simply broadcasting state messages. They provide for a much richer world through a mechanism for sending large objects reliably and supporting VE partitioning.

In our model there exists four methods for communication within the context of VEs:

*Light-weight interactions.* These messages are composed of the same state, event, and control PDUs used in the DIS paradigm but implemented with multicast. They are light-weight because the complete semantics of the message are encapsulated in the maximum transfer unit (MTU) of the underlying data link to permit asynchronous real-time interactive use. Therefore, these PDUs are not segmented. They are either received completely or not at all because they are communicated via connectionless and unreliable (unacknowledged data) networks. The MTU for Ethernet is 1500 bytes and 296 bytes for 9600 point-to-point (PPP) links.

*Network pointers.* Proposed are light-weight references to resources, in a similar way to Uniform Resource Identifier (URI) as defined in the Hypertext Transfer Protocol (HTTP)[27]. Pointers are multicast to the group so that they can be cached by members. Therefore, common queries need not be resent and the server can direct the responses to other members of the group. We make a distinction between pointers and light-weight interactions (e.g. Join Request PDU) because they do not completely contain a object but rather its reference. Pointers provide a powerful mechanism for referencing not only the current aggregate state of the group but also terrain, model geometry, and entity behaviors defined by a scripting language. In the context of the World Wide Web, network pointers have revolutionized Internet communication.

*Heavy-weight objects.* These objects require reliable, connection-oriented communication. For example, an entity may require model geometry after joining a group that does

not exist in its database. The entity would multicast a request for the geometry and the response would be a multicast pointer to the source. If efforts such as the Virtual Reality Modeling Language (VRML) are successful, heterogeneous systems may be able to exchange this type of information [28].

*Real-time streams.* Video and audio traffic provide continuous streams of data that require real-time delivery, sequencing and synchronization. Moreover, these streams will be long-lasting, persisting from several seconds to days. They are multicasted on a particular "channel" to a functional class. In contrast with the current DIS protocol, we propose the use of pointers to direct entities to these channels rather than, for example, forcing the VE, which may be as simple as a text-based application, to receive both light-weight DIS PDUs as well as video streams. Moreover, the VE can spawn a separate process which incorporates an adaptive receiver and which separates the handling of bursty simulation message from real-time streams.

## ENTITY INTERACTIONS

Entities can only interact if they are aware of and can communicate with each other. Entity A becomes aware of entity B only if B is an active member of a group that A belongs to -- and therefore, in the AOI of A. If both are only passive members of the same groups then each one is beyond the view or influence of the other.

In a combat simulation, it is possible that if tank A fired a non-guided munition (which is not instantiated as an entity) at tank B, then B's AOI might not overlap the cell in which A was an active member tank. A must become an active member of the target area cell and forward a detonation PDU to that cell. According to the DIS protocol, entities assess for themselves the effects of the detonation and report via an ESPDU any state changes which are the result.

## ADVANTAGES

**Reduced latency for new entrant learning.** Assuming an even distribution of entities in our example, for each cell joined an entity must receive data for about 40 other entities--approximately 40 Kbits. At 10 Mbps data transfer rates, it would take 4 ms to update a new entrant versus 5 seconds under the current DIS scheme.

**Reduced bandwidth requirements.** This architecture eliminates the need for entity keep-alives. New entrants are informed by the Join procedure of who exists in their particular groups. Multicast association further reduces the traffic demands on the tail links by confining the scope of an entity's communication to its area of interest and implicitly directing it traffic to a subset of hosts on the network.

**No need for a centralized server.** Using the oldest member of a group to serve Join requests is logical because it is the entity that should know all of the other entities and the past events that have occurred in the group. We expect that serving the group will be relatively undemanding with respect to Input/Output processing for the group leader because of the small number of active members in a group/cell and relatively slow transitions due to the expected real world transition rates for vehicles. Moreover, the server, through the pointer mechanism, can assign other entities to the task of serving the request. This provides an opportunity for exploring different algorithms for load balancing purposes.

**Solves the static and dead entity problem.** Likely candidates for the group leader will be static entities such as those representing buildings or bridges which can change state (i.e. collapse). Servers for these destructible entities will be the originating members of a spatially associated group and remain with the group for its entire existence. Moreover, static or dead entities are no longer a major burden to the VE with respect to wasting bandwidth with update ESPDUs. They need only to transmit PDUs upon initialization and when changing state.

**Localization of reliability problems.** large-scale VEs will naturally have some degree of unreliability. Partitioning the VE into groups prevents problems from impacting on the entire simulation. Currently, an entire DIS simulation involving hundreds of entities can fail because of a single rogue application because all communication is broadcast.

**Maintains the current DIS semantics.** The AOIM can be run as a separate thread or process and eliminates the need to change current DIS PDU semantics. The application simulating an entity is not required to have knowledge of the partitioning or the AOIM.

## STATUS OF WORK

We have developed an IP Multicast version of the NPSNET-IV 3D vehicle simulator using a network library developed by Paul Barham and John Locke that supports multiple threads and dynamic creation of multicast groups [23]. Furthermore, we are including the algorithms to support the AOIM concept presented here and developing a simulation to predict the results.

## CONCLUSION

This paper describes a concept that provides a network software architecture for solving the problem of scaling very large distributed simulations. The fundamental idea behind our approach is to logically partition virtual environments by associating spatial, temporal, and functionally related entity classes with network multicast groups. This is

accomplished by exploiting the actual characteristics of the real-world large-scale environments that are to be simulated, and by focusing an entity's processing and network resources to its area of interest via an Area of Interest Manager.

Finally, we present an example of how we would implement this concept for spatial classes. We have begun design and construction of the AOIM for use with the NPSNET 3D vehicle simulator. NPSNET is currently the only DIS compliant simulator using IP Multicast communications and is suitable for operation over the Internet.

## ACKNOWLEDGMENTS

This work would not have been possible without the support of our research sponsors: USA ARL, ARPA, DMSO, USA STRICOM, USA HQDA AI Center-Pentagon, USA TRAC.

## RESOURCES

Many of the references noted below are available via the NPSNET Research Group's WWW home page:

ftp://taurus.cs.nps.navy.mil/pub/NPSNET_MOSAIC/npsnet_mosaic.html

1. Chung, J.W., *An Assessment and Forecast of Commercial Enabling Technologies for Advanced Distributed Simulation*, technical report, Institute for Defense Analysis, Arlington, VA (October 1992).

2. Deering, S. *Host Extensions for IP Multicasting.* RFC 1112 (Aug 1989).

3. Doris, K. Issues Related to Multicast Groups. In *Proceedings of the Eighth Workshop on Standards for the Interoperability of Defense Simulation* (March 1993), pp. 269-302.

4. Harvey, E.P., Schaffer, R.L., *The Capability of the Distributed Interactive Simulation Networking Standard to Support High Fidelity Aircraft Simulation*, technical report, BMH Associates, Inc. and BBN Systems and Technologies, Norfolk VA, Cambridge, MA. (July 1992).

5. Institute of Electrical and Electronics Engineers, International Standard, ANSI/IEEE Std 1278-1993, *Standard for Information Technology, Protocols for Distributed Interactive Simulation* (March 1993).

6. Institute for Simulation and Training, IST-TR-93-20, *Communication Architecture for Distributed Interactive Simulation (CADIS) [Final Draft]*, University of Central Florida, Orlando, FL. (June 1993).

7. Loral Systems Company, *Strawman Distributed Interactive Simulation Architecture Description Document Volume 1*, technical report, Advanced Distributed Simulation Technology Program Office, Orlando, FL (March 1992).

8. Miller, D.C., Pope, A.C., and Waters, R.M. Long-Haul Networking of Simulators. In *Proceedings of Tenth Interservice/Industry Training Systems Conference* (December 1989), p. 2.

9. Partridge, C. *Gigabit Networking.* Addison-Wesley. Reading, MA. 1994, pp. 191-193

10. Perlman, R. *Interconnections: Bridges and Routers.* Addison-Wesley, NY, 1992, p. 258.

11. Pope, A., BBN Report No. 7102, *The SIMNET Network and Protocols*, technical report, BBN Systems and Technologies, Cambridge, MA, (July 1989).

12. Pratt, D.R., *A Software Architecture for the Construction and Management of Real Time Virtual Environments*, dissertation, Naval Postgraduate School, Monterey, CA, (June 1993).

13. Van Hook, D.J. *Simulation Tool for Developing and Evaluating Networks and Algorithms in Support of STOW 94.* Presented for Scalability Peer Review, (August 1993).

14. Zyda, M.J., Pratt, D.R., Falby, J.S. Barham, P.T., Kelleher, K.M. The Software Required for the Computer Generation of Virtual Environments. *Presence*, **2**, 2 (Summer 1993) 130-140.

15. Dunnigan, J. and Macedonia, R.M. *Getting It Right*, Morrow. New York, NY, 1993, p. 211.

16. Helmbold, R. L. *Rates of Advance in Historical Land Combat Operations*, technical report, CAA-RP-90-1, US Army Concepts Analysis Agency, Bethesda, MD, (June 1993), pp. 5-2,3.

17. McQuie, R. *Historical Characteristics of Combat for Wargames*, technical report, CAA-RP-87-2, US Army Concepts Analysis Agency, Bethesda, MD, (July 1988), p. 13.

18. Samet, H. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley. Reading, MA, 1989, pp. 20-21.

19. Macedonia, M.R., Brutzman, D.P. MBone Provides Audio and Video Across the Internet. *IEEE Computer*, 27, 4 (April 1994), 30-34.

20. Feldmeier, D. C., Multiplexing Issues in Communication System Design. In *Proceedings of ACM SIGCOMM '90* (September 1990), ACM, pp. 209-19.

21. Pentland, A.P., Computational Complexity versus Simulated Environments. *Computer Graphics. 1990 Symposium on Interactive 3-D Graphics* 24, 2 (March 1990), 185-192.

22. Benford, S., Fahlen, L.E. and Bowers, John. Supporting Social Communication Skills in Multi-Actor Artificial Realities. In *Proceedings of The Fourth International Conference on Artificial Reality and Tele-Existence (July 14-15,* Tokyo, Japan), 1994, pp. 205-223.

23. Macedonia, M. R., Zyda M.J., Pratt D.R., Barham P.T., Zeswitz S. NPSNET: A Network Software Architecture for Large-scale Virtual Environments. *Presence* 3,4 (Winter 94).

24. Curtis, P., Nichols, D.A. MUDs Grow Up: Social Virtual Reality in the Real World. 1994. ftp:// ftp.parc.xerox.com/pub/MOO/papers/MUDsGrowUp.ps.

25. Benford, S., Bowers, J., Fahlen, L. and Greenhalgh, C. Managing Mutual Awareness in Collaborative Virtual Environments. In *Proceedings of VRST '94,* World Scientific Publishing Company, NJ, pp. 223-236.

26. Carlsson, C. and Hagsand, O. (1993). DIVE - a Multi User Virtual Reality System. In *Proceedings of VRAIS '93* (September 18-22, Seattle, WA) IEEE, NJ, 1993. pp. 394-400.

27. Berners-Lee, T. Hypertext Transfer Protocol (HTTP), A Stateless Search, Retrieve and Manipulation Protocol, Internet Engineering Task Force Draft, ftp://nic.ddn.mil/ internet-drafts/draft-fielding-http-spec-01.ps, (19 December 1993).

28. Pesce, M. The Virtual Reality Modeling Language. http://www.eit.com/vrml/vrmlspec.html.

# EM - An Environment Manager For Building Networked Virtual Environments

Qunjie Wang, Mark Green and Chris Shaw
Department of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2H1, CANADA
email: {qunjie,mark,cdshaw}@cs.ualberta.ca
WWW: http://www.cs.ualberta.ca/~graphics

**ABSTRACT**

The Environment Manager (EM) is a high-level tool for constructing both single user and multi-user virtual environments. A script file is used to initialize and run virtual worlds. Independent applications can share information and cooperate with each other across the Internet. EM reduces the effort required to produce a networked virtual world by providing high-level support for application replication, network configuration, communication management and concurrency control. This paper describes the architecture and implementation of EM.

**KEYWORDS:** Multi-User, MR Toolkit, Peers, Interpreted Language, Networked Virtual Reality

## 1 INTRODUCTION

With many single-user Virtual Reality(VR) applications being successfully implemented [3, 13, 15, 16, 20], Networked Virtual Reality is now becoming a hot topic. *Networked Virtual Reality* refers to virtual environments where multiple users connected by a network can share information with each other. The production of good VR worlds, whether single user or networked, requires a considerable amount of design and programming time. Expertise is required in device handling, user interface design, network programming, graphics programming, and interaction techniques.

We have been building software tools that reduce the amount of development work for VR, allowing the VR implementer to tackle a wider range of applications. These tools are briefly summarized:

The MR Toolkit [18] provides standard software facilities required by VR user interfaces. It provides support for common VR devices such as 3D trackers, Head Mounted Displays (HMDs), gloves, and 3D mice, and supports distribution of the user interface and data over several workstations. A single-user MR application consists of one or more UNIX-style processes, with one designated as the master process, and the others as slave or computation processes. Slaves are used to perform output tasks on non-master machines, such as rendering the other eye's image for a HMD, and computation processes perform CPU-intensive tasks on computation server machines. MR applications are written in C or FORTRAN, and the graphics programming is done using the machine's native graphics library such as GL, Phigs or Starbase.

The MR Toolkit Peer Package is an extension to the MR Toolkit that provides the connection level facilities to allow multiple independent MR applications to exchange data with one another across the Internet [17]. The master process (the **peer**) can transmit device data to other remote master processes and receive device data from them. Application-specific data can also be shared between peers. Any MR Toolkit program may start up the peer package at any time, and may initiate and quit communications with other processes at will. Peers are connected pairwise and one peer may send a message to any or all other peers using procedures.

JDCAD+ [12, 14] is a solid modeling and animation computer-aided design system. It uses a Polhemus or Ascension 3D tracker to sweep out 3D canonical shapes such as boxes, cylinders, cones and the like. These shapes can be reshaped, joined together and connected in kinematic chains. JDCAD+ has a keyframe animation facility that can be used to animate various motions of an object. JDCAD+ automatically generates OML animation code, and most animations can be created without the user having to write an OML program.

OML (the Object Modeling Language) is a procedural programming language we have designed, with fundamental data types and operations for geometry, object-oriented programming and behavior specifications [11]. It is used to describe the geometries and behaviors of 3D objects used in virtual worlds. The geometry processor within the OML interpreter supplies efficient collision detection between objects selected by the world designer, and performs efficient object culling to maximize visual update rate. OML is designed to be portable to any platform, so its geometric modeling aspects are independent of any particular graphics package.

An OML *object* corresponds to a C++ Class, and contains code to generate the geometry of the 3D object, to control how the 3D object is to appear (color, texture, etc), and behavior code. An OML *instance* corresponds to a C++ object instance. An OML *behavior* is a procedure (method) that reacts to an incoming event or combination of events, and typically generates some sort of change in the state and appearance of the 3D object. Behaviors trigger other behaviors by sending an event to the behavior to be triggered. The built-in *tick* event triggers ongoing behaviors like walking and so on, and an internal time value can be used to interpolate between keyframes.



0-8186-7084-3/95 $04.00 

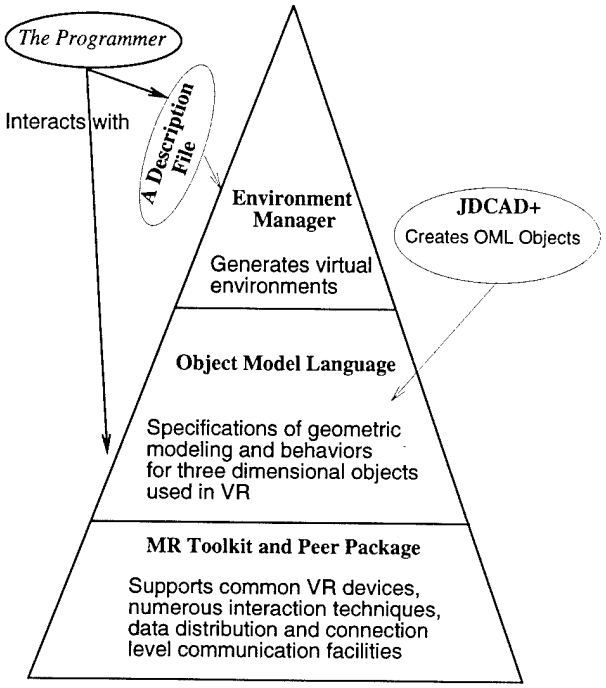



Figure 1: A VR Tool Architecture And Its Component Relationships

The OML compiler produces an interpretable version of the object specification (called the *object prototype*) and the OML interpreter is linked with the application program at run-time. One can create an arbitrary number of instances at run-time and have high level control over their behaviors. Different instances for the same object can have different geometries and behaviors, since object specification can be parameterized.

To create a Virtual Environment using OML, an MR Toolkit program (written in C) loads compiled OML code for each of the objects, sets up the VR devices that are to be used, dispatches device-related events to the OML interpreter as appropriate, and calls the interpreter every graphical update. The interpreter evaluates the behaviors of each instance, and draws each instance. Writing a new C program for each new OML-based Virtual Environment is a rather error-prone and tedious process, so a high-level tool called the **Environment Manager** (EM) was developed on top of our existing tools to eliminate the programming effort of initializing and running OML-based virtual environments. EM can run both single user and networked virtual environments, as specified on the command line. EM constructs a virtual environments using only a script file, without any lower level programming. The architecture of our VR tool package and the relationship between its components is illustrated in Figure 1.

## 2 RELATED WORK

In this section, we focus on existing networked VR systems and tools. Most of the newly developed tools and systems use the *application replication architecture* – each participating process has a copy of a replicated application database and changes are propagated to the other processes.

The most famous example of multi-user virtual environment is SIMNET [1, 4], which is a distributed interactive virtual world for battlefield simulation and training. In SIMNET, an object broadcasts an event to all objects without calculating which other objects might be interested in the event, or how the receiving objects might be affected by it. The receiving object decides what it is going to do about the received event. Objects transmit information only about changes in their state (position, orientation), and the *dead reckoning* algorithm is used to extrapolate state for objects. NPSNET [20] and VERN [2] use SIMNET's DIS protocol to perform military simulations.

The Distributed Interactive Virtual Environment(DIVE), developed at the Swedish Institute of Computer Science, is a platform for heterogeneous multi-user virtual environments [5, 6]. A process group in DIVE is a set of processes which can be addressed as one entity: atomic multi-cast protocols can be used to relay the messages addressed to the group as one, so each process in the group can receive exactly the same updates with reduced network traffic. In DIVE, there are three mechanisms to ensure consistency in the replicated database: mutual exclusive locks, reliable source ordered multi-cast and distributed locks.

IBM's VR-DECK [7] is a C++ class library that extends IBM's Rubber Rocks system [8] by distributing the role of the central dialog manager among a collection of peer processes called modules. Modules are the building blocks of the virtual environment, including graphical objects, trackers, and renderers. Each module has a set of compiled-in rules which determine which events to accept, what type of event to generate, and how the module reacts to a certain event. For each receiver, a module maintains a list of events that the receiver is known to be interested in, minimizing inter-module network traffic. Similar to visual programming, a 2D graphical tool called *vbuild* is used to create a virtual environment by connecting the application's modules to each other and to the device and rendering modules. Multi-user virtual worlds can be created by manually connecting each user's application modules to each other and to each of the other user's renderers. The configuration can be saved, so that startup easier the second time around. One clear disadvantage if this system is that a single module must start all the others, which requires that the this process have execution privileges on each user's machine. Also, it is not clear how a user at another location can join an already-running virtual environment.

BrickNet is a networked virtual reality toolkit developed at the National University of Singapore [19]. Different from DIVE and the MR Peer Package, which have a peer-to-peer communication scheme, BrickNet has a client/server communication configuration. A virtual world developed using BrickNet is a *client*, which connects to a *server* to request objects of interest and communicate with other clients. BrickNet virtual worlds are not restricted to having identical local databases (set of objects), as is the case with SIMNET and DIVE (they are multi-user-same-content applications). Similar to MR Peers, BrickNet uses UDP to transmit messages.

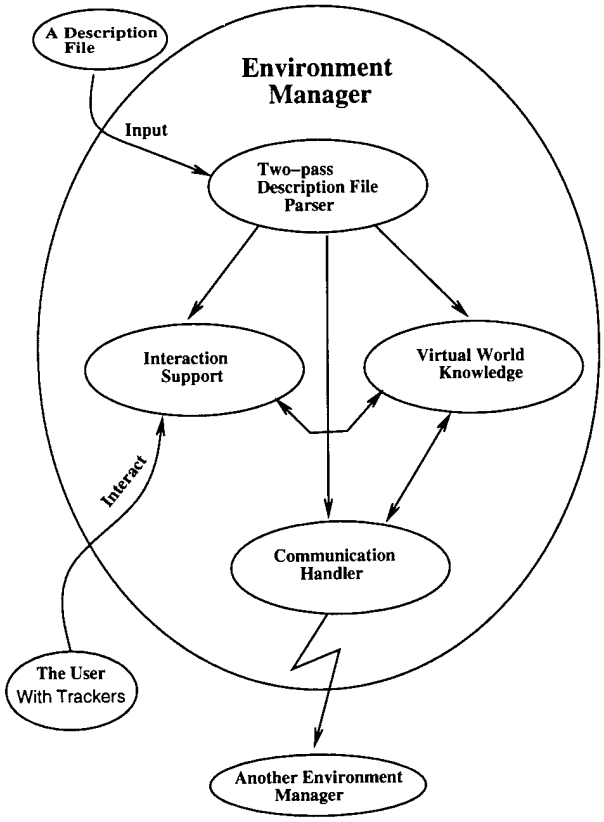



Figure 2: Architecture Of The Environment Manager

## 3 EM ARCHITECTURE

EM binds together multiple OML objects into a virtual world. It sets up the VR devices, dispatches events, and calls the OML interpreter every update. In the networked (multi-user) case, EM implements the replicated architecture, allows identical or different content network configuration, performs shared information communication management, concurrency control and network bandwidth reduction.

EM is in charge of the initialization and management of the local application as well as communications between the other applications. Each user in a multi-user virtual environment runs a local EM which communicates with other EMs across the network. The OML objects managed by EM need not be aware whether the world is single-user or multi-user, since EM handles all the event dispatching that triggers OML behaviors. This allows an object designer to create OML objects without having to worry about what context they may be used in.

The virtual world built by EM (called an *EM world*) is comprised of OML objects and instances that embody their graphical, behavioral and network properties. Unlike SIMNET, EM worlds are not restricted to sharing an identical set of objects. An EM world manages its own set of objects and instances, some or all of which may be shared with other EM worlds on the network. *Local* objects and instances are those which reside on the user's home machine. *Shared* objects or instances may be loaded by other EM worlds on the network. The remote versions of local objects are called *ghosts*.

EM consists of four subparts: (see Figure 2) the description file parser, interaction support, the VR world and the communication handler.

### 3.1 The Description File Parser

EM starts by reading an EM world description file that states the configuration if this EM world. A two-pass parser reports syntax errors in the EM file, and collects information about device configuration, object files, instances, and behaviors. This information is processed by the remaining EM subparts.

The following information is specified in an EM description file:

1. Device configuration.

   This part specifies the local devices to use.

2. Object prototype files

   The local OML object prototype files are listed here. EM loads these files into the OML interpreter.

3. Instances

   Each instance of a local object is specified here. Each entry consists of the name of the object, the name of the instance, the parameter values used to create the instance, the mapping from events to behaviors, and the shared variables for the instance, if there are any.

4. Object behaviors

   If an instance does not specify an event for a behavior, it will inherit the event for that behavior from its object. These object behaviors will be inherited by remote instances of this object if this object is shared. This inheritance eliminates the tedium of having to specify the same event for each instance of an object.

5. Networked EM world description.

   This part specifies the shared objects and instances.

For a single-user EM world, networked descriptions (item 5) are not included. The networked EM world description specifies the shared objects and instances that can be exported to or imported from other EM worlds. This specification has four parts:

1. Concurrency control scheme.

   The EM programmer may choose one of two concurrency schemes, as outlined in section 4.

2. Shared object information.

   This is a list of the names of the objects that are required by this EM world. The local EM will solicit remote EMs for these objects. Hopefully the remote EMs which have them and are willing to *export* them.

3. Shared instance information.

   This information is provided when an EM world wants to send shared instances to remote EM worlds. Remote EMs should have the objects that the instances are created from. The information is a subpart of the EM description file containing instance entries, identical to item 3 in the previous list.

4. Expected object information.

   This is a list of objects that the local EM world expects to receive from remote EMs. *Expected* differs from *required* in that required objects are solicited by the local EM while expected objects are not. If the local EM receives an object that is neither required nor expected, it is discarded.

### 3.2 Interaction Support

The MR Toolkit manages devices such as 3D position and orientation trackers, hand digitizers, and sound I/O using the client-server model. The EM interaction support simply states what devices are to be used, and automatically performs MR's data collection. EM also creates an instance of a special object called "body", which provides instance variables for the user's eye position and orientation, the position, orientation and finger shape of the hand(s), and tracker button state.

### 3.3 The VR World

This subpart of EM manages objects and instances, either *local* or *shared*, to form the content of the virtual world. EM loads in all the OML object prototypes, creates, positions and orients the instances within the environment, and builds the event table for all instance behaviors. The mapping from events in the environment to behaviors is handled at run-time. At each time step, EM checks the events in the event table to see if any behavior needs to be activated. If an event has occurred, EM activates the corresponding behavior, puts it on the active behavior list, and runs it at the next time step.

An EM world can receive shared objects in the form of OML code during the simulation. These objects are loaded into the interpreter by EM as they are received. Due to the interpreted nature of OML, an object can be loaded multiple times during the simulation, allowing incremental updates. An EM world can also receive shared instances in the form of an EM description file, which is parsed by the EM parser to create the ghosts of shared instances.

### 3.4 The Communication Handler

In BrickNet, the communication is based on the client/server architecture, whereas in EM the communication is peer-to-peer. We put the "server box" of BrickNet's client/server architecture down in each peer. From the command line, EM gets the information about the remote peer that the local application wants to connect to. The local process does not block while starting communications with a remote peer. All connection solicitation and other peer traffic takes place asynchronously, so any EM application may start up the peer package at any time.

EM parses shared data items from the description file, and allocates shared data storage for each item. EM handles application data updates based on the concurrency control scheme stated in the description file. The peer shared data facility also allows a callback to be defined whenever a shared data item arrives from a remote peer. The callbacks are used by EM to update the actual application variables and to activate the EM description file parser if a piece of EM description file code is received.

The peer package and EM maintain a complete graph connection topology by default, which means that each peer is connected to all other peers explicitly. Each peer has a list containing all the active remote peers, and it informs the connected peers when it receives a new remote connection. Any EM world may start up the peer package at any time and may initiate and quit communications with other EM worlds at will. A *quit* command is sent out when an EM world wants to quit, causing the recipient to delete this remote peer from its connected-peer list. EM defines a callback for the quit message, which deletes the shared instances and objects owned by the remote peer sending out the quit message.

### 3.5 Unique Instances

One of the main features of EM networked worlds is the ability to create multiple new instances while the simulation is running. It should be possible to communicate between instances regardless of the locations of the sender and the receiver. Each peer must be able to determine the instances associated with shared messages. Therefore EM assigns a unique identifier to an instance which can then be used to communicate with it.

### 3.6 Reducing Bandwidth

The main limitation on maximum performance for distributed VR systems is the bandwidth of the connections between processors in the system. It is necessary to reduce the communication between processors as much as possible. In EM, instances transmit only shared variables whose states have been changed. Messages are sent only to relevant EM worlds. Like SIMNET's dead reckoning, EM supports local simulation of the behavior of shared instances. EM also provides *quenching* and *unquenching* messages (similar to BrickNet [19]) to eliminate unnecessary communications, if an EM world decides to stop collaborative work for a period of time and does not wish to be informed of any updates for all imported objects during the break.

## 4 CONCURRENCY CONTROL

The replicated architecture needs concurrency control to resolve conflicts between participants' simultaneous operations. Concurrency control algorithms used in distributed database systems, such as explicit locking and transaction processing, are not appropriate for networked virtual environment under certain circumstances [10]. Networked VR systems and distributed systems are similar in that they are distributed over a network and they are shared by multiple users. However, a networked VR system is required to be *responsive* [18]. Under certain circumstances, responsiveness can be lost when *locking* (applied in DIVE) or a *centralized controller* (applied in BrickNet) are used.

We have found that different applications may need different concurrency control schemes. It is not necessary to find a

generic control scheme for every type of application; instead, we have implemented two schemes, and leave the choice of scheme to the users:

### 4.1 Simulation Ownership Token Passing

Some networked VR applications (e.g. the multi-player handball game [17]) are intended for situations where only one participant at a time owns the simulation and is active, while the other participants watch the simulation, and wait for their turns. We maintain consistency of a distributed virtual world database by restricting manipulation in such a way that only one site can perform operations to alter the status of the virtual world.

### 4.2 Ownership and Access Permissions

Instance or instance variable ownership indicates that only one EM world is permitted to have control over that value. The ownership may be fixed at inception, or it may shift between EM worlds as the application demands. For example, the ownership of a tank in SIMNET is never transferred, while the handball game transfers ball ownership.

As mentioned in section 3, an EM world may have local objects and instances which have no relation to other EM worlds, or it may export objects or instances to the network and share them with other EM worlds. Using ownership, we can restrict sharing in various ways. For example, an instance might be visible to other EM worlds, but updatable only by its current owner.

EM assigns access permissions to each instance shared variable, similar to file access permissions. EM defines two kinds of permission for a shared variable:

- Writable Permission
  Other interested EM worlds have permission to write to this shared variable. Every EM world is permitted to write its local shared variables.
- Readable Permission
  Other interested EM worlds have permission to read this shared variable, and the variable owner will send out the current value if necessary.

These concurrency control algorithms are managed and enforced by EM, and do not require special OML coding to implement. For example, if a local instance variable is writable, the OML interpreter does not know whether this local variable has been written by a remote EM. This allows for reuse of objects in both single and multiple user virtual environments without the need for rewriting the OML code for an object. Because instance variables are updated by EM before it calls the OML interpreter, all behaviors will run with the new values. External changes to instance variables will not occur in the midst of OML behavior execution.

To allow responsiveness, each EM world has its own copy of the simulation including the shared information. When an operation is requested (e.g. a button press causing an instance to move), the copy locally performs the operation immediately, which may cause an inconsistent state if another EM is also operating on this instance. Writable variables are the usual suspected cause of inconsistency, but this depends on the semantics of the variable.

One way to solve this problem is to use a technique called *operation transformations* [9], which allows responsiveness by executing local operations immediately. After the execution, the VR world sends out the operation, along with a *state vector* indicating how many operations it has recently processed from other VR worlds. Each VR world has its own state vector, which it compares to incoming state vectors. If the received and local state vectors are equal, the receiving operation is executed as requested; otherwise it is *transformed* before execution.

The specific transformation is dependent on the operation type (e.g., an *add* or *delete*) and on a log of operations already performed. However, it is not clear whether a user immersed in a virtual environment is happy with both *operation transformations* and being told the action he/she has made is ineffective, so we have not yet implemented this. In reality, some actions are irreversible.

Ownership token passing solves the concurrency problem by enabling a process to locally update a variable and then distribute the changes while holding the token. Unlike the ownership concurrency control in BrickNet [19] and the distributed locks in DIVE [5, 6], there is always one user who owns the instance or the variable token at any time. The current token owner is broadcast to every other user, so that whenever a user wants to modify an instance or a variable, he/she knows where the request should be sent before the actual operation is performed.

The effectiveness of a concurrency control scheme is highly application dependent. Each scheme may yield good performance for one application but poor performance for others. We have not yet made a formal study of the relative performance merits of these schemes.

## 5 EXAMPLES

We introduce three networked demonstrations built using EM: (1) a dynamic target shooting game, (2) a simple tank battle, and (3) the East Edmonton Mall design environment.

The shooting game allows networked players to shoot dynamic targets selected by the opposing players. One player shoots at a time, while being watched by the remaining players. The *simulation token passing* concurrency control is used in this demonstration.

In the Tank Battle demonstration (figure 3), each EM world represents a soldier owning a tank and its view-scope. The object space is the same across all the participants. Every soldier can enter or leave the battle simulation at any time. Each EM world broadcasts a tank instance (a ghost instance) representing the local tank to all the other EM worlds, so that each soldier can see all of the tanks which are currently in the battle. The instance ownership and access permission are used in this demonstration. The OML interpreter's collision detection facility generates collision events between tanks and enemy bullets, which in turn allows a bullet to update the tank's writable hit variable. The appendix shows the EM script for the red tank in this demonstration, and figure 3 shows two simultaneous screenshots from two users. The black tank's EM script is identical except for the fifth line,

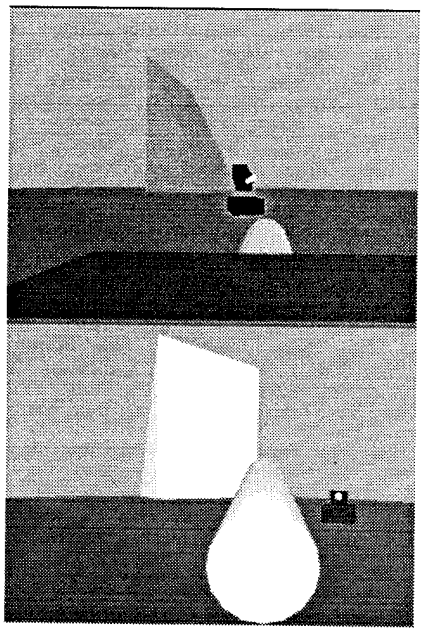


Figure 3: Simplified tank battle. In the upper picture, the black tank has just blown up. The lower picture is the black tank's simultaneous view of this.

which instances **tank2** at a different location and color.

The East Edmonton Mall Design Environment is representative of networked design systems that can be supported by EM. Unlike the previous two "same-content" networked VR worlds, this environment consists of three EM worlds, each of which has different objects and content.

The Airplane Museum is an EM world that contains a number of airplane objects on display to its user. The second EM world is the Sculpture Studio, which contains a collection of dynamic sculpture. The Mall Design Studio allows its user to design a building in which various sculptures and airplane models are to be placed. Because the Mall does not have its own sculptures or airplanes, it sends out a request to the network for the specified plane models and sculptures. The Plane Museum and the Sculpture Studio export their objects, along with the object behaviors onto the network upon receiving the request. The Mall Design Studio takes the received models, places them in the desired spots and shows their dynamic features (behaviors). Updates for these shared pieces are sent out by their owners whenever any new design idea has been applied.

## 6 DISCUSSION

To compare with other multi-player VR systems, SIMNET [1, 4], DIVE [5, 6] and VR-DECK [7] support only multi-user-same-content applications that have the same object space across all the users; whereas EM provides support for both multi-user-same-content and multi-user-different-content applications. EM enables the real-time sharing of object code, since like BrickNet [19], EM uses an interpreted object representation language.

In EM, messages are sent only to virtual worlds that have the affected object or instance and have expressed interest in receiving updates. This strategy is very different from broadcasting messages to all worlds whether the messages are relevant or not. The broadcasting approach leads to a significant amount of redundant network traffic in large-scale VR environments. Network bandwidth can also be reduced with EM dead-reckoning support.

DIVE uses multicast protocols to transmit messages. The MR Toolkit peers package does not currently use multicasting, but with the steady proliferation of the multicast backbone, the peer package can now be usefully re-implemented to use multicasting without the programmer being aware of the change. The only outstanding issue with multicast is a reliable multicast protocol. *Locking* is used in DIVE to prevent data inconsistency, EM provides a wider and more flexible concurrency control solution for users to choose.

The major difference between EM and BrickNet is that BrickNet uses the *client-server* arrangement for the creation of networked virtual worlds, while EM follows the *complete distribution* strategy. Complete distribution is more complex, but less prone to failure and provides less opportunity for communication bottlenecks to occur. BrickNet has several distributed cooperating servers running on high performance machines. EM takes this distribution trend one step further by allowing any EM process to function as a server.

## 7 CONCLUSIONS

The Environment Manager (EM), Object Modeling Language (OML), JDCAD+, and the Minimal Reality (MR) Toolkit provide a rich set of functionalities geared towards expediting the creation of both single-user and networked, multi-user virtual worlds. They eliminate the need to learn about low level graphics, device handling and network programming. This is achieved by providing higher level support for graphical, behavioral and network modeling of virtual worlds.

The MR Toolkit provides support for common VR devices and numerous interaction techniques. OML provides geometry and behavioral modeling for three dimensional objects used in virtual worlds. JDCAD+ is used to interactively create and animate dynamic 3D objects, automatically creating OML code and an associated EM script. EM sits on top of OML and MR, allowing for the easy construction of virtual environments. Instead of asking the developer to start from scratch, EM generates both single-user and networked virtual worlds using a simple script file where the device configuration, object and instance information (including *parameters*, *behaviors* and *network-shared information*) are specified.

Existing single-user OML applications that once required a C program to load OML code and start up the devices can now use an EM script. For example, an OML demo program that once required a 400 line C program now uses a 20 line EM script. EM's multi-user support radically simplifies the programming of networked virtual worlds by appropriately packaging the network facilities. Adding another user to a running multi-user virtual world is as easy as running EM from the command line with options to rendezvous with the appropriate remote machine, using a description file with

unique local instance names.

## REFERENCES


1. Earl A Alluisi. The Development of Technology for Collective Training: SIMNET, a Case History. *Human Factors*, 33(3):343–362, June 1991.

2. Brian Hughes Blau, Charles E. Moshell, and C Lisle. Networked Virtual Environment. *ACM SIGGRAPH - 1992 Symposium on Interactive 3D Graphics*, pages 157–160, 1992.

3. Frederick P. Brooks Jr. Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings. In *Proceedings 1986 Workshop on Interactive 3D Graphics*, pages 9–21, Chapel Hill, North Carolina, 1986. ACM SIGGRAPH.

4. James Calvin, Alan Dickens, Bob Gaines, Paul Metzger, Dale Miller, and Dan Owen. The SIMNET Virtual World Architecture. In *IEEE Virtual Reality Annual International Symposium (VRAIS 93)*, pages 450–455, Seattle, Washington, September 18-22 1993. IEEE.

5. Christer Carlsson and Olof Hagsand. DIVE - a Multi-User Virtual Reality System. *IEEE Virtual Reality Annual International Symposium (VRAIS 93)*, pages 394–400, 1993.

6. Christer Carlsson and Olof Hagsand. DIVE - a Platform For Multi-user Virtual Environment. *Computers and Graphics*, pages 663 – 669, 1993.

7. Christopher F Codella, Reza Jalili, Lawrence Koved, and J. Bryan Lewis. A Toolkit for Developing Multi-User, Distributed Virtual Environments. *IEEE Virtual Reality Annual International Symposium (VRAIS 93)*, pages 401–407, 1993.

8. Christopher F. Codella, Reza Jalili, Lawrence Koved, J. Bryan Lewis, Daniel T. Ling, James S. Lipscomb, David A. Rabenhorst, Chu P. Wang, Alan Norton, Paula Sweeney, and Greg Turk. Interactive Simulation in a Multi-Person Virtual World. In *Human Factors in Computing Systems CHI'92 Conference Proceedings*, pages 329–334, Monterey, California, May 1992. ACM SIGCHI.

9. C. A. Ellis and Simon J. Gibbs. Concurrency Control in Groupware System. *SIGMOD Record*, 18(2):399–407, 1989.

10. C. A. Ellis, Simon J. Gibbs, and G. L. Rein. Groupware: Some Issues and Experiences. *Communications of the ACM*, 34(1):39–58, 1991.

11. Mark Green. *Object Modeling Language (OML) Manual*. Department of Computing Science, University of Alberta, 1994.

12. Sean Halliday and Mark Green. A Geometric Modeling and Animation System for Virtual Reality. In Gurminder Singh, Steven K Feiner, and Daniel Thalmann, editors, *Virtual Reality Software and Technology (VRST 94)*, pages 71–84, Singapore, August 23-26 1994. World Scientific.

13. Larry F Hodges, Jay Bolter, Elizabeth Mynatt, William Ribarsky, and Ron van Teylingen. Virtual Environments Research at the Georgia Tech GVU Center. *PRESENCE: Teleoperators and Virtual Environments*, 2(3):234–243, 1994.

14. Jiandong Liang and Mark Green. JDCAD: A Highly Interactive 3D Modeling System. *Computers and Graphics*, 18(4):499–506, 1994.

15. Michael W McGreevy. The Presence of Field Geologists in Mars-Like Terrain. *PRESENCE: Teleoperators and Virtual Environments*, 1(4):375–403, 1992.

16. Warren Robinett and Michael Naimark. Artists Explore Virtual Reality: The Bioapparatus Residency at the Banff Centre for the Arts. *PRESENCE: Teleoperators and Virtual Environments*, 1(2):248–250, 1992.

17. Chris Shaw and Mark Green. The MR Toolkit Peers Package and Experiment. In *IEEE Virtual Reality Annual International Symposium (VRAIS 93)*, pages 463–469, Seattle, Washington, September 18-22 1993. IEEE.

18. Chris Shaw, Mark Green, Jiandong Liang, and Yunqi Sun. Decoupled Simulation in Virtual Reality with The MR Toolkit. *ACM Transactions on Information Systems*, 11(3):287–317, July 1993.

19. Gurminder Singh, Luis Serra, Willie Png, and Hern Ng. BrickNet: A Software Toolkit for Network-Based Virtual Worlds. *PRESENCE: Teleoperators and Virtual Environments*, 3(1):19–34, Winter 1994.

20. Michael Zyda, David Pratt, John Falby, Paul Barham, and Kristen Kelleher. NPSNET and the Naval Postgraduate School Graphics and Video Laboratory. *PRESENCE: Teleoperators and Virtual Environments*, 2(3):244–258, 1994.


## 8 APPENDIX

### EM Description File for the Tank Battle Demonstration

```
world tank_simulator
 simulation shared information

  round_robin off
  broadcast_instances
   instance Tank tank1 (10, 10, 0.5, (1, 0, 0), 0)
   actions
    collision ( hill_OBJ ) explosion
    collision ( Bomb ) got_it
    collision ( Tank ) explosion
    need_to_explode explosion
    tick marching
   end
   shared information
    heading double writable
    tx double writable dead_reckoning
```

```
      ty double writable dead_reckoning
      hit integer readable writable
      start_marching integer writable
    end
  end
   objects expected
    Tank
  end
end
object_files

    world.obj tank.oml.obj body.obj
    hill.obj bomb.obj
end
instance WORLD terrain (-99, 99, -99, 99, 30, 30)

instance test_body body

actions
        tick navigation
end
instance Tank tank1 (10, 10, 0.5, (1, 0, 0), 1)

actions
        tick signal_marching
        begin_marching marching
        tick turning_with_hand
        tick signal_stop
        collision ( hill_OBJ ) explosion
        collision ( Bomb ) got_it
        collision ( Tank ) explosion
        need_to_explode explosion
end
shared information
        tx double readable
        ty double readable
        hit integer readable writable
        heading double readable
        start_marching integer readable
end
instance Bomb bomb1

actions
        tick stay
        tick start_to_shoot
        need_to_bomb bombing
        collision ( hill_OBJ ) stop_shoot
        collision ( Tank ) stop_to_shoot
        need_to_stop stop_to_shoot
end
instance hill_OBJ hill_1 (50, 50, (0.9, 0.3, 0.8) )

instance hill_OBJ hill_2 (40, -30, (0, 1, 0.5) )

instance hill_OBJ hill_3 (-80, 70, (0.8, 1, 0.5) )

instance hill_OBJ hill_4 (-65, -75, (0.3, 0.4, 1) )

end
```

# BrickNet: Sharing Object Behaviors on the Net

*Gurminder Singh, Luis Serra, Willie Png, Audrey Wong, Hern Ng*

Institute of Systems Science
National University of Singapore
Kent Ridge, Heng Mui Keng Terrace
Singapore 0511.
+65 772-3651
{gsingh, luis, wpng, awong, nghern}@iss.nus.sg

**ABSTRACT**
In a majority of networked virtual worlds, object sharing is limited to object geometries only. The BrickNet toolkit extends the sharing of objects to include dynamic object behaviors. This is achieved by combining a structured organizational paradigm for virtual worlds with an interpreted language. Sharing in virtual worlds is handled by transferring the program code that builds the structure and executes the behavior. The range of behaviors that can be shared in BrickNet include simple behaviors, virtual world dependent behaviors, reactive behaviors and capability-based behaviors.

**KEYWORDS:** Virtual reality, network-based virtual worlds, client-server computing, object management

## 1.0 INTRODUCTION

Networked virtual worlds allow multiple virtual worlds connected on a network to share information with one another. This information describes the static geometrical structure of the worlds and its behavior, that is, the time-variant aspects of the worlds. In a majority of networked virtual worlds, object sharing is limited to object geometrical structure only. Our toolkit, BrickNet, extends the sharing of objects to include dynamic object behaviors. The range of behaviors that can be shared include simple behaviors, virtual world dependent behaviors, reactive behaviors and capability-based behaviors. Examples of simple behaviors include canned animations and linearly interpolated animations. In virtual world dependent behaviors, the behavior of the object depends on certain properties of the virtual world in which they reside. Reactive behaviors react to events generated by the user or other objects. For capability-based behaviors, their execution depends on the receiving client having a particular capability. This paper describes how object behaviors are shared among virtual worlds. For a general discussion of the BrickNet architecture, see [2].

BrickNet enables graphical objects to be maintained, managed, and used efficiently, and permits objects to be shared by multiple virtual worlds or "clients". A client can connect to a "server" to request objects of its interest. These objects are deposited by other clients connected to the same server or another server on the network (see Figure 1). Depending on the availability and access rights of objects, the server satisfies client requests.

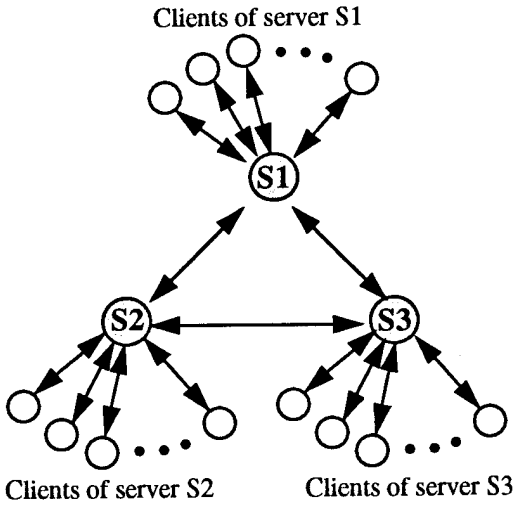



Figure 1. A BrickNet server handles several clients. Servers communicate with each other to satisfy client requests

BrickNet's object sharing strategy allows users to set-up their own private work-spaces, populated by shared and private objects. BrickNet virtual worlds are not restricted to sharing an identical set of objects; a virtual world manages its own set of objects, some or all of which may be shared with the other virtual worlds on the network. This basic arrangement can be used to implement several types of applications including collaborative, interactive learning systems (see Section 2), collaborative design environments [12], location-based entertainment (LBE) systems, concurrent engineering systems and other asynchronous network-based interactive, graphical environments.

BrickNet currently runs on a network of Silicon Graphics workstations. It has been implemented in the Starship [7]

and C programming languages. Starship is a general-purpose, interpretive, frame-based language. It provides both the object-oriented model and the frame model. The communication part of BrickNet has been implemented using UDP. Our I/O devices include Virtual Research EYEGEN 3 Head Mounted Display, Crystal Eyes stereo glasses, Ascension's The Bird, Logitech 6D mouse, and Immersion's Probe.

## 2.0 A QUICK EXAMPLE - COLLABORATIVE, INTERACTIVE LEARNING

To help understand the types of applications that BrickNet is aimed at, we describe a collaborative, interactive learning environment which has been built using BrickNet. This environment is representative of systems supported by BrickNet, although it has been simplified here for expository purposes.

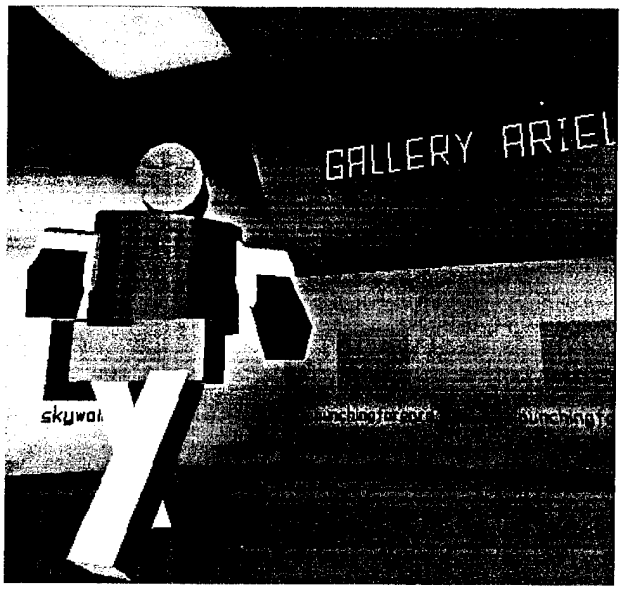



Figure 2. A student learning about mechanical joints and building objects in his virtual world.

Our example environment allows three students, Audrey, Willie and Ming, to build objects with mechanical joints and to share these objects with each other to understand how various types of joints work. An instructor on the network can enter the students' environments to examine the state of their work. Students have private workspaces of their own in which they designs objects they are interested in (see Figure 2). They can share objects, including object behaviors, with their classmates on the network. The types of behaviors that the students can use in the example system are:

**TABLE 1.** Types of Joints

| Type of Joint | Degrees of Freedom | Type of Freedom |
|---|---|---|
| Sliding | 1 | 1 Translation |
| Pin | 1 | 1 Rotation |
| Universal | 2 | 2 Rotations |
| Cylindrical | 2 | 1 Translation |
| | | 1 Rotation |
| Ball Socket | 3 | 3 Rotations |

The Sliding joint is useful for implementing piston-like movements whereas the Pin joint is for scissor-blade-like movements. An example of Universal joint can be found on the axle of a car while the Cylindrical joint can be found on the contractible drill. The human lower-arm is joined to the upper-arm by a BallSocket joint.

Students can request objects from their classmates by typing object names. To do so, they have to know the correct names of objects which they can get from the server. Alternatively, they can "enter" their classmates virtual worlds and bring back the (copies of) objects they like in their own virtual world. This is achieved by using "dynamic portals". Dynamic portals enable a user on the network to visualize the status of the current server-client configuration (refer to [12], for a discussion on dynamic portals). When the user hits the desired portal, the user ends up in the virtual world represented by the dynamic portal on the node. This is accomplished by transferring that virtual world's contents over the net to the current user who can then select the objects of interest. These can then be copied into the user's own virtual world (of course, whether an object can be copied or not, and the other permissions on the mode of copying, are controlled by the original owner of the object).

When Willie enters AudreysWorld and selects, for example Piston, Audrey's world transfers the Piston object to WilliesWorld. This way a shared Piston object is created in WilliesWorld. When Audrey modifies the Piston object, AudreysWorld sends updates to the BrickNet server, which then automatically forwards them to the WilliesWorld. AudreysWorld does not have to know about the clients that have its Piston object. The BrickNet server maintains this information and decides to forward updates to whichever client should get them. Since MingsWorld does not have the Piston object, it is not sent Piston updates. This filtering of messages is done by the BrickNet server.

The instructor can also enter the student's workspaces to inspect their objects and interact with them, in the same fashion that students enter each others workspaces to copy objects.

## 3.0 RELATED WORK

Multi-user virtual worlds can be divided into two groups. In the first group we put all those systems in which the content (or the set of objects) of all virtual worlds is the same. Individual users may be looking at different parts of the virtual space but they all have the same objects loaded in their worlds. In the second group, we put those systems in which virtual worlds are not restricted to sharing an identical set of objects. The objects loaded in a virtual world may differ from the objects loaded in another virtual world.

EM [13] is built on top of MR [10] and uses MR's Peer package [11] for communication between virtual worlds. In EM virtual worlds are described using OML [6], an interpreted C-like programming language for describing object geometries and behaviors. EM uses replicated architecture (each virtual world has a complete copy of the object set described in OML) and peer-to-peer communication between virtual worlds.

VR-DECK [5] allows multiple users to share a single simulation (or virtual world content). In VR-DECK, virtual worlds are created using a mixed object-oriented and event-based paradigm for defining system behavior. The modules (objects, operations, functions, and users) that make up a virtual world communicate with each other by producing and consuming events. Modules are defined at a high-level using rules written in C++ that determine how events are handled.

SIMNET [3] uses an object- and event-based approach to distributed, interactive virtual worlds for battlefield simulation and training. In SIMNET, virtual worlds consist of objects that interact with each other by broadcasting a series of events. An object initiating an event does not calculate which other objects might be affected by it. It is the receiving object's responsibility to determine whether the event is of its interest or not. To minimize communication processing and bandwidth requirements, objects transmit only changes in their behavior. Until an update is received, the new position of a remote (or a network-based) object is extrapolated from the states last reported by those objects.

NPSNET [9] uses as a subset of the DIS protocol for communication between virtual worlds. Early versions of NPSNET used the SIMNET protocol. DIS uses "remote entity approximation" rather than "dead reckoning" as used in SIMNET; dead reckoning uses constant velocity which has been found too limiting for modeling many types of military vehicles in NPSNET. Pratt et al. [8] discusses NPSNET's networking technology in detail.

VERN [1] is based on the networking technology of SIMNET. A distinguishing feature of VERN is its extensible object-oriented class hierarchy which abstracts communication and process control to the highest levels of the hierarchy.

DIVE [4] uses peer-to-peer communication to implement shared virtual worlds. A DIVE world consists of a set of objects to which DIVE processes can connect. DIVE processes connected to the same world are all identical in their content. If there are multiple DIVE worlds running on the network, a DIVE process can dynamically change worlds, but at any one time it can be a member of exactly one world.

VEOS [2] is an extendable, user-level framework for prototyping distributed VR applications. It facilitates coarse-grained parallelism by using heavyweight sequential processes, similar to UNIX processes. The VEOS application programmer's interface is provided by XLISP.

### 3.1 Relation to BrickNet

BrickNet allows multiple clients to share object geometries and behaviors with one another through BrickNet servers. BrickNet clients (usually) differ in their content from each other. Clients connected to different BrickNet servers can share objects through their servers.

Unlike VR-DECK, BrickNet does not force users to share a common object space, although such a scenario is possible to implement with BrickNet. Like VR-DECK, SIMNET, NPSNET and VERN all focus on supporting multi-user-same-content applications, although their software organization and communication mechanisms are different. In terms of object and behavior sharing, SIMNET behaviors are simpler but more continuous than BrickNet behaviors. This is the result of the different application focus of these systems: SIMNET is focussed at simulation-type applications whereas BrickNet is aimed at coarser-grained collaboration required in design or learning applications.

EM implements many of the same concepts in object sharing implemented in BrickNet, albeit using replicated architecture and peer-to-peer communication. In EM, object geometry and behavior sharing is facilitated by OML. OML seems to provide the same kind of power and flexibility as provided in BrickNet, but one has to learn a new language. In BrickNet, the main programming language, Starship, is used for everything from object definition to computation to sharing.

In DIVE, all processes (or clients) connected to a world (or server) are identical, but they can change their worlds dynamically. Once a process connects to a new world, it loses contact with its old world and switches its content completely to that of the new world. In BrickNet, clients cannot dynamically change their server, but they can share information across servers.

## 4.0 OVERVIEW OF BrickNet

This section provides a brief overview of BrickNet. For an in-depth discussion of the BrickNet architecture, see [12].

The BrickNet toolkit provides a rich set of functionalities geared towards expediting the creation of network-based virtual worlds. It eliminates the need to learn about low level graphics, device handling and network programming. This is achieved by providing higher level support for graphical, behavioral and network modeling of virtual worlds. Instead of asking the developer to start from scratch, as is the case with most other toolkits, BrickNet provides a "virtual world shell" which is customized by populating it with objects of interest, by modifying its behavioral properties, and by specifying the objects' network behavior. This makes it possible for the developer to create network-based virtual worlds quickly and easily.

BrickNet supports the layered architecture for network-based virtual worlds shown in Figure 3. A BrickNet server is designed to handle multiple clients in an asynchronous mode. A client executes autonomously and has a virtual world of its own. Client virtual worlds usually contain two types of objects: local and remote (or network-based). The local objects, as their names suggest, are local to the virtual world and not shared with other clients on the network. So the state and existence of such objects is not affected by the other clients on the network. Usually, local objects are used as background objects or as scratch objects. The network-based objects enable clients to share information with one another. Updates on the state of such an object are sent to all the clients who have the object and have expressed interest in updates on the object. In the rest of this section we will focus on network-based objects only.

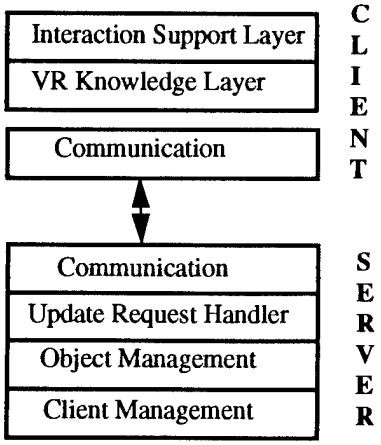



Figure 3. Architecture of BrickNet Clients and Servers

A server can be seen as an object request broker and a communication manager. As an object request broker, it enables objects to be maintained, managed, and used efficiently. A server also permits objects to be shared by multiple clients. It keeps track of client requests for objects and object updates, and services these requests when possible. As a communication manager, a BrickNet server keeps track of clients' status (active or passive), their addresses and port information, and manages the sending and receiving of packets of information.

### 4.1 Client Layers

The Interaction Support Layer is the user interface part of the virtual world. The user interacts with the virtual world through devices and interaction techniques that are part of this layer. For example, this layer decides how a virtual object is picked and moved.

The VR Knowledge Layer manages objects that form the "content" of the virtual world. In a network-based virtual world, most of these objects are obtained from the server, and hence are network-based. The knowledge layer assumes that a person (the VR end-user) is moving and interacting within a world populated by solid objects that are governed by a set of physical and environmental laws. Each of these components is represented by a class which is interconnected to the other classes by means of semantic links such as is-part-of. Based on these classes, a VR application is constructed by first selecting the components of the world, assigning them behaviors, choosing a set of laws and then running the simulation loop.

### 4.2 Server Layers

The client management layer in the server is responsible for maintaining client-specific information. When a client is registered with the server, a new record for the client is created and managed by this layer.

The object management layer keeps track of objects deposited by various clients and object access rights and updates. When a client makes a request for an object, this layer sends the object if access rights permit.

The next layer, called the update request handler, manages clients' update requests. This layer ensures that clients have the correct permissions for updates. For example, if a client sends a position update for an object, this layer goes through the object database maintained by the object management layer and sends the update to other clients who may have expressed interest in such updates.

The communication layer in BrickNet servers implements the communication protocol to receive messages from, and to send messages to, clients.

## 5.0 SHARING OBJECT BEHAVIORS

As noted earlier, BrickNet allows clients to not only share object geometries but object behaviors as well. The types of behaviors that can be shared among BrickNet clients include simple behaviors, environment dependent behaviors, reactive behaviors and capability-based behaviors. The following sections describe these behaviors and give examples of their use.

Behavior sharing in BrickNet is implemented by transferring the Starship program code that implements

behaviors. When an object is transferred, not only its geometry but also methods generating its behavior are transferred. Since Starship is an interpretive language, our object behaviors can be as complex as needed. Because Starship is our main programming language, the programmer does not have to learn a new object or behavior language such as OML [6] in EM [13].

The structured organizational paradigm for virtual worlds in BrickNet combined with its interpreted nature is ideal for networking operation: every object in the VR Knowledge Layer (see Figure 3) can be transferred easily as Starship code to be interpreted at the receiving end. In this way, object structure and procedural behavior can be duplicated with minimal effort from station to station.

This sharing can take advantage of the inheritance in BrickNet: behaviors can be shared either from the classes themselves (and thus becoming applicable to all existing and subsequent instances of that class), or from particular instances (and thus being applicable to that instance alone).

BrickNet's object structure is key to sharing complex behaviors between and within objects. Hierarchies are constructed by means of 'cloning' procedures, which include both structure and behavior methods. Inverse kinematics are supported in a similar fashion, by encapsulating the structure of the closed loops relating the objects into a transferable method, and then invoking a resident generalized equation solver for run-time solution.

In the following sections we will give examples of the different types of behaviors available in BrickNet. The behavior code will be placed in the `animate` method, which is called once per loop during the execution of the program.

### 5.1 Simple Behaviors

The simplest behavior that can be shared in BrickNet is based on repetitive, unconstrained changes. Simple behaviors are not synchronized between clients: once the object has been transferred to a receiving virtual world, there is no communication between the sender and the receiver. A slightly more complex type of behavior uses sequencing tables of position/orientation/size triplets which the object has to follow. Constrained animations of the type described in Section 2 (Sliding, Cylindrical, Ball, Socket, Universal, and Pin) are implemented procedurally.

The following program example describes a Piston. The Piston is first instanced from `SolidClass` and then its geometry is created. In this particular example, the geometry is generated by executing code rather than by reading from a file. After that, a simple back and forth motion procedure is described within the `animate` method. In this simple example, the motion of the piston translates by a fixed amount every iteration (1 unit in the 'y' direction, between +10 and -10 unit boundaries).

Notice that the procedural approach to geometry construction used in this example could in itself serve as an animation technique, although it is not used for that purpose here.

```
(Piston SolidClass).proto {

%% Create Piston geometry procedurally

  :geometry #geometry {
  % Add the Piston head
    + (#geometry {
      :shape cylindrical;
      :material (#material orange);
      scaleTo #vector[10 10 10];
      translate #vector[20 -10 -20];
    }); % geometry
  % Add the Piston handle
    + (#geometry {
    :shape cylindrical;
    :material (#material yellow);
    scaleTo #vector[30 10 10];
  });}; % geometry

%% Translate Piston by 1 unit per iteration

method animate declare
[{arg self}
 self:upVector   @vector[0 1 0];
 self:downVector @vector[0 -1 0];
 self:topLimit @vector[0 10 0];
 self:lowLimit @vector[0 -10 0];
 self.position > (self.topLimit) then [
   self:jumpVector (self.downVector);
 ] else [
  self.position < (self.lowLimit) then [
    self:jumpVector (self.upVector);
  ];
 ]; % else
 self translateBy (self.jumpVector);
]; % method animate

};% PistonSolidClass proto
```

### 5.2 Environment-Dependent Behaviors

Environment-dependent behaviors take into account the clients' run-time environment: machine configuration, time of execution, etc. This class of behaviors changes object properties depending on attribute values encountered in the world they populate.

The piston described in Section 5.1 is an example where environment dependent behaviors prove useful. In this example, the piston translates by a fixed amount (1 unit) every iteration. Here the piston speed is dependent on the frame rate of the client; for clients running at a high frame rate, the piston would appear to move faster than in clients running at a lower frame rate. To correct this undesired behavior, the client should figure out the amount of translation per iteration required in order to maintain an

equivalent playback speed to the originating virtual world. This can be achieved by either sending to the client the machine configuration of the origination world, or by encoding in the animate method the playback speed, instead of a fixed translation per iteration. The latter approach is described in the following example:

```
%% Translate Piston 1 unit/sec;
%% Client environment variable
%% 'env.iterationRate' indicates performance
%% of 10 frames per second

method animate declare
[{arg self}
 self:jumpRate 1; % 1unit/second
 self.jumpRate =
   (self.jumpRate / env.Iterationrate);
 self.jumpVector =
   (@vector[0 1 0] / self.jumpRate);
 self:upVector (self.jumpVector);
 self:downVector -(self.jumpVector);
 self:topLimit @vector[0 10 0];
 self:lowLimit @vector[0 -10 0];
 self.position > (self.topLimit) then [
   self:jumpVector (self.downVector);
 ] else [
  self.position < (self.lowLimit) then [
    self:jumpVector (self.upVector);
  ];
 ]; % else
 self translateBy (self.jumpVector);
]; % method animate
```

This animation method computes the rate of translation of the Piston based on the current value of the iterationRate, which can vary with system load.

A more complex example of an environment dependent behavior would be jumping of an object based on the elasticity of the walls in the receiving virtual world.

### 5.3 Reactive Behaviors

Reactive behaviors react to events generated by the user or other objects. Each object checks for the occurrence of events that have behavior functions associated with them. When such an event is detected, its associated behavior function is executed. Using this facility one could for example implement a behavior that shakes an object when the user touches it.

```
method animate declare
[{arg self}
  (self.position near
   self.world.person.hand.position) then [
    self shake;
  ];
];
```

### 5.4 Capability-Based Behaviors

In BrickNet, it is possible to share behaviors whose execution depends on the receiving virtual world having a particular capability. For example, for a shared treasure box, the receiving virtual world must have the key that can open it to see what is inside the box. This kind of capability-based-behavior sharing can be used to implement semantically interesting scenarios. In one of our demonstrations, a robot needs two legs to start walking. As soon as both legs are attached to the robot, its walking behaviors becomes active.

Checking for capabilities is implemented by using pre-conditions. Unless the pre-conditions for a behaviors are satisfied, the behavior cannot be executed. Behavior pre-conditions are coded in the Starship programming language and can be as complex as required.

## 6.0 SYNCHRONIZING OBJECTS

In simulation applications, there is often a need to synchronize object states among clients. BrickNet provides a finer grain of synchronization than most other systems. When a BrickNet client, who owns an object, wants to ensure that other clients who have leased out the object are synchronized with it, it sends a sync message to the server. The server in turn checks with each client who has leased out the object. These clients then send a sync-acknowledgment to the server. The server, after it has received acknowledgments from all relevant clients, sends a sync-confirm message to the requesting client.

Object behaviors in BrickNet are synchronized by the client that controls the shared object sending a message that contains synchronizing information for the clients that have leased out the object. The receiving clients then use the information to change the state of object behavior. Using this strategy, it is possible to implement a variety of synchronization algorithms including dead-reckoning. Due to the fact that BrickNet behaviors can be as complex as desired (and not just position and velocity as in DIS), synchronized behaviors can be used to implement advanced simulations.

## 7.0 DISCUSSION

We have found BrickNet's structured organization of virtual worlds combined with its interpreted nature suitable for building a variety of networked virtual worlds. Objects in BrickNet's VR Knowledge Layer can be easily transferred as code to be interpreted at the receiving end. In this way, object geometry and behavior can be duplicated easily and rapidly from station to station resulting in powerful and flexible object sharing in networked virtual worlds.

BrickNet's approach to behavior sharing requires that all virtual worlds be able to execute the program code that gets transferred. It is possible to do away with this requirement by identifying a set of high-level behavior primitives (similar to animation messages used in [14]) and encoding shared behaviors in these primitives. Individual virtual worlds could then implement the identified set of behavior primitives in their programming language of choice. This approach provides more "portability" at the cost of both flexibility and power of behaviors that can be shared. By identifying a set of behavior primitives, we restrict the range of shared behaviors to the ones that can be coded in this set. The BrickNet approach on the other hand is more general and allows arbitrarily complex behaviors to be shared among virtual worlds.

In addition to the collaborative, interactive learning system described in secrion 2, we have developed a location-based entertainment system and a collaborative design system using BrickNet. The range and power of shared object behaviors supported by BrickNet has made it possible for us to easily and rapidly implement these systems.

**References**


1. Blau, B., Hughes, C.E., Moshell, J. M., & Lisle, C. (1992). Networked Virtual Environments. *Proc. 1992 Symp on Interactive 3D Graphics*, Cambridge, Massachusetts, 29 March - 1 April 1992, pp: 157-160.

2. Bricken, W., & Coco, G. (1994). The Veos Project. *Presence: Teleoperators and Virtual Environments*, MIT Press, Boston, 3(2), 111-129

3. Calvin, J., Dicken, A., Gaines, B., Metzger, P., Miller, D., & Owen, D. (1993). The SIMNET Virtual World Architecture. *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS'93)*, Sept. 18-22, 1993, Seattle, Washington, USA, pp: 450-455

4. Carlsson, C. & Hagsand, O. (1993). DIVE - a Multi-User Virtual Reality System. *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS'93)*, Sept. 18-22, 1993, Seattle, Washington, USA, pp: 394-400

5. Codella, C., Jalili, R., Koved, L., & Lewis, B.J. (1993). A Toolkit for Developing Multi-User, Distributed Virtual Environments. *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS'93)*, Sept. 18-22, 1993, Seattle, Washington, USA, pp: 401-407

6. Green, M. (1994) *Object Modeling Language (OML), Version 1.1*, Dept. of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1.

7. Loo, P.L. (1991). The Starship Manual (Version 2.0), ISS TR#91-54-0. Institute of Systems Science, National University of Singapore, Kent Ridge, Singapore 0511.

8. Pratt, D.R., Locke, J., Macedonia, M.R., Zeswitz, S.R., Young, R.D., & Zyda, M. (1994) NPSNET: Ensuring World Consistency in a Shared Networked Virtual Environment. *Proc. Networked Reality'94*, May 13-14, 1994, NTT Media Lab, Tokyo, Japan, 11 pages.

9. Macedonia, Michael R., Zyda, Michael J., Pratt, David R., Barham, Paul T. & Zeswitz, Steven (1995) NPSNET: A Network Software Architecture for Large Scale Virtual Environments. *Presence: Teleoperators and Virtual Environments*, MIT Press, Boston, 3(4), (to appear)

10. Shaw, C., Green, M., Liang, J., & Sun, Y. (1993). Decoupled Simulation in Virtual Reality with the MR Toolkit. *ACM Transactions on Information Systems*, 11(3), 287-317

11. Shaw, C. & Green, M. (1993). The MR Toolkit Peers Package and Experiment. *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS'93)*, Sept. 18-22, 1993, Seattle, Washington, USA, pp: 463-469

12. Singh, G., Serra, L., Png, W. & Ng, H. (1994) BrickNet: A Software Toolkit for Network-Based Virtual Worlds. *Presence: Teleoperators and Virtual Environments*, MIT Press, Boston, 3(1), 19-34

13. Wang, Q. (1994) *Networked Virtual Reality*. M.Sc. Dissertation, Dept. of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1.

14. Zeleznik, R.C. et al (1991) An Object-Oriented Framework for the Integration of Interactive Animation Techniques. *Computer Graphics (Proc. SIGGRAPH'91)*, 25(4), 105-111

# Human Factors

# Realizing the Full Potential of Virtual Reality: Human Factors Issues That Could Stand in the Way

*Kay Stanney*
University of Central Florida
Industrial Engineering & Management Systems Department
4000 Central Florida Blvd.
Orlando, FL 32816
(407) 823-5582
stanney@iems.engr.ucf.edu

**ABSTRACT**
This paper reviews several significant human factors issues that could stand in the way of virtual reality realizing its full potential. These issues involve maximizing human performance efficiency in virtual environments, minimizing health and safety issues, and circumventing potential social issues through proactive assessment.

**KEYWORDS:** human factors, human-virtual environment interaction

## INTRODUCTION

Virtual reality (VR) technology will be used to advance many fields, including medicine, education, design, training, and entertainment. The reality is, however, a considerable amount of systematic research must be done before VR technology receives widespread use [10]. If VR systems are to be effective and well received by their users, researchers need to focus significant efforts on addressing a number of human factors issues [26]. This paper provides an overview of many of these human factors issues, including: human performance efficiency in virtual worlds; which is likely influenced by tasks characteristics, user characteristics, human sensory and motor physiology, multi-modal interaction, and the potential need for new design metaphors; health and safety issues, of which cybersickness may pose the most concern; and the social impact of the technology.

## HUMAN PERFORMANCE EFFICIENCY IN VIRTUAL WORLDS

Computer speed and functionality, image processing, synthetic sound, and tracking mechanism have been joined together to provide realistic virtual worlds. A fundamental advance still required for virtual environments (VEs) to be effective is to determine how to maximize the efficiency of human task performance in virtual worlds. While it is difficult to gauge the importance of the various human factors issues requiring attention, it is clear that if humans cannot perform efficiently in virtual environments, then further pursuit of this technology may be fruitless. Focusing on understanding how humans can perform most effectively in VEs is thus of primary importance in advancing this technology.

Human performance in VEs will likely be influenced by several factors, including: task characteristics; user characteristics; design constraints imposed by human sensory and motor physiology; integration issues with multi-modal interaction; and the potential need for new visual, auditory and haptic design metaphors uniquely suited to virtual environments.

### Task Characteristics

One important aspect that will directly influence how effectively humans can function in virtual worlds is the nature of the tasks being performed. Some tasks may be uniquely suited to virtual representation, while others may not be effectively performed in such environments. It is important to determine the types of tasks for which VEs will be appropriate. In order to obtain this understanding the relationship between task characteristics and the corresponding virtual environment characteristics which effectively support their performance (e.g., stereoscopic 3D visualization, real-time interactivity, immersion, etc.) must be attained.

While there is limited research on the types of task characteristics that are uniquely suited to human-virtual environment interaction (HVEI) (a notable exception is [29]), there is extensive literature on task characteristics in general. In order to identify tasks which are appropriate for virtual environment training, this body of knowledge on task characteristics must be explored and its relation to virtual task performance needs to be identified. For example, task characteristics which lend themselves to

perceptual understanding through three-dimensional visualization in a virtual world should be distinguished. Bennett, Toms, and Woods (1993) research supports the use of such 3D displays for tasks requiring information integration. On the other hand, focused attention tasks tend to be more effectively performed using 2D displays. Thus, displaying such tasks in 3D stereoscopic virtual worlds could potentially hinder performance.

Task characteristics which are suitable for representation as displayable virtual objects which can be manipulated through perceptual and motor processes also need to be determined. For example, Sollenberger and Milgram (1993) found optimal path tracking performance when using a 3D, rotating, stereoscopic display. Texturing, the surface rendering available on virtual objects, has been found to be effective for representing additional data dimensions, such as emergent features. These relationships need to be further explored in order to clearly delineate the specific characteristics of virtual worlds which support and enhance task performance as compared to other visualization approaches such as real-time simulations, animations, and non-interactive three-dimensional visualizations.

A taxonomy of virtual task characteristics would be instrumental in providing designers with a tool to guide and direct their design efforts in order to maximize human performance. Such a tool would classify tasks according to the types of information displays (e.g., 2D, 3D stereoscopic, point, line, angle, area, volume, etc.) and interactions (e.g., passive, enactive, interactive) which maximize human-performance efficiency in virtual worlds. The influences of user characteristics (e.g., high versus low spatial individuals) would also need to be considered. Such a taxonomy could assist in guiding VE designers by imposing order [23] on the complex interactions between user, task, and system phenomena.

**User Characteristics**

An important aspect influencing human VE performance is the affect of user differences. Significant individual performance differences have already been noted in early studies [13]. User characteristics that significantly influence VR experiences need to be identified in order to design VR systems that accommodate the unique needs of users. In order to determine which user characteristics are influential in VEs one can examine studies in human-computer interaction (HCI). In HCI one of the primary user characteristics which interface designers adapt to is level of experience. Experience level influences the skills of the user, the abilities which predict performance, and the manner in which users understand and organize task information [6]. In examining the influences of experience on HVEI, one could thus predict that experience would influence the skill with which users interact with the VE and the manner in which users mentally represent a virtual environment over time. The implication being that designers must design the VE interface to be appropriate for the level of expertise of the target user population. Understanding what is an "appropriate" VE interface for novices versus experts is a challenge.

Technical aptitudes (e.g., spatial visualization, orientation, spatial memory, spatial scanning) are generally significant in predicting HCI performance [6]. These studies indicate that individuals who score low on spatial memory tests generally have longer mean execution times and more first try errors. These studies also suggest that the difficulties experienced by low spatial individuals are particularly related to system navigation issues -- users often report being "lost" within hierarchical menu systems [21]. These findings are particularly relevant to VEs which may often place a high demand on navigation skills. In fact, users are already known to become lost in virtual worlds [18]. The issue is thus how to assist low spatial users with maintaining spatial orientation within virtual worlds. New design metaphors could potentially be developed to assist with this issue.

Other aptitudes, such as verbal and motor ability, and traits, such as personality, that have not been found to consistently predict human computer performance [6], may become more influential during HVEI. Particularly with the emphasis on audio and haptic interaction modes in VEs [10, 14], it is essential that human factors analysis be devoted to understanding the influences of these other aptitudes on HVEI.

**Design Constraints Imposed by Human Sensory and Motor Physiology**

In order for designers to be able to maximize human efficiency in VEs it is essential to obtain an understanding of design constraints imposed by human sensory and motor physiology. Without a foundation of knowledge in these areas there is a chance that the multi-modal interactions provided by VE systems will not be compatible with their users. Such design incompatibilities could place artificial limits on human VE performance. VE design requirements and constraints aimed at maximizing human VE performance should thus be developed by taking into consideration the abilities and limitations of humans [10]. The physiological and perceptual issues which directly impact the design of multi-modal VEs, include: visual perception, auditory perception, and haptic perception.

*Visual Perception.* The design of visual presentations for VEs is complicated because the human visual system is

very sensitive to any anomalies in perceived imagery, especially in motion scenes [14]. During virtual motion scenes, minute, nearly imperceptible scene anomalies become dreadfully apparent because of the unnatural appearance of visual flow field cues [10]. In order to avoid this issue, more research is needed to develop guidelines that assist designers in fabricating approximate optical flow patterns. In general, human visual perception needs to be better understood in order to ensure that the most effective visual scenes are developed for virtual worlds.

It is also important to determine what a viewer can see in a VE, that is to determine the viewer's visual field when wearing a Head Mounted Display (HMD). In order to determine exactly what individuals can see in HMDs, visual field graphical dimensions must be overlaid onto obscuration plots imposed by HMDs. HMDs substantially reduce the field of view (FOV) of a user, thus obscuring the perception of motion in the peripheral vision. Current systems are generally limited to a FOV of 70 degrees per eye and do not provide peripheral vision [14]. Kalawsky (1993) has suggested, but not yet proven, that many virtual tasks will require FOVs of 100 degrees or more in order to achieve immersive environments. These suggestions needed to be further studied in order to determine what FOV is required to perform different kinds of virtual tasks effectively. Then the extent to which FOVs need to be enlarged can be specified.

*Auditory Perception.* In order to synthesize a realistic auditory environment it is important to obtain a better understanding of how the ears receive sound, particularly focusing on 3-D audio localization. Although it is known that audio localization is primarily determined by intensity differences and temporal or phase differences between signals at the ears, such localization is affected by the presence of other sounds and the direction from which these sounds originate [10]. In addition, while auditory localization is understood in the horizontal plane (left to right), localization in the median plane (intersection between front and back) and discrimination of sounds from front to back are not well understood. Thus, much work is needed in order to effectively synthesize 3D auditory environments.

In order to study 3-D audio localization, binaural localization cues received by the ears can be represented by a Head Related Transfer Function (HRTF), phase differences, and overtones [5]. The HRTF represents the manner in which sound sources change as a listener moves his/her head and can be specified with knowledge of the source position and the position and orientation of the head. Personalized HRTFs may need to be developed because these functions are dependent on the physiological makeup of each individual listener's ear. Ideally, a more generalized HRTF could be designed that would be applicable to a multitude of users.

*Haptic Perception.* A haptic sensation (i.e., touch) is a mechanical contact with the skin [10]. Three mechanical stimuli produce the sensation of touch: a displacement of the skin over an extended period of time; a transitory (few millisecond) displacement of the skin; and a transitory displacement of the skin which is repeated at a constant or variable frequency. Even with this understanding of global mechanisms, however, the attributes of the skin are difficult to characterize in a quantitative fashion. This is due to the fact that the skin has variable thresholds for touch (vibrotactile thresholds) and can perform complex spatial and temporal summations which are all a function of the type and position of the mechanical stimuli. So as the stimulus changes so does the sensation of touch, thus creating a challenge for those attempting to model synthetic haptic feedback.

Another haptic issue is that the sensations of the skin adapt with exposure to a stimuli. More specifically, the effect of a sensation decreases in sensitivity to a continued stimulus, may disappear completely even though the stimulus is still present, and varies by receptor type. Surface characteristics of the stimulus (e.g., hard, soft, textured) also influence the sensation of touch.

In order to communicate the sensation of synthetic remote touch it is thus essential to have an understanding of: the mechanical stimuli which produce the sensation of touch; the vibrotactile thresholds; the effect of a sensation; the dynamic range of the touch receptors; and the adaptation of these receptors to certain types of stimuli. The human haptic system needs to be more fully characterized, potentially through a computational model of the physical properties of the skin, in order to generate synthesized haptic responses.

**Integration Issues with Multi-Modal Interaction**

While developers are focusing on synthesizing effective visual, auditory, and haptic representations in virtual worlds, it is also important to determine how to effectively integrate this multi-modal interaction. One of the aspects that makes VEs unique from other interactive technologies is its ability to present the user with multiple inputs and outputs. This multi-modal interaction may be a primary factor that leads to enhanced human performance for certain tasks presented in virtual worlds. Early studies have already indicated that sensorial redundancy can enhance human performance in virtual worlds [16]. There is currently, however, a limited understanding on how to effectively provide such sensorial parallelism [3]. When

sensorial redundancy is provided to users it is essential to consider the design of the integration of these multiple sources of feedback. One means of addressing this integration issue is to consider (1) the coordination between sensing and user command and (2) the transposition of senses in the feedback loop.

Command coordination considers the user input as primarily mono-modal (e.g., through gesture or voice) and feedback to the user as multi-modal (i.e., any combination of visual, auditory, and/or haptic). There is limited understanding on such issues as (1) is there any need for redundant user input (e.g., voice and direct manipulation used to activate the same action); (2) can users effectively handle parallel input (e.g., select an object with a mouse at the same time as directing a search via voice input); and (3) for which tasks is voice input most appropriate, gesture most appropriate, and direct manipulation most appropriate.

Sensorial transposition occurs when a user receives feedback through other senses than those expected. This may occur because a VE designer's command coordination scheme has substituted unavailable system sensory feedback (e.g., force feedback) with other modes of feedback (e.g., visual or auditory). Such substitution has been found to be feasible (e.g., Massimino and Sheridan (1993) successfully substituted vibrotacticle and auditory feedback for force feedback in a peg-in-hole task). VE designers thus need to establish the most effective sensorial transposition schemes for their virtual tasks. The design of these substitutions schemes should be consistent throughout the virtual world to avoid sensorial confusion.

### Virtual Environment Design Metaphors

It is known that well-designed metaphors can assist novice users in effectively performing tasks in human-computer interaction [4]. Thus, designing effective VE metaphors could similarly enhance human performance in virtual worlds. Such metaphors may also be a means of assisting in the integration of multi-modal interaction. For example, affordances may be designed that assist users in interacting with the virtual world much as they would interact with the multi-modal real world. Unfortunately, at the present time many human-VE interface designers are using old metaphors (e.g., windows, toolbars), that may be inappropriate for HVEI.

Oren (1990) suggested that every new technology goes through an initial incunabular stage, where old forms continue to exist which may not be uniquely suited to the new medium. Currently, virtual technology appears to be in such a stage. For example, many users of virtual environments don their high tech helmet and gloves and enter the virtual world only to find floating menus awaiting them! Virtual environments are in need of new design metaphors uniquely suited to their characteristics and requirements.

McDowall (1994) has suggested that the design of interface metaphors may prove to be the most challenging area in VE development. VR sliders (3D equivalents of scroll bars), map cubes (3D maps which show space in a viewer's vicinity), and tow planes (where a viewer's navigation is tied to a virtual object which tows him/her about the VE) are all being investigated as potential visual metaphors for virtual environments.

Beyond the need for new visual metaphors, VEs may also need auditory metaphors which provide a means of effectively presenting auditory information to users. Cohen (1992) has provided some insight into potential auditory metaphors through the development of "multidimensional audio windows" or MAW. MAW provides a conceptual model for organizing and controlling sound within traditional window-icon-menu-pointing device (WIMP) interfaces. In addition, Hahn, Gritz, Darken, Geigel, and Won Lee (1993) have developed the concept of 'timbre trees' which are general representations of sound. Hahn et al. (1993) suggest that timbre trees can be used as a construction methodology for representing any new synthetic sound.

Metaphors for haptic interaction may also be required. Limited work has been done in this area to date and no noted haptic metaphors have been presented.

## HEALTH AND SAFETY ISSUES IN VIRTUAL ENVIRONMENTS

Maximizing human performance in VEs is essential to the success of this technology. Of equal importance is ensuring the health and welfare of users who interact with these environments. If the human element in these systems is ignored or minimized it could result in discomfort, harm, or even injury. It is essential that VE developers ensure that advances in VE technology do not come at the cost of human well being.

There are several health and safety issues which may affect users of VEs. These issues include both direct and indirect affects [27]. The direct effects can be looked at from a microscopic level (e.g., individual tissue) or a macroscopic level (e.g., trauma). The indirect effects are primarily psychological.

There are several microscopic direct effects which could affect the tissues of VE users. The eyes, which will be closely coupled to HMDs or other visual displays used in

VEs, have the potential of being harmed. The central nervous system (CNS) could be affected by the emfs of VE systems.

Some individuals are susceptible to "flicker vertigo" -- when they are exposed to flickering lights (usually in the range of 8 to 12 Hz) they experience a seizure. VE displays flickering at this rate could lead to a seizure in a few users, even in some unaware that they have the condition.

Phobic effects may result from VE use, such as claustrophobia (e.g., HMD enclosure) and anxiety (e.g., falling off a cliff in a virtual world). Viirre (1994) suggests, but has yet to prove, that no long term phobic effects should result from HVEI, except potential avoidance of VE exposure.

The auditory system and inner ear could be adversely affected by VE exposure to high volume audio (e.g., the "Walkman" effect). One of the possible affects of such exposure is noise induced hearing loss. Prolonged repetitive VE movements could also cause overuse injuries to the body (e.g., carpal tunnel syndrome, tenosynovitis, epicondylitis). The head, neck and spine could be harmed by the weight or position of HMDs [10, 27].

Limited or eliminated vision of natural surroundings when wearing HMDs could lead to falls or trips that result in bumps and bruises. Sound cues may distract users causing them to fall while viewing virtual scenes. Imbalance of body position may occur due to the weight of VE equipment or tethers that link equipment to computers causing users to fall [26, 27]. Obstacles in the real world, that may not be visible in the virtual world, could pose a threat to the safety of users. If haptic feedback systems fail a user might be accidentally pinched, pulled or otherwise harmed. Another direct macroscopic effect that could prevent VR from realizing its full potential is that many users of VEs experience motion sickness (i.e., cybersickness). Such sickness may prevent users from seeking further VE interactions.

The use of VEs may produce disturbing after-effects, such as head spinning and delayed onset of sickness. Delayed effects from virtual experiences must be investigated in order to ensure the safety of users once interaction with a virtual world concludes.

If a system fails, the sudden disruption of "presence" may cause disorientation, discomfort, and/or harm. Finally, psychological or emotional well-being could be negatively influenced by VE interaction (e.g., addiction, transfer- of-training from violent VEs). All of these health and safety issues must be addressed in order to ensure the well being of users interacting with virtual worlds.

**Cybersickness**

One of the most important health and safety issues that may influence the advancement of VE technology is cybersickness. Cybersickness (CS) is a form of motion sickness that occurs as a result of exposure to VEs. Cybersickness poses a serious threat to the usability of VE systems. Users of VE systems generally experience various levels of sickness ranging from headaches to severe nausea [10]. Although there are many suggestions about the causes of motion sickness, to date there are no definitive theories of cybersickness. Research needs to be done in order to identify the specific causes of CS and their inter-relationships in order to develop methods which alleviate this malady. If CS is not adequately addressed, many individuals may avert VE experiences in order to avoid becoming sick.

Motion sickness is considered to be the product of a cue conflict acting upon the visual and/or vestibular systems [9]. The user's body perceives this conflict as a poison and attempts to remove this "poison" by making itself sick [19]. Motion sickness may manifest itself in the form of headaches, blurred vision, salivation, burping, eye strain, dizziness, vertigo, disorientation, or even severe vomiting. It has been shown that between 10 to 60% of users demonstrate some form of simulator sickness [12]. For those who do become sick, research has shown that CS may prevent a person from wanting to reenter a virtual world [1]. Currently, however, system developers cannot prevent such sickness from occurring because the exact causes of motion sickness are not well defined.

While it is known that users adapt to VE experiences and become less sick over time [8], the first impressions of users may influence their attitudes towards this technology. If users become very ill during their initial experience, they may avoid future VE interactions. Relying on adaptation alone as a remedy for CS may thus not prove effective.

There have been several studies focused on understanding the factors that may contribute to motion sickness (e.g., vection, lag, field of view, etc.), yet no general theory of motion sickness has resulted from this research. In fact, contradictory evidence among the existing studies leads to skepticism about the actual impact of each of these factors. The reason for these contradictions may be due to the fact that in some of these studies users were in control of their moment about the simulated world, while in others they were confined to a predestined course. Control may provide users with a means of adapting to or accommodating cue conflicts by building conditioned

expectations through repeated interactions with a virtual world (e.g., when a user's head turns the user learns to expect the world to follow milliseconds behind). Lack of control would not allow such expectations to be established since users would not be aware of which way they were turning at any particular moment (i.e., the course would be determined by the system). Thus, without control, users would not be expected to adapt to cue conflicts. User control in conjunction with adaptation may provide a means of minimizing the influences of cybersickness.

Research on CS needs to be conducted in order to fully specify the relationships between control, adaptation, and CS. Control also needs to be tested against varying degrees of other factors to see what level of freedom is necessary to potentially negate their affects. The research should focus on developing a general theory of CS which would allow for the prediction of the combinations of factors which would be disruptive and lead to CS; those which would be easy or hard to adapt to; and the relationship of these levels of adaptation to the level of user control. Such a theory would provide VE developers with the knowledge necessary to minimize the adverse effects of VE interaction.

## THE SOCIAL IMPACT OF VIRTUAL TECHNOLOGY

While researchers are often concerned about human performance and health and safety issues when developing a new technology, an often times neglected effect of new technologies is their potential social impact. Virtual reality is a technology, which like its ancestors (e.g., television, computers, video games) has the potential for negative social implications through misuse and abuse [11]. Its higher level of user interaction may even pose a greater threat than past technologies. Through a careful analysis, some of the problems of VEs may be anticipated and perhaps prevented. A proactive, rather than reactive, approach may allow researchers to identify and address potentially harmful side-effects related to the use of VE technology. Such an approach requires that researchers and developers prioritize social issues early on in VE development, rather than taking a wait-and-see attitude. Most VR conferences have yet to even recognize and address that social issues may exist.

Currently the potential negative social influences resulting from VE exposure are not well understood. There are many open issues [11, 22, 25, 28], such as: What will be the psychological and character effects of VE use? How will interaction in the virtual world modify behavior? What will the 'transfer of training' be for violent virtual interactions? Will individuals transfer violent virtual experiences to the real world? Will people turn their backs on the real world and become "contented zombies" wandering around synthetic worlds which fulfill their whims but disregard their growth as a human being? Will VR users experience traumatic physical or psychological consequences due to a virtual interaction? Will people avoid reality and real social encounters with peers and become addicted to escapism? Is continual exposure to violent virtual worlds similar to military training, which through continued exposure may desensitize individuals to the acts of killing and maiming? Could the behaviors of soldiers after intense military training events provide an indication of the influences of intense violent VE interactions? How will VE influence young children who are particularly liable to psychological and moral influence? Does VE raise issues which are genuinely novel over past media due to the salience of the experience and the active interaction of the user? These issues need to be proactively explored in order to circumvent negative social consequences from HVEI.

## CONCLUSIONS

This paper has presented many of the human factors issues which must be addressed in order for VR technology to reach its full potential without inflicting harm along the way. VR technology promises to permeate both professional and personal aspects of our lives. If this influx is to be a positive influence rather than a forceful intrusion, it is essential that each of these human factors issues receive significant systematic research.

## REFERENCES


1. Barfield, W. and Weghorst, S. The sense of presence within virtual environments: a conceptual model. In G. Salvendy and M. Smith (Eds.), *Human-Computer Interaction: Software and Hardware Interfaces.* Elsevier Science Publishers, Amsterdam, Netherlands, 1993, pp. 699-704.

2. Bennett, K.B., Toms, M.L., and Woods, D.D. Emergent features and graphical elements: Designing more effective configural displays. *Human Factors*, 35, 1 (1993), 71-97.

3. Burdea, G. and Coiffet, P. *Virtual Reality Technology*. Wiley, New York, 1994.

4. Carroll, J.M. and Mack, R.L. Metaphor, computing systems, and active learning. *International Journal of Man-Machine Studies*, 22 (1985), 39-57

5. Cohen, M. Integrating graphic and audio windows. *Presence*, 1, 4 (1992), 468-481.

6. Egan, D.E. Individual differences in human-computer interaction. In M. Helander (Ed.), *Handbook of*

*Human-Computer Interaction*. North Holland, Amsterdam, Netherlands, 1988, pp. 543-568.

7. Hahn, J.K., Gritz, L., Darken, R., Geigel, J., and Won Lee, J. An integrated virtual environment system. *Presence*, 2, 4 (1993), 353-360.

8. Held, R. and Durlach, N. Telepresence, time delay, and adaptation. In S. Ellis (Ed.), *Pictorial Communication in Virtual and Real Environments*. Taylor and Francis, London, 1993, pp. 232-246.

9. Hettinger, L.J., Berbaum, K.S., Kennedy, R.S., Dunlap, W.P., and Nolan, M.D. Vection and Simulator Sickness. *Military Psychology*, 2, 3 (1990), 171-181.

10. Kalawsky, R.S. *The Science of Virtual Reality and Virtual Environments*. Addison-Wesley, Wokingham, England, 1993.

11. Kallman, E.A. Ethical evaluation: A necessary element in virtual environment research. *Presence*, 2, 2 (1993), 143-146.

12. Kennedy, R.S., Fowlkes, J.E. and Hettinger, L.J. *Review of Simulator Sickness Literature* (Tech Report NTSC TR89-024). Naval Training Systems Center, Orlando, FL, 1989.

13. Lampton, D.R., Knerr, B.W., Goldberg, S.L., Bliss, J.P., Moshell, J.M., and Blau, B.S. The virtual environment performance assessment battery (VEPAB): Development and evaluation. *Presence*, 3, 2 (1994), 145-157.

14. Larijani, L.C. *The Virtual Reality Primer*. McGraw-Hill, New York, 1994.

15. Massimino, M.J. and Sheridan, T.B. Sensory substitution for force feedback in teleoperation. *Presence*, 2, 4 (1993), 344-352.

16. Massimino, M.J. and Sheridan, T.B. Teleoperator performance with varying force and visual feedback. *Human Factors*, 36, 2 (1994), 145-157.

17. McDowall, I. 3D Stereoscopic data for immersive displays. *AI Expert*, 9, 5 (1994), 18-21.

18. McGovern, D.E. Experience and results in teleoperation of land vehicles. In S. Ellis (Ed.), *Pictorial Communication in Virtual and Real Environments*. Taylor and Francis, London, 1993, pp. 182-195.

19. Money, K.E. and Cheung, B.S. Another function of the inner ear: facilitation of the emetic response to poisons. *Aviation, Space, and Environmental Medicine*, 54, 3 (1983), 208-211.

20. Oren, T. Designing a new medium. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design*. Addison-Wesley, Reading, MA, 1990, pp. 467-479.

21. Sellen, A. and Nicol, A. Building user-centered on-line help. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design*. Addison-Wesley, Reading, MA, 1990, pp. 143-153.

22. Sheridan, T.B. My anxieties about virtual environments. *Presence*, 2, 2 (1993), 141-142.

23. Shneiderman, B. *Designing the User Interface* (2nd ed.). Addison-Wesley, Reading, MA, 1992.

24. Sollenberger, R.L. and Milgram, P. Effects of stereoscopic and rotational displays in a three-dimensional path tracing task. *Human Factors*, 35, 3 (1993), 483-499.

25. Stone, V.E. Social interaction and social development in virtual environments. *Presence*, 2, 2 (1993), 153-161.

26. Thomas, J.C. and Stuart, R. Virtual reality and human factors. In *Proceedings of the Human Factors Society 36th Annual Meeting* (October 12-16, Atlanta, GA). Human Factors Society, Santa Monica, CA, 1992, pp. 207-210.

27. Viirre, E. A survey of medical issues and virtual reality technology. *Virtual Reality World*, (August, 1994), 16-20.

28. Whiteback, C. Virtual environments: Ethical issues and significant confusions. *Presence*, 2, 2 (1993), 147-152.

29. Wickens, C.D., Merwin, D.H., and Lin, E.L. Implications of graphics enhancements for the visualization of scientific data: dimensional integrality, stereopsis, motion, and mesh. *Human Factors*, 36, 1 (1994), 44-61.

# Implications of Balance Disturbances Following Exposure to Virtual Reality Systems

Robert S. Kennedy, Ph.D.
Vice President, Essex Corporation

Michael G. Lilienthal, CDR (MSC)USN
Defense Modeling Simulation Office

*Abstract*

*As we move our heads and bodies, and travel through our real world, the human organs of equilibrium (the vestibular system) integrate the information from the visual and proprioceptive systems and compare them. A useful analogue is the fire control system for a gun mounted on a moving vehicle such as a battleship. It is our opinion that many virtual reality (VR) systems, like many ground based flight simulators, alter the natural correspondences between these sensory inputs and when the exposure to the VR environment is protracted, the sensory systems are recalibrated to accommodate the new relationships. These recalibrations, when they involve the vestibular system, can result in balance disturbances, and these latter can outlast the period that an individual remains under the control of the person or entity that exposed that individual to the VR system. If a person should trip when leaving the building, or later when driving home, safety could be compromised and products liability could be incurred. We review our experiences with balance disturbances in flight trainers and describe recent findings with an automated postural equilibrium assessment system which can be employed before and after exposure in order to certify that no observable changes are evident in a subject or user.*

## INTRODUCTION

Under ordinary circumstances, as we move our heads and bodies, and travel through our real world, the human organs of equilibrium (the vestibular system) integrate the information from the visual and proprioceptive systems and compare them. A useful analogue is the fire control system for a gun mounted on a moving vehicle such as a battleship. Figure 1 shows an example of just such a system, and the various human systems and their analogues (central nervous system = computer; rangefinder and elevation = vision; stable element = vestibular system; etc.) can be clearly seen. Usually, within tolerances, the visual and inertial forces are concurrent, and predictions of physical reality can be made from correlations of these sensory inputs. Occasionally, one or more of these systems go out of correlation such as when mirrors are used (which reverses the correlation) or vestibular signals of motion are not corroborated by the visual scene, as happens below deck on a moving ship. When this occurs, under the "right" circumstances, individuals can encounter symptoms of motion sickness.

Experience over the past twenty years with ground based flight trainers has presented a complication called simulator sickness, a form of motion sickness characterized by symptoms such as nausea, sweating, dizziness, headache, eyestrain, and balance disturbances. It was advertised by early developers of these systems that as these simulators became more compellingly realistic, and fidelity was improved, sickness incidence would be reduced; this has not happened.

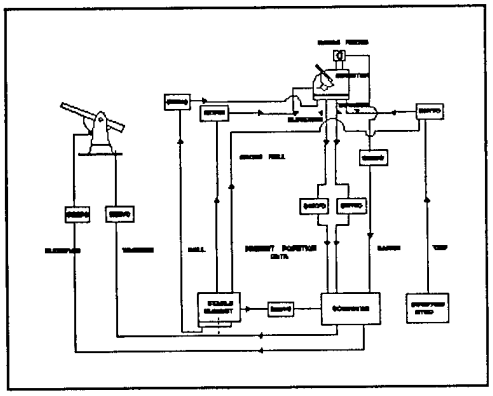


Figure 1. Diagram of a sea going fire control system. Analogue to the human control systems (4).



0-8186-7084-3/95 $04.00 © 1995 IEEE

Nearly every developer of emerging technology virtual reality systems has experienced similar symptoms with their devices. Usually, one or more members of a development team is more prone to sickness than another. This individual is usually spared the more severe exposures, although with time and experience with a particular system's dynamics and visuals, the member may become adapted to that system.

We believe these experiences and those of the military pilot simulator community vastly underestimate the problem. Moreover, one of the symptoms that is experienced AFTER leaving the simulator is an assortment of balance disturbances, disorientation, dizziness and vertigo. Recent findings imply that the same kinds of problems are about to be encountered in the usage of virtual reality systems. Indeed, if they are more visually elaborate, but do not convey sufficient vestibular and proprioceptive cuing, and entail longer exposures, it is likely that balance disturbances will occur in VR systems to a greater extent. Since these problems can outlast the period of time an individual spends in the building where the system is housed, and since balance disturbances can relate to navigation errors (viz., going in the wrong direction or falling), we have been looking at postural disturbances in simulators and some virtual reality systems. Beginning with a floor-based test battery, easily accomplished by lay individuals with minimal training, and recently using a video-based automatic data acquisition system of head position, we have collected a moderate data base of post effects in connection with several different virtual reality and flight simulator systems (2). We found that in general, longer hops, off-axis viewing, moving-bases, and domes are the most disruptive of walking and standing steadiness. However, in many cases, we found that performance improved over pre/post testing and many of these differences were only statistically significant when postural changes were compared to a control group.

## Method

In order to develop a more sensitive measure, we developed a video-based test of postural equilibrium. This device (described more completely elsewhere - (3) makes use of a video record of head position, a framegrabber, a reticle, and a computer as shown in figure 2 and 3. The subject assumes a heel-to-toe standing position with eyes closed and arms folded. The subject is asked to stand for 30 seconds (or until the subject falls) and velocity is recorded in z and y (cf., figure 4) and size change of the reticle is used to index movement in x. These data are then automatically scored using a program developed by us, and the data are placed into a statistical package for analysis (cf., figure 5).

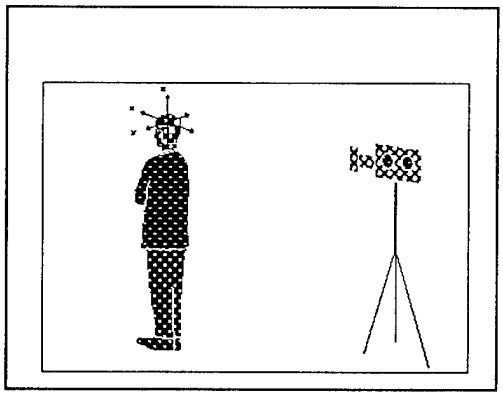


Figure 2. Schematic showing subject, reticle and video camera recording set-up

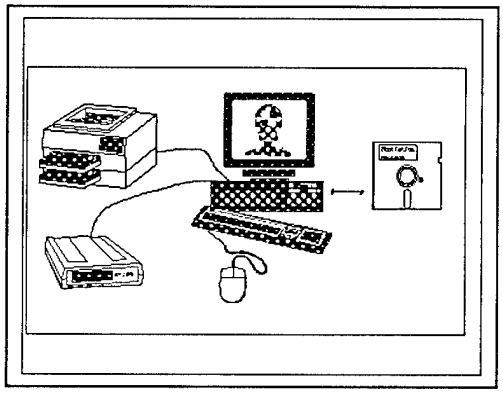


Figure 3. Schematic showing data analytic set-up.

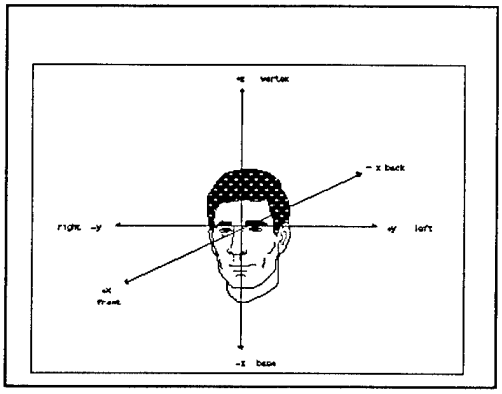


Figure 4. Vestibular nomenclature for cardinal axes and planes for human recording (1).

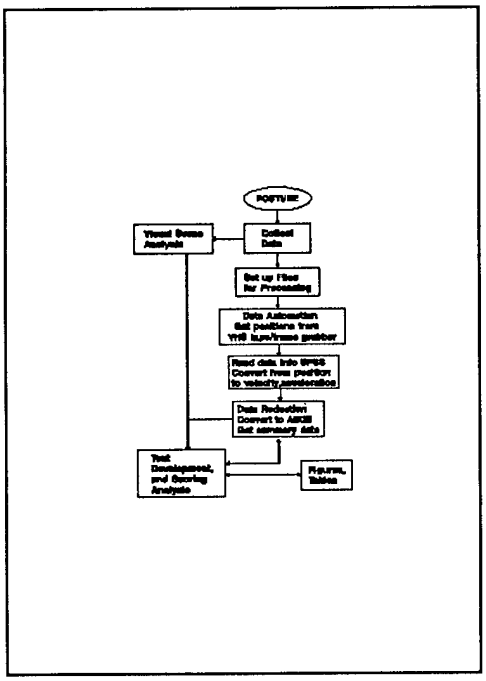


Figure 5. Flowchart for Video Posture Data Analysis

To date we have studied several different groups of subjects employing two different helicopter simulators and also graded dosages of alcohol. We have also studied a control group of college students. The first simulator device (Simulator #1) located at the Mayport, Florida Naval Station was an SH-60B. The visual displays were generated by a "Vital IV" calligraphic dusk/night computer image generator (CIG). The visual was a 6-window, 6 channel, folded on-axis virtual-image CRT with approximately a 130 degree (H) x 30 degree (V) field-of-view. Motion was generated with a 6-degree-of-freedom motion-base system. Visual displays were presented on the pilots' forward and side window and the copilots' forward window. Limited visual information was presented on both chin or bubble windows.

The HH-60J simulator device (Simulator #2) at USCG station, Mobile, Alabama is an Operational Flight Trainer (OFT) with a 6-degree-of-freedom hydraulic motion platform. The visual displays were also generated by the "Vital IV". The visual hardware is a Wide Field of View (WOF) projection dome. The visual scene is focused and aligned primarily for the pilots' viewing perspective, but the scene content is accessible by all parties in the confines of the device compartment. The distance from the pilots' eyes to the visual scene plane is approximately 10 feet at the center of the dome.

## Results

### Stability

1. Six pilots in simulator #1 were tested twice before, during, and after four flight simulator hops distributed over a 14 day period. The pretest scores for these subjects were averaged over the four flights and no improvement between sessions in posture performance was shown over four pairs of trials which implies that this test exhibits minimal practice effects. This result was expected and is certainly reasonable since the task, heel-to-toe, standing, involves movements and positions well practiced by humans.
2. There were no mean differences between the two pilot groups ($p > .4$) from simulators #'s 1 & 2 and the difference between pilots and the college students was barely significant ($p = .04$) and due entirely to the differences between males and females ($p < .01$) in the college population where males' balance scores showed less sway.
3. To examine within session learning effects, performances on trial one were compared to trial two for all subjects. It appeared again that mean scores did not improve from trial one to trial two ($p > .5$) and they were averaged. The retest reliability from trial one to trial two for 22 subjects was statistically significant ($p < .001$) and metrically acceptable ($r = .69$).

### Validity:

4. The pilots in simulator #1 showed differences between pre/post performance depending on whether the flight was long or short. It was seen that reductions of 20% were found with longer hops, but no change or slight improvement (< 10%) was found with shorter flights.
5. Post/Pre changes in posture performance in flight simulator #2 showed that the pilots using the co-pilot seat, which was slightly (20°) offset from the pilot's design eye position, exhibited considerable disequilibrium (60% loss). The pilot showed only a 10% loss when exiting the flight simulator. These differences (in 8 subjects) were significant ($p < .05$).
6. In the third experiment, graded dosages of alcohol were administered to 10 subjects while their postural performances were obtained. These data were regressed within subjects and had an average correspondence of posture to ETOH concentration of $r = .70$.

Then taking the subjects' raw posture scores as data points, we calibrated a predicted ETOH score for each subject at the .025, .050, .075, and .100 BAC levels. These predicted scores were averaged to form a group alcohol regression line which takes the form of: Blood alcohol concentration

(y) = the raw posture score

x .0079 - .0121

This regression line is shown in figure 6.

We then used the regression line of figure 6 to index the changes we saw from pre and post testing in the on-axis and off-axis subjects referred to in item 5 above. As shown in figure 7, it was found that both groups of pilots had an "equivalent" BAC level of <.025 BAC when they entered the flight simulator. But, when they emerged from the simulator the pilots had an equivalent alcohol level of .035 BAC, and the co-pilots of .052 BAC. The latter is what the literature on alcohol intoxication research indicates is a post with which behavioral effects begin to become measureable, and in the present case is illustrative of the methodology as well as suggestive of potential difficulties from such simulator exposures. One might therefore use these data to advise individuals regarding restrictions on these subsequent activities, if any.

Using these values, it should be possible in future applications to index change in posture performance (for whatever causes) to an equivalent blood alcohol concentration. Future work should evaluate persons exposed to virtual reality devices and thus relate the size of the changes to a blood alcohol concentration.

We believe that video based analysis of stance and gait has considerable utility for certification of changes due to VR and simulator exposure. Components of the proposed system that we have selected are readily available "off the shelf" on the open market, and are acquirable without excessive delay. The reliability of the system is defined by the reliability of the components (the camcorder and the computer), as well as the reliabilty of the human response system of the "head kinematics control system". Because these hardware components are so ubiquitous, costs for components and their maintenance are small. Costs for test administration are also very low. It would involve setting up a tripod and video recorder and the cost of video tape. Scoring and interpretation could be implemented using a computer program on a personal, perhaps a portable, computer which is fitted with a framegrabber and a VCR. In future work we plan to have the system capable of operating on internal battery power from within the camcorder and portable computer. The system should be adaptable to emerging technological advances in videotaping, framegrabbing, personal, and workstation computers. These technologies driven by the market are the fastest developing of any of those available. In other words, "real time" analyses are possible. Video cameras, video recorders, and portable computers are already used in flight, at sea, and in microgravity. We predict these strengths will make the selected design superior to all others where portability, price, and ease of use are important.

If VR-induced postural instability is more than occasional, it has direct implications for safety whether in connection with vehicles operated after exposure (e.g., cars, aircraft) or other activities (e.g., roof repair, mountain climbing). Virtual reality systems, which use helmet-mounted displays, we believe are particularly vulnerable since they employ wraparound displays which can be expected to occasion similar problems. Greater usage in the future of such systems is expected in the private sector, and it can be anticipated that such aftereffects could present considerable medical and safety problems.

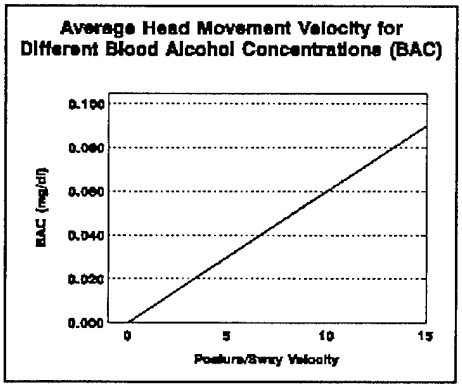



Figure 6. Average head movement velicity for different blood alcohol concentrations.

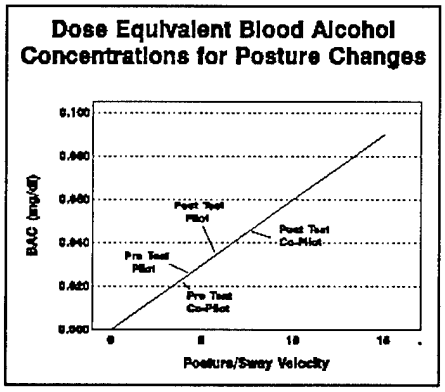



Figure 7. Dose equivalent blood alcohol concentration for posture changes

## References


1. Hixon, W. C., Niven, J. I., & Correia, M. J. (1966). Kinematices nomenclature for physiological accelerations. Monograph 14, Naval Aerospace Medical Institute, Naval Aerospace Medical Center, Pensacola, FL. 8 August, 1966.

2. Kennedy, R. S., Fowlkes, J. E., & Lilienthal, M. G. (1993). Postural and performance changes in Navy and Marine Corps pilots following flight simulators. Aviation, Space, and Environmental Medicine, 64, 912-920.

3. Kennedy, R. S. (1993). Device for Measuring Head Position as a Measure of Postural Stability. Final Report No. 9260166, National Service Foundation, Washington, DC.

4. Whitteridge, D. (1960). Central control of eye movements. In J. Field, H. W. Magoun, & V. E. Hall (Eds.), Handbook of neurophysiology (pp. 1089-1109). Washington, DC: American Physiological Society.

# The Use of Sketch Maps to Measure Cognitive Maps of Virtual Environments.

*Mark Billinghurst and Suzanne Weghorst*
Human Interface Technology Laboratory
FJ-15, University of Washington,
Seattle, WA 98195.
+1-206-616-1493
{grof,weghorst}@hitl.washington.edu

**ABSTRACT**
Cognitive maps are mental models of the relative locations and attributes of phenomena in spatial environments. Understanding how people form cognitive maps of virtual environments is vital to effective virtual world design. Unfortunately, such an understanding is hampered by the difficulty of cognitive map measurement. The present study tests the validity of using sketch maps to examine aspects of virtual world cognitive maps. We predict that subjects who report feeling oriented within the virtual world will produce better sketch maps and so sketch map accuracy can be used as an external measure of subject orientation and world knowledge. Results show a high positive correlation between subjective ratings of orientation, world knowledge and sketch map accuracy, supporting our hypothesis that sketch maps provide a valid measure of internal cognitive maps of virtual environments. Results across different worlds also suggest that sketch maps can be used to find an absolute measure for goodness of world design.

**KEYWORDS** Cognitive Mapping, Virtual Environments, Sketch Maps, Mental Models.

**INTRODUCTION**
Whether in real or virtual space we form cognitive maps to deal with and process the information contained in the surrounding environment. Cognitive mapping is formally defined by Downs and Stea [6] as:

> "..a process composed of a series of psychological transformations by which an individual acquires, codes, stores, recalls, and decodes information about the relative locations and attributes of phenomena in their everyday spatial environment."

An individual's cognitive map is an active information seeking structure of which spatial imagery is but one aspect [14]. Cognitive maps are also made up of memories of objects and kinesthetic, visual and auditory cues [8].

The fundamental importance of an effective cognitive map is that it allows two questions to be answered quickly and efficiently: Where is that? How do I get to there from here? Thus human spatial behavior relies upon and is determined by the individual's cognitive map of the surrounding environment. In addition, the perception of the environment itself is always guided by some sort of cognitive map, so an inaccurate or incomplete cognitive map leads to disorientation and confusion[14].

Designing virtual worlds through which subjects can navigate and orientate themselves successfully requires an understanding of cognitive map formation in virtual environments. Considerable research which might be brought to bear on this topic has been conducted on the development of cognitive maps and how they affect real world behavior.

In exploring how people formed mental images of a city Briggs[4] has identified three complementary ways in which cognitive maps are created:

- Through an individual's sensory modalities.
- From symbolic representations such as maps.
- From ideas about the environment which are inferred from experiences in other similar spatial locations.

Of these, an individual's sensory modalities provide direct sources of information and are more effective in cognitive map formation than indirect sources[6].

Cognitive maps are created as the result of active and passive modes of information processing [14]. Generally, active information processing gives the greatest meaning to the information processed and produces more information for the moving perceiver. Thus the information produced by locomotion is fundamental to an individual's spatial orientation.

An individual's cognition of the environment is not only a function of the behavior by which information is obtained but also depends on the characteristics of the environment [4]. The amount of information gained by each sensory modality is also environmentally dependent [16].

Aside from the way cognitive maps are formed, the types of information stored in a cognitive map are also of interest. Kuipers[10] suggests that a cognitive map consists of five different types of information, each with it's own representation: Topological, Metric, Route Descriptions, Fixed Features and Sensory Images. Different techniques are needed to measure each different information type.



0-8186-7084-3/95 $04.00 

Finally, Lynch[12] notes the uniquely personal nature of cognitive maps. Across different cultures he found that different groups may have widely different images of the same outer reality. Also, on an individual level, what an observer sees is based on a common exterior form, but how the observer interprets and organizes this form is unique. This interpretation governs how the observer directs his attention and this in turn affects what is seen. So at both a societal level and a cultural level cognitive maps are highly individualistic.

## COGNITIVE MAPPING - THE VIRTUAL EXPERIENCE

As suggested above, cognitive maps are most effectively formed by active interaction with the environment using many different sensory modalities. However, in a virtual world there is typically sensory degradation and a lack of many of the perceptual cues used in the real world. Downs and Stea [6] point out that any filtering of information before it reaches the sensory modalities affects the cognitive map. This is the case for virtual environments. For example, the visual modality may suffer from low image resolution, poor image quality or a reduction of the peripheral field. In real environments, Alfano and Michel[1] have shown the reduction of peripheral vision impairs perception and visuomotor performance, both of which are essential for cognitive mapping ability. In addition there are rarely any tactile or olfactory cues and often only limited auditory feedback. The study presented here examines some of the factors influencing cognitive map construction given current immersive technology.

## METHODS FOR ASSESSING COGNITIVE MAPS

One of the difficulties in studying cognitive mapping is the problem of extracting an external representation of an individual's internal map. By definition a cognitive map is highly subject-specific and, although individuals often record the same things in their cognitive maps, there is no evidence that they record them in the same way. Golledge[7] identifies four distinct methods for extracting environmental cognition information :

- Experimenter observation of subject behavior
- Historical reconstruction
- Analysis of external representations
- Indirect judgment tasks

In our experiment we assess the subject's cognitive map through subject self-reporting and analysis of external representation.

We are particularly interested in the subject's topological understanding of the virtual environment, i.e. knowing where they are and where everything else is, as compared with metric knowledge - knowing precise object location and distance between objects. Topological knowledge is generally more important than metric knowledge for effective navigation.

A common approach for measuring topological knowledge was suggested by Lynch[12], who had subjects sketch maps to represent the mental models of their local cities. Lynch finds that sketch maps are more accurate when used for topological rather than metric analysis.

Golledge[7] points out, however, that caution must be taken that sketch maps are not over analyzed. The disadvantages of sketch maps include trying to represent a three-dimensional cognitive map in two dimensions and the difficulties of quantitative analysis. They may also measure more than just spatial understanding of an environment, e.g. drawing or memory ability. Conversely, Blades[3] finds them reliable over time and Newcombe[15] comments that they are no less accurate than other cognitive techniques.

Other common techniques used for cognitive map analysis include distance and angle estimation. However, Henry[9] found that distances were consistently underestimated in virtual environments and that angle estimation produced wildly varying results. Moreover, his subjects' sketch maps are topologically accurate even when the sketched distances are not. In a prior work we used a different technique for distance estimation and found similar results[18].

The present study is designed to asses the validity of sketch maps as a tool for measuring cognitive maps of virtual environments, particularly the topological knowledge of the cognitive maps. We predict that subjects who report feeling oriented and unconfused in the virtual world will later produce relatively accurate sketch maps, whereas subjects who report feeling disoriented and confused in the virtual world will produce less accurate sketch maps. In other words, if sketch maps are an accurate external representation of the subject's cognitive map then we would expect a correlation between the sketch map scores and subjective ratings of how oriented subjects felt within the virtual world.

## EXPERIMENT DESIGN

Eighty four subjects experienced a number of simple virtual worlds and then produced maps. The worlds were constructed using Swivel and Body Electric software, and rendered on an SGI VGX. Participants wore VPL Eyephones and interacted with the virtual environment using a VPL Dataglove. Movement through the virtual environment was achieved by the users pointing in the desired direction they and making a "fly" gesture with the Dataglove. This movement was completely unconstrained so participants could be as close or far away from the world as they wanted. Collision detection was not used so participants could travel through objects.

Each subject was initially trained on the same immersive virtual environment until they felt comfortable with moving and interacting within a virtual environment. Following this training, they were given a 24-question survey which asked for responses on a range of navigation, orientation, interaction, presence and interface questions. Survey responses were indicated on a 10-point anchored scale. These survey questions are reproduced in the appendix. Participants were also invited to comment about the experience in general.

| Name | Density | Object number | Object Types | Object classes | Number of Subjects |
|---|---|---|---|---|---|
| Virtual Valley | Dense | high | logical | high | 35 |
| Cloudlands | Sparse | low | abstract | low | 25 |
| Neighborhood | Cluttered | high | logical | high | 24 |

Table 1.0 : The Different Characteristics of the Three Test Worlds.

After the training session, subjects experienced one of three different virtual worlds for 10 minutes and were told to explore it as fully as they could. They were then asked to produce a map of the world that someone unfamiliar with the world could use to navigate around the world. The subjects also completed the same survey that was administered after the training world and were video taped for later observation of behavior patterns.

If the sketch maps are an accurate external representation of the subjects cognitive map then we would expect a correlation between the sketch map scores and subject survey scores for orientation within the virtual world.

**World Differences**

Three different worlds were used to explore how differences in world design might affect the cognitive map formed and the resultant sketch maps. According to Darken and Silbert's[5] world classification, each of them are "small", in that all of the world can be seen from a single viewpoint. They are also static, all their objects having positions and values which don't change over time. However, the density of each of the worlds varied considerably as detailed below. Each subject experienced only one of the test worlds.

*Virtual Valley*

Under Darken and Silbert's scheme this is a dense world: it has a large number of objects and spatial cues; however, they are all placed in a logical manner. The world is bound on either side by tall mountain ranges that direct attention to the objects contained in the valley below. Objects within the world are all representative of what would be expected in a real valley and there are no hidden objects. Objects are clearly distinguishable by color and size, and there are a number of distinctive objects that could serve as landmarks. This world design would make it difficult for subjects to become disoriented.

*Cloudlands*

Cloudlands is a sparse world containing few objects. It contains a dominant ground plane with clusters of objects floating above it in cloud groups. One of these clouds contains a fish and star while the others are empty. The objects are incongruous and surprising - there is a floating cactus, stacks of multicolored planes, cones and small gray rocks. The are no environmental cues to direct attention other than the object clusters themselves. However, the sparcity of the world would also make it difficult for subjects to become disoriented.

*Neighborhood*

Neighborhood is a cluttered world containing clusters of buildings all closely grouped and each containing other objects. The buildings are largely the same size and color making it hard to distinguish between them, and the objects within them are almost all the same color as the buildings. The objects are all those that would be logically found in a neighborhood, such as trees, tables, glass and a piano but the similarity of the buildings makes it hard to precisely locate them. This world is generally confusing and disorientating.

Table 1.0 summarizes the characteristics of the three test virtual worlds.

## SKETCH MAP ANALYSIS METHOD

As mentioned before, one of the challenges of using sketch maps is analyzing the results. The maps produced are as individualistic as each of the cognitive maps of the subjects. Although sketch maps are commonly used in real world cognitive mapping there is no generally accepted method for their analysis. Useful approaches have been reported in Appleyard[2], Ladd[11], Moore[13], and Walsh *et. al.*[17], among others; however these are used to analyze maps of large scale urban environments. Adapting these methods, we use a simple, purely topological technique. Each sketch map was given a set of goodness, object class and object positioning scores as detailed below:

**Map Goodness**

Maps were ranked for goodness on a scale of 1-3 by two researchers who were experienced in virtual environments but blind to subject identity and other correlated measures. The researchers were told to rank the maps on how useful they would be as a navigational tool if they were taken with them into the virtual environment. They were told to ignore the participants drawing ability and focus on how well the map represented the virtual world and the locations of the objects within it.

**Object Classes**

Each map was given a score according to the number of object classes present - for example, trees, rocks and mountains are each counted as separate classes. Using object classes is a way to assess completeness of a sketch map for a given virtual world.

**Relative Object Positioning**

To provide a measure of differences in cognitive maps for different worlds we scored maps according to relative object positioning. We used topological positioning and so scored objects if they were correctly positioned to the right or left, above or below, or clockwise or counterclockwise, depending on the specific world being represented. The specific object position was not important, only its position relative to other objects in the sketch map.

| | Virtual Valley | | Neighborhood | | Cloudlands | |
|---|---|---|---|---|---|---|
| | Class No. | Map Goodness | Class No. | Map Goodness | Class No. | Map Goodness |
| World Knowledge | .480 | .635 | .427 | .405 | .242 | .193 |
| World Orientation | .567 | .738 | .397 | .524 | .353 | .290 |
| | n = 12, $p < 0.05$, r = 0.56 | | n = 21, $p < 0.05$, r = 0.38 | | | |

Table 2.0: Goodness and Class Number correlations with virtual world orientation and knowledge across the test worlds.

Maps were given two positioning scores: a total object position score in which all the objects were scored, and a significant object position score where the five most commonly drawn objects for each world are scored. Relative object positioning is a way to assess the accuracy of sketch maps.

## RESULTS

Although subjects were given no instructions on how to produce their maps, almost all of them drew three dimensional representations of the virtual world. This may be due to the small size of the worlds - sketch maps produced of large scale real world environments are usually two dimensional. Figure 1.0 shows typical sketch maps produced for the Cloudlands world.

### Within World Correlation

If sketch maps can be used as an external measure of subjects' cognitive maps then there should be a strong correlation between map goodness scores and subject scores for the survey questions "Knowing where everything is in the Virtual World" and "Orientation in the Virtual World". To investigate this we correlated the object class and map goodness scores with the survey responses. Table 2.0 shows the correlation values of the map scores and survey scores. Although the map goodness rankings are higly subjective, the correlation between the scores given by the two researchers was very high; ( r = 0.86, 0.71, 0.70, for three worlds respectively, significant at $p < 0.01$).

In the Virtual Valley and Neighborhood worlds object class and map goodness were both significantly correlated with the subjects' reported sense of orientation in the virtual world. For these two worlds, the map goodness score is also significantly correlated with subjects' knowledge of where everything is. However, this isn't the case with the Cloudlands world. The sparse nature of Cloudlands may make it difficult to produce an accurate sketch map. Cloudlands was also more three-dimensional that the other worlds with most objects placed high above the dominant ground plane, adding to the difficulty of producing a two-dimensional representation.

Since cognitive maps are most effectively formed by active interaction with the environment, there should also be a relationship between map scores and the survey questions relating to interaction. This is indeed the case with Virtual Valley, where the map goodness rankings correlate significantly with the subjects survey score for ease of interaction (r = 0.882), ease of navigation (r = 0.865), ease of movement within the virtual world (r = 0.814) and ease of use of the Data Glove (r = 0.645). However, in the other two worlds the correlation between these survey questions and the map rankings were not significant.

### Between World Differences

A two factor ANOVA was done on the survey results to identify world differences and possible gender-linked factors. There was a significant difference between worlds in subject's understanding of where everything was ($F[2,22]=4.49$, $p < 0.025$), and how oriented the subjects felt within each of the worlds ($F[2,22]=3.314$, $p < 0.05$). For both of these questions subjects rated Neighborhood world significantly lower than the two other worlds, as shown in figure 2.0. There was also a significant difference between the sense of dizziness reported by subjects, with those in Neighborhood registering the most dizziness, ($F[2,22] = 3.95$, $p < 0.025$). These results reflect the particularly disorienting nature of Neighborhood world.

If sketch maps are representative of subjects virtual world cognitive maps, they should also reflect these world differences. The relative object position scores can be used to compare across worlds. For each world we defined the five most commonly drawn objects as "significant objects" and a relative positioning ratio was then calculated for each map:

$$\text{Ratio} = \frac{\text{Correctly placed significant objects.}}{\text{Total number of significant objects in map.}}$$

An ANOVA revealed a statistically significant world difference for the significant object relative positioning ratios, ($F[2,22] = 4.004$, $p < 0.025$). A similar ratio was calculated for the relative positioning for all objects drawn in the sketch maps. In this case an ANOVA showed no significant world difference, ($F[2,22] < 1.0$ NS). Figure 3.0 shows the relative positioning ratios for both sets of objects.

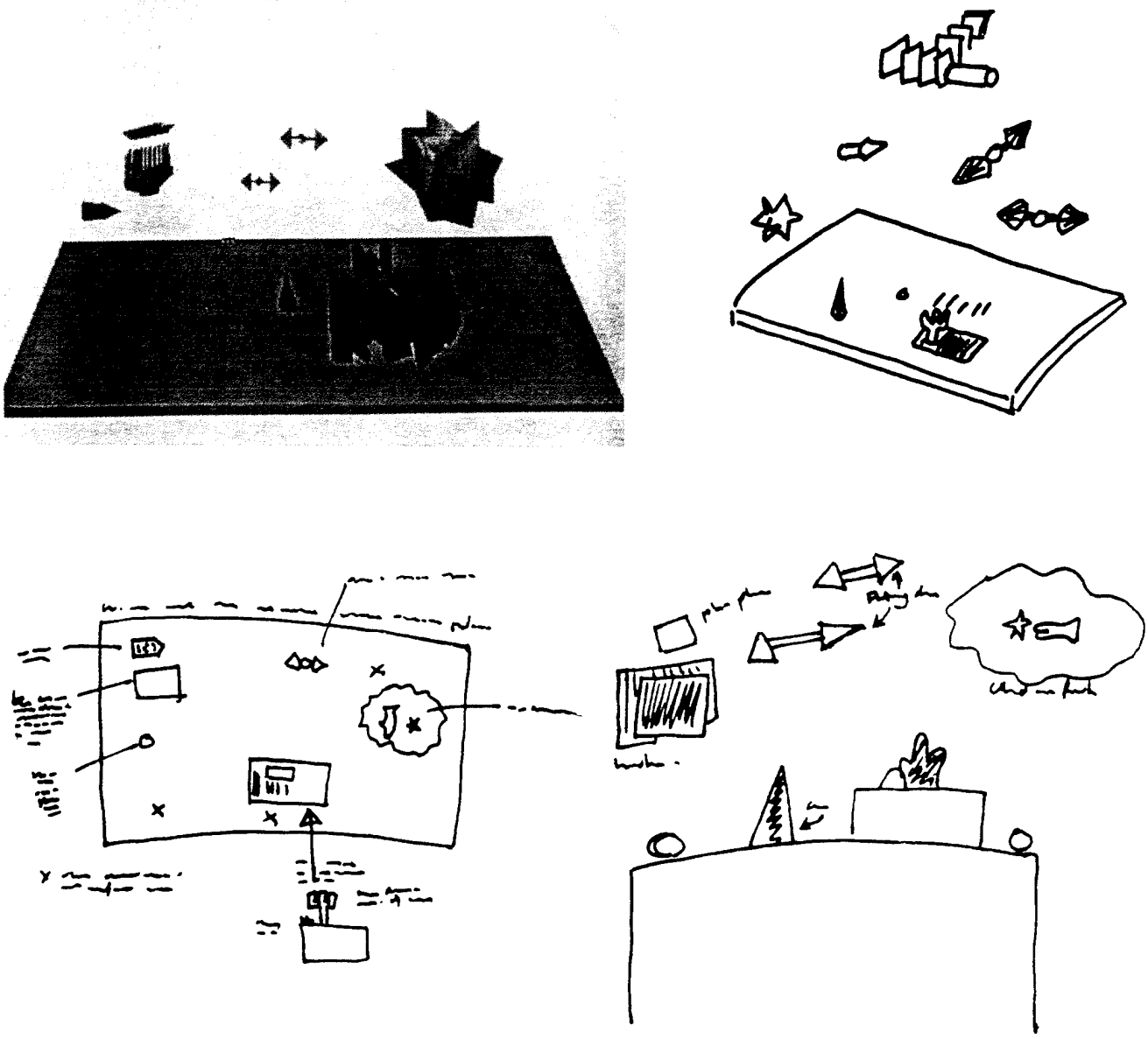


Figure 1.0: Cloudlands world (upper left) and three typical sketch maps.

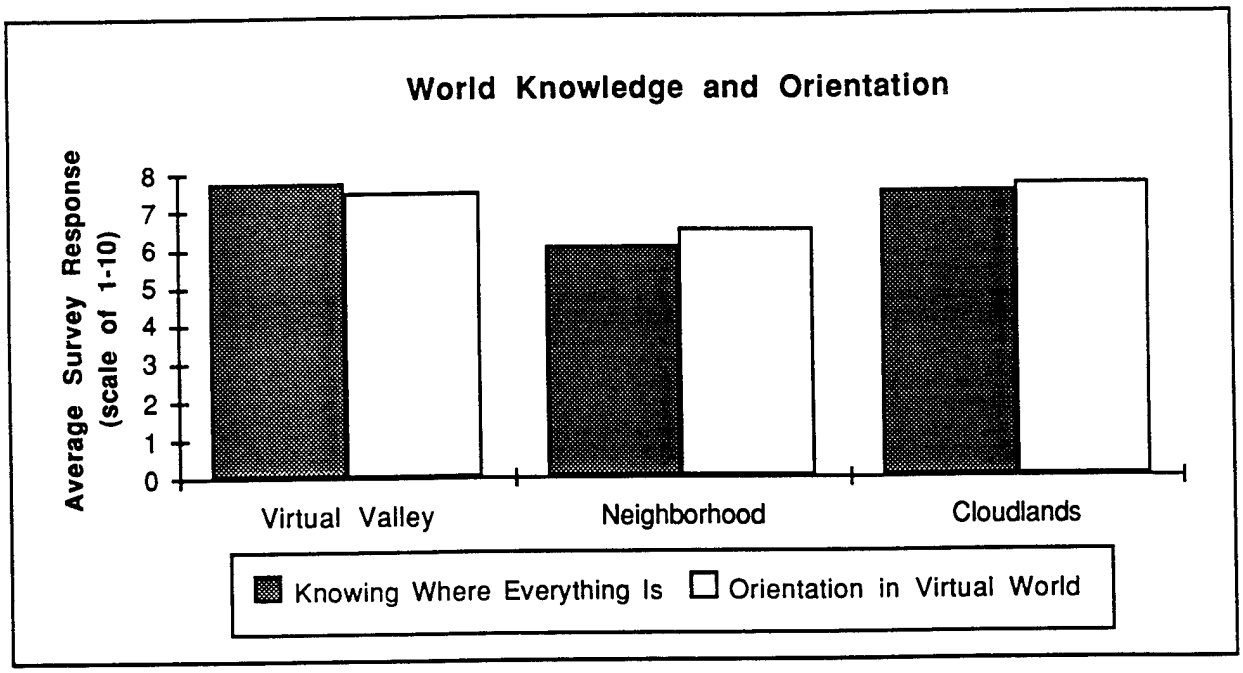



Figure 2.0 Average subject orientation and world knowledge survey scores across the three test worlds

In Virtual Valley over 90% of the significant objects that are placed are placed correctly, reflecting the well designed nature of the world. The difference in ratios from "significant" object placement to "all" object placement in Virtual Valley is largely due to a number of landmark objects which almost all of the subjects positioned correctly. The similarity of the "significant" and "all" object placement ratios in the other worlds may mean that there are fewer, if any, landmark objects.

The difference in Virtual Valley and Neighborhood map scores correspond to the difference in subjects' orientation scores shown in figure 2.0. This suggests that "significant object" positioning scores may be used as a simple absolute measure of map accuracy and goodness of world design. It also implies that the sketch maps for these worlds accurately represents the topological knowledge stored in the subjects' cognitive maps.

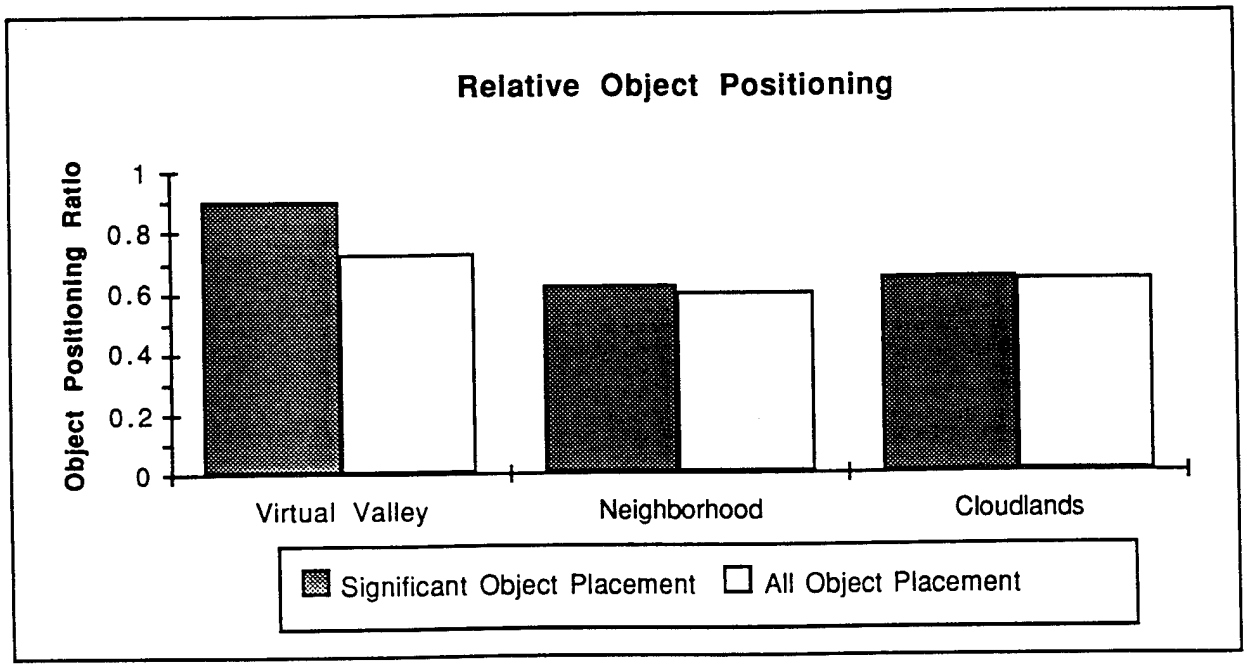



Figure 3.0: Average subject relative object positioning ratios across the three test worlds.

## CONCLUSIONS

In this study we have investigated the applicability of sketch maps as an external representation of an individual's cognitive map of a virtual environment. We have found that sketch maps reflect differences both between worlds and within worlds.

We used three methods to score the sketch maps, chosen for their simplicity and general applicability: map goodness and object class number for comparing maps from a given world, and the relative object positioning ratio for comparing maps across a range of worlds.

In two of our test worlds, Virtual Valley and Neighborhood, map goodness and object class number scores correlated significantly with the subjects' self-reported sense of orientation within the virtual world. The relative object positioning ratio also matched the difference in reported orientation between Virtual Valley and Neighborhood worlds. These two results suggest that sketch maps do indeed accurately represent the topological aspects of subjects cognitive maps.

The "significant object" ratio appears useful for comparing across worlds, while the map goodness and object class scores are useful for comparing subjects within worlds. The difference between the "significant" and "all" object placement ratios may also be used to identify worlds that have well defined landmarks.

However, the low correlation with the Cloudlands results may indicate that sketch mapping is more useful for relatively dense worlds, or that more complicated forms of sketch map analysis is needed for sparse worlds.

## ACKNOWLEDGEMENTS

The authors would like to than the reviewers for their useful comments, and Hunter Hoffman and William Winn for help with analysis of the experimental results.

## REFERENCES

1. Alfano, P.L. and Michel, G.F. Restricting the Field of View: Perceptual and Performance Effects. *Perceptual and Motor Skills*, 1990, 70, pp35-45.

2. Appleyard, D. Styles and Methods of Structuring a City. *Environment and Behavior*, 1970, vol2, pp100-118.

3. Blades. M. The Reliability of Data Collected From Sketch Maps. *Journal of Environmental Psychology*, 1990, vol10, pp327-339.

4. Briggs, R. Urban Cognitive Distance. In *Image and Environment*, (Downs, R.M. and Stea, D. Eds.), Aldine Publishing Co., Chicago, 1973, pp361-388.

5. Darken, R.P. and Silbert, J.L. A Toolset for Navigation in Virtual Environments. In *Proceedings of the ACM Symposium on User Interface Software and Technology, UIST '93*, 1993, pp157-166.

6. Downs, R.M. and Stea, D. Cognitive Maps and Spatial Behavior: Process and Products. In *Image and Environment*, (Downs, R.M. and Stea, D. Eds.), Aldine Publishing Co., Chicago, 1973, pp8-26.

7. Golledge, R.G. Methods and Methodological Issues in Environmental Cognition Research. In *Environmental Knowing*, (Golledge, R.G. and Moore, G.T. Eds.), Dowden, Hutchinson and Ross, Inc., Pennsylvania, 1976, pp300-313.

8. Griffin, D.R. Topographical Orientation. In *Image and Environment*, (Downs, R.M. and Stea, D. Eds.), Aldine Publishing Co., Chicago, 1973, pp296-299.

9. Henry, D. *Spatial Perception in Virtual Environments: Evaluating an Architectural Application*. M.S.E. Thesis, College of Engineering, University of Washington, 1992.

10. Kuipers, B. The Cognitive Map: Could it have been any other way. In *Spatial Orientation: Theory, Research and Application*, (Pick, H.L. and Acredolo, L.P. Eds.), Plenium Press, New York, 1983, pp345-360.

11. Ladd, F.C. Black Youths View their Environment: Neighborhood Maps. *Environmental Behavior*, 1970, vol 2, pp64-79.

12. Lynch, K. *The Image of the City*, MIT Press, Cambridge, Massachusetts, 1960.

13. Moore, G.T. Theory of Research and the Development of Environmental Knowing. In *Environmental Knowing*, (Golledge, R.G. and Moore, G.T. Eds.), Dowden, Hutchinson and Ross, Inc., Pennsylvania, 1976, pp138-164.

14. Neisser, U. *Cognition and Reality*, WH Freeman, San Francisco, 1976.

15. Newcombe, N. Methods for the Study of Spatial Cognition. In *The Development of Spatial Cognition*, (Cohen, R. Ed.), Hillsdale, New Jersey, Lawrence Erlbaum Associates, 1985, pp277-300.

16. Stea, D. and Blaut, J.M. Some Preliminary Observations on Spatial Learning in School Children. In *Image and Environment*, (Downs, R.M. and Stea, D. Eds.), Aldine Publishing Co., Chicago, 1973, pp226-234.

17. Walsh, D.A., Krauss, I.K., and Reginer,V.A. Spatial Ability, Environmental Knowledge, and Environmental Use: The Elderly. *In Spatial Representation and Behavior Across the Life Span: Theory and Application*, (Liben, L.S., Patterson,A.H. and Newcombe, N. Eds.), Academic Press, New York, 1981, pp321-357.

18. Weghorst, S. and Billinghurst, M. *Spatial Perception of Immersive Virtual Environments*, HIT Lab Technical Report, University of Washington, 1993.

## APPENDIX: SUBJECT SURVEY

The 24 survey questions given to subjects are listed below. For each of the questions subjects were asked to rank their responses on a scale from one to ten. The anchors for these scales are shown under the each of the questions. Responses were collected automatically using a Hypercard stack on a Macintosh computer and participants were also given theopportunity to add their own comments at the end of the survey.

### Questions

1. Sense of being there:
None -> Total

2. Ease of interaction:
Impossible -> Effortless

3. Comfort of the display hardware:
Unbearable -> Comfortable

4. Enjoyment:
Boring -> Very enjoyable

5. How easy was it to navigate?
Very difficult -> Very easy

6. Sense of orientation relative to the laboratory:
No sense of direction -> Completely orientated

7. Sense of orientation in the virtual world:
No sense of direction -> totally orientated

8. Feeling of being lost:
All the time -> Never

9. Sense of dizziness:
Never -> All the time

10. Image brightness:
Way too dim -> Way too bright

11. Color quality:
Very poor -> Very good

12. Ease of use of the glove:
Very difficult -> Very easy

13. Feeling of inclusion in the world:
Totally removed -> Actually there

14. Overall physical comfort:
Very uncomfortable -> Very comfortable

15. Understanding of where everything was in the world:
Total confusion -> Total understanding

16. Invites exploration:
Not at all -> Very much so

17. Invites introspection:
Not at all -> Very much so

18. Ease of movement around the world:
Very difficult -> Very easy

19. Ease of getting where you wanted to go:
Very easy -> Impossible

20. How engaging was it?
Not at all -> Totally

21. Image clarity:
Extremely fuzzy -> Extremely sharp

22. How comfortable are you with using computers?
Totally uncomfortable -> Totally comfortable

23. Your experience in Virtual Reality:
First time -> Very Experienced

24. Sense of presence within the Virtual World:
Very low -> Very High

# Virtual-Reality Monitoring

*Hunter G. Hoffman, Keith C. Hullfish, and Stacey J. Houston*
Human Interface Technology Laboratory, FJ-15
University of Washington
Seattle, WA., 98195
hunter@hitl.washington.edu

## ABSTRACT
We investigated whether subjects could separate memories of events experienced in virtual reality from real and imagined events: a decision process we term virtual-reality monitoring. Participants studied 8 separate spatial configurations of real geometric objects arranged on a life-sized chessboard, 8 configurations in virtual reality (an immersive, computer-simulated world), and imagined objects in 8 other configurations. On a later source identification memory test, subjects were generally able to correctly identify the sources of the events. A Memory Characteristics Questionnaire was administered to assess differences in qualitative characteristics of memories for virtual, real and imagined events. Differences were found that could potentially serve as cues to help people decide where their memories originated. Results are interpreted within the Johnson-Raye [7] theoretical framework.

**KEYWORDS:** Reality monitoring, virtual reality, source memory, presence

## INTRODUCTION
In virtual reality, life-like changes in visual imagery occur in response to the participant's own actions. Such realistic feedback often leads subjects to report that they feel "in a place" when navigating the computer-simulated world. For instance, subjects experiencing a computer-simulated Sharkworld may have the feeling they are in the ocean, moving around a shark-infested ship wreck. This subjective experience of "presence" in the virtual environment is thought to be the essence of virtual reality.

There are speculations in the virtual reality literature that presence improves sensori-motor or cognitive performance within virtual reality, and improves the efficiency of training and planning (15). It may also improve transfer of training to the real world and enhance learning. There are likely degrees of presence. Despite their immersion, at some level subjects presumably remain aware that they are only standing in the laboratory wearing a helmet. The more subjects focus their attention on the simulated environment, the more present they are likely to feel in the virtual world. On the other hand, if they were allowed to hear people whispering or telephones ringing in the laboratory, these distractions are likely to draw their attention away from the virtual world, reducing their sense of presence. We need a good measure of presence to explore speculations about the virtues of virtual reality, and to optimize the quality of the human-computer interface. Reality monitoring, a successful paradigm from Cognitive psychology, may prove helpful.

## REALITY MONITORING
Johnson and Raye [7] have developed a paradigm in which to study reality monitoring (RM), the decision process by which memories of real and imagined events are distinguished (see [6] for a review). The decision process by which people distinguish memories of real, virtual and imagined events is a related phenomenon we term virtual-reality monitoring. The present experiment is the first in a series of studies aimed at developing memory source identification confusions (virtual-RM errors) into an objective measure of presence.

Johnson and Raye propose that differences between real and imagined events as originally experienced are preserved in memory and can later serve as cues to where the memory originated. That is, memory source is inferred by the subject at the time of retrieval, based on cues associated with the target memory. RM decisions take advantage of differences in qualitative characteristics of memories from different sources. For example, compared with memories for imagined events, memories of real events tend to include more perceptual, spatial and temporal, semantic and affective (emotional) information and less information about mental effort. Consequently, a memory with a great deal of visual and spatial detail and very little evidence of mental effort would be judged to have been real. In contrast, a target memory with few perceptual cues, but abundant evidence of mental effort is likely to be identified as imagined.

In one experiment, Johnson and Raye showed subjects a list of word pairs. For some of these stimuli, the subject's task was to self-generate (imagine) the second

0-8186-7084-3/95 $04.00 © 1995 IEEE

word in the pair. For example, given the word HOT they had to think of the opposite word COLD and fill it in. For other items, the second word was already filled in and all subjects had to do was read it. Later, they took a source identification (ID) test. The test list consisted of imagined and read (perceived) words from the study list as well as new distracter items. When subjects were asked to identify the source of each item by responding "perceived" "imagined", or "new", they were quite accurate at doing so. In a second experiment, the amount of effort required to generate the imagined word was manipulated. For some subjects, the task was made easier by giving subjects the first letter of the second word in the pair. This was intended to decrease the amount of mental effort required to self-generate the second word in the pair. As predicted by Johnson and Raye, this manipulation increased confusions between memories of perceived and imagined events. This finding that confusion increases with decreases in the amount of mental effort associated with imagined events has now been demonstrated in a number of studies [2][3[4][8].

Increasing the perceptual similarity between memories of real and imagined events has also been shown to increase RM confusions [5][9]. For example, if an artist with a vivid imagination generates an image of a fruit bowl that contains many characteristics typically associated with real events (clarity of detail, vividness, rich colors etc.), these cues could later mislead the artist when she attempts to remember whether she actually saw a fruit bowl or only imagined one. More generally, memories with qualities atypical of their class are likely to result in RM errors.

According to a number of investigators [1], researchers presently lack a theoretical framework within which to study VR, and lack an objective measure of "presence", the sensation of being in a place while experiencing a VE. The present study explores the use of the RM paradigm to fill these needs. We began by investigating whether subjects can discriminate memories of events experienced in VR from real and imagined events. Subjects studied the spatial locations of geometric objects located on a chessboard. The objects were either real, imagined, or virtual. Subjects later took a source ID memory test. After the test, we explored qualitative similarities/differences between the 3 types of memories by asking subjects to rate each class of memories on an adaptation of the Memory Characteristics Questionnaire (the MCQ) developed by [10]. Our questionnaire consisted of a number of scales designed to assess a wide range of characteristics of memories (e.g., visual detail, spatial and temporal information, emotional intensity, and the subjective experience of presence). Following the logic used by Johnson and colleagues, we predict that some of the differences in the qualities of the experiences as originally experienced will get stored in association with the target memories, and will allow subjects to infer the sources of their memories on a later memory test. More specifically, we predict that cues associated with memories of virtual events will generally allow the virtual source of these memories to be identified. That is, subjects will be able to discriminate memories of virtual events from memories of real and imagined events. Our version of the MCQ was administered to investigate the virtual-RM decision process. Ratings of the phenomenological qualities associated with memories of real, virtual and imagined events may point out differences in qualities which subjects could be using in their virtual-reality decisions, helping us to understand how these decisions are made.

## THE EXPERIMENT

### Method

**Subjects.**

Twenty four college educated subjects, as well as graduate and undergraduate students from the University of Washington participated in the 2 hour experiment.

**Materials and equipment.** A configuration consisted of four identical 3-dimensional objects (either triangles, cubes, half-cylinders, or t-squares), placed in one of 32 possible relations on a life-sized chessboard. In the real and imagined world conditions the 8 x 8 square chessboard was 12' x 12'. The objects seen on the chessboard in the real condition, that subjects were asked to imagine on the chessboard in the imagined condition were approximately 14" x 14" x 14". For the imagined condition, subjects were given written directions to determine which objects should be imagined to exist on which squares of the chessboard. In the virtual world, the checkerboard and objects were scaled to approximately the same dimensions as real world stimuli. Configurations were rotated through conditions such that each configuration occurred in each of the three worlds (real, virtual, and imagined) and as new items equally often in the experiment.

The VR system consisted of a Virtual Research stereoscopic head mounted display and a hand held joystick. Both devices were equipped with Polhemus 6 DOF electromagnetic trackers, which allowed the computer to track head movements and to track the position of the joystick. Subjects moved in the worlds with the joystick.

**Design and Procedure.**

Stimuli were counterbalanced such that each of 8 possible paths, and each of the 4 object shapes, appeared in each world type (Real, Imagined, Virtual, and New) equally often for each subject.

Each subject experienced 24 of the 32 configurations in the study phase (8 real, 8 virtual, and 8 imagined). The remaining 8 configurations served as distracters (new items) in a memory test to be described shortly. The order in which subjects encountered each world type was counterbalanced. Subjects were randomly assigned to one of 3 orders such that each subject was equally likely to encounter each world first, second or third. Once in a

world, subjects studied 8 consecutive configurations before changing worlds.

Each subject practiced navigating through "shark world", a VR game, for approx. 5 min. The experiment was described as a study of cognitive abilities in real, virtual and imagined worlds. In each world they would follow a path around 4 objects (or imagined objects) placed in various positions on a life-sized chessboard (see Figures 1, 2, and 3).

They were to study where the objects were placed in relation to each other and to the chessboard in preparation for the upcoming memory test (the exact nature of which was unspecified; see Figure 4). Prior to beginning a new world, subjects were given 2 practice trials (Subjects who received the "imagined" condition first were shown what the real objects looked like). After the study phase, subjects played "Sharkworld" again for approx. 5 min. as a distracter task.

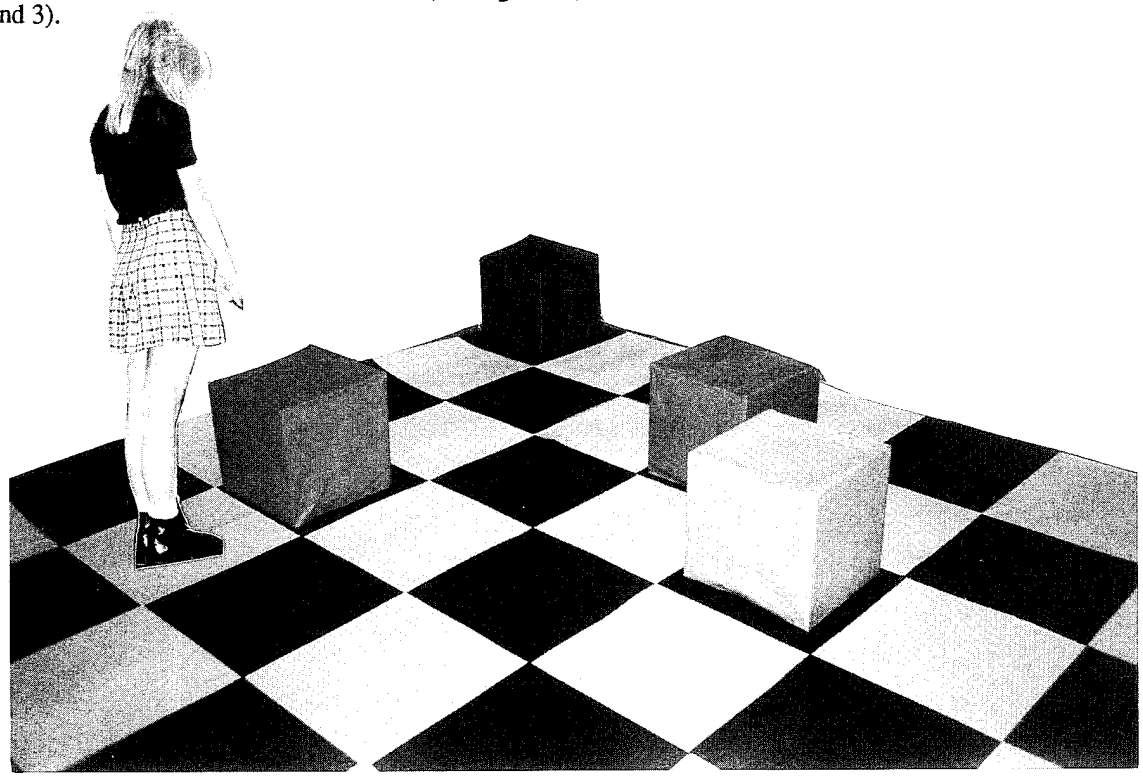


Figure 1: Subject pointing at real objects in a real world.

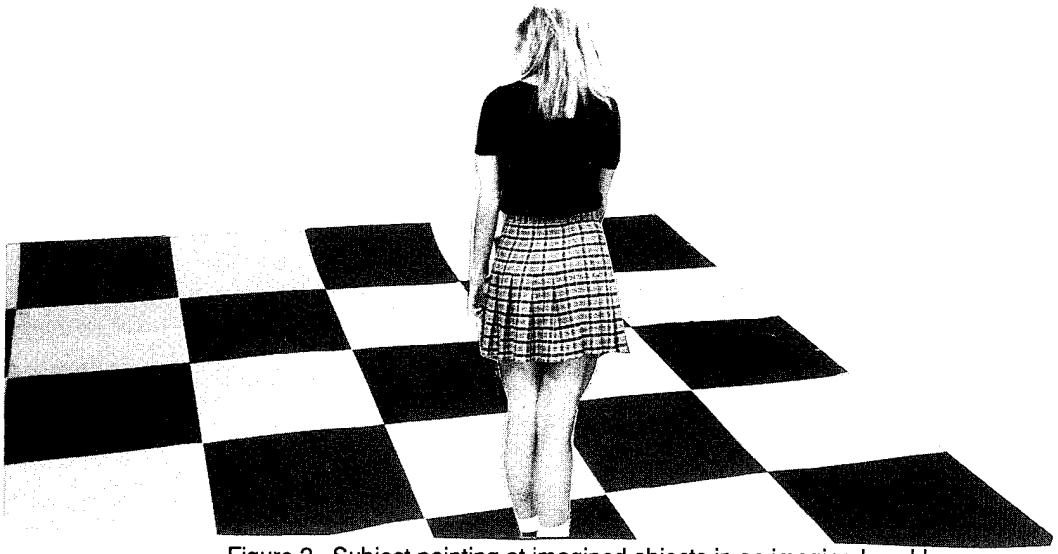


Figure 2. Subject pointing at imagined objects in an imagined world.

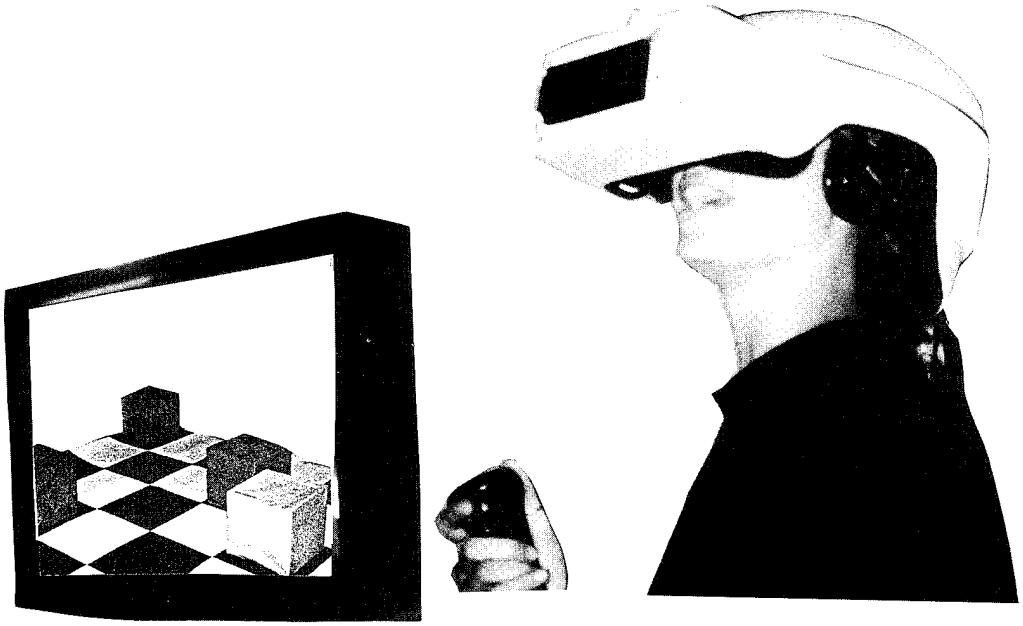


Figure 3. Subject pointing at virtual objects in a virtual world.

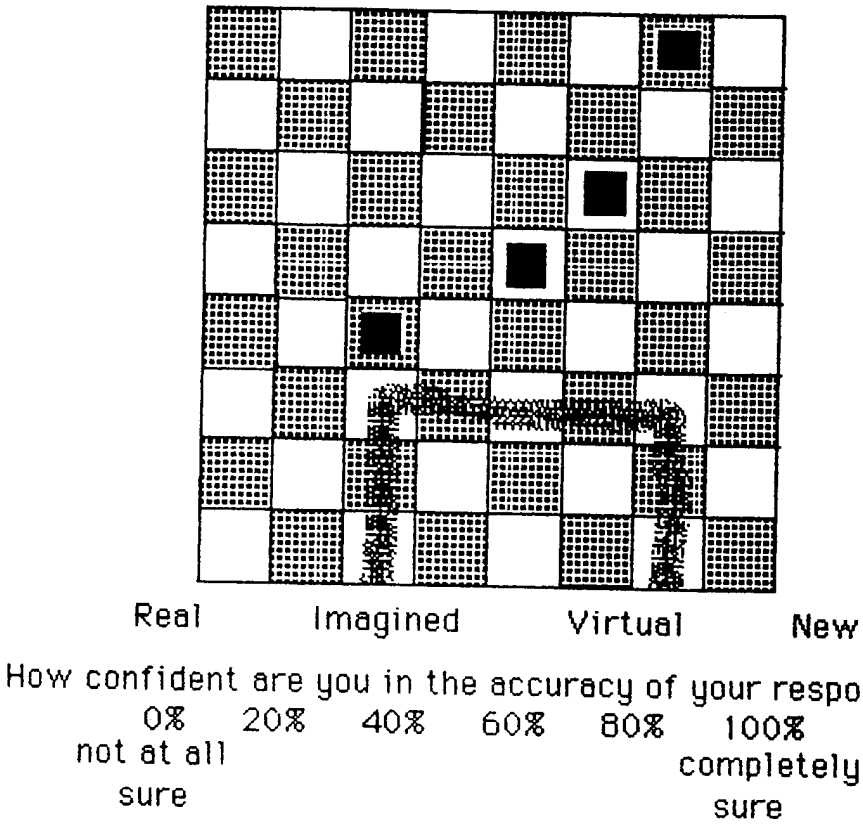



Figure 4. An example of a source identification test item (with path subject walked shown in the lower half).

All study items, and 8 new distracter items were presented for the source I.D. test. The order of the test trials was randomized. Subjects were asked a) to identify the source of origin of each test item, and b) to indicate how confident they were in the accuracy of their responses as a percentage of either 0%, 20%, 40%, 60%, 80%, or 100%. The test was self-paced.

After all memories had been identified, subjects completed a subset of the MCQ. These questions assessed a wide range of memory characteristics (e.g., visual detail, spatial and temporal information, and feelings) on a 7-point scale (e.g., the relative spatial arrangement of objects in my memory for the event is: *vague* 1 2 3 4 5 6 7 *clear/distinct* ). A questionnaire introduced by [16], supplemented by two new presence questions was then administered to get subjective self-assessments of subjects' psychological experience of "presence" in each of the environments. Each question was asked 3 times in a row, once for each world. The order in which the questions were asked were always real, imagined, then virtual.

## RESULTS

Collapsing across conditions, subjects' source ID accuracy was significantly higher than chance (.51 vs. .25 respectively), $t(23) = 7.53$, $p < .001$. No significant difference was found in the proportions of correct responses, nor of confidence ratings for items correctly attributed to real, imagined, and virtual sources. No significant differences were found for false recognitions, misses, or source monitoring errors. And there was no correlation between source monitoring errors, and subjective ratings of presence.

### Memory ratings.

The mean memory ratings were calculated for each item adapted from the MCQ (see appendix). For each rating, the following planned comparisons were conducted: real vs. virtual events, virtual vs. imagined events, and imagined vs. real events. The outcomes of these comparisons are summarized in the next two sections. Consistent with our predictions, significant differences were found for 5 of the memory characteristics for the real vs. virtual ratings, for 6 of the memory characteristics for the virtual vs. imagined events, and for 11 of the memory characteristics for the real vs. imagined comparison. Although not conclusive, this pattern suggests that there may be more cues to distinguishing real from imagined events than for the other types of decisions.

### Real vs. Virtual.

Real events were given significantly higher ratings than virtual events on the following characteristics: Clarity, spatial location, doubts/certainty. Virtual events were rated as having more color, and felt more intense. No difference in ratings of presence were found.

### Virtual vs. Imagined.

Virtual events were rated higher than imagined events on the following characteristics: color, vividness, visual detail, event detail, feeling of intensity, and presence.

### Real vs. Imagined.

Real events were rated significantly higher than imagined events on the following characteristics: clarity, color, visual detail, vividness, event detail, spatial location, event duration, +/- of feelings, overall memory, doubts/certainty, and presence questions.

## CONCLUSION

The results of the source ID test showed that subjects can distinguish memories of virtual events from real and imagined events. Combined with the MCQ ratings, these results support the Johnson-Raye theoretical framework; cues associated with memories of virtual events allow subjects to distinguish memories for these events from memories of real events and imagined events. Apparently, differences in the qualitative characteristics associated with the memories enabled subjects to infer sources at time of retrieval.

Previous research on RM has primarily used words, phrases/sentences, geometric stimuli, pictures of objects, and simple actions [6], [10]. The present results generalize these findings to memories for spatial configurations on life-sized chessboards and to memories for virtual events.

Subjects' ratings on a wide range of characteristics suggested some phenomenalogical cues that could potentially be useful in these decisions. Of particular interest was the finding that subjective ratings of "presence" for real events and virtual events were dramatically higher than ratings of "presence" for imagined events. Future research could investigate the extent to which RM and VR monitoring decisions are based on differential feelings of presence. Presumably, presence is most indicative of an event experienced in the real world. Manipulations that increase the phenomenological experience of presence associated with memories of virtual and imagined events may make them more difficult to distinguish from memories of real events on source ID tasks, and may have predictable effects on characteristics associated with memories for these events, as evident from ratings on the MCQ.

There are presently a number of limitations of VR technology (e.g., field of view, update rate, resolution) that likely reduce the realness of the experience. Future research should explore whether these qualities of the virtual experience are stored in memory, and later help subjects identify the source of these memories as "virtual". For example, one could manipulate field of view to determine whether doing so improved performance on the source ID test and increased differences in ratings (between groups) on the MCQ. It may also be interesting

to add some questions regarding these limitations to the MCQ. Findings that these artifacts of the present limitations of VR technology help subjects accurately identify the source of their virtual memories would further support the logic of using VR monitoring as a metric of presence discussed below. As the technology improves and the artifacts are reduced, the cues to source will also be reduced.

One application of VR technology is for training (e.g., training astronauts how to fix the Hubble, [12]). The more closely the virtual worlds simulate the real worlds, the more likely it is that training will transfer from one world to the other. For training purposes, the ideal (but perhaps unattainable) goal is for the virtual world to be made indistinguishable from the real world. We are presently attempting to develop virtual-RM into a metric for assessing how closely virtual events simulate real events, a sort of Turing test for quantifying the fidelity of virtual technologies. The more convincing the virtual world, the more "present" subjects will feel, and the more likely they will be to misattribute their memories for this event to a "real" source.

The Johnson-Raye theoretical framework also suggests ways to minimize transfer effects in situations where transfer is undesirable. For example, there are (admittedly controversial) concerns that violent entertainment contributes to violent behavior in the real world [11]. The advent of violent games to be played in virtual realities will likely heighten such concerns. Undesirable transfer of training could perhaps be minimized by making the phenomenal experience in the virtual world distinctively unreal (e.g., vertical flight enablement, permeable walls etc.). Doing so may help players compartmentalize these entertainment experiences separate from their pool of knowledge about the real world [6],[13],[14].

The data from this exploratory experiment were encouraging in that they showed it is reasonable to study the psychological experience of virtual reality using the Johnson-Raye framework. Unfortunately, these results contained too much variance and performance was too low to assess whether virtual-reality monitoring might provide a good objective measure of presence (e.g., no correlations between virtual-reality monitoring confusions and subjective measures of presence were found). In future research, meaningful stimulus objects (and a larger number of them) will be employed in an attempt to stabilize variance and raise accuracy and confidence.

An objective measure of presence would be a valuable tool for VR researchers, allowing us to assess the value of presence, and the conditions under which it occurs most intensely. In the process, we hope to gain a better understanding of reality monitoring, a central process in cognition.


## ACKNOWLEDGMENTS

This research was supported by AFOSR grant #F49620-93-1-0339. The authors would like to thank Drs. Woodrow Barfield, John Dunlosky, Tom Furness, Sheree Kwong See, Elizabeth Loftus and William Winn for their precious input. Special thanks to Dr Max Wells. Finally, we extend our thanks to the U.W. students and members of the community who took part in this study with such complete sincerity.


## REFERENCES


1. Barfield, W., Sheridan, T., Zeltzer, D., & Slater, M. (1995). Presence within virtual environments. In W. Barfield & T. Furness III (Eds.) *Virtual Environments and Advanced Interface Design.* London: Oxford University Press.
2. Durso, F.T., & Johnson, M.K. (1980). The effects of orienting tasks on recognition, recall, and modality confusion of pictures and words. *Journal of Verbal Learning and Verbal Behavior, 19,* 416-429.
3. Finke, R.A., Johnson, M.K., & Shyi, G.C.-W. (1988). Memory confusions for real and imagined completions of symmetrical visual patterns. *Memory & Cognition, 16,* 133-137.
4. Foley, M.A., Durso, F.T., Wilder, A., & Freidman, R. (1991). Developmental comparisons of explicit versus implicit imagery and reality monitoring. *Journal of Experimental Child Psychology, 51,* 1-13.
5. Johnson, M.K., Foley, M.A., & Leach, K. (1988). The consequences for memory of imagining in another person's voice. *Memory & Cognition, 16,* 337-342.
6. Johnson, M.K., Hastroudi, S., & Lindsay, D.S. (1993). Source monitoring. *Psychological Bulletin, 114,* 3-28.
7. Johnson, M.K., and Raye, C.L. (1981). Reality monitoring. *Psychological Review, 88,* 67-85.
8. Johnson, M.K., Raye, C.L., Foley, H.J., & Foley, M.A. (1981). Cognitive operations and decision bias in reality monitoring. *American Journal of Psychology, 94,* 37-64.
9. Johnson, M.K., Raye, C.L., Wang, A.Y., & Taylor, T.H. (1979). Fact and Fantasy: The roles of accuracy and variability in confusing imaginations with perceptual experiences. *Journal of Experimental Psychology: Human Learning and Memory, 5,* 229-240.
10. Johnson, M.K., Suengas, A.G., Foley, M.A., & Raye, C.L. (1988). Phenomenal characteristics of memories for perceived and imagined autobiographical events. *Journal of Experimental Psychology: General, 117(4),* 371-376.
11. Lazar, B.A. (1994). Why Social work should care: Television violence and children. *Child Adolescent Social Work Journal, 11(1),* 3-19.
12. Loftin, R.B. & Kenney, P.J. et al. (1994). *Virtual Environments in Training: NASA's Hubble Space Telescope Mission.* Proceedings of the 16th Interservice/Industry Training Systems & Education Conference, Orlando Florida.

13. Potts, G.R., & Peterson, S.B. (1985). Incorporation versus compartmentalization in memory for discourse. *Journal of Memory and Language, 24,* 107-118.
14. Potts, G.R., St. John, M.F., & Kirson, D. (1989). Incorporating new information into existing world knowledge. *Cognitive Psychology, 21,* 303-333.
15. Sheridan, T. (1992). Musings on telepresence and virtual presence. *Presence: Teleoperators and Virtual Environments, 1 120-126.*
16. Slater, M., Usoh, M., & Steed, A. (in press). Depth of presence in virtual environments, *Presence: Teleoperators and Virtual Environments.*
17. Zeltzer, D. (1992). Autonomy, interaction, and presence. *Presence: Teleoperators and Virtual Realities, 1,* 127-132.

## APPENDIX

**-each question was rated on a scale from 1 to 7.**

1. My memories for the real objects are: dim...sharp/clear.
2. My memories for real objects are: black & white...entirely color.
3. My memories for real objects involve visual detail: little or none...a lot.
4. Overall vividness for real objects is: vague...very vivid.
5. My memory for the real objects are: sketchy...very detailed.
6. Relative spatial arrangement of the real objects in my memories are: vague...clear/distinct.
7. The real events seem: short...long.
8. In the real events I was: a spectator...a participant.
9. At the time, the real events seemed like they would have serious implications: not at all...definitely
10. Feelings at the time (for real events) were: negative...positive
11. Feelings at the time (for real events) were: not intense...intense
12. I remember what I thought at the time (real event): not at all...clearly
13. Overall, I remember the real events: hardly...very well
14. Do you have any doubts about the accuracy of your memories for the real events: yes, a great of doubt...no doubt whatsoever
15. How strong was your sense of presence in the real chess world? weak presence...strong presence.
16. To what extent could you "feel" the presence of the geometric objects (excluding the chessboard). in the real world: weak...strong presence
17 Please rate your sense of "being there" in the real chess world: not at all...very much so.
18 To what extent were there times during the real chess world experience when the chess world became the "reality" for you, and you almost forgot about the real world outside: at no time...almost all the time.
19 The real chess world seems to me to be more like: something that I saw...somewhere that I visited.
20. How strong was your sense of presence in the real chess world? weak presence...strong presence.
21. To what extent could you "feel" the presence of the geometric objects (excluding the chessboard) in the real world? weak presence...strong

# Perception and Presence

# Quantification of Adaptation to Virtual-Eye Location In See-Thru Head-Mounted Displays

*Jannick P. Rolland*
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175, USA
+1-919-962-1901
rolland@cs.unc.edu

*Frank A. Biocca*
Center for Research in
Journalism and Mass Communication
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3365, USA
+1-919-966-7024
fbiocca@email.unc.edu

*Todd Barlow*
Department of Psychology
North Carolina State University
Raleigh, NC 27695-7801

*Anantha Kancherla*
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599, USA

## ABSTRACT

An experiment was conducted on the effect of a prototype see-thru head-mounted display (HMD) on visuo-motor adaptation. When wearing video see- thru HMDs in augmented reality systems, subjects see the world around them through a pair of head-mounted video cameras. The study looked at the effects of sensory rearrangement caused by a HMD design that displaces the user's "virtual" eye position forward (165 mm) and up (62 mm) toward the spatial position of the cameras. Measures of hand-eye coordination and speed on a manual task revealed substantial perceptual costs of the eye displacement, but also evidence of adaptation. Upon first wearing the video see-thru HMD, subjects' pointing errors increased significantly along the spatial dimensions displaced (the y and z dimensions). Speed of performance on a manual task decreased by 43% compared to baseline performance. Pointing accuracy improved by about a 1/3 as subjects adapted to the sensory rearrangement but did not reach baseline performance. When subjects removed the see-thru HMD there was evidence that their hand-eye coordination had been altered by the see-thru HMD. Negative aftereffects were observed in the form of greater errors in pointing accuracy compared to baseline. Although these effects are temporary, the results may have serious practical implications for the use of see-thru HMDs by user populations who depend on accurate hand-eye coordination such as surgeons.

## 1. INTRODUCTION

Nowhere is the promise of see-thru head-mounted displays (HMDs) and augmented reality more exciting than in medical imaging applications. "Medical Imaging, since its birth, has provided a valuable and yet non-surgical possibility to see what was unseen before: the internal world of the human body" [24, p14]. See-thru HMDs will take us one step further by simulating x-ray vision -- the internal world of the human body will be seen superimposed on the patient. Doctors will not divert their vision to a side monitor or viewing screen to see inside a patient's body. The virtual image of the internal organs and the real body of the patient will be merged. Doctors have always used natural observation of the body for diagnosis. Now that natural ability to observe the symptomology of a body will be extended and augmented by fusing normal observation with the visualizing power of the x-ray, the magnetic resonance machine, and the ultrasound machine. At least, that is the vision.

But a number of design challenges must be overcome before the promise of see-thru HMDs becomes reality. One of the challenges of such devices is providing depth information that accurately merges the virtual scene with the real scene. As we will see in this study, another is building a system that minimizes sensory rearrangement and the need for adaptation.

Two approaches to hardware design are now common. Real and virtual views of the world can be merged either: 1) via a semi-transparent mirror as with optical see-thru HMDs [4, 2, 6, 19], or 2) via video cameras mounted on the helmet as with video see-thru HMDs [1]. A discussion of design issues and the relative merits of each approach can be found in one of our recent studies [20].

This paper will describe the consequences of one key design feature of existing video see-thru HMDs -- *visual displacement* of the user's eyes to a virtual position -- the



0-8186-7084-3/95 $04.00 

entrance pupil of the HMD's cameras. We report on an experimental study of adaptation to visual displacement using a video see- through system designed and built at the University of North Carolina at Chapel Hill [7].

## 2. SENSORY REARRANGEMENT, INTERSENSORY CONFLICT AND ADAPTATION TO VIRTUAL ENVIRONMENTS

Immersive virtual environment (VE) and telepresence systems are likely to induce some form of sensory rearrangement for the foreseeable future. Video see-thru head-mounted displays are a good example of a virtual reality (VR) component that requires some form of sensory rearrangement. Sensory rearrangement is a change in the normal relationship between body movements and the resulting inflow of sensation to the central nervous system. It can also result from discoordination of one sensory inflow pattern with that of another sense -- also known as intersensory conflict [15, 17]. In VEs, sensory rearrangement and intersensory conflict can result from a discoordination of displays to the various senses. According to Welch [21], "it is not so much the absence of certain stimuli that causes serious perceptual and behavioral difficulties with telesystems, but the presence of intersensory discrepancies, such as mismatches between sensory modalities and delays of sensory feedback" (p. 1).

Intersensory conflict puts a stress on the user's body, especially when the conflict involves the vestibular system [16]. The stress has cognitive, behavioral, and physiological manifestations. For example, performance is slowed down immediately after entering a HMD-based virtual environment. Movements are short and tentative. The user may be slightly uncoordinated. Reaching behavior is uncertain and inaccurate.

The heightened effects of intersensory conflict and rearrangement can also manifest themselves as the physiological reactions of simulation sickness [3]. During extended use, users may experience sweating, eye strain, stomach awareness, and vomiting [11]. To minimize the noxious effects, susceptible users may limit their movements and actions to minimize the experience of intersensory conflict. This is a concern in all training environments. Inappropriate behaviors learned in response to the simulator can negatively transfer to the real environment where they are inappropriate.

In this human factors study we wanted to explore how a user's motor system would adapt to VE induced sensory conflict between the visual and kinesthetic-proprioceptive systems. We focused on the relationship between the eyes and the hands because intersensory conflict between vision and sensed hand position (proprioception) is critical to performance in VEs. A central component of medical, military, and other training systems is learning subtle, coordinated hand-eye movements.

### 2.1. Research Questions

The problem of adaptation is particularly important to the practical problem of see-thru HMD design. It is difficult, if not impossible, for video-based see-thru HMDs to perfectly match the natural viewpoint of the user without trading for field of view (FOV) [7]. Therefore, for large FOV systems, some adaptation will most likely be necessary. But, inevitably, there will be some perceptual costs. What are they?

This study sought to answer the following questions:

*How much will user motor performance deteriorate because the present design of video see-thru head-mounted displays displaces the eyes forward and upward?*

We predicted that the intersensory conflict initiated by the visual displacement of our see-thru HMD will extract some cost on motor performance. We were interested in getting an *exact quantitative measure* of the performance cost as a benchmark that can be used to compare the human factors performance of future designs of see-thru HMDs. We also wanted an estimate of how the cognitive and motor cost would be lessened over time by practice and adaptation to the eye displacement.

*Will users adapt to see-thru head-mounted displays and, if so, how quickly?*

The extensive literature on adaptation [18,22] -- especially research on prism displacement -- suggests that users should adapt. But much of the relevant research involves adaptation to prism goggles that displace vision to the side [e.g., 8, 18 , 5] while our video see-thru HMD displaces the eyes to a spot higher and further out than the natural location of the eyes. It was a practical design question to see how *quickly* and *fully* users would adapt to this unnatural eye location.

*Will adaptation to see-thru HMDs lead to negative aftereffects, and what is the exact extent of those aftereffects?*

If users adapt to the altered eye location of the video see-thru HMD, then the users' perceptual systems might be miscalibrated for the real world once they remove the see-thru HMD. This negative aftereffect might be manifested by altered visuo-motor coordination. Again, the literature on prism adaptation suggests that negative aftereffects were

likely [13, 22].
The presence of negative aftereffects has tremendous practical significance for the use of VEs, especially in medical applications. Consider, for a moment, the use of see-thru HMDs by surgeons. Some form of safety protocol would be necessary if use of a video see-thru HMD were to temporarily alter the hand-eye coordination of a surgeon! But the issue of negative aftereffects extends to many other VR applications as well. What detrimental negative aftereffects might influence user performance in applications requiring high levels of hand-eye coordination: e.g., engine repair, athletics, weapon aiming.

If this study indicates that some perceptual cost is evident, a program of gradual user immersion and adaptation might reduce these to tolerable limits [12, 21] or promote dual adaptation [14, 23 ] to the natural and virtual world.

## 3. METHOD

This adaptation study involved an experiment. The experiment used a 3 X 2 mixed, experimental design with three within-subjects and two between-subjects levels. The main within-subjects factor was type of HMD. The three levels of this factor were:

1) baseline task measures using *no HMD*,
2) tasks using the *see-thru HMD*, and
3) the same tasks using a *control-model of the HMD* (see description in apparatus section below).

The between subjects factor was the order in which the subjects used the HMDs: see-thru HMD or the control HMD was used first. The dependent measures were: (a) time to complete a manual task (enter pegs in a pegboard) and (b) pointing accuracy ( x, y, z coordinate space) in a pair of open loop (no feedback) pointing tasks.

### 3.1. Subjects

Fourteen subjects participated in the study, 12 were males and 2 were females. All subjects were right handed and had an interpupillary distance (IPD) of 64 mm (+/- 1mm). The latter requirement was set to match the parameters of the equipment as described in the next section. Seven had no previous experience, 1 had very little experience, 4 had some experience, and 2 had a lot of experience with HMDs. All subjects had 20/20 vision or corrected vision.

### 3.2. Apparatus & Measures

*Video see-thru head-mounted display.* The study focused on the adaptation effects of UNC's video see-thru HMD, especially the effect of eye displacement to a "virtual" location (See Figure 1). The main components of the system are a flight helmet from Virtual Research, opaque HMD using LEEP optics [10], and two miniature custom made fisheye lens video cameras. Viewers see the real world through these cameras which are located 62 mm higher and 165 mm forward from the viewer's natural eye point (see Figure 2). The cameras are laterally separated by 64 mm, which was set according to the separation of the LEEP optics of the viewer itself. The fisheye lenses were custom designed and built to match the FOV of the LEEP optics when integrated in the flight helmet, and to precisely pre compensate the optical distortion of the optical viewer as described by Edwards et al. [7]

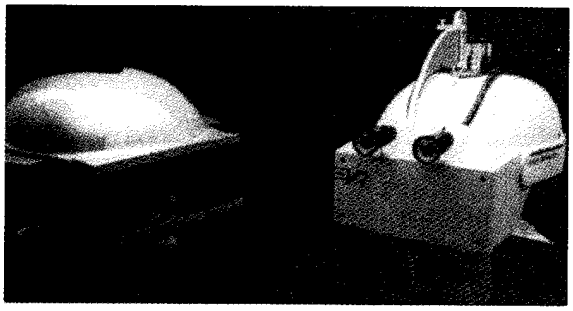


Figure 1: On the right is UNC's see-thru HMD used in the experiment. Note the camera located on the top of the helmet. On the left is a control HMD also used in the study. The control HMD was designed to match the effects of the weight and field of view of the test HMD, but without any visual displacement.

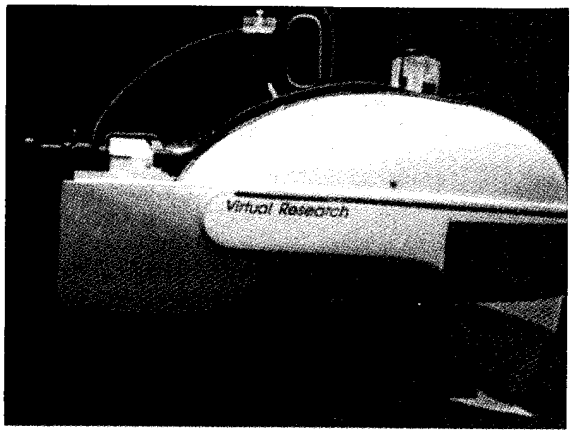
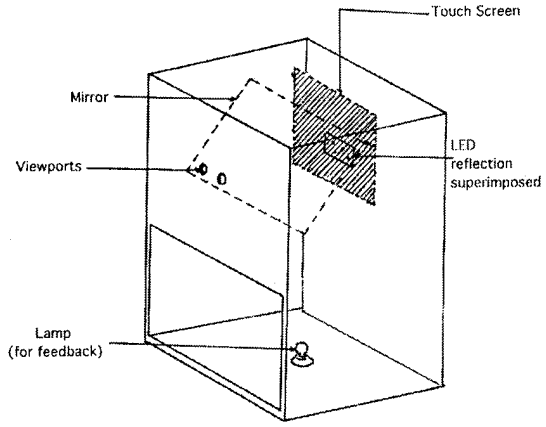


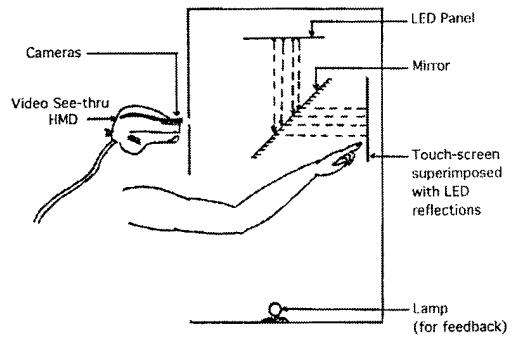

Figure 2: This picture shows the exact location of the cameras on the see-thru HMD: 62 mm higher, and 165 mm forward from the viewers natural eyepoint.

*Control head-mounted display.* The control HMD -- also shown in Figure 1 -- was designed to control for the effects of the weight and field of view of the test HMD on task performance. The control HMD matched the weight (7 lb.), and field of view (73.7 x 60.8 dg.) of the see-thru HMD. The actual field of view for each subject varied depending the size of

the subject's head since the subject's eyes varied in their distance to the window. Our estimates calculated that the field of view of the control HMD would be withing 10% on the X dimension and approximately 11% on the Y dimension. Further, we estimated that the field of view would be smaller and the bias, therefore, would be against any performance advantage for the control-HMD. Beside equating the two HMDs in weight, the location of the center of gravity of both devices was matched.




3 (a)




3 (b)

Figure 3: Diagram of the X-Y pointing accuracy measure which allowed users to point straight ahead at an object without seeing their hand. This light-sealed box had an opening at the bottom (See 3a). Subjects looked through view ports to see one of 4 LEDs reflected off a 45 degree two-way mirror (See 3b). To the subjects, the LEDs appeared to shine from the back of the box. Subjects touched the virtual LEDs without seeing their hands and receiving feedback. Their pointing accuracy was recorded on a touch screen as X-Y coordinates.

*Open Loop X-Y Pointing Accuracy Measure.* Studies of adaptation require accurate, independent measures of coordination of visual spatial position and haptic/proprioceptive location. The X-Y pointing accuracy measure used in this study was an improved version of a reliable and valid measure of adaptation with a long history [e.g., 9]. Viewing through a pair of holes, subjects saw one of four, randomly lit, red LED lights inside the dark interior of a light sealed box (Figure 3). Subjects were instructed to touch the light. A calibrated touch screen captured the exact location touched and provided a measure of X-Y pointing error. A mirror set at 45 degrees gave the subjects the illusion of seeing a light straight in front of them while preventing them from seeing their hand. This feature kept subjects from using sight of their finger to "home in" on the target or from obtaining feedback as to their accuracy.

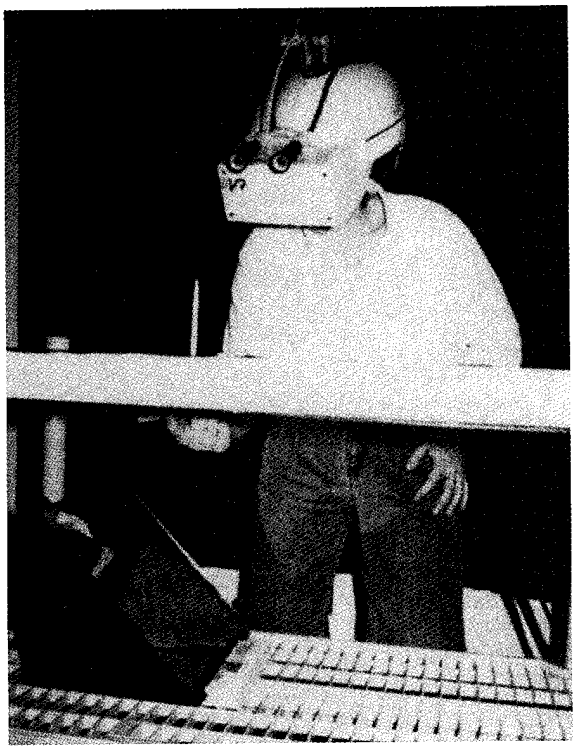


Figure 4. Representation of the measure of pointing accuracy along the Z (depth) axis. Subjects pointed at the location of the white peg underneath a shelf. Subjects received no tactile or visual feedback of their pointing accuracy. A mirror at 45$^{o}$ allowed data recording using a video camera.

*Open Loop Z Pointing Accuracy Measure.* This apparatus measured pointing accuracy along the Z axis. Subjects were seated in front of a dark shelf from which a white rod protruded as shown in Figure 4. The subject's task was to touch the point on the bottom of the shelf where the rod

would protrude if pushed through the shelf. The shelf prevented subjects from seeing their hand and gauging their accuracy (no feedback). Their pointing accuracy was recorded by a camera aimed at a polar grid pasted on the bottom of the shelf.

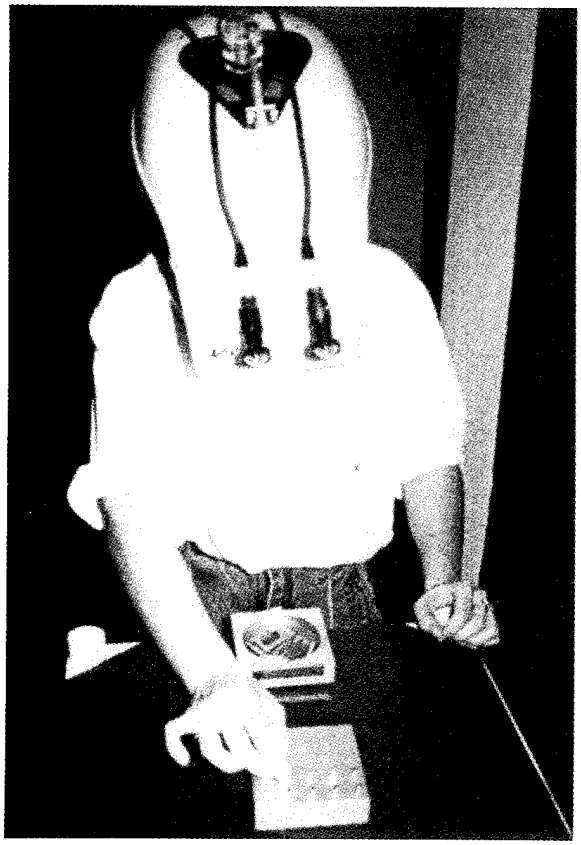


Figure 5. This pegboard task is a standard measure of manual dexterity and hand-eye coordination. In this experiment it also gave subjects immediate sensory feedback of the discrepancy between the visual sense of spatial location and their kinesthetic-proprioceptive sense of location, causing a recalibration of the latter.

*Pegboard Task.* This is a standardized test of manual dexterity (Lafayette Pegboard, model 32027). See Figure 5. The bowl of pegs was placed in front of the subjects and the board was 1 ft. away from the bowl.

### 3.3. Procedure

The order of the experimental procedure is outlined in Table 1. Following instructions various physiological and behavioral trait measurements were taken (interpupilary distance, depth perception, previous exposure to HMDs). These are not reported here.

| |
|---|
| 1. Baseline performance (no HMD)<br>X,Y,Z measures + task measures |
| 2. Performancs wearing see-thru HMD or control HMD first<br>X,Y,Z measures + task measures + X,Y,Z measures |
| 3. After-effects (no HMD)<br>X,Y,Z measures |
| 4. Performance wearing alternate HMD (control or see-thru)<br>X,Y,Z measures + task measures + X,Y,Z measures |
| 5. After-effects (no HMD)<br>X,Y,Z measures |

Table 1. Experimental Procedure.

*Baseline Procedure.* Prior to putting on any HMD, subjects were measured for their baseline performance on pointing accuracy (5 trials each) and speed on the pegboard task (10 trials). For each pegboard trial, subjects began by pressing the button on a stopwatch. After they inserted all the pegs in a left-to-right and top to bottom order, the subject turned off the stopwatch. The experimenter recorded the time. Subjects could not see the face of the stopwatch, nor were they given any feedback about their performance.

*HMD Procedure.* Depending on the order to which subjects had been assigned, subjects either put on the see-thru HMD or the control HMD following the baseline tasks. Subjects were pretested on the pointing accuracy measures (pretest X,Y,Z measures: 5 trials each). Subjects then performed 10 timed trials of the pegboard task following the same procedure used for baseline measurement before putting on a HMD. After 10 trials of the pegboard task, subjects were measured once again for their pointing accuracy while still wearing the HMD (posttest X,Y,Z measures: 5 trials each). Subjects removed the HMD and were then measured for the presence of visuo-motor aftereffects using the pointing accuracy measures (aftereffect X,Y,Z measures: 5 trials each).

Following a 5 minute rest, subjects repeated the same sequence of tasks and measures wearing the other HMD. If they wore the see-thru HMD helmet in the first part of the experiment, they now wore the control HMD and vice-versa.

After the pretest, task, and posttest using the other HMD, subjects removed the HMD and were again measured for the presence of visuo-motor aftereffects using the pointing accuracy measures.

After the experiment subjects were treated to a sequence of closed loop (feedback) trials of the X-Y pointing measure to recalibrate their visuo-motor coordination to normal. They then filled out a short simulation sickness questionnaire.

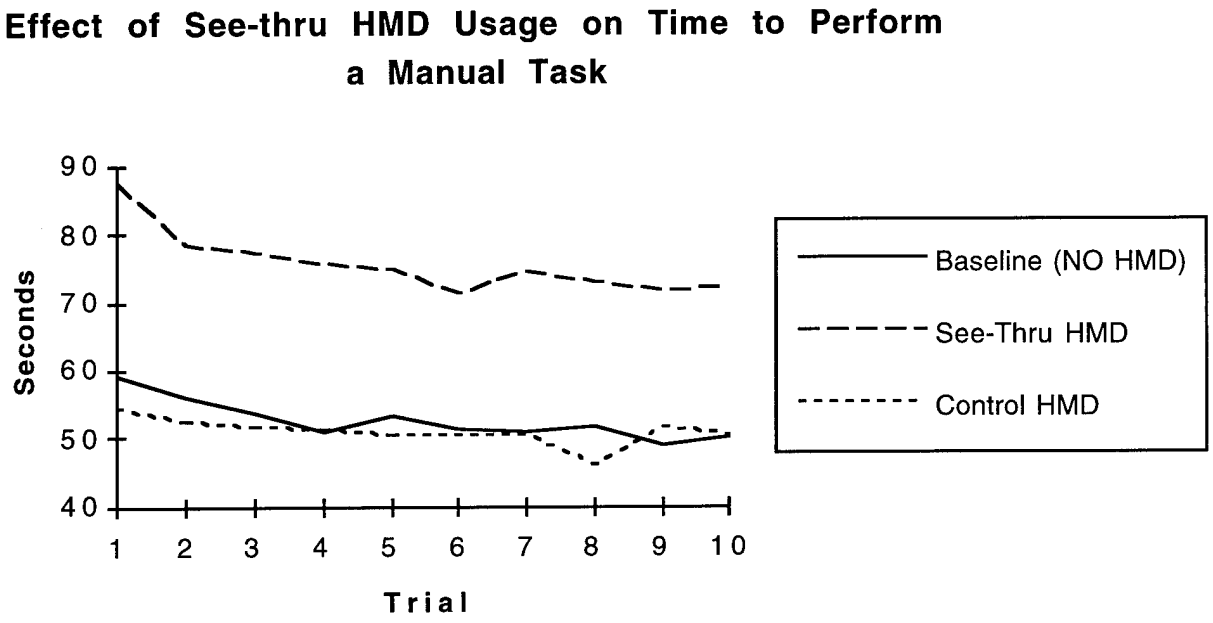



Figure 6. Effects of video see-thru HMD usage on time to perform a manual Task.

## 4. RESULTS

### 4.1. Effect of See-Thru HMD Usage on Manual Task Performance

The times to complete the manual pegboard task are reported in Figure 6. A 3 X 2 X 10 (type of HMD X HMD order X repetition) mixed, repeated measures analysis of variance was conducted on the dataset of completion times for the pegboard task. By HMD type we mesn: a) no HMD (baseline), b) the see-thru HMD, and c) the control HMD. HMD type significantly affected subjects' time to perform the manual task [$F$ (2, 22) = 102.45, $p$ < .0001]. When using the see-thru HMD (Mean = 76 sec.), subjects took an average 43% longer than their baseline performance with no HMD (Mean = 53 sec.) or the control HMD (Mean = 51 sec.) Subjects' performance times improved over the 10 trials. There was no effect for the order in which the subjects used the HMDs [$F$ (1, 11) = 1.21, $p$ = .21].

### 4.2. Effect of See-Thru HMD Usage on Hand-Eye Coordination

Figure 7 shows the amount of error subjects made when pointing at a target without visual feedback of their hand location. The pointing errors are presented for each spatial dimension: (a) X dimension, left-or-right pointing errors; (b) Y dimension, up-or-down pointing errors; and (c) Z dimension, front-or-back pointing errors. The first value in each graph (Figures 7a,b,c) is the baseline value. This value was obtained at the beginning of the experiment when the subjects had not yet put on *any HMD*. This is followed by bars for pointing errors when the subjects wore the control HMD and see-thru HMD. In some dimensions there was a significant effect when subjects used the control HMD either *before or after* the see-thru HMD. In the graphs the control HMD data are shown for both orders of HMD use. As evidence of some adaptation, bars are plotted to show differences in error levels before and after the subjects completed the manual task (pretest versus posttest).

A 2 X 2 X 3 X 5 (type of HMD X HMD order X measurement stage X repetition) mixed, repeated measures analysis of covariance was conducted. The measurement stages were: a) to before conducting the pegboard task, b) after the task, and c) after removing the HMD. The covariate was baseline pointing error (no HMD). The between subjects factor was order of HMD use. The dependent variable was pointing accuracy along each of the three spatial dimensions. Separate analyses were conducted for errors along the X, Y, and Z dimensions.

Pointing Errors Along the X Dimension (left-right of target) See Figure 7a. Although errors appear slightly higher when subjects used the control HMD, type of HMD had no effect on subjects' ability to point accurately on a target along the X dimension [$F$ (1, 11) = .98, $p$ = .35]. Effects for order of HMD usage [$F$ (1, 11) = 1.83, $p$ = .20] and measurement stage [$F$ (2, 22) = 1.01, $p$ = .38] are not significant.

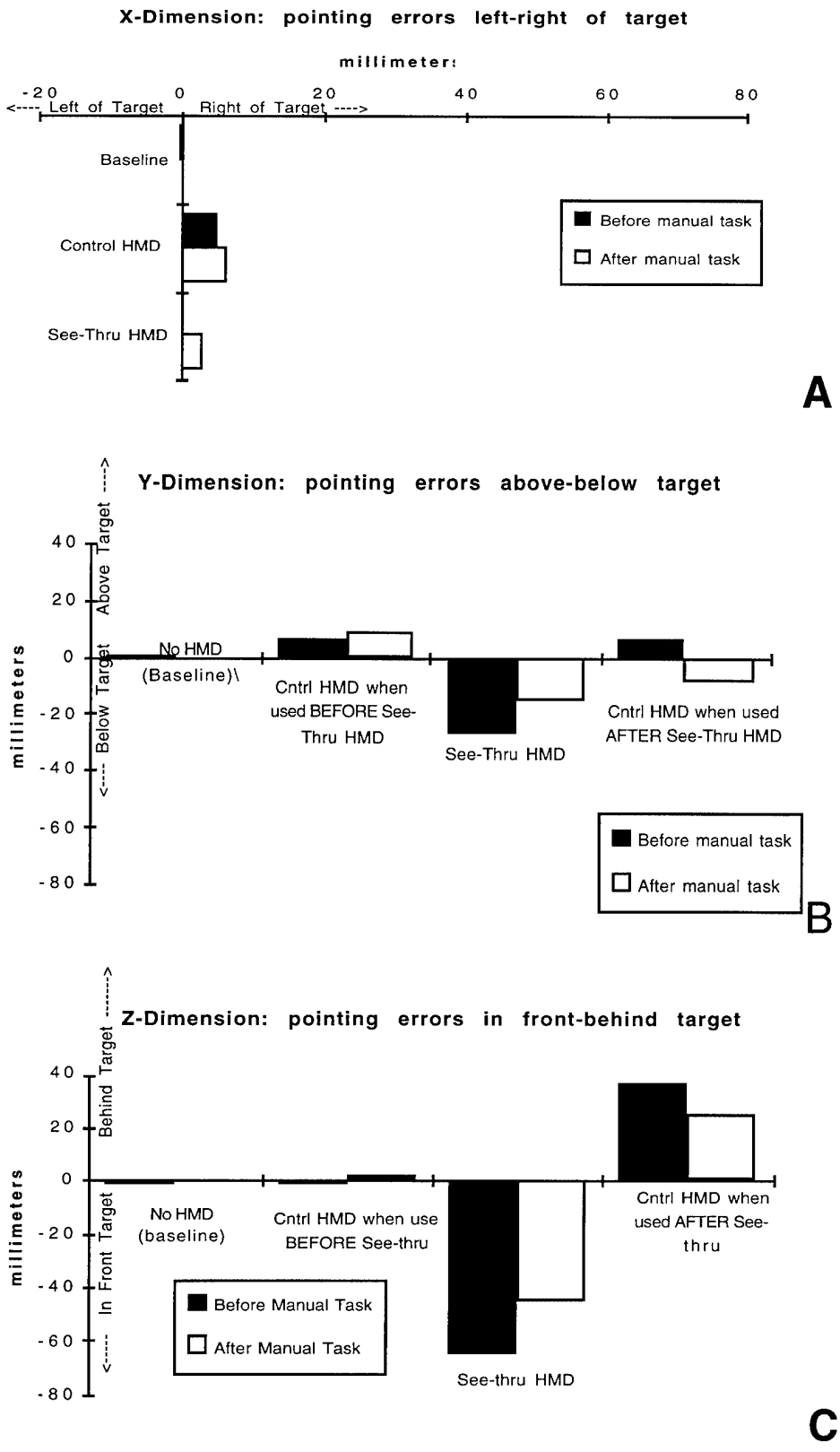
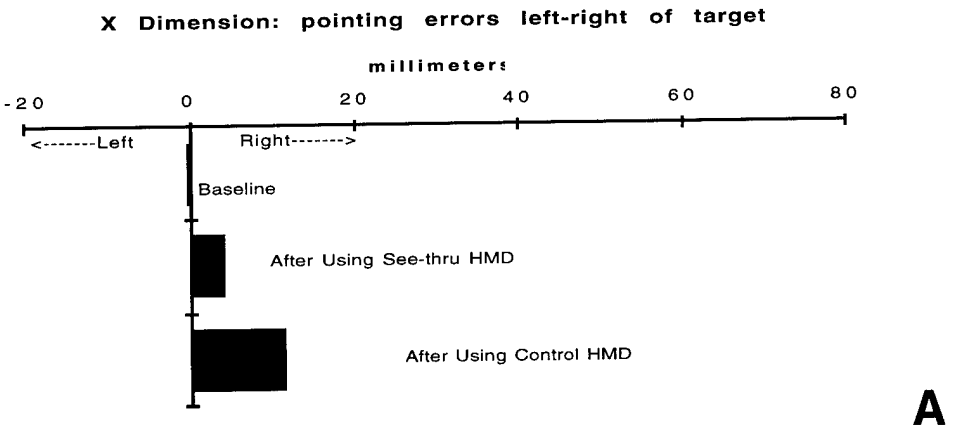
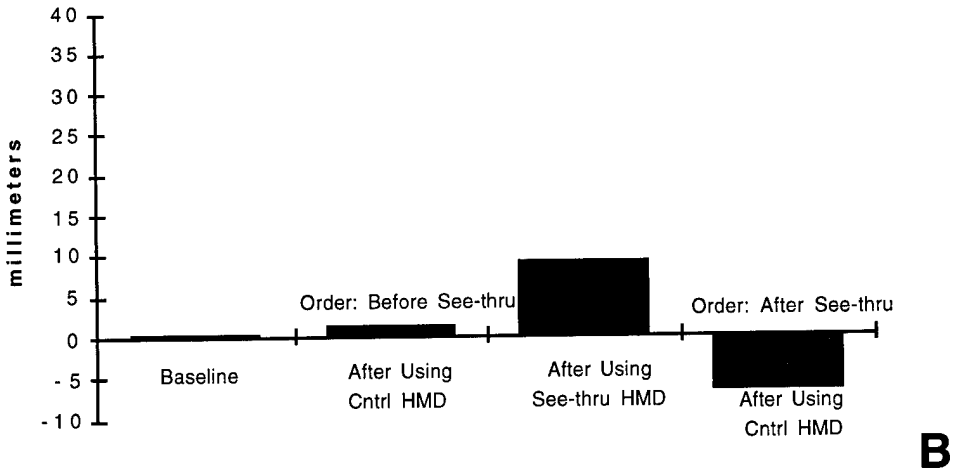


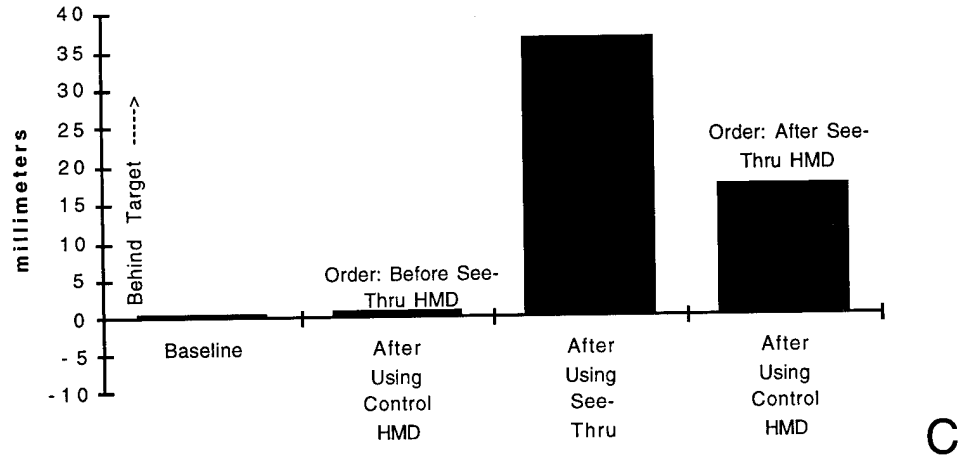

Figure 7. Effect of HMD Type on Pointing Errors By Spatial Dimension.




A




B




C

Figure 8. Aftereffect of HMD Type on Pointing Errors by Spatial Dimension.

Pointing Errors Along the Y Dimension (up-down of target). There was a significant main effect for type of HMD on pointing accuracy along the Y dimension [F (1, 11) = 9.77, p < .01] See Figure 7b. When subjects were wearing the see-thru HMD that displaced their vision upwards, they tended to point downward of the actual target position. There also was a main effect of measurement stage [F (2, 22) = 8.21, p < .002] as well as an interaction of type of HMD by measurement stage [F (2, 22) = 30.85, p < .0001]. Subjects' errors tended to decrease following their completion of the manual task while wearing a HMD, but this adaptation effect appears restricted to usage of the see-thru HMD.

Pointing Errors Along the Z Dimension (front-back of target). See Figure 7c. There was a significant main effect of type of HMD on pointing accuracy along the Z dimension [F (1, 7) = 63.29, p < .0001]. When subjects wore the see-thru HMD that displaced their vision forward, they tended to point short of the target. There was a main effect of measurement stage [F (2, 14) = 174.76, p < .0001]. Subjects' errors were less pronounced after they conducted a manual task using the HMD. Although there was no main effect for the order of HMD usage [F (1, 7) = 1.35, p < .28], there was an interaction of measurement stage and order [F (2, 14) = 4.92, p < .03]. as well as an interaction of type of HMD by measurement stage [F (2, 22) = 28.25, p < .001]. There appears to be no effect when the control HMD preceded use of a see-thru HMD. When the control HMD was used after, there appears to be an effect on pointing error. This may be due to residual after-effects from the see-thru HMD. See below and discussion.

### 4.3. HMD Use And Negative Aftereffects On Hand-Eye Coordination

The presence of negative aftereffects is commonly used as one of the more telling indicators of adaptation [16]. Figure 8 compares pointing accuracy at four times when subjects are wearing *no* HMD. After subjects removed the see-thru HMD, they displayed evidence of negative aftereffects in their hand-eye pointing accuracy as compared to their baseline performance. The different values of the control HMD along the Y and Z dimension indicate the presence of the order effects reported above. Subjects had the highest level of aftereffects following usage of the see-thru HMD. The aftereffects appear to persist and are still present when the subject uses the control HMD, but only when the latter follows use of the see-thru HMD.

## 5. DISCUSSION

The see-thru HMD appeared to have a significant effect on the visuo-motor system. Subjects' motor performance decreased. There was evidence that subjects' visuo-motor systems attempted to adapt to the display: (1) initial pointing errors decreased as subjects adapted during the manual task and, (2) they displayed significant aftereffects when they removed the see-thru HMD.

### 5.1. Effects Of Video See-Thru HMDs On Manual Performance

This study was designed to test the short-term human factors costs of visual displacement typical of large FOV, video see-thru HMDs. The see-thru HMD in this study displaced the subject's eyes to a virtual eye position, 165 mm forward and 62 mm up. The study found pronounced decrement in human performance with this generation of video see-thru HMDs. Performance on a manual task requiring hand-eye coordination took 43% longer with the see-thru HMD.

This drop in human performance appeared to be caused by intersensory conflict between the visual system and the kinesthetic system. After the subjects put on the see-thru HMD, their hand motions were uncertain and tentative. With their altered eye position pushed forward, subjects significantly overshot the pegboard as well as the peg holes in the initial trials. Errors stabilized near the end of the 10 trials as can be seen in Figure 6. Previous research on adaptation suggests that with continued practice the subjects could have performed at speeds close to their baseline speeds. But the lines in Figure 6 are somewhat parallel suggesting that movement towards baseline performance might take quite a few rounds of extended practice. But even with adaptation to visual displacement, the poorer resolution and the more limited field of view of the see-thru HMD may prevent subject performance from reaching performance levels exhibited normally without the HMD.

### 5.2. The Effect Of Video See-Thru HMDs On Hand-Eye Coordination While Using An Augmented Reality System

The discoordination of visual space and kinesthetic space appeared to be the cause of the drop in human performance. The presence of the discoordination is reflected in the data showing pointing errors. Subjects could not accurately point at objects that they saw because their eyes and hands were discoordinated by the visual displacement of the see-thru HMD.

As expected, the pointing errors were greatest along the spatial dimensions displaced by the see-thru HMD: the Y and Z spatial dimensions. The errors were systematic. Because their virtual eye position was moved up, subjects failed to compensate and pointed lower than the target before they had time to adapt. With their virtual-eye position also pushed forward, they under reached for objects

before adaptation. Errors, which were on average low at baseline, increased by several 100% after putting on the see-thru HMD. The amount of error dropped by about 1/3 as subjects began to adapt to the sensory rearrangement and would have probably dropped further over longer periods of time.

### 5.3. The Problem Of Negative Aftereffects Once the HMD is Removed

In the previous section, there was some good news: subjects began to adapt to the visual displacement of the see-thru HMD. This is a positive note for designers of video see-thru HMDs. Humans can adapt to imperfections in see-thru HMDs. But there may be a cost. Unfortunately, this positive change in the virtual environment is linked to a negative change in the real world: there are significant negative aftereffects when the subjects remove the see-thru HMD. The subjects' brains automatically recalibrated the visuo-motor system to meet the altered spatial dimensions of the virtual environment. The virtual-eye location led them to automatically rearrange their body (visuo-motor system). The visuo-motor system was still calibrated for the virtual environment once the see-thru HMD was removed. Subjects found this adaptation interfered with their performance in the "real" world. The HMD removed, subjects exhibited a negative aftereffect, overshooting the target in the pointing task in a direction opposite of the errors they made when they "entered" the virtual environment.

The presence of negative aftereffects has some potentially disturbing practical implications for the diffusion of see-thru HMDs. Surgeons and other medical professionals are the intended early users of these HMDs. Hand-eye sensory recalibration for highly skilled users like surgeons could have potentially disturbing consequences if the surgeon were to perform surgery within some period after use of a HMD.

How long might the negative aftereffects persist? It is an empirical question. In this experiment the effect of the see-thru HMD lasted long enough to disrupt the performance of those subjects who wore the control HMD after the see-thru HMD. Effects might be minimized by a program of gradual adaptation [21] in which users develop dual adaptation [23] to the real and virtual environment. Like scuba divers, users might be able to switch from one environment to another and quickly readapt.

### 5.4. Some Limitations Of The Study

Although our control HMD was able to match the weight, field of view and discomfort of the see-thru HMD, we were not able to control for the poorer resolution of the unit. Some of the effect on task performance times may be attributable to poor visual resolution, although all subjects said they could definitely see the holes on the pegboard. The actual light conditions in the real world were kept the same in all conditions. But some subjects reported that their hand was casting a shadow on the pegboard when placing the pegs, a shadow that seemed to only affect them while wearing the video see-thru HMD. While poor resolution or lighting effects might have contributed slightly to the poorer performance on the pegboard task, it is highly unlikely that poorer resolution of the see-thru HMD or lighting effects contributed in any significant way to the strong displacement in pointing observed in the subjects.

## 6. CONCLUSION

Adaptation studies have shown that the human perceptual system is relatively plastic [22]. Faced with most altered perceptual environments, users can adapt partially, if not always fully. The future use of immersive virtual environments in training and entertainment may rest on: 1) this amazing ability of the human perceptual system to adapt to altered environments *and* 2) the creation of VR hardware/software that minimizes sources of intersensory conflict. It is an empirical and practical question whether the present generation of immersive and see-thru virtual environments will provoke levels of intersensory conflict that limit the extent of their utility. While trying to engineer the technology to overcome its limitations, a parallel effort might focus on understanding how well users can adapt to the limitations of the systems [e.g., 21]. Because VE technology will not be able to produce a seamless sensory "reality" for decades to come, research on the adaptive power of the human user is likely to be of continued value in the foreseeable future.

## ACKNOWLEDGMENTS

The authors would like to acknowledge the help of R. Welch for his valuable advice before the study and on the draft, Y. Wojtkowych for work on the measurement apparatus, and Terry Yoo and David Harrison for their assistance with calibrating the miniature video cameras. Finally, we must acknowledge the important contribution of the subjects who volunteered to participate in this physically tiring study. This work was supported under an ARPA grant DABT 63-93-C-0048 and an ONR grant N00014-94-1-0503.

## REFERENCES

1. Bajura, M., H. Fuchs, and R. Ohbuchi (1992), Merging Virtual Objects with the Real World. Computer Graphics, 26, 203 -10.

2. Berman, A.L., and Melzer, J.E. (1989) Optical collimating apparatus. U.S. Patent Number 4,859,031.

3. Biocca, F. (1993). Will simulation sickness slow down the diffusion of virtual environment technology. Presence, 1 (3), 334-343.

4. Buchroeder, R.A., Seeley, G.W., and Vukobradatovich, D. (1981), Design of a Catadioptric VCASS Helmet-Mounted Display. Optical Sciences Center, University of Arizona, under contract to U.S. Air Force Armstrong Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Dayton, Ohio, AFAMRL-TR-81-133.

5. Dolezal, Hubert (1982), Living in a world transformed: perceptual and performatory adaptation to visual distortion, Academic Press.

6. Droessler, J.G., and Rotier, D.J. (1990), "Tilted cat helmet-mounted display," Optical Engineering, 29 (8), 849-854

7. Edwards, E.K., J.P. Rolland, and K.P. Keller (1993), "Video see-through design for merging of real and virtual environments," Proceeding of IEEE VRAIS'93, 223-233.

8. Harris, C.S. (1965). Perceptual adaptation to inverted, reversed, and displaced vision. Psychological Review, 72, 419-444.

9. Held, R., & Gottlieb, N. (1958). Technique for studying adaptation to disarranged hand-eye coordination. Perceptual and Motor Skills, 8, 83-86.

10. Howlett, E.M. (1983), "Wide angle color photographymethod and system. U.S. Patent Number 4,406,532.

11. Kennedy, R. , Lane, N., Lilitenthal, M.G., Berbaum, K.S., Hettinger, L.J. (1992). Profile analysis of simulator sickness symptoms: Applications to virtual environment systems. Presence, 1, 3, 295-302.

12. Kennedy, R., Berbaum, K.S., Liilenthal, M.G., Dunlap, M.P., Mulligan, B.E., Funaro, J.F. (1987). Guidelines for alleviation of simulator sickness symptomatology. Orlando, FL: Navy Training Systems Center, NAVTRA SYSCEN TR-87-007.

13. Kornheiser, A.S. (1976). Adaptation to laterally displaced vision: A review. Psychological Builletin, 83, 783-816.

14. McGonigle, B.O. & Flook, J.P. (1978). Long-term retention of single and multistate prismatic adaptation by humans. Nature, 272, 364-366.

15. Oman, C. (1991). Sensory conflict in motion sickness: an observer theory approach. In Ellis, S. R., Kaiser, M., & Grunwald, A.. Pictorial communication in virtual and real environments (pp. 363-376). London: Taylor & Francis.

16. Reason, J. T. (1975). Motion sickness. London: Academic Press.

17. Reason, J. T. (1978). Motion sickeness adaptation: A neural mismatch model. Journal of the Royal Society of Medicine, 71, 819-829.

18. Rock, I (1966), The nature of perceptual adaptation, New York: Basic Books.

19. Rolland, J.P. (1994), Head-mounted displays for virtual environments: the optical interface. Presented at the Internationale Optical Design Conference 94, Proc. OSA 22 (in press).

20. Rolland, J.P., Holloway, R.L., and Fuchs, H. (1994), A Comparison of Optical and Video See-thru Head-Mounted Displays. Proc. SPIE 2351 (in press).

21. Welch, R. B. (1995). Adaptation to telesystems. Unpublished manuscript, NASA-Ames Research Center, Moffett Field, CA.

22. Welch, R.B, Bridgemen, B., Sulekha, A., & Browman, K.E.. (1978). Perceptual modification: Adapting to altered sensory environments. New York: Academic Press.

23. Welch, R.B. (1993). Alternating prism exposure causes dual adaptation and generalization to a novel displacement. Perception & Psychophysics, 54 (2), 195-204.

24. Yao, J. (1994), Model Observers for Predicting Human Performance on Signal-Detection Tasks, Ph.D. Dissertation, University of Arizona.

# Visual Resolution and Spatial Performance:

## The trade-off between resolution and interactivity

Gerda J.F. Smets & Kees J. Overbeeke
Laboratory for Form Theory
Faculty of Industrial Design Engineering
Delft University of Technology
Jaffalaan 9, NL-2628 BX
The Netherlands

**Abstract**

*A series of experiments is reported in which subjects performed a search-and-act spatial task in conditions of reduced resolution and exploratory freedom. Images were produced using miniature cameras, comparing static camera position, passive camera movement, and head-coupled immersive VR/teleoperation conditions. By using cameras and real light, time lags could be avoided. Videoprocessors were used to artificially reduce spatial, and temporal resolutions. Results show that although spatial and intensity resolutions are very important in static viewing conditions, like those of traditional image-producing computer graphics, subjects can complete the puzzle in head-mounted (VR-like) conditions with resolutions as little as* $18 \times 15$ *pixels. Furthermore results show that animation of the image viewpoint does not always improve spatial performance when the animation is not user-controlled; in some conditions performance actually got worse by adding passive movement.*

## 1 INTRODUCTION

VR displays are usually far behind classical computer graphics displays where static image quality parameters, such as resolutions, are concerned. Often, both in the popular press and in scientific papers alike, it is stressed that resolutions will (have to) go up greatly before virtual environments can be experienced as 'the real thing'. Nevertheless, it is already possible to do some useful work in VR environments. The point this paper wants to make, and demonstrate experimentally, is that resolution is much less important for the interactive tasks where immersive VR is typically brought to bear, (where the user can explore his environment by moving his head and body), than it is in classical computer graphics applications (where static detail becomes important because the user can only explore, by directing his gaze, over this single picture).

Swartz, Wallace, & Tkacz [11] have shown, in the context of Unmanned Aerial Vehicles, that frame rate (read: passive camera movement) is more important for target detection, recognition, designation, and tracking, than is resolution. They call these results "surprising".

In the experiments reported here[1] we investigated the relative importance of various image parameters like spatial resolution (number of pixels per video frame), intensity resolution (number of grey levels per pixel), and temporal resolution (number of frame updates per second). Most experimental data concerning these resolutions have been obtained from classical psychophysics. Experimental conditions in classical psychophysics, featuring stationary observers looking at short-term pointlike flashes on stationary displays, however are far more representative of human interaction with pictures and photographs than of highly interactive systems like those employed in virtual reality – our senses did not develop while we were sitting still. Recent years have shown a growing interest in the study of our senses as perceptual systems (the approach initiated by Gibson; see [3, 4]), where the perceptual capabilities of human observers are studied while they are exploring or performing tasks involving perceptual and motor skills. These studies, recently dubbed 'active psychophysics' [13, 12, 2], show much more potential for achieving measurements of human capabilities and the technical demands that must be satisfied to support them. The present experiments fall into this line of research. Although they are not aimed to determine perceptual thresholds with great accuracy, they clearly demonstrate the order of magnitude of resolutions needed in VR conditions as compared to those of static image presentation. See Figure 1.

## 2 EXPERIMENT I

### 2.1 Apparatus

Subjects were fitted with a head mounted system containing a display and, depending on the experimental condition, a micro camera (Figure 2). The camera image is fed through a video-processor to manipulate aspects of the image stream, e.g. the number of grey values, pixels, and frames per second. The camera is a Panasonic WV-CD1 micro-camera. The camera head has a diameter of 17 mm and a length of 48 mm. It weighs approximately 20 g. The obtained PAL 625 lines video signal is manipulated with a Panasonic WJ-MW10 production mixer. The display is an electronic viewfinder of a Sony Video Hi-8 camera type CCD-V900 E. The size of the screen is 11.0 mm $\times$ 8.2 mm and it weights about 75 g. The helmet mounted system including camera, viewer and helmet, weight 350 g.

[1] Note — Details of Experiment I, discussed in the context of perceptual theory and applications to visual search strategies, can be found in [8].



0-8186-7084-3/95 $04.00 

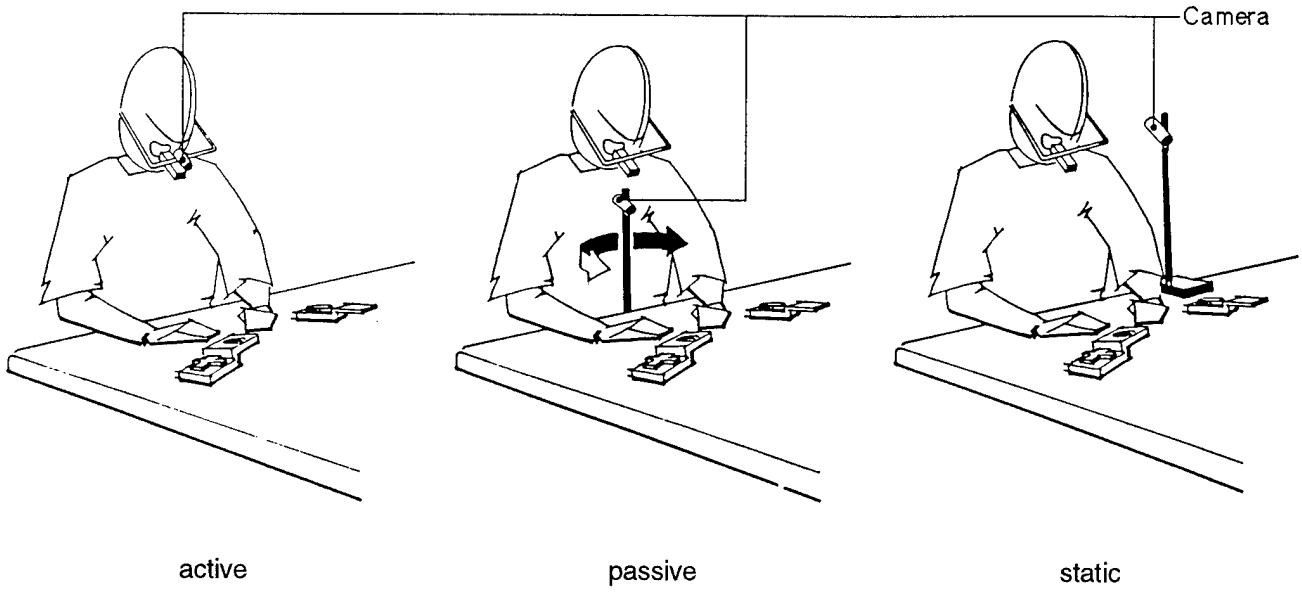



Figure 1: Image viewing conditions varying in level of interactivity.

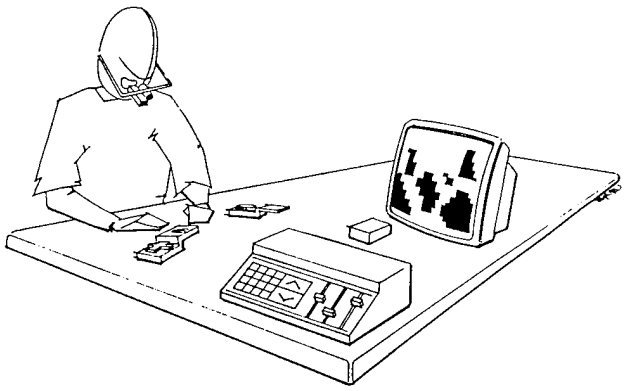


Figure 2: Apparatus.

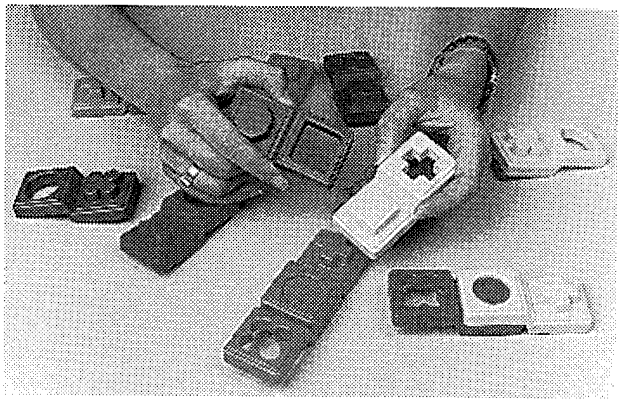


Figure 3: Puzzle in Experiment I.

The visual angle obtained by the combination of the camera and the ocular lenses is 40 degrees. The enlargement factor is 0.78. This means that the visual input does not entirely fill the field of view and that the hands of the subject appear to be further away visually. Although the same lenses were used throughout the whole of experiment I, the lens setup was changed for the replication experiment II reported below.

## 2.2 Procedure

Performance in a search-and-act task is compared for subjects whose visual information is artificially impoverished, and in three interactivity conditions of static camera, passive camera motion, head- coupled camera movement. The subjects had to complete a jigsaw- like puzzle for 4-year-old children, as shown in Figure 3. This puzzle was used with the explicit instruction and supervision that the blocks only be touched by the side so as to prevent tactile exploration. The dependent variable was the time needed to complete this puzzle as measured with a stop watch. If after 10 minutes the subject had not combined two puzzle blocks the trial was stopped and a zero result recorded.

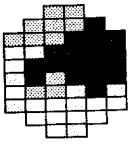
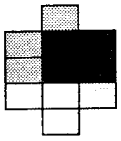
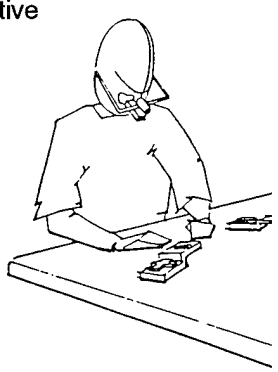
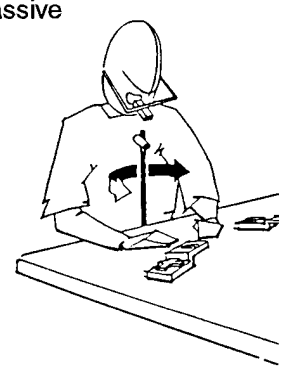
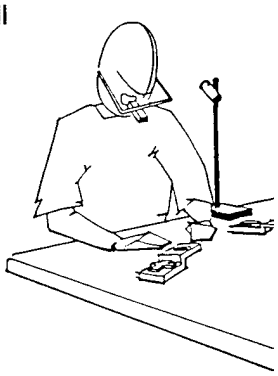
## 2.3 Subjects

Subjects were four junior members of staff who had no previous experience in psychological experiments. None had an uncorrected visual problem.

## 2.4 Design

Table 1 shows the experimental design. Under all conditions 4 grey values were used (black, white and two inter-

Table 1: Design and data Experiment I. (t.o.: trial ended on time-out when not more than two blocks had been linked after 600 *s*)

| | | | spatial resolution | | |
|---|---|---|---|---|---|
| | | | PAL 625 lines  | Mosaic 36×30  | Mosaic 18×15  |
| interactivity | temp.res. | SS | | | |
| active  | real-time | 1 | 39 | 114 | 200 |
| | | 2 | 42 | 353 | t.o. |
| | | 3 | 43 | 86 | 486 |
| | | 4 | 41 | 67 | 290 |
| | strobo-scopic | 1 | 48 | 102 | 302 |
| | | 2 | 57 | 291 | t.o. |
| | | 3 | 47 | 372 | 481 |
| | | 4 | 57 | 94 | 157 |
| passive  | real-time | 1 | 75 | 396 | 875 |
| | | 2 | 92 | 416 | t.o. |
| | | 3 | 62 | 367 | 368 |
| | | 4 | 55 | 230 | t.o. |
| | strobo-scopic | 1 | 147 | 414 | t.o. |
| | | 2 | 54 | 701 | t.o. |
| | | 3 | 108 | 370 | t.o. |
| | | 4 | 84 | 236 | t.o. |
| still  | real-time | 1 | 47 | 204 | 418 |
| | | 2 | 92 | 206 | t.o. |
| | | 3 | 77 | 263 | 492 |
| | | 4 | 64 | 713 | 486 |
| | strobo-scopic | 1 | 55 | 268 | t.o. |
| | | 2 | 80 | 877 | t.o. |
| | | 3 | 47 | 350 | 469 |
| | | 4 | 100 | 495 | t.o. |

mediate grey values). The subjects went through all conditions. Conditions were counterbalanced as to avoid order effects.

There are three independent variables, spatial resolution, temporal resolution, and interactivity. See Table 1. Spatial resolution has three levels: a PAL 625 video image, an image consisting of a 36x30 mosaic and an image of a 18x15 mosaic. Temporal resolution has two levels: real-time PAL (25 Hz), and stroboscopically sampled (5 Hz). Interactivity is manipulated by the correspondence between the observer's exploratory movements and the visual input. This is manipulated on three levels. At the first level the camera records the scene from a single viewpoint (still). At the second level a small electrical motor moves the camera on a steady track around the scene (passive). At the third the camera is attached to the observer's head (active).

### 2.5 Hypothesis

Self-generated optic flow, where the observer's exploratory movements cause shifts in the optic array, is only present when the camera is head-mounted. In this condition we expect performance to stay high, even with low spatial resolution. Therefore we predict a significant interaction, a trade off, between the independent variables. Furthermore we predict a significant main effect of both.

### 2.6 Results

Results are shown in Tables 1 (raw data) and 1 (ANOVA). Only two levels of spatial resolution were included as the $18 \times 15$ condition was clearly too difficult. However, the data show that in the active condition 3 out of 4 subjects were able to solve the puzzle. The main effect of spatial resolution was significant. This, of course, is not new. The interactivity condition is also significant, indicating the importance of actively controlled visual input. The temporal resolution effect was not significant. It might be that the differences in image update rate were not large enough. Now for our main concern. The interaction between spatial resolution and interactivity was not significant at the 0.05 level, although we expected low resolution performance to be better in the active condition. Two reasons may explain this. First, there was a strong learning effect due to the contamination of the puzzle blocks, resulting in large intersubject variances within the cells. Second, we excluded the third level of spatial resolution because of missing data. This level was too difficult for the subjects in some of the conditions. Yet if the third level of spatial resolution could have been included, the interaction would turn out significant.

## 3 EXPERIMENT II

This experiment was a partial replication of Experiment I, optimised so as to eliminate the unwanted factors identified above. The following changes were made:

### 3.1 Apparatus

The lenses of the viewer/camera system were changed so the visual angle obtained by the combination of the camera and the ocular lenses was 60 degrees. The resulting enlargement factor is 1.00. This means that the visual input fills the field of view and that the hands of the subject look as distant as usual.

Table 2: Analysis of Experiment I. (SR=Spatial Resolution; TR=Temporal Resolution; IT=Interactivity; *: $p < 0.05$, **: $p < 0.01$;***: $p < 0.001$).

| Source | SS | df | MS | F |
|---|---|---|---|---|
| SR | 845883.00 | 1 | 845883.00 | 46.46*** |
| IT | 170468.79 | 2 | 85234.40 | 4.68** |
| TR | 35752.08 | 1 | 35752.08 | 1.96 |
| SR×IT | 101418.88 | 2 | 50709.44 | 2.79* |
| SR×TR | 20833.33 | 1 | 20833.33 | 1.14 |
| IT×TR | 3283.29 | 2 | 1641.65 | 0.09 |
| SR×IT×TR | 6769.04 | 2 | 3384.52 | 0.19 |
| Residual | 655469.50 | 36 | 18207.49 | |
| Total | 1839877.92 | 47 | | |

### 3.2 Task

The subjects have to complete a specially designed puzzle, depicted in Figure 4. This puzzle excludes any learning by place, since the location of each piece was randomly varied throughout the trials. It was used with the explicit instruction and control that the pieces only be handled by the pegs as to prevent tactile exploration.

### 3.3 Subjects

Subjects were five volunteer students in Industrial Design engineering, who had no previous experience in psychological experiments. None had an uncorrected visual problem.

### 3.4 Design and Results

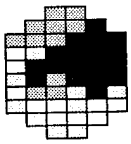
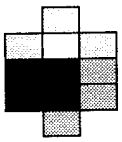
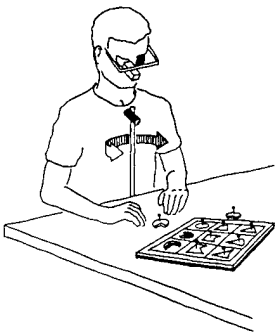
The design was simplified by leaving out the temporal resolution manipulation. The independent variables spatial resolution and interactivity were retained with identical levels.Design and results are shown in Table 3 (raw data) and Table 4 (ANOVA). The lowest spatial resolution condition was again excluded from the ANOVA, since it contained a lot of 'time-out' cells. Again, it can be seen however that in the active condition three out of five subjects still can resolve the puzzle, a much better result than in both other movement conditions (passive and still).

The main spatial effect is again significant ($p < 0.01$). Interactivity is also significant ($p < 0.05$). Now we return to the point of our major concern. The interaction between spatial resolution and interactivity is significant ($p < 0.05$). With decreasing spatial resolution observers perform better when actively controlling the camera with their head movements as compared to conditions were the camera was passively moved or still.

## 4 DISCUSSION

The results of both experiments show that the added interactivity of VR can compensate for losses in spatial resolution in a way that passively animated images cannot, but that the perception/action feedback loop is cut off if the temporal resolution (strobe rate manipulation) is too low. The advantages of VR conditions match up well with similar results from medical prosthetics, teleoperation, and Gibsonian and Gestalt perception theory (e.g., [4]). For instance,

Table 3: Design and data Experiment II. (t.o.: trial ended on time-out when no piece of the puzzle had been placed after 600 *s*)

| | | spatial resolution | | |
|---|---|---|---|---|
| | | PAL 625 lines | Mosaic 36×30 | Mosaic 18×15 |
| | |  |  |  |
| interactivity | SS | | | |
| active  | 1 | 33 | 73 | 242 |
| | 2 | 35 | 141 | 259 |
| | 3 | 32 | 51 | 210 |
| | 4 | 24 | 190 | t.o. |
| | 5 | 27 | 88 | t.o. |
| passive  | 1 | 56 | 143 | t.o. |
| | 2 | 51 | 596 | t.o. |
| | 3 | 41 | 153 | t.o. |
| | 4 | 48 | 531 | t.o. |
| | 5 | 101 | 600 | t.o. |
| still  | 1 | 35 | 422 | t.o. |
| | 2 | 49 | 123 | t.o. |
| | 3 | 41 | 123 | 384 |
| | 4 | 97 | 472 | t.o. |
| | 5 | 56 | 436 | t.o. |

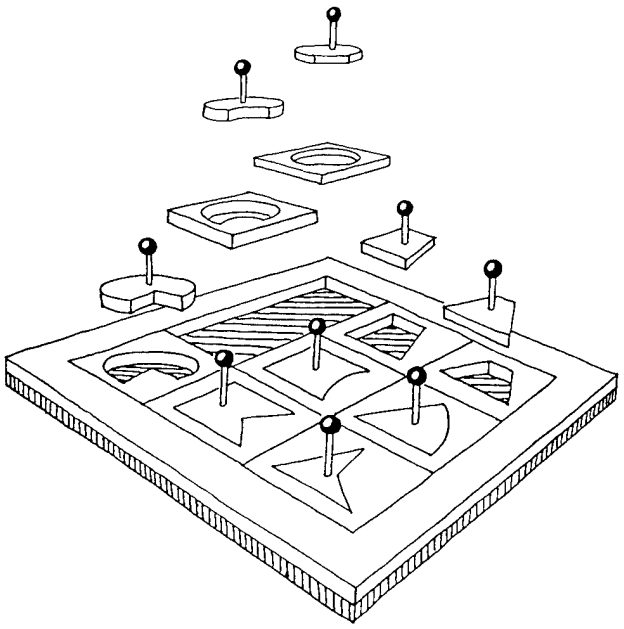


Figure 4: The redesigned puzzle used in Experiment II.

Bach-y- Rita found in his Tactile Visual Substitution System TVSS (see [1]), in which digitized camera images are presented to a congenitally blind subject by means of an array of vibrating pins placed against the skin of his back, that no recognition takes place unless the subject himself controls the movements of the camera. Also, the trade-off fits in well with Sheridan's three factor model of telepresence, where three independent factors together add up the quality of presence realized by a teleoperator system [6]. These three factors are (i) the extent of sensory information (such as resolution), (ii) the amount of control over sensors (called 'interactivity' in this paper), and (iii) the ability to modify one's environment. In our own applied research we are using the above experiments also for the development of non-immersive systems for teleoperation and non- immersive surgery using the Virtual Window technique [5, 9, 10],

Table 4: Analysis of Experiment II. (SR = Spatial Resolution; IT = Interactivity; *: $p < 0.05$, **: $p < 0.01$).

| Source | SS | df | MS | F |
|---|---|---|---|---|
| SR | 388968.53 | 1 | 388968.53 | 29.26** |
| IT | 140221.07 | 2 | 70110.53 | 5.27* |
| SR×IT | 92785.87 | 2 | 46392.93 | 3.49* |
| Blocks | 98595.53 | 4 | 24648.88 | 1.85 |
| Residual | 265844.87 | 20 | 13292.24 | |
| Total | 986415.87 | 29 | | |

in which movement parallax is produced by adapting the viewpoint of a real or virtual camera to match the displacements of the observer's head in front of the display. For several application areas, e.g., X-ray inspection, these easily outperform static binocular displays [7].

# 5 CONCLUSIONS

The experiments reported in this paper provide behavioural evidence about the relative importance of spatial and temporal resolution factors (pixels per frame and frames per second, respectively) in static, dynamic, and interactive display conditions. Although the experiments were performed using real light and cameras, the results apply equally well to computer-based display systems. Results show that especially in interactive viewing conditions of Virtual Reality, static resolution qualities are a relatively minor concern for (some) spatial orientation and performance tasks, as compared to their prominence for static and passive animation displays.

## References


[1] Bach-y-Rita, P. (1972) *Brain Mechanisms in Sensory Substitution.* Academic Press, London.

[2] Flach, J.M. (1990) Control with and eye for perception: Precursors to an active psychophysics. *Ecological Psychology*, 2(2): 83-110.

[3] Gibson, J.J. (1966) *The Senses considered as Perceptual Systems.* Houghton Mifflin, Boston, MA.

[4] Gibson, J.J. (1979) *The Ecological Approach to Visual Perception.* Houghton Mifflin, Boston, MA.

[5] Overbeeke, C.J., & Stratmann, M.H. (1988) *Space through Movement.* Doctoral dissertation, Delft University of Technology, Delft, The Netherlands.

[6] Sheridan, T. (1992) Musings on telepresence and virtual presence. *Presence* 1(1), 120-126.

[7] Smets, G.J.F. (1992) Designing for Telepresence: The interdependence of movement and visual perception implemented. In *Fifth IFAC/IFIP/IFORS/IEA symposium on analysis, design, and evaluation of man-machine systems.* (International Federation on Automatic Control), 9- 11 June 1992. The Hague, Netherlands, p. 1-7.

[8] Smets, G.J.F., & Overbeeke, C.J. (1993) Trading off spatial versus temporal resolution: About the importance of actively controlled movement for visual perception. In: Brogan, D., Gale, A., & Carr, K. (Eds.) *Visual Search 2.* Taylor & Francis, London, England.

[9] Smets, G.J.F., Stratmann, M.H., & Overbeeke, C.J. (1988) Method of causing an observer to get a three-dimensional impression from a two- dimensional representation. *US Patent 4,7575,380.*

[10] Smets, G.J.F., Stratmann, M.H., & Overbeeke, C.J. (1990) Method of causing an observer to get a three-dimensional impression from a two-dimensional representation. *European Patent 0189232.*

[11] Swartz, M., Wallace, D., & Tkacz, S. (1992) The influence of frame rate and resolution reduction on human performance. *Proceedings of the human factors society 36th annual meeting, 1440-1444.*

[12] Warren, R. (1988) Active Psychophysics: Theory and Practice. In: Ross, H.K. (Ed.) *Fechner Day '88: Proceedings of the 4th Annual Meeting of the International Society for Psychohysics*, p 47-52. Stirling, Scotland.

[13] Warren, R. & McMillan, G.R. (1984) Altitude control using action- demanding interactive displays: Toward an active psychophysics. In *Proceedings of the 1984 Image III Conference*, p 37-51, Air Force Human Resources Laboratory, Phoenix, AZ.

# Presence in Virtual Environments as a Function of Visual and Auditory Cues

Claudia Hendrix and Woodrow Barfield
Laboratory for Interactive Computer Graphics and Human Factors
Department of Industrial Engineering, FU-20
University of Washington, Seattle, WA
barfield@u.washington.edu

## Abstract

This paper reports the results of two experiments each investigating the sense of presence within visual and auditory virtual environments. The variables for the studies included the presence or absence of head tracking, the presence or absence of stereoscopic cues, the geometric field of view (GFOV) used to design the visual display, the presence or absence of spatialized sound and the addition of spatialized versus non-spatialized sound to a stereoscopic display. In both studies, subjects were required to navigate a virtual environment and to complete a questionnaire designed to ascertain the level of presence experienced by the participant within the virtual world. The results indicated that the reported level of presence was significantly higher when head tracking and stereoscopic cues were provided, with more presence associated with a 50 and 90 degree GFOV when compared to a narrower 10 degree GFOV. Further, the addition of spatialized sound did significantly increase ones sense of presence in the virtual environment, on the other hand, the addition of spatialized sound did not increase the apparent realism of that environment.

## 1. Introduction

The concepts of virtual presence and telepresence have received considerable attention in light of recent advances in display technologies and virtual world development. One of the objectives of virtual world designers is to create virtual worlds which lead to a strong sense of presence within the computer-generated environment. Presence, the sense of you being there, typically at a remote site as in teleoperation; or of the sense that something virtual is here, as in the case of a virtual object viewed using a head-mounted display (HMD), is generally considered to be a desirable outcome for virtual environment participants. Presence, however, is not an entirely new concept nor is it a phenomenon limited to complex immersive display technologies; it has long since been a preoccupation for authors, artists, and film producers to create mediums which will draw in and enhance the level of presence of its observers [1].

A great emphasis has been placed on creating methods for measuring presence. There are several reasons why developing a metric for presence is an important goal for designers of virtual environments. First, a measure of presence will aid software designers in evaluating the effectiveness of different features used to design virtual worlds. For example, once a valid measure of presence is available, it will be possible to determine what field of view, frame rate, or pixel resolution for an HMD is necessary to induce a given level of presence. Second, a method for measuring presence will aid human factors engineers in designing studies which relate presence in virtual environments to performance. This is an important goal because it may not be necessary to always induce the highest level of presence for a given task. Some tasks may require a strong sense of presence, others may not. Finally, the development of models focusing on what subject, hardware, and software factors produce presence may form the theoretical base for the work being done in the virtual environment field.

This paper reports the results of two experiments each consisting of a series of comparative studies investigating the effect of various display parameters on reported levels of presence in computer-generated virtual environments. The first experiment investigated the effect of visual display parameters while the second investigated the effect of auditory display parameters. Both experiments used a subjective questionnaire approach to measure presence, specifically in the context of visual and auditory display parameters for virtual environments. It was hypothesized that visual factors that resulted in an increased spatial realism of the virtual environment would increase presence, and for the auditory domain, the use of spatialized sound would increase presence.

Future studies from our laboratory are focusing on performance as well as subjective measures of presence due to the idea that presence is affected by the task performed within the virtual world.



0-8186-7084-3/95 $04.00 

**Table 1. Summary of each question used in the current study by category.**

| |
|---|
| Presence |
| (1) If your level of presence in the real world is "100", and your level of presence is "1" if you have no presence, rate your level of presence in this virtual world. |
| (2) How strong was your sense of presence, "being there", in the virtual environment? |
| Stereopsis |
| (3) How realistic did the virtual world appear to you? |
| (4) To what degree did the room and the objects in the room appear to have realistic depth/volume? |
| (5) Did you feel that you could have reached into the virtual world and grasped an object? |
| Headtracking |
| (3) How realistically did the virtual world appear to interact with you? |
| (4) How realistically did the virtual world move in response to your head motions? |
| GFOV |
| (3) How realistic did the virtual world appear to you? |
| (4) Did the objects appear to be compressed or magnified as compared to real world objects? |
| (5) Did your view of the world seem too narrow or wide as compared to the real world? |
| (6) Did you feel that the objects in the virtual world appeared proportionately correct, that is, did they have about the right size and distance in relation to you and other virtual objects? |

## 2. Experiment 1: Visual Cues and Presence

### 2.1Method

Twelve subjects volunteered to participate in the study. The group consisted of university students with a mean age of 27 years and included 6 male and 6 female students. All subjects had normal or corrected-to-normal visual acuity. The first study was run as a 1 x 2 within subjects design. The independent variable for the study was monoscopic versus stereoscopic viewing conditions. The second study was also run as a 1 x 2 within subjects design with the independent variable consisting of head tracking (present or absent). The third study was run as a 1 x 3 within subjects factorial design. The independent variable was the geometric field of view (GFOV) used to design the visual display, either 10, 50, or 90 degrees. A larger GFOV minifies a scene, a smaller GFOV magnifies a scene.

The dependent variables represented responses to questions evaluating the degree of presence and display fidelity within the virtual world. Table 1 summarizes the questions used for each study. The first two questions relating to presence were repeated across all studies. The following questions were presented separately for each of the three studies. Each question was numbered so that it can be identified by its numerical reference in the results section of the current experiment.

The questionnaire was given to each subject at the beginning of each of the studies. Subjects were instructed to navigate about the environment, ad lib, and familiarize themselves with the environment. Once subjects felt familiar with the environment, they were asked to answer the questionnaire. No attempt at counterbalancing was done within each study since subjects were given the option of going back and forth between virtual worlds while answering the questionnaire. No counterbalancing was done across experiments since there was no indication of order effects during pilot studies.

Unless otherwise specified as a variable above, the standard viewing conditions for the virtual worlds was a stereoscopic mode with a 50 degree GFOV and with headtracking. Headtracking was added to the display using a Polhemus 3Space Fastrak magnetic tracking device. The headtracking device had three degrees of freedom; front / back, lateral left / right, and up / down. Rotation was not incorporated into the device given that the display was non immersive and in a fixed location. Furthermore, the virtual worlds consisted of a 10 x 10 meter computer-generated room which contained familiar objects and included such things as tables and chairs, a bookshelf, a soda machine, a photocopier machine, paintings, etc. (Figure 1). These objects were scaled to match the size and proportion of similar, real world objects. The camera eyepoint elevation was set at 1.10 meters above the floor, matching that of an average sized person sitting in an averaged sized chair. All virtual objects were the same for all experimental conditions. The layout of the virtual objects in the computer-generated world, however, did change for each of the experimental conditions.

The virtual worlds were generated using an in-house imaging software and a Silicon Graphics Indigo Extreme2 computer workstation. The images were viewed on a 6' x 8' rear projection screen using a GE 610 projection system. The stereoscopic condition was displayed using a time multiplexed shutter technique (StereoGraphics Corporation) with 1280 x 512 pixel resolution. Monoscopic conditions were also presented in this manner, however, for these conditions the image disparity was set to zero. This was done to avoid having subjects take off the stereo glasses for the monoscopic conditions and thereby avoiding any additional cues that might influence their responses. Headtracking was added to the display using a Polhemus 3Space Fastrak magnetic tracking device. Subjects were seated in front of the projection screen such that their position subtended a 90 degree GFOV, with the

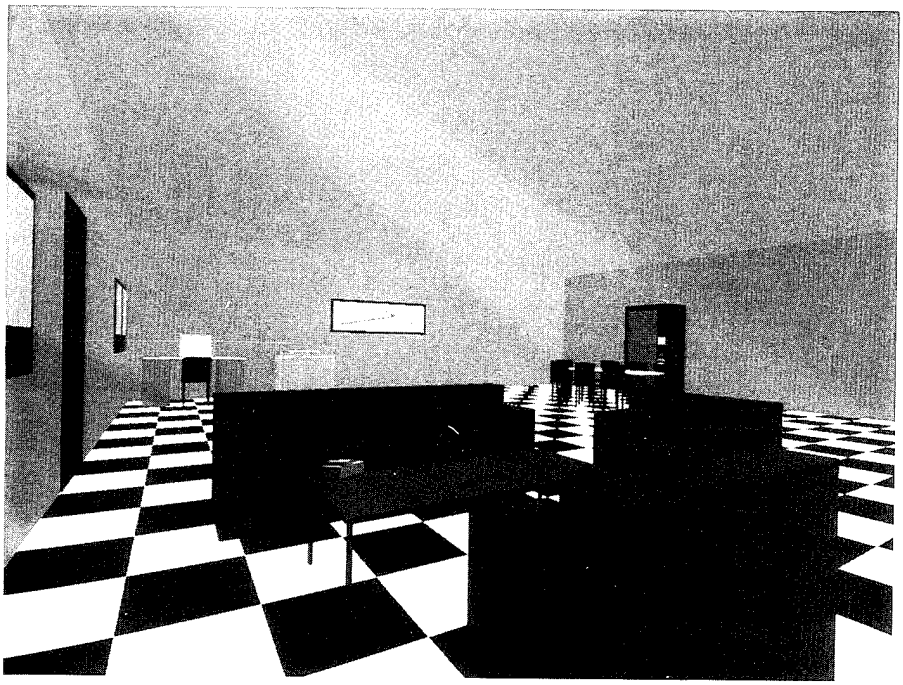


**Figure 1. Photograph illustrating the virtual world used in the comparative studies. This particular photograph shows a virtual world with a 90 degree GFOV representing a monoscopic display format.**

display screen. Subjects navigated through the environment using a mouse located on a small table in front of them.

## 2.2 Results

The results of the analysis are reported in tabular format in the following sections. Each question is referenced by number and by a brief descriptive of the question content. For a more detailed description of the questions reported in the following section, please refer to the methodology section of the current experiment. The questions for the subjective presence questionnaire used ordinal response scales and thus nonparametric statistics were used to analyze the data (for a discussion of scaling techniques, see[2]). Specifically, the Wilcoxon matched-pairs signed-ranks test was used to access the difference in the reported level of presence for independent variables with two levels. For independent variables with three levels the Friedman nonparametric two-way ANOVA on ranks was used, as was a nonparametric multiple comparison test.

The results indicated that there was a significant difference in reported levels of presence and interactivity of the virtual environment when a stereoscopic display was used (Table 2). However, the addition of stereopsis did not significantly increase the overall realism of the virtual environment. In addition to the questionnaire, subjects were encouraged to add any additional comments relating to the effect of adding binocular disparity to the display. The most prevalent comments included the complaint that the addition of binocular disparity caused the images to appear fuzzy and less defined when examined up close (the effect of ghosting) as compared to far away and, as a result, detracted from their sense of realism of the virtual world.

A significant difference in reported levels of presence, realism and interactivity in which the user interacts with the virtual environment resulted when headtracking was added to the stereoscopic display (Table 3). Most subjects expressed an increase in enjoyment when headtracking was added to the virtual world. Reactions to the addition of headtracking included, among others, subjects standing on their chair to see the top of objects, standing and stooping, turning sideways or backwards and looking back over their shoulder at the display screen, tilting their head sideways, and leaning forwards and backwards in their chair. Several subjects commented that they thought that their sense of presence would increase even more with the addition of headtracking given some sort of interactive searching task while in the virtual world.

The results indicated there was a significant difference in reported levels of presence for the different GFOVs used to design the display. In addition, questions relating the overall realism and correctness of proportionality were also found to be significantly different for the three GFOVs. For those questions where significance was found between the three GFOVs used in the current study, further nonparametric analysis using the Wilcoxon matched-pairs signed ranks test (a nonparametric multiple comparison test) indicated that the average responses were significantly different between the 10 degree GFOV and the 50 and 90 degree GFOVs. The results

**Table 2. Results of the effect of monoscopic versus stereoscopic displays on presence and display fidelity.**

| Question Number | Significance Level* | Average Response Monoscopic Display | Average Response Stereoscopic Display |
|---|---|---|---|
| 1. Presence (1-100) | T = 2.57, p < .03, N = 12 | 47.67 (std = 23.62) | 63.58 (std = 19.08) |
| 2. Presence (1-5)** | T = -2.7, p < .02, N = 12 | 3.08 (std = 0.90) | 2.33 (std = 0.78) |
| 3. Realism** | T = -1.59, p > .14, N = 12 | 3.17 (std = 1.03) | 2.67 (std = 0.89) |
| 4. Depth/Volume** | T = -3.39, p < .006, N = 12 | 3.08 (std = 0.79) | 1.92 (std = 1.00) |
| 5. Reach/Grasp** | T = -3.77, p < .003, N = 12 | 3.25 (std = 0.97) | 2.17 (std = 0.72) |

* Wilcoxon matched-pairs signed-ranks ** where 1 = very much so and 5 = not at all.

**Table 3. Results of the effect of headtracking versus non-headtracking displays on presence and display fidelity.**

| Question Number | Significance Level* | Average Response No Headtracking | Average Response Headtracking |
|---|---|---|---|
| 1. Presence (1-100) | T = 4.32, p < .001, N = 12 | 43.33 (std = 20.15) | 63.33 (std = 17.27) |
| 2. Presence (1-5)** | T = -3.45, p < .005, N = 12 | 3.17 (std = 0.83) | 2.42 (std = 1.00) |
| 3. Realism ** | T = -4.75, p < .0006, N = 12 | 3.83 (std = 0.83) | 2.92 (std = 1.08) |
| 4. Responsiveness** | T = -2.68, p < .02, N = 12 | 4.42 (std = 0.67) | 3.08 (std = 1.16) |

* Wilcoxon matched-pairs signed-ranks ** where 1 = very much so and 5 = not at all.

are listed in Table 4.

Subjects expressed a great deal of discomfort while navigating in the 10 degree GFOV world. Most commented on the difficulty in navigating in this world because the room appeared very cluttered and small. In addition, subjects commented that the world was difficult to navigate in because one was always "bumping into" objects. When teleported into the other worlds with 50 and 90 degree GFOVs, subjects commented on having a higher comfort level because the rooms appeared less cluttered and larger as compared to the 10 degree GFOV world.

As described in the methodology of the current experiment, there were two presence related questions which were repeated across all three comparative studies. In addition, identical display parameters were repeated across all three comparative studies. That is, a virtual world with a 50 degree GFOV, headtracking, and a stereoscopic display format was repeated across all three studies. It was of interest to determine if subjects responded consistently across all comparative studies with respect to the presence related questions and for the same virtual world display parameters. A Friedman two-way ANOVA on ranks indicated that there was no significant difference in the subjects' rating of their "level of presence" of the repeated condition across the three comparative studies of the current experiment ($x^2 = .04$, df = 2, p > .97). The average responses for study 1, 2, and 3 were 63.58 (std = 19.08), 63.33 (std = 17.27), and 64.25 (std = 15.94) respectively. A Friedman two-way ANOVA on ranks indicated that there was no significant difference in the subjects "sense of presence" of the repeated condition across the three comparative studies of the current experiment ($x^2 = 0.12$, df = 2, p > 0.93). The average responses for study 1, 2, and 3 were 2.33 (std = 0.78), 2.42 (std = 1.00), and 2.42 (std = 1.00) respectively.

## 2.3 Discussion

The results indicated that one's feeling of presence in a virtual environment may be indicative of the perceived realism of that virtual environment. Realism, however, does not necessarily imply the "visual-realism" of the virtual objects located in the virtual environments per say, but rather the realism with which the user interacts with the environment, or, the human-computer interaction (see [3]). This aspect of presence is compatible with Zeltzer's [4] discussion of presence, in which he defined presence in terms of the degree to which the input and output parameters of the human machine interaction are matched. An example of this can be clearly seen in the second comparative study where headtracking versus non-headtracking displays were used. The addition of headtracking to the virtual environment not only provided subjects with interactive feedback as they moved their head and upper body, but also increased the subject's enjoyment level and tendency to "play" more with the virtual environment (such as standing on their chair or crouching on the floor to look above and below virtual objects).

As noted, the addition of binocular disparity to the virtual world display increased the level of presence in the virtual world. While the addition of binocular disparity increased the level of realism of the virtual world's apparent depth and volume as well as one's sense of being able to reach into the

**Table 4. Results of the effect of GFOV on presence and display fidelity.**

| Question Number | Significance Level* | Average Response |
|---|---|---|
| 1. Presence (1-100) | $x^2 = 18.38$, df = 2, $p < .000$<br>T = 6.45, $p < .0001$, N = 12†<br>T = 6.48, $p < .0001$, N = 12††<br>T = 0.69, $p > .50$, N = 12††† | 135.33 (std = 18.50) - 10° GFOV<br>64.25 (std = 15.94) - 50° GFOV<br>66.17 (std = 14.86) - 90° GFOV |
| 2. Presence (1-5)** | $x^2 = 18.00$, df = 2, $p < .0001$<br>T = -7.7, $p < .0001$, N = 12†<br>T = -9.94, $p < .0001$, N = 12††<br>T = 0, $p > 1.0$, N = 12††† | 3.92 (std = 0.79) - 10° GFOV<br>2.42 (std = 1.00) - 50° GFOV<br>2.42 (std = 0.67) - 90° GFOV |
| 3. Realism ** | $x^2 = 11.54$, df = 2, $p < .003$ | 3.67 (std = 0.49) - 10° GFOV<br>2.33 (std = 0.89) - 50° GFOV<br>2.50 (std = 0.67) - 90° GFOV |
| 4. Compression / Magnification** | $x^2 = 1.04$, df = 2, $p > .05$ | 2.83 (std = 1.47) - 10° GFOV<br>3.08 (std = 0.51) - 50° GFOV<br>2.67 (std = 0.49) - 90° GFOV |
| 5. View** | $x^2 = 0.29$, df = 2, $p > 0.05$ | 2.75 (std = 1.42) - 10° GFOV<br>3.33 (std = 0.65) - 50° GFOV<br>3.50 (std = 1.00) - 90° GFOV |
| 6. Proportionally Correct** | $x^2 = 9.12$, df = 2, $p < .01$<br>T = -4.71, $p < .0006$, N = 12†<br>T = -4.02, $p < .002$, N = 12††<br>T = -0.27, $p > .80$, N = 12††† | 3.83 (std = 1.03) - 10° GFOV<br>2.25 (std = 0.75) - 50° GFOV<br>2.17 (std = 0.72) - 90° GFOV |

*Friedman two-way ANOVA
** where 1 = very much so and 5 = not at all.
† Wilcoxon matched-pairs signed ranks test comparing 10° and 50° GFOVs
†† Wilcoxon matched-pairs signed ranks test comparing 10° and 90° GFOVs
††† Wilcoxon matched-pairs signed ranks test comparing 50° and 90° GFOVs

environment, it did not, however, contribute to a subject's overall sense of realism of the environment. Subjects commented that the addition of binocular disparity detracted from their sense of realism of the environment due to the ghosting effect of the left and right eye stimulus images thereby making the image "less crisp" as compared to the monoscopic display. These results are compatible with the findings of [5] who indicated that ghosting resulted in head coupled displays without stereo being preferred somewhat more often than head coupled displays with stereo.

As discussed, the results indicated that presence does not necessarily imply the "visual-realism" of the virtual objects located in the virtual environments per say, but rather the realism with which the user interacts with the environment. In other words, even though subjects did not rate the stereoscopic display as appearing more realistic, they did rate it as being more realistic in conveying depth/volume spatial information and in increasing their sense of being able to interact with the environment (i.e. "reach in and grasp an object"). The results indicate that future studies need to be conducted which examine: 1) presence as a function of the tradeoff between the field of view and spatial resolution which exists for current virtual environment equipment [6][7], and 2) presence as a function of the variables which define stereoscopic displays such as the location of the zero plane of parallax and the amount of lateral disparity between images.

It was also shown that the computer graphics field of view with which the virtual world was designed had a great effect on the subject's sense of presence in that virtual environment. While the 10 degree GFOV resulted in a lesser sense of presence, lesser realism of the virtual world, and lesser "correctness of proportionality" of the virtual objects as compared to the 50 and 90 degree GFOVs, no significant differences were found between the three GFOVs for "perceived object

compression." When considering the results of the 10 degree versus 50 and 90 degree GFOVs, one might question if there is a strong engineering or economic reason for even considering the use of small GFOVs in display design. Since changing the GFOV of an image is equivalent to zooming in or out of a scene, it is believed that many practical applications in both telemedicine and teleoperations could benefit from such a design feature. For example, telemedicine would certainly benefit from a system which would allow for the ability to navigate and search internal cavities in a wide angle, large GFOV mode, then operate in a small GFOV, zoom mode. In addition, subjects found that regardless of the computer graphics GFOV, virtual objects appeared consistently slightly more compressed than real world objects. This result is consistent with previous studies which have shown that images viewed using lenses must be magnified to appear at their objectively correct distance [8].

Furthermore, the verbal feedback received from the majority of the subjects were of frustration and lack of comfort when navigating throughout the 10 degree GFOV virtual world as compared to the 50 and 90 degree GFOV rooms. This feedback seems to indicate the importance of one's sense of comfort, and/or frustration level, in determining one's sense of presence in virtual environments. As with the concept "enjoyment" level described above, the results indicate that future studies should examine the relationship between one's sense of presence and one's comfort and frustration levels when interacting with the virtual environment.

## 3. Experiment 2: Presence as a Function of Auditory Cues

### 3.1 Method

Sixteen volunteer subjects were used for this study. The group consisted of university students with a mean age of 29.9 years and included 14 male and 2 female students. All subjects reported having normal or corrected-to-normal visual acuity in addition to normal binaural hearing. The majority of the subjects consisted of naive virtual world users. However, four subjects had previously participated in presence related studies using computer-generated virtual environments.

The experiment consisted of two 1 x 2 within subjects factorial designs, each design being a comparative study. The independent variable for the first comparative study was a no-sound display versus a spatialized-sound display. For the second study, the independent variable was a spatialized sound display versus non-spatialized sound display. The virtual world design, task, and apparatus were similar to those described for the first experiment. Sound was added to the display using a Crystal River Engineering Beachtron audio spatializer card installed in a 386 PC. Beachtron default coefficient values were used for the HRTF, atmospheric absorption, as well as the spread rolloff. So that sound sources would appear to be omni-directional, a uniform radiation pattern was used. Due to hardware limitations, only two sound sources were displayed in the virtual world and included: (1) a live, continuous radio broadcast of progressive light rock music, and (2) a discrete recording of a monetary exchange with a soda machine (insertion of coins, item selection, and the delivery of a can of soda) repeated at 10 second intervals. The radio signal was delivered via a Realistic receiver/amplifier. The soda machine sounds were obtained using an Ensoniq digital sound sampler. The sound sources were combined in a 12 channel mixer and delivered to the Beachtron for spatialization. The object coordinates relative to the subject's location in the virtual room were transmitted from the SGI computer to the 386 PC via a RS-232 serial communications connection at a rate of 9600 baud. Sound output from the Beachtron was presented to the subject through Yamaha YH-1 orthodynamic headphones.

The questionnaire used for the first comparative study consisted of the first three questions described in experiment 1 while the questionnaire for the second comparative study consisted of the same first three questions in addition to two additional questions relating to spatialized sound (Table 5). In addition, the questionnaires used in this second experiment were administered in the same fashion as described for the first experiment.

### 3.2 Results

The results indicated that there was a significant difference in reported levels of presence of the virtual environment when a spatialized sound was added to a stereoscopic display (Table 6). However, the addition of spatialized sound did not significantly increase the overall realism of the virtual environment.

**Table 5. Summary of each question used in the current study by category.**

| |
|---|
| Presence |
| (1) If your level of presence in the real world is "100", and your level of presence is "1" if you have no presence, rate your level of presence in this virtual world. |
| (2) How strong was your sense of presence, "being there", in the virtual environment? |
| (3) How realistic did the virtual world appear to you? |
| Spatialized Sound |
| (4) How realistically did the sound sources interact with you as you navigated throughout the virtual world? |
| (5) To what degree did the sound sources in the virtual room appear to come from specific locations? |

**Table 6. Results of the effect of the addition of spatialized sound to a stereoscopic display on presence.**

| Question Number | Significance Level* | Average Response No Sound | Average Response Spatialized Sound |
|---|---|---|---|
| 1. Presence (1-100) | $T = 2.29, p < .04, N = 16$ | 45.45 (std = 19.42) | 56.09 (std = 21.00) |
| 2. Presence (1-5)** | $T = -2.33, p < .04, N = 16$ | 3.45 (std = 0.82) | 2.73 (std = 0.90) |
| 3. Realism ** | $T = -1.4, p > .2, N = 16$ | 3.36 (std = 0.67) | 2.91 (std = 0.83) |

*Wilcoxon matched-pairs signed-ranks ** where 1 = very much so and 5 = not at all.

**Table 7. Results of the effect of the addition of spatialized and non spatialized sound to a stereoscopic display on presence.**

| Question Number | Significance Level* | Average Response Non Spatialized | Average Response Spatialized Sound |
|---|---|---|---|
| 1. Presence (1-100) | $T = 2.7, p < .02, N = 16$ | 50 (std = 21.10) | 60 (std = 18.84) |
| 2. Presence (1-5)** | $T = -1.8, p < .08, N = 16$ | 3.18 (std = 0.87) | 2.64 (std = 0.81) |
| 3. Realism ** | $T = -1.6, p > .13, N = 16$ | 3.36 (std = 0.67) | 2.91 (std = 0.70) |
| 4. Interaction** | $T = -3.5, p < .006, N = 11$ | 4.00 (std = 0.89) | 2.64 (std = 0.92) |
| 5. Emanation** | $T = -5.2, p < .0004, N = 11$ | 4.36 (std = 0.81) | 2.00 (std = 1.00) |

* Wilcoxon matched-pairs signed-ranks ** where 1 = very much so and 5 = not at all.

In addition, the results indicated that there was a significant difference in reported levels of presence of the virtual environment when a spatialized sound versus non-spatialized sound was added to a stereoscopic display. However, the addition of spatialized sound did not significantly increase the overall realism of the virtual environment (Table 7).

As described above, identical experimental conditions were repeated across the two comparative studies. Recall that a 50 degree geometric field of view, spatialized sound, and a stereoscopic display format and was repeated across both comparative studies. It was of interest to determine if subjects responded consistently in rating their level of presence across both comparative studies given the same virtual world display parameters. The results indicated that there was no significant difference in the reported levels of presence between the two studies ($T = -1.68, p > .11, N = 16$ for the "1 to 100" rating of presence and $T = 0.70, p > .50, N = 16$ for the "1 to 5" rating of presence). The average responses for the "1 to 100" rating of presence for study 1 and 2 were 56.09 (std = 21.00) and 60.00 (std = 18.84) respectively. The average responses for the "1 to 5" rating of presence for study 1 and 2 were 2.73 (std = 0.90) and 2.64 (std = 0.81) respectively.

### 3.3 Discussion

The results indicated that the addition of spatialized sound to a stereoscopic display, with or without non-spatialized sound, significantly increased ones sense of presence in the virtual environment. In addition, spatialized sound significantly increased the fidelity of the sound sources' apparent interaction with the subject as well as one's sense that sounds emanated from specific locations. However, the addition of spatialized sound did not increase the overall realism of that environment. These results are surprising as one would expect that as new sensory channels of information are added to a display medium, one would associate with that display medium a greater sense of overall realism. This would seem to be especially true in light of the fact that subjects: 1) associated a greater sense of presence with the display medium having both visual and auditory feedback and, in the second study, 2) perceived a greater sense of display fidelity of the sound sources' apparent interaction with the subject, as well as 3) augmented one's sense that sounds emanated from specific locations. The results suggest that, while presence and the fidelity of the interaction of the acoustic images can be improved through the addition of spatialized sound, the overall reported "realism" of a virtual environment may be influenced by other factors.

One such factor might be that term "realism" has some additional semantic load which implies the "visual-realism" of the environment in the user's mind. As a result, if the user does not perceive a change in the visual scene, then they may not perceive a change in the overall realism of the environment even though they have reported higher levels of presence and interactive fidelity for that environment. When developing subjective questionnaires, it may be a consideration to use terminology that does not include such words as "realism", "realistic", and "real" so as to avoid semantic loading of the questions. In the current experiment, however, the majority of the subjects used in the study were novice computer users unfamiliar with the technical jargon used by the research community. Therefore, the wording of the questions were purposefully simplistic and general in nature and included

variations of the term "realism". In addition, responses to the questions were left entirely to the interpretation of the users.

A second factor contributing to the results of the current experiment might also stem from the notion that subjects associate degrees of realism predominantly with changes in the visual information channel over other sensory information channels. For example, in the first experiment of this paper, subjects were ask to rate presence and realism as a function of visual cues such as geometric field of view, stereopsis, and headtracking. The results indicated changes in both presence and realism as these visual cues were varied or added to the display; subjects reported higher levels of presence and realism when headtracking was added to the display as well as when the geometric fields of view approximating the actual field of view were used to design the display. In the current study however, changes in auditory cues did effect presence but did not effect the perceived realism of that environment, be it in the addition of spatialized sound over no sound (first comparative study) or in the addition of spatialized sound over non-spatialized sound (second comparative study). In the first experiment of the current paper, it was speculated that perceived realism does not necessarily imply the "visual-realism" of the virtual objects located in the virtual environments per say, but rather the realism with which the environment interacts with the user, or, the human-computer interaction. Perhaps "presence" is associated with the realism with which the environment interacts with the user, regardless of the information channel used while perceived realism is perhaps more closely associated to the realism with which the user interacts with the environment in the visual channel and/or the visual fidelity of the image.

In addition to determining a subject's sense of presence and perceived realism, it was of interest to examine the magnitude in increase, or decrease, of presence across the comparative studies. For example, the magnitude of increase in presence from no-sound to spatialized sound (10.64 units) was approximately the same as the increase in magnitude of presence from non-spatialized sound to spatialized sound (10.00 units). One might expect that the increase in magnitude of presence would be much greater for the first study since a new channel of information was provided to the user (the auditory channel) over the existing visual channel of information while the second study "only" introduced additional auditory cues, i.e. pinnae and binaural cues, to the existing auditory and visual channels of information. These results may suggest that, when new channels of sensory information are provided to the user, unless that additional channel can provide the user with sufficient cues to externalization, that channel may be redundant in contributing to the user's sense of presence in that environment. Obviously, the contribution of the additional channel of information and the cues it provides is highly task specific. For example, if the purpose of the auditory channel is to provide the user with a warning signal or verbal information, externalization cues may not be necessary in providing an increase in presence.

There are certain aspects of the current study which may have contributed to lower significance levels of reported presence and realism than was expected across the two comparative studies. First, generic HRTFs were used in the current study. Studies have consistently show that the use of generic HRTFs over individualized HRTFs can result in increased front-back reversals, decreased performance in spatial localization, as decreased externalization of sound sources [9]. Second, reverberation was not incorporated into the model. Reverberation is one of the main cues contributing to the externalization of acoustic images [10][11][12[13]. Third, headtracking, per say, was not incorporated into the display. Acoustic image locations were simulated as a function of joystick operations and not head movements. In other words, subjects perceived changes in both visual and acoustic image locations when navigating about the virtual environment but not when moving their head. Since a chin rest was not used in the current study, it was possible for subjects to rotate their head position independent of the joystick thereby giving the observer conflicting visual and auditory feedback of the associated object location. [14] theorizes that the externalization of a sound source is maximized when head movements occur and when binaural processing of a stimulus is altered in a natural way as a function of these head movements. [14] further stipulate that it should be possible to weaken or create ambiguous externalized images by varying the head position and binaural stimulus in an unnatural way. It is possible that subjects of the current experiment experienced ambiguous externalization and therefore a decreasing level of presence and perceived realism of the virtual environment.

## 4. Conclusions

Subjects' reported physical, verbal, and written responses to variations of visual and auditory display parameters described in the current paper suggest that future studies investigating presence in virtual environments should take into consideration the following elements. First, future presence questionnaires should explore the link between reported levels of presence and enjoyment, comfort, as well as frustration levels while operating in virtual environments. Second, specific tasks should be included as part of the experiment so as to encourage as much interaction as possible between the subject and the virtual world. Specifically, it is believed that subjects can be further "pulled into" a virtual environment by giving the subject a task to achieve while in the virtual environment which would exploit the benefits of the parameters used to design the display and, in the process, result in additional cognitive resources being allocated to the virtual world as compared to the real world environment. Third, the results indicate that future studies on auditory virtual environ-

ments should investigate the relationships between externalization and presence. For example, one might consider subjective questionnaires which includes a subjective assessment of the "quality" of the externalized images in addition to the localization of those images. In addition, cues that contribute to externalization can be varied and tested as a function of both task performance and subjective measures of presence and perceived realism. Finally, future questionnaires evaluating presence within virtual environments should focus on the interactivity of the input devices used to manipulate virtual objects, and system features such as delays in update rates, and sensor delays in response to human movements, on presence.

The results of the two experiments reported here allow us to comment on the reliability of the questions used in the current study to access presence given a navigation task. Test reliability refers to the consistency of scores obtained by the same persons when reexamined with the same test on different occasions, or with different sets of equivalent items, or under other variable examining conditions (Anastasi, 1982). Quite importantly, the two differently worded presence questions produced repeatable results when comparing the same display variables. Equally important was that subjects were quite consistent when answering the same question across different studies using similar virtual environments. This was shown by the consistency of the responses for the identical display parameters used across the three comparative studies of the first experiment as well as the across the two comparative studies for the second experiment. Additional studies from our laboratory using different subjects, virtual environments, and tasks will allow us to access the reliability and validity of the questionnaire for these variables.

## Acknowledgments

This work was supported in part by the Air Force Office of Scientific Research (contract # 92-NL-225 and INST PROP NO:78216), and the National Science Foundation (DMC-8857851, CDA-8806866). We thank Paul Schwartz for developing the in-house software program "Precept", which was used to design the virtual worlds as well as Eric Danas and Bob Futamura for the auditory interface development.

## 6. References

[1] Sheridan, T. B., Musings on Telepresence and Virtual Presence, *Presence: Teleoperators and Virtual Environments, Vol. 1,(1)*, 120-125, 1992.

[2] Shepard, T., Romney, A. K., & Nerlove, S. B. (1972). *Multidimensional Scaling: Volume I*, Theory, New York, Seminar Press.

[3] Hirose, M., Myoi, T., Liu, A., and Stark, L., Object Manipulation in Virtual Environment, *6th Symposium on Human Interface*, Oct. 24-26, Tokyo, 571-576, 1990.

[4] Zeltzer, D,. Autonomy, Interaction, and Presence, *Presence: Teleoperators and Virtual Environments, Vol. 1(1)*, 127-132, 1992.

[5] Ware, C., Arthur, K., and Booth, K. S., Fish Tank Virtual Reality, *INTERCHI*, 37-42, 1993.

[6] McKenna, M. D., and Zeltzer, D., Three Dimensional Visual Display Systems for Virtual Environments, *Presence: Teleoperators and Virtual Environments, Vol. 1(4)*, 421-458, 1992.

[7] Barfield, W., Hendrix, C., Bjorneseth, O., Kaczmarek, K., and Lotens, W., Comparison of Human Sensory Capabilities with Technical Specifications of Virtual Environment Equipment, *In press: Presence Teleoperators and Virtual Environments*, 1995.

[8] Roscoe, S. N., Judgments of Size and Distance with Imaging Displays, *Human Factors, Vol. 26(6)*, 617-629, 1984.

[9] Wenzel, E. M., Wightman, F. L., Kistler, D. J., & Foster, S. H., Acoustic origins of individual differences in sound localization behavior, *Journal of the Acoustic Society of America, Vol. 84*, S79, 1988.

[10] Loomis, J. M., Herbert, C., & Cicinelli, J. G., Active localization of virtual sounds. *Journal of the Acoustical Society of America, Vol. 88*, 1757-1764, 1990.

[11] Begault, D. R., Perceptual effects of synthetic reverberation on 3-D audio systems. *91st Convention of the Audio Engineering Society*, Reprint 3212 (W-6), 1991

[12] Plenge, G., On the difference between localization and lateralization. *Journal of the Acoustical Society of America, Vol. 56*, 944-951, 1974.

[13] Wenzel, E. M., Localization in virtual acoustic displays, *Presence: Teleoperators and Virtual Environments, Vol. 1(1)*, 80-107, 1992.

[14] Durlach, N. I., Rigopulos, A., Pang, X. D., Woods, W. S., Kulkarni, A., Colburn, H. S., and Wenzel, E. M., On the externalization of auditory images, *Presence: Teleoperators and Virtual Environments, Vol. 1(2)*, 251-257, 1992.

# Tools: HMDs, Head Tracking, and TeleSurgery

# Design and Applications of a High-Resolution Insert Head-Mounted-Display

Akitoshi Yoshida

Lehrstuhl für Informatik V
Universität Mannheim
68131 Mannheim, GERMANY
+49-621-292-5707
yoshida@mp-sun1.informatik.uni-mannheim.de

Jannick P. Rolland

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599, USA
+1-919-962-1901
rolland@cs.unc.edu

John H. Reif

Department of Computer Science
Duke University
Durham, NC 27708, USA
+1-919-660-6568
reif@cs.duke.edu

**Abstract**

*In this paper, a new Head Mounted Display (HMD) that provides a large field of view with a high-resolution insert is proposed and designed. Previously, this type of HMD has been designed using mechanical or sequential scanning devices, which are bulky and expensive. The proposed High-Resolution Insert HMD (HRI-HMD) is innovative and it uses only electronic devices that can be easily integrated with the optical components. The potential benefit of the HRI-HMD comes not only from its improved visual quality via a high-resolution insert but also from its increased human-computer interaction capability via eye tracking. The design principles and envisioned applications of the HRI-HMD are described, and the feasibility of the HRI-HMD is demonstrated by designing a prototype model.*

## 1. Introduction

The field of Virtual Reality (VR) has recently received considerable attention, due to the potential to create unique capabilities for human-computer interaction [13, 15]. Such advanced interaction can include: interactive control and diagnostics systems, educational and training systems, teleoperation systems, and entertainment systems [5, 7, 11, 12, 17, 22]. For these applications, HMDs are typically used to provide visual information to the user [6]; however, conventional HMDs usually do not utilize the full potential of VR technology. In particular, they do not provide enough resolution or field of view to give the user the realistic feeling of being immersed in the computer-simulated virtual environment, nor support integrated effective interaction capabilities combining head and eye tracking.

For some applications, the feeling of being immersed in the computer-simulated world is critical for properly performing the required task. To give the user the feeling of immersion, two features of the HMD must be met. First, the display must provide a field of view large enough to surround the entire view. Second, it must provide resolution high enough to render the fine detail of the image. Too narrow of a field of view causes the user to see the frame of the display and to have the sense of looking through a window. To remove this perhaps annoying window effect, the field of view must be at least 80 degrees [19, 25]. Too low of a resolution causes the user to see the individual pixels or raster scans of the display device and fine details of the image are lost. To match human visual acuity, the pixel size must be about 1 arc minute [8]. However, the human retina does not provide uniform visual acuity [21, 24]. The high visual acuity is only available at the fovea, a small area of about 5° in angular extent at the center of the retina. The visual acuity degrades rapidly as the distance from the fovea increases; at an angular distance of 5° from the center of the fovea, it is about a quarter of the highest acuity, and at an angular distance of 15°, it becomes only one seventh [8]. Therefore, the resolution does not have to be 1 arc minute over a large field.

For a fixed number of pixels in a display, these two features are contradictory. A large field of view leads to low resolution, and high resolution leads to a small field of view. Consequently, most HMDs do not provide these two features adequately. To overcome this dilemma, HMDs that combine a low-resolution, large-field background image with a high-resolution, small-field insert image have been developed [3, 23]. Because of the property of the human visual system to have high visual acuity only over a narrow region around the fovea, a small area of high-resolution insert can be superposed on a large field of low-resolution image to virtually create a large field of view with high resolution. In this case, the position of the insert is dynamically controlled by the gaze point.

The approach taken by these systems is to use large high-resolution displays or light valves to generate the high-resolution insert and to use optics combined with a bundle of optical fibers to transport the images to the eyes. These systems provide significant improvements over ordinary displays and are considered the best displays



0-8186-7084-3/95 $04.00 

available, in spite of the fact that they are very heavy and extremely expensive.

Thus, currently available HMDs are either low-cost low-performance or high-cost high-performance models. Our approach is to develop an hybrid type, a low-cost high-performance insert HMD system that uses fully optoelectronic components. The use of fixed optoelectronic components allows the whole system to be fabricated with fewer alignment errors, to be immune to mechanical failure and, in general, to be more tolerant to vibrations.

The interaction capability currently integrated to HMDs is typically limited to the use of head tracking to measure the position and orientation of the user's head and to generate scenery from the user's perspective [10]. The user can navigate through the virtual world and interact with its objects by using three-dimensional manual input devices. For some situations that require very fast response time or difficult coordinated skills, the interaction capability supported by such manual input devices becomes inadequate. For those cases, eye movement can be used in conjunction with manual input devices to provide effective fast and flexible interaction methods.

The basic concept of the High-Resolution Insert HMD (HRI-HMD) described in this paper is to optically duplicate the insert image and to select one copy by blocking the other copies. The selected copy of the insert image is then optically superposed on the background image. The insert image traces the gaze point, thus the user sees the whole field at high resolution. The whole system uses fixed optical components and ordinary display devices. The availability of such a low-cost high-performance HMD will significantly increase the potential of many virtual reality applications. In addition to the apparent advantage of having a large-field, high-resolution image, the HRI-HMD provides effective interaction methods through eye tracking. Thus, combined with appropriate computer software, the whole system will become an Active Vision HMD (AV-HMD) system that gives the user the feeling of being immersed in the virtual environment and provides effective gaze-point-oriented interaction methods.

## 2. Applications

The primary advantage of the HRI-HMD is its large field of view with the existence of a high-resolution insert at the user's gaze point. The user can observe dynamic scenery over a large field at high resolution. Updates in the image do not have to occur simultaneously at high resolution. The portion of the image near the gaze point may be updated quickly at high resolution. However, other portions of the image may be updated less frequently or at lower resolution to reduce both the computational load and transmission bandwidth. The HRI-HMD can potentially improve the performance of real-time applications that require generation of complex computer graphics images, decompression of hierarchically compressed images, or transmission of remotely sensed images. Another use of the insert is to superpose different kinds of images at the gaze point. X-ray, ultrasound, or infrared heat images may be superposed over the regular visual spectrum images. The HRI-HMD with its built-in insert capability becomes advantageous for those applications.

The additional advantage is its increased interaction capability. The use of eye tracking is not limited to finding the gaze point for positioning the insert. The eye can respond to stimulus much faster than the hands [16]. Thus, the eyes can be used for fast and effective input, selection, and control methods. Various interaction methods can be realized through the use of hand, body, and eye movements [2, 4, 20].

## 3. Sytem Description and Objective

The HRI-HMD described in this paper inserts a small area of the high-resolution image on a large field of the low-resolution image, as shown in Figure 1.

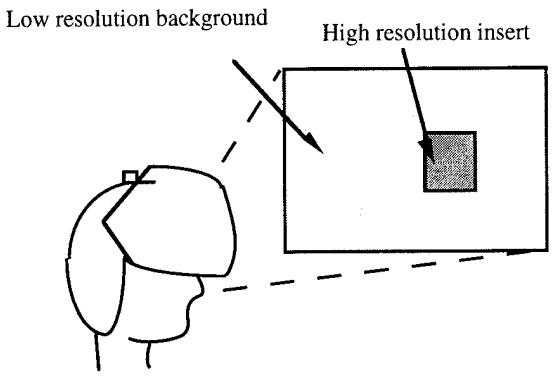



Figure 1. High-resolution insert HMD.

Using eye tracking information, the system dynamically places the high-resolution insert at the gaze point. Thus, in principle, the HRI-HMD visually provides the user with both high-resolution imagery and a large field of view. Several methods that may differ in accuracy can be used to track eye movements and the gaze points [1, 26]. The electro-oculography method detects the change of electric field in the tissue surrounding the eye and determines the orientation of the eye. This electric tracking method is the simplest method with moderate accuracy. The limbus tracking method detects the position of the limbus, which is the boundary between the iris and the sclera, and determines the gaze point. The pupil-corneal reflection method measures the corneal reflection with respect to the center of the pupil. These two optical tracking methods are much more accurate than the electric tracking method, however they require more complex hardware for

processing. In order to determine the gaze point for superposing the high-resolution insert, the accuracy of the electro-oculography may be sufficient. To implement some of the complex human-computer interaction methods using the gaze point, the more accurate limbus tracking or pupil-corneal methods may be necessary. In either case, once the gaze point is determined, the superposition of the high-resolution insert over the low-resolution background is carried out using liquid crystal devices and fixed optical components. This will result in a low cost, reliable system. The schematic diagram of our HMD is shown in Figure 2.

There are two displays: one for the background and the other for the insert. The image of the insert display is optically duplicated to fill the entire background display, and a liquid crystal device array is used to select one element of the array. This means that only one copy of the insert display image passes through the liquid crystal array, and all the other copies are blocked. The images of the insert display and the background display are then combined using a beam splitter.

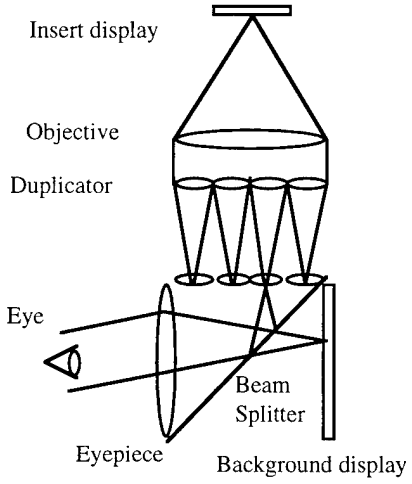



Figure 2. Schematic diagram of the HMD.

More specifically, the light from theinsert display is collimated by the objective. The collimated light is divided and focused by the duplicator to a set of identical images of the insert display, and the chief rays from these image points are set parallel to the optical axis. An array of liquid crystal shutters placed at the duplicator passes only one of these images and blocks the other images. These duplicated images are placed symmetrically to the background display with respect to the beam splitter so that the eye can see the superposed image through the eyepiece. The diagram represents the objective with a single lens, the duplicator with two arrays of lenses, and the eyepiece with another single lens.

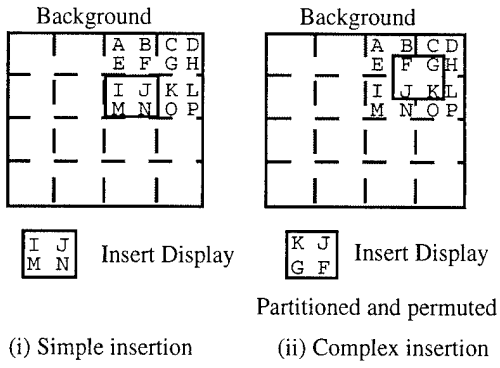



Figure 3. Superposition of the insert display.

The superposition of the insert and the background is depicted in Figure 3. In this figure, the shaded areas correspond to the background and the bright areas correspond to the insert. The character symbols represent contents of the image, and the dashed lines represent the cell boundary of the duplicated images. For a simple system, the insertion may be made at these discrete non-overlapping cell locations. In this case, the liquid crystal array may be placed anywhere inside the duplicator and blocks all the duplicated images except for one copy. When the insert image of "I J M N" is desired at a particular cell location, as shown in Figure 3-(i), this image can be directly displayed from the insert display. The duplicated images of "I J M N" fill every cell, and the copy of this image at that cell location exits the duplicator to be superposed with the background. For a complex system, the insertion may be made at continuous locations (up to the pixel level of the liquid crystal array). However, the size of the insert must be no larger than the size of a single duplicated image. In this case, the liquid crystal array must be placed near the duplicated image plane and blocks all the duplicated images except for some portions of up to four copies. When the insert image of "F G J K" is desired at a particular location, as shown in Figure 3-(ii), this image may be partitioned and permuted to "K J G F", and this transformed image can be displayed from the insert display. The duplicated images of "K J G F" fill every cell, and the portions of the four adjacent copies at that location which form the image of "F G J K" exit the duplicator to be superposed with the background.

The final goal of the design is to build a HMD that provides both a field of view large enough to surround the entire view and resolution high enough to match human visual acuity. The background must have a field of view of at least 80 degrees and resolution of 10 arc minutes, and the insert must have a field of view of at least 15 degrees and resolution of 1 arc minute. It must provide a stereoscopic view in color. The whole system must be integrated, folded, and packed in small volume so that the user can wear it without difficulty.

## 4. Prototype Design

This prototype is a scaled-down version of the final model. First, the design parameters are determined from the constraints. To clarify the principles, an ideal thin-lens model of the system with the determined parameters is presented. Finally, a real model that was designed using an optical design tool, Zemax from Focusoft [14], is described.

### 4.1. Basic Configuration

The main component of the high-resolution insert is an optoelectronic system for duplicating the insert image and bringing the image to the eye. The basic configuration of this component can be organized in three stages, as shown in Figure 4.

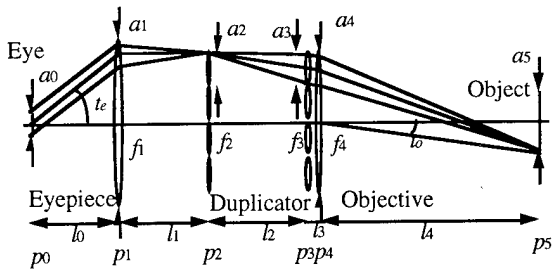



Figure 4. System parameters of the HMD.

The rightmost element is the insert display and the leftmost element is the eye. The first stage is the objective which collimates light from the display. The second stage is an array of telecentric systems which duplicate the display image from the collimated light and set the chief rays from the duplicated images parallel to the optical axis. We call this array of telecentric systems the duplicator. The third stage is the eyepiece which produces collimated light at the eye pupil from the duplicated image.

Miniature displays such as liquid crystal displays or viewfinders typically have 300 to 500 pixels in each direction. The largest field angle for an ordinary large field of view eyepiece is about 50 to 60 degrees. Thus, we arrive at the following basic design parameters:

| | Background | Insert |
|---|---|---|
| Field | 50° | 12.5° |
| Pixels | 400 x 400 | 400 x 400 |
| Resolution | 7.5 arc minutes<br>8 pixels/degree | 1.875 arc minutes<br>32 pixels/degree |

Table 1 : Basic design parameters

The distance between the eyepiece and its first focal plane where the duplicated images are located must be at least the physical size of the background display so that a beam splitter can be placed between the eyepiece and the duplicator to combine the insert and background images.

### 4.2. Ideal Thin-Lens Model

The ideal thin-lens model assumes ideal lenses of zero thickness. In other words, parallel beams to an ideal thin-lens with focal length $f$ converge to a focal point at distance $f$ from the lens. The height of the focal point from the optical axis is equal to $f \tan t$, where $t$ is the angle that the parallel beams make with the optical axis.

In this ideal thin-lens model, $p_0, p_1, \ldots, p_5$ represent planes along the optical system, as shown in Figure 4. More specifically, the eye pupil resides at $p_0$ and the display object at $p_5$. The eyepiece uses a lens with focal length $f_1$ placed at $p_1$. The intermediate object at $p_2$ is viewed by the eye placed at $p_0$. The telecentric system used in the duplicator uses two lenses with focal lengths $f_2$ and $f_3$ placed at $p_2$ and $p_3$, respectively. Collimated beams at $p_3$ are imaged at $p_2$. The objective uses a lens with focal length $f_4$ placed at $p_4$. Beams from the display object at $p_5$ are collimated at $p_4$. The distance between planes $p_i$ and $p_{i+1}$ and the diameter of the aperture at $p_i$ are denoted by $l_i$ and $a_i$, respectively. The number of duplicated images along the vertical or horizontal axes is denoted by $k$. The largest chief ray angle at the eye pupil is $t_e$ and the largest angle of the insert object subtended at the apex of the objective lens is $t_o$. These two parameters play essential roles in the ideal thin-lens model described.

The distances of both the eye and the duplicated images to the eyepiece lens must be equal to the focal length of the lens in order to obtain collimated beams at the eye pupil. Thus, we have

$$f_1 = l_0 = l_1 \quad . \tag{1}$$

Furthermore, the chief ray with the largest angle at the eye pupil is associated with the highest object point of the highest duplicated object. Thus, we have

$$l_0 \tan t_e = \frac{k\, a_2}{2} \quad . \tag{2}$$

For the telecentric system of the duplicator, collimated beams at $p_3$ must be focused at $p_2$, and beams passing the centers of the lenses at $p_3$ must exit parallel to the optical axis after passing $p_2$. Thus, we have

$$f_2 = f_3 = l_2 \quad . \tag{3}$$

Furthermore, the chief ray with the largest angle to the telecentric system at $p_3$ gives the highest image point for that element at $p_2$. Thus, we have

$$l_2 \tan t_o = \frac{a_2}{2} \quad . \qquad (4)$$

For the objective, the distance of the insert object to the lens must be equal to the focal length of the lens in order to obtain collimated beams from the object. Thus, we have

$$f_4 = l_4 \quad . \qquad (5)$$

Furthermore, the ray with the largest angle subtended by the insert object at the lens apex is associated with the highest object point. Thus, we have

$$l_4 \tan t_o = \frac{a_5}{2} \quad . \qquad (6)$$

Collimated beams at $p_4$ produce collimated beams at $p_0$. From this telescopic relationship, we have

$$\frac{a_0}{a_3} = \frac{l_1}{l_2} \quad . \qquad (7)$$

From the relationship among the lens apertures, we get

$$a_1 > l_0 \tan t_e + \frac{a_0}{2} \quad , \qquad (8)$$

$$a_4 > k\, a_2 \quad , \qquad (9)$$

and

$$a_3 = c\, a_2 \qquad \text{where } 0 < c \le 1 \, . \qquad (10)$$

The largest field angle of the eyepiece $t_e$ and the number of duplicated images $k$ can be calculated from the parameters given in Table 1. The exit pupil diameter $a_0$ must be at least 5 mm, but to allow some eye movements, a larger value is preferred. The insert display size $a_5$ may be about 25 mm for a typical miniature display. The ratio of the two lenses in the telecentric system, c, is ideally set close to 1.0 to maximize the brightness of the duplicated images, but physical constraints restrict it to be somewhat lower than this value. As the distance between the object and the objective lens decreases, the field angle of the objective increases, and so do field aberrations. Thus, a smaller field angle is preferred for minimizing aberration, but for compactness a larger angle is preferred. Thus, the largest field angle of the objective $t_o$ may be limited to 6°. In summary, the following parameter values are assumed for the prototype:

$a_0 = 8$ mm, $a_5 = 25$ mm, $t_e = 25°$, $t_o = 6°$, $k = 4$, $c = 0.8$.

Solving the above equations, we determine the other parameter values as follows.

Combining Equations (1), (6), (7), and (10), the expression for $f_1$ is

$$f_1 = l_0 = l_1 = \frac{a_0}{2\, c \tan t_o} \quad . \qquad (11)$$

From Equation (6), we have

$$l_4 = \frac{a_5}{2 \tan t_o} \quad . \qquad (12)$$

Combining Equations (2) and (11), $a_2$ is given by

$$a_2 = \frac{a_0 \tan t_e}{k\, c\, \tan t_o} \quad . \qquad (13)$$

Combining Equations (10) and (13), $a_3$ is given by

$$a_3 = \frac{a_0 \tan t_e}{k\, \tan t_o} \quad . \qquad (14)$$

Finally, combining Equations (4) and (13), yields

$$f_2 = f_3 = l_2 = \frac{a_0 \tan t_e}{2\, k\, c \tan^2 t_o} \quad . \qquad (15)$$

Table 2 lists the determined parameter values in millimeters:

| $i$ | $a_i$ | $f_i$ | $l_i$ |
|---|---|---|---|
| 0 | 8.000 | - | 47.572 |
| 1 | 52.366 | 47.572 | 47.572 |
| 2 | 11.092 | 52.765 | 52.765 |
| 3 | 8.873 | 52.765 | 1.000 |
| 4 | 44.368 | 118.930 | 118.930 |
| 5 | 25.000 | - | - |

Table 2 : Determined prototype parameters

The ideal thin-lens model layout at two different telecentric positions is shown in Figures 5-(i) and (ii).

### 4.3. Real Model

Since the purpose of designing a prototype is to show the feasibility of our approach, monochromatic light is assumed and our design is limited to the use of only spherical lenses of the same glass material (BK7). For more complete systems, chromatic aberration must be corrected by using different glass materials or more surfaces. Our design is further limited to the use of identical telecentrics in the duplicator.

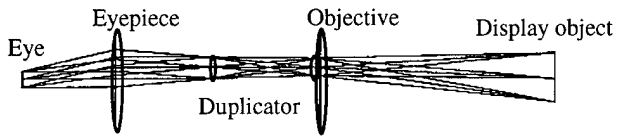



Figure 5-(i). Ideal thin-lens model layout at position 1.

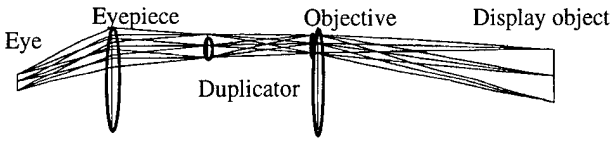



Figure 5-(ii). Ideal thin-lens model layout at position 2.

In this prototype design, the three stages are independently designed. The performance of the system may be improved by optimizing the system entirely by balancing aberrations from each stage.

The eyepiece uses four lenses and its layout is shown in Figure 6-(i). The object plane of the eyepiece, which symbolizes the duplicated images of the display formed by the objective and the duplicator, is the rightmost plane in Figure 6-(i). Light from this object is collimated and directed toward the eye pupil at the leftmost plane. The eye clearance is 15 mm. There is enough room for a beam splitter to be placed between the back lens surface and the focal plane. The performance of this eyepiece is shown in Figures 6-(ii) to (iv).

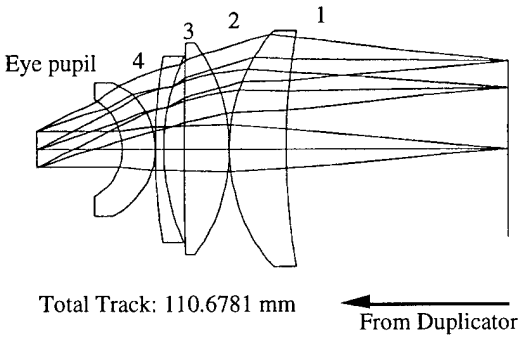



Figure 6-(i). Eyepiece layout.

The duplicator uses an array of telecentric systems. All the telecentric systems are identical and each uses four lenses. These lenses are imbedded in a square form. The layout of a single telecentric system is shown in Figure 7-(i). Collimated beams from the objective at the leftmost plane are focused at the rightmost plane. The rays passing the center of the first lens emerge parallel to the optical axis after passing the fourth lens. This guarantees that these rays, after passing the eyepiece, cross the optical axis at the eye pupil as shown in Figure 4. Thus, the first lens acts as an aperture and the fourth lens acts as a field stop. The performance of this telecentric system is shown in Figure 7-(ii).

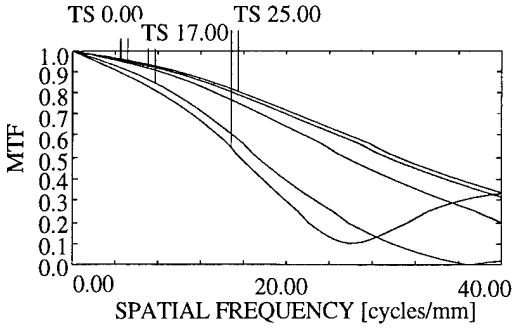
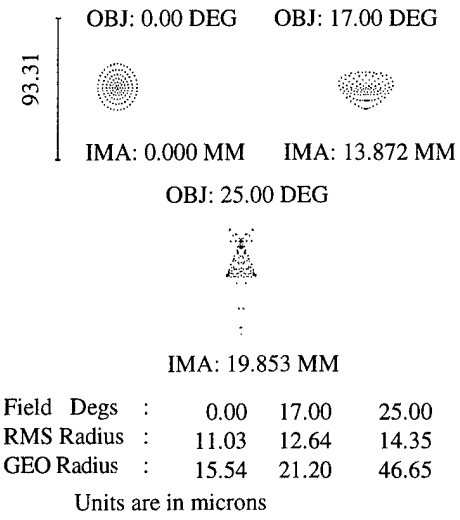



Figure 6-(ii) Eyepiece MTF.




| | | | | |
|---|---|---|---|---|
| Field Degs | : | 0.00 | 17.00 | 25.00 |
| RMS Radius | : | 11.03 | 12.64 | 14.35 |
| GEO Radius | : | 15.54 | 21.20 | 46.65 |

Units are in microns

Figure 6-(iii) Eyepiece spot diagram.

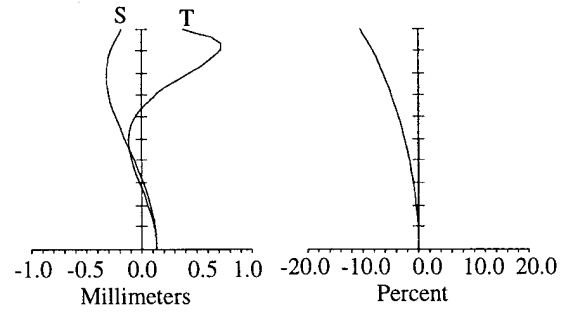



Figure 6-(iv) Eyepiece field curvature / distortion plot.

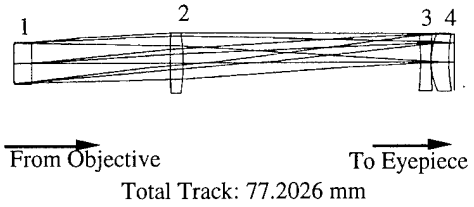



Figure 7-(i). Duplicator layout.

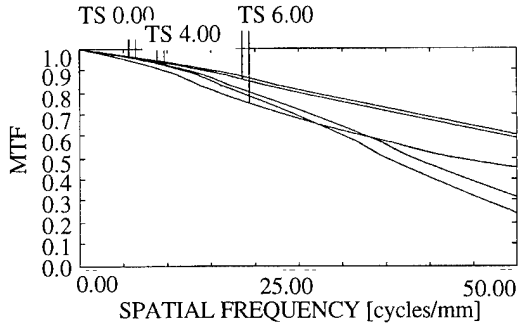



Figure 7-(ii). Duplicator MTF.

The objective uses four lenses and its layout is shown in Figure 8-(i).

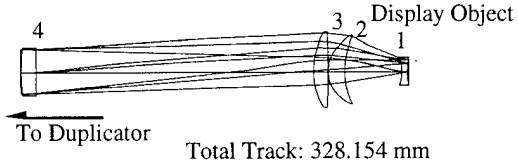



Figure 8-(i). Objective layout.

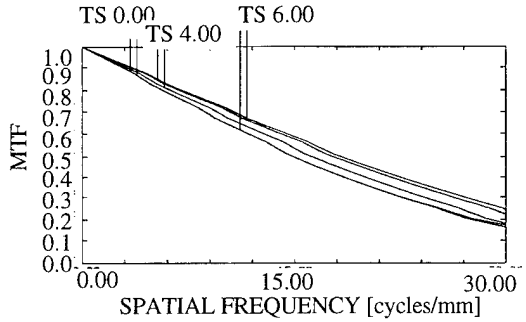



Figure 8-(ii). Objective MTF.

The display object is at the rightmost plane. Beams from this object are collimated at the leftmost plane. The performance of this objective is shown in Figure 8-(ii).

The layout of the entire system at two different telecentric positions is shown in Figures 9-(i) and 10-(i). For simplicity, the figures do not show the folding of the system nor combining the insert and the background. The folding and combining can be done between the eyepiece and the duplicator. Further folding can be made within the duplicator and the objective. The performance of the entire system at these two telecentric positions is shown in Figures 9-(ii) to (iv) and 10-(ii) to (iv).

Because of the symmetry of the system, the performance at positions 3 and 4 is equivalent to that of 1 and 2.

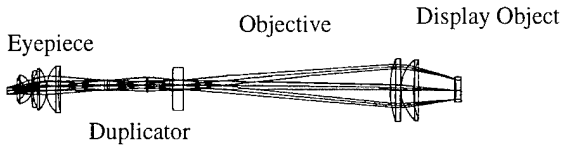



Figure 9-(i). Real model layout at position 1.

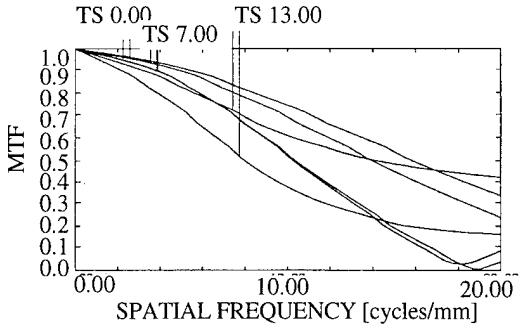



Figure 9-(ii). Real model MTF at position 1.

## 5. Discussion and Future Research

### 5.1. Prototype Performance

The insert image has 200 line-pairs in each direction. The size of this insert image at the insert display plane at $p_5$ is 25 mm, which results in a spatial frequency of 8 lp/mm. At the duplicated image plane at $p_2$, the size of this insert image is 11.092 mm, which results in a spatial frequency of 18 lp/mm. Each stage as well as the entire system are analyzed from their collimated object planes to their focused image planes. For the eyepiece, its focused image plane is at $p_2$, and thus the required spatial frequency is 18 lp/mm. For the telecentric system in the duplicator, the required spatial frequency is also 18 lp/mm, since its focused image plane is also at $p_2$. For the objective, its focused image plane is at $p_5$, and thus the required spatial frequency is 8 lp/mm. For the entire system, the required

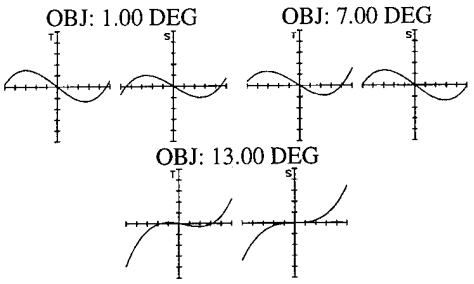
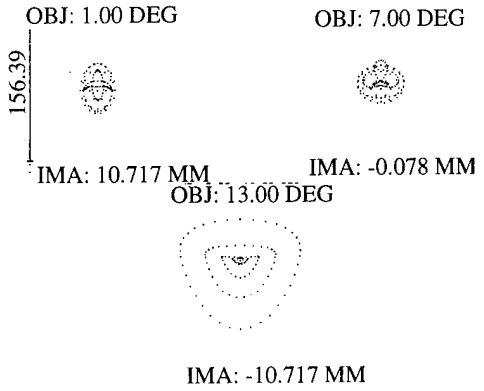



Figure 9-(iii) Real model rayfan plot at position 1.




| Field Degs: | 1.00 | 7.00 | 13.00 |
|---|---|---|---|
| RMS Radius | 17.62 | 19.37 | 40.44 |
| GEO Radius: | 29.75 | 30.92 | 78.19 |

Units are in microns

Figure 9-(iv) Real model spot diagram at position 1.

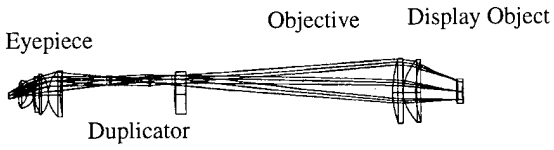



Figure 10-(i). Real model layout at position 2.

spatial frequency is also 8 lp/mm, since its focused image plane is also at $p_5$. If there is no balancing of aberrations from different stages, the MTF of the entire system is the product of the MTF of each stage. Thus, to get a MTF of 0.5 for the entire system at this required spatial frequency, the MTF of each stage at this spatial frequency must be at least 0.8. The MTFs of the duplicator and the objective

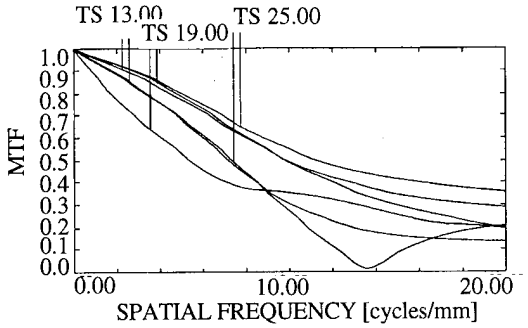



Figure 10-(ii). Real model MTF at position 2.

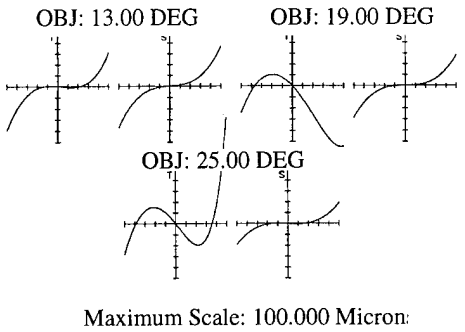
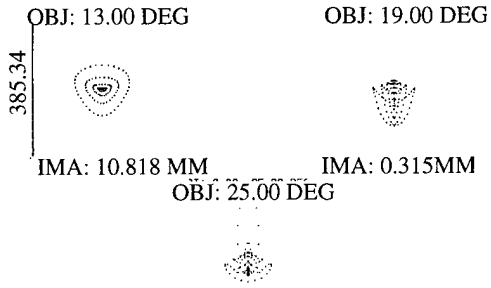



Figure 10-(iii). Real model rayfan plot at position 2.




| Field Degs: | 13.00 | 19.00 | 25.00 |
|---|---|---|---|
| RMS Radius: | 44.83 | 51.72 | 50.03 |
| GEO Radius: | 79.08 | 108.66 | 192.67 |

Units are in microns

Figure 10-(iv). Real model spot diagram at position 2.

show that the system is capable of resolving this limit. The eyepiece is limited by astigmatism which limits the performance of the entire system at large angles. For the entire system, the required spatial frequency is 8 lp/mm. The MTFs of the entire system show that they are capable of resolving the required spatial frequency.

The limitations of the entire system currently comes from mainly the coma, astigmatism, and field curvature of each stage. Between the objective and the duplicator, these aberrations can be balanced, since they form a point-to-point imaging system as a pair. The astigmatism and field curvature of the eyepiece may be reduced by using telecentric systems with different focal lengths or displacements to bend the object plane of the eyepiece. In our prototype, the duplicator uses lenses at the image plane (the object plane for the eyepiece) to align the principal rays parallel to the optical axis. This may introduce a practical problem, since any scratches or dusts on these lenses significantly degrade the image quality. To avoid this problem, these lenses may be placed away from the image plane as long as the collimated beams reach the pupil of the eye.

### 5.2. Future Systems

Our prototype design is the first step to a more complete HMD system that can be used in the described applications. Extensions to our prototype design include:

Use of a color display,
Use of a display with more pixels,
Use of non spherical surfaces or binary optics,
Integration with an eye tracker,
Fabrication and packaging,
System demonstration for the described applications.

To use a color display, chromatic aberration must be corrected using glasses with different refractive indices. The resolution and field of view can be increased by using a display with more pixels as long as the optical system supports its spatial frequency.

For a higher performance at the expense of more complex structures, non spherical lenses or binary optical elements may be used to optimize the system. Such complex surfaces are particularly suited for the eyepiece, since each small portion of the angular area can be independently optimized. The eyepiece also serves for the background, however the quality of the background image is not critical. Therefore, the eyepiece can be tuned for the insert image.

As mentioned earlier, the ordinary eyepieces have their field of view limited to about 60°. To obtain a larger field of view, eyepieces with intentionally distorted imaging properties may be used [18]. In this case, both the background and insert images for the HRI-HMD have to be predistorted so that the correct imagery can be viewed by the user.

Even if such an intentionally distorted eyepiece is assumed, in order to keep enough room between the eyepiece and its image plane for folding, a significant tradeoff between the image quality and the field of view may remain. In this case, the image quality of both the background and insert at a larger angle may be sacrificed for some extent to maintain the large field of view. A relatively high image quality may be retained for the insert by optimizing each angular section of the eyepiece.

All components including the eye tracker must be integrated and fabricated as an HMD.

Finally, the system performance for the described applications must be demonstrated. A set of software tools has to be developed to utilize the functionality of the HMD.

## 6. Conclusion

We introduced a new high-resolution insert HMD, named the High-Resolution Insert HMD (HRI-HMD), and designed its first prototype model. The HRI-HMD uses only optoelectronic devices and no mechanical devices. The apparent benefit of the HRI-HMD is its potential for providing a large-field, high-resolution image. The additional benefit is its potential for supporting various gaze point oriented interaction methods. We presented its principles by formalizing the system design parameters, and demonstrated its feasibility by presenting the design of a prototype system. We also described potential applications of the HRI-HMD systems.

## Acknowledgements

The authors thank Prof. Gerhard Schlimbach and Mr. Claus Neubronner at the FH Rheinland-Pfalz, Worms, Germany for their kind support by allowing the use of their equipment.

## References

1. Applied Science Laboratories, Eye tracking systems handbook, (Applied Science Laboratories, Waltham, MA, 1992).

2. R. A. Bolt, "Gaze-orchestrated dynamic windows," Computer Graphics, 15(3), 109-119 (1981).

3. D. Burbidge and P. M. Murray, "Hardware improvement to the helmet mounted projector on the visual display research tool (VDRT) at the Naval Training Systems Center," Proc. SPIE Vol. 1116, 52-60 (1989).

4. S. Bryson, "Interaction of objects in a virtual environment: a two-point paradigm," Proc. SPIE Vol. 1457, 180-187 (1991).

5. G. Burdea and P. Coiffet, Virtual Reality Technology, (Wiley & Sons, New york, NY, 1994).

6. J. C. Chung, M. R. Harris, F. P. Brooks, H. Fuchs, M.T. Kelley, J. Hughes, M. Ouh-young, C. Cheung, R. L. Holloway, and M. Pique, "Exploring virtual worlds with

head-mounted displays," Proc. SPIE Vol. 1083, 42-52 (1989).

7. R. E. Cole, C. Ikehara, and J. O. Merritt, "A low cost helmet-mounted camera/display system for field testing teleoperator tasks," Proc. SPIE Vol. 1669, 228-235 (1992).

8. H. Davson, Physiology of the eye, 5th ed. (Pergamon Press, New York, 1990).

9. S. Feiner, S. Nagy, and A. van Dam, "An experimental system for creating and presenting interactive graphical document," ACM Trans. Graphics 1(1), 59-77 (1982).

10. F. J. Ferrin, "Survey of helmet tracking technologies," Proc. SPIE Vol. 1456, 86-94 (1991).

11. S. S. Fisher, M. McGreevy, J. Humphries, and W. Robinett, "Virtual environment display system," ACM Workshop on Interactive 3D Graphics, Oct 23-24, Chapel Hill, North Carolina (1986).

12. S. S. Fisher, M. W. McGreevy, J. Humphries, and W. Robinett "Virtual interface environment for telepresence applications," in Proceedings of the ANS International Topical Meeting on Remote Systems and Robotics in Hostile Environments, J. D. Berger, ed. (1987).

13. S. S. Fisher, "Virtual interface environments," in The Art of Human-Computer Interface Design, B. Laurel ed. (Addison-Wesley, Menlo Park, CA, 1990), pp.423-438.

14. Focusoft, Zemax optical design program user's guide ver. 3.0, (Pleasanton, California 94566, 1994).

15. D. Foley, "Interfaces for Advanced Computing," Scientific American 257(4), 126-135 (1987).

16. H. Girolamo, "Notional helmet concepts: A survey of near-term and future technologies," US Army NATICK Technical Report NATIK/TR-91/017 (1991).

17. C. Herot, "Spatial management of data," ACM Trans. Database Systems 5(4), 493-514 (1980).

18. E. M. Howlett, "Wide angle orthostereo," Proc. SPIE Vol. 1256, 210-223 (1990).

19. E. M. Howlett, "High-resolution inserts in wide-angle head-mounted stereoscopic displays," Proc. SPIE Vol. 1669, 193-203 (1992).

20. R. H. Jacoby and S. R. Ellis, "Using virtual menus in a virtual environment," Proc. SPIE Vol. 1668, 39-47 (1992).

21. R. A. Moses, Adlers Physiology of the eye, The C.V. Mosby Company (1970).

22. D. Thalmann, Using virtual reality techniques in the animation process, in Virtual Reality Systems, R. A. Earnshaw, M. A. Gigante, and H. Jones ed. (Academic Press, Reading, MA, 1993).

23. M. L. Thomas, W. P. Siegmund, S. E. Antos, and R. N. Robinson, "Fiber optic development for use on the fiber optic helmet mounted display," Proc. SPIE Vol. 1116, 90-101 (1989).

24. G. Westheimer, The eye as an optical instrument, in Hanbook of Perception and Human Performance Vol I., 4 (Wiley-Interscience, New York, NY, 1986).

25. H. Yamaguchi, A. Tomono, and Y, Kobayashi, "Proposal for a large visual field display employing eye movement tracking," Proc. SPIE Vol. 1194, 13-20 (1989).

26. L. Young and D. Sheena, "Survey of eye movement recording methods," Behavior Research Methods, Instruments and Computers Vol. 7, 397-429 (1975).

# A Vision-Based Head Tracker for Fish Tank Virtual Reality – VR without Head Gear –

Jun Rekimoto
Sony Computer Science Laboratory Inc.
Takanawa Muse Building,
3-14-13, Higashi-gotanda, Shinagawa-ku,
Tokyo 141 Japan
E-mail: rekimoto@csl.sony.co.jp

## Abstract

*A practical and robust head-position tracking method using computer vision is presented. By combining two simple image processing techniques, this tracker can report the position of the user's head in real time. Whole image processing is performed by software running on normal mid-range workstations. This tracker can support desk top virtual reality (also referred to as "fish tank VR"), thereby enabling a user to use a wide range of 3D systems without having to put on any equipment. An experiment conducted by the author suggests this tracker can improve the human's ability in understanding complex 3D structures presented on the display.*

## 1 Introduction

One of the major drawbacks in virtual reality (VR) is its cumbersome devices. A typical VR system requires a user to wear goggles and a position tracker on the head for 3D immersion, and a DataGlove for gesture recognition. Although VR has great potential, such equipment prevents users from accessing its capability in normal situations. Aside that a head-mounted display (HMD) shields a user from the real world, these devices require time to put on and take off, thus making it impossible to quickly switch between *VR mode* and real life mode. The HMD's impact on human health is not yet clear, especially when it is used for long periods of time. It is still impractical and not yet acceptable to wear VR equipment in an office environment.

To overcome these limitations, another approach has emerged recently which uses a normal display screen (either monocular or binocular) coupled with a head tracker that dynamically updates a 3D projection matrix according to the viewer's head position [4, 1, 3]. Arthur, Ware and Booth coined the term "fish tank virtual reality" for this kind of system [1, 16]. With such systems, the user looks through the screen as if looking into a fish tank. Fish tank VR does not provide strong immersion, but is suitable for certain applications such as 3D-CAD or visualization systems because of its ease of use and ability to present high-quality images. For example, Liang's JDCAD system [7], which is a mechanical 3D-CAD system using a 6-degree-of-freedom (DOF) input device, employs a fish tank configuration instead of a normal VR environment.

However, most fish tank VR systems still require a device to track the user's head position. Arthur et al.'s system uses ADL-1, which is a mechanical position tracker and the user is connected to a mechanical rod. Deering's system uses an ultrasonic tracking device; a user must wear an ultrasonic transmitter on their head. As in immersion VR systems, these head trackers also limit the usability of the VR systems.

Due to progress in hardware and the recent boom in multimedia, many of today's workstations and personal computers include a video capturing unit as a standard input device. Using real-time video processing as a method for human-computer interaction is a natural idea, and has finally become practical. As Aukstakalnis and Blatner claimed in their book, vision-based position tracking "shows great promise for virtual reality systems because of its relative simplicity of use."(*Silicon Mirage* [2], page 36)

Although estimation of human position and orientation using video images under uncontrolled conditions is still only a research topic in computer vision, vision-based head tracking used only for fish tank VR is within reach of today's technology. There are two reasons for this: (1) We can assume the rough position of a user, because the user sits in front of the screen, and (2) we can omit estimation of orientation, because the user is looking at the screen most of the time. These assumptions make it easier to apply head tracking techniques based on image processing in actual 3D systems.

In this paper, I describe a vision-based head position tracker and a fish tank VR system as its application. With this system, a user does not need to wear any special gear. The vision system automatically tracks the position of the user's head while the user is sitting at the desk. A tracking system uses the simple image processing techniques of frame subtraction and template matching based on correlation. Even though image processing is performed by software, the system can achieve 15 frames per second (fps) on a compara-



0-8186-7084-3/95 $04.00 © 1995 IEEE

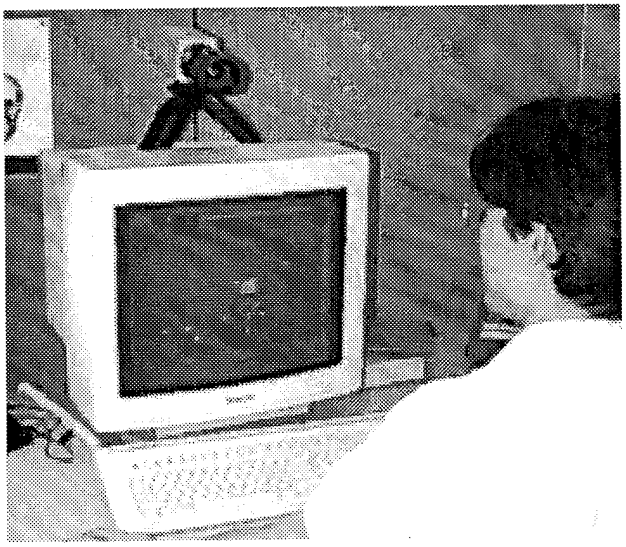


Figure 1: A snapshot of the system

tively slow workstation using a MIPS R3000 CPU.

## 2 Head Tracking Using Computer Vision

Figure 1 shows the system in use. A 3D scene is displayed on a screen monitor. A video camera on top of the screen captures a user's images and estimates its position in real time. The system updates the transformation matrix with respect to the user's head position, and updates 3D images according to it. This causes an illusion that the user is looking at a 3D object through the display screen. Motion parallax caused by head position movement enriches that illusion.

### 2.1 Head position estimation

Head position estimation takes two steps of image processing techniques as illustrated in Figure 2.

First, to detect the user's face area, a pre-stored background image is subtracted from a captured image, pixel by pixel (i.e., a pixel that exceeds a threshold is treated as the user's image). To increase robustness, we use distance in YUV space instead of intensity distance space. By using YUV thresholding, we can get a clear segmentation of the user's image under complex background images (Figure 3).

Second, the system searches for the center of the user's face by using template matching. A partial area of the user's face is stored as a template. Typically, the area between the left and the right eyebrows is used (the user can change the area at any time by clicking a mouse button on the face image). The system calculates correlation coefficients between the template and every area in the face image, and assumes the area with the highest score as the center.

Finally, using $(u, v)$ position on the captured image plane, the system estimates the users head position $(x, y, z)$ by using the following equation:

$$x = C_x - \frac{D}{F}u,$$

$$y = C_y + \frac{D}{F}v,$$

$$z = C_z - D.$$

Where $C_{x,y,z}$ is a camera position in the world coordinates, $F$ is the focal length of the camera, and $D$ is a distance between the user and the camera. Note that we assume the distance between the user and the display if fixed. It is not always true and the system performs incorrect perspective transformation when the user is too close to (or too far away from) the display. We will discuss this problem later.

### 2.2 Template matching with background elimination

Although only template matching seems to be sufficient for position estimation, we combine frame subtraction with template matching as a preprocessing step for the following reasons:

*Increasing performance.* We can simply omit the background area from the correlation calculation since it is obvious that this area does not include a pattern we are looking for. Doing this greatly improves the performance of position tracking. Figure 4 shows a correlation map for before and after background subtraction.

*Increasing robustness.* Background elimination also lowers the possibility of mismatching, because the search area is limited to the user's face and body images and the background area is excluded. Doing this increases the robustness of head tracking. Even when the user does not turn their head toward the camera correctly, or tilts their head, the template can still find the correct position. The reason is no other area is as good as that position in calculating correlation values. A simple silhouette-based technique (which uses the gravity center of the silhouette as a head position) for locating the head position would fail in such a case (Figure 5).

### 2.3 Off axis perspective projection

Extracted head positions are used to construct a transformation matrix that projects 3D objects onto a display screen.

Traditional perspective projection used in computer graphics assumes that a viewer's line of sight is perpendicular to the view plane (i.e., the screen), and that a view volume is a symmetric frustum. With a fish tank configuration, however, the viewer might look at the screen from a slanted position, thus making a view frustum asymmetric. In such a case, the system must generate an image that looks correct from the user's viewpoint, but might be skewed from a frontal position.

To implement such projections, the system uses a variant of transformation matrices that supports an asymmetric view frustum which is similar to those described in Deering's paper [4]. Using OpenGL [10] or

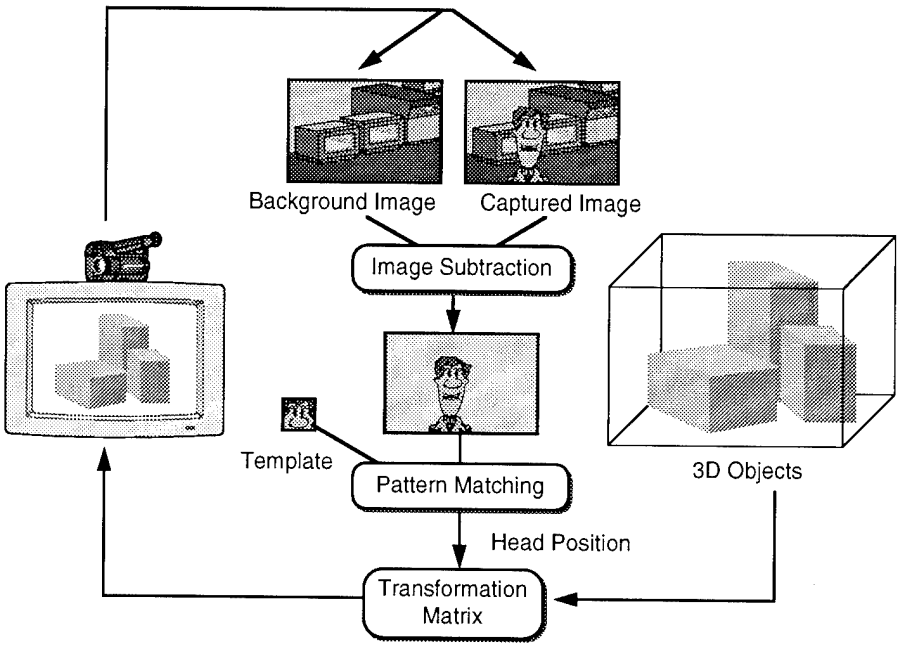
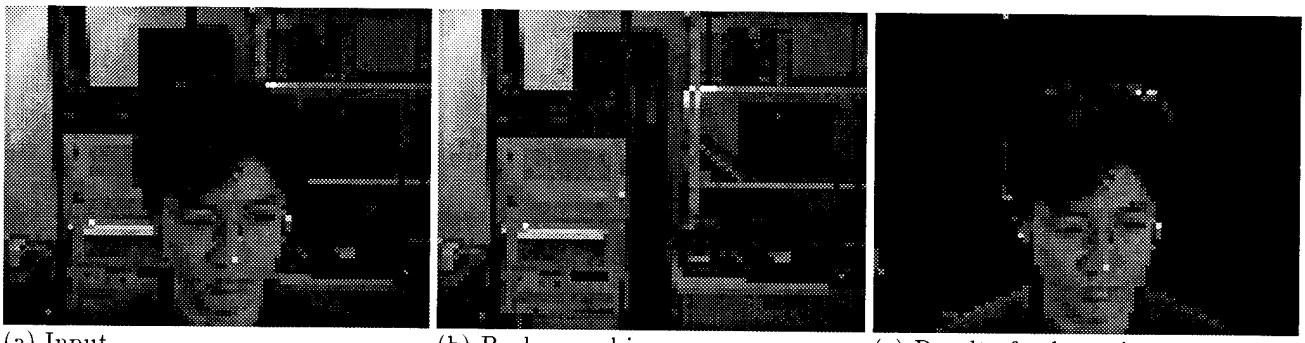
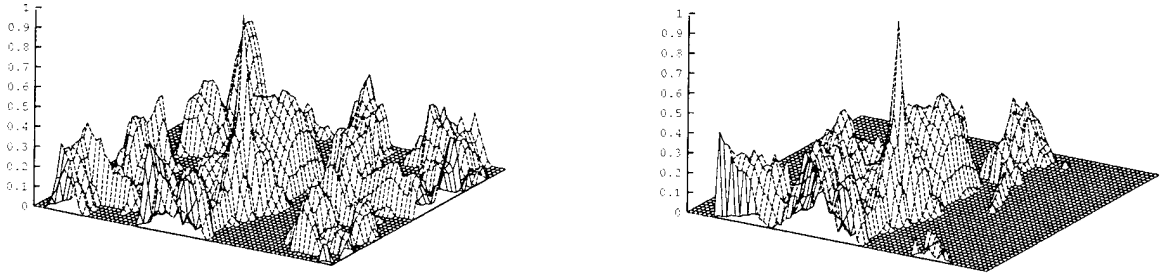



Figure 2: The system overview



(a) Input (b) Background image (c) Result of subtraction

Figure 3: Image Subtraction



(a) Captured image (b) After frame subtraction

Figure 4: Correlation Intensity values

(a) (b)

Figure 5: Comparison of two position tracking methods. A cross mark between eyes is based on the correlation method, and a mark below is based on the gravity center method. This is shown in (a). When the face is upright, the two methods give similar results. However, when the face is tilted (b), the gravity center method is less accurate than the correlation method.

GL [12], such perspective transformation is easily realized by calling the library functions **glFrustum** or **window**, respectively.

### 2.4 Implementation details and performance

The image processing is done entirely by software except for RGB to YUV conversion. The current system uses 160 × 120 pixel images for head tracking, and a 12 × 12 pixel image as a template. With a comparatively slow workstation (SGI IRIS 4D 320VGX using a MIPS R3000 CPU), the system can process at about 15 frames per seconds (fps) for incoming images. To increase performance, the system assumes that the head does not move too fast, and first searches the neighborhood area around the previous head position. When there is a point in the neighborhood area that exceeds the predefined correlation threshold, it is taken as the next head position. If the nearby search fails, the system switches its mode to global search. The current implementation uses a 32 × 32 pixel area around the previous position for the nearby search.

Although the current frame rate is not as fast as other position trackers such as magnetic or ultrasonic trackers, the user was able to experience a good illusion of motion parallax. Of course, since the processing rate depends on the speed of the CPU, the performance could reach the video frame rate (30 fps) by using faster CPUs which will be available within a few years.

## 3 Evaluation

To study how our optical head tracker helps a viewer's 3D perception skill, we conducted a task analysis which is originally designed by Sollenberger and Milgram [13]. Similar experiment was also achieved by Arthur, Booth and Ware [1], to evaluate their fish-tank virtual reality system.

### 3.1 The experiment

In this experiment, three 3D trees standing at the corners of an equilateral triangle are presented to a subject (Figure 6). A leaf of one of the trees is labeled by a small square mark and color. The subject's task is to detect which tree contains this leaf and give the answer via the keyboard. One session consists of 50 questions. The subject alternatively answers these questions with and without head tracking. The system generates the marked leaf and shapes of the trees randomly each time.

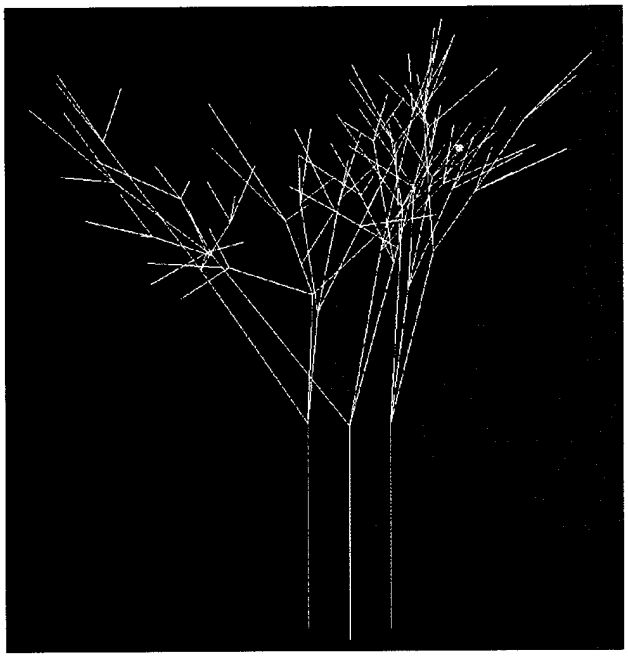


Figure 6: A snapshot of the tree test.

Six subjects participated in the experiment. All of them were computer scientists, but were not familiar with 3D computer graphics systems. Each subject had two sessions, answering 100 questions in total. No practice trials are given prior to the experiment. The second session consisted of exactly the same sequence of questions (marked edges and shapes of the trees), but switched between the tracking methods. The sub-

| Method | Response Time (sec) | errors (%) |
|---|---|---|
| With head tracking | 5.95 | 5.0 |
| Without head tracking | 3.50 | 21.3 |

Table 1: Experimental results

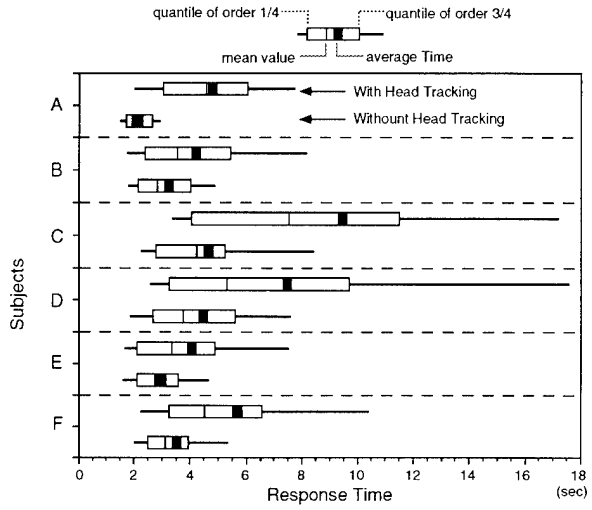



Figure 7: Response times for each subject

ject thus answered two times for each question, once with head tracking and once without head tracking. To make sure that subjects did not memorize the trial sequence, the sessions were at least a day apart.

### 3.2 The results of the experiment

The results of the experiment are listed in Table 1, and in Graphs 7 and 8. With head tracking, the subjects took a longer time to answer, but had lower error rates. There are significant differences at the 0.01 level of significance between the two methods for both (average times and error rates). This result is quite similar to Arthur et al.'s experiment (though they used a mechanical head tracker). We thus assume our vision-based head tracker caused the same effect on the subject as Arthur et al.'s mechanical tracker.

A simple explanation for why head tracking was slower is that it requires time for the subject to move their head. Let us discuss this phenomenon more carefully.

As shown in the graph (Figure 9), we did not observe any learning effects in this experiment, though no training trials ware given to the subject. Thus, we can conclude that each response time roughly reflects how difficult the question was. In addition, we often observed that subjects without head tracking often gave up in difficult cases while subjects with head tracking kept trying by moving their head repeatedly. This was confirmed during informal interviews after the experiment.

The graph in Figure 10 reveals this situation clearly.

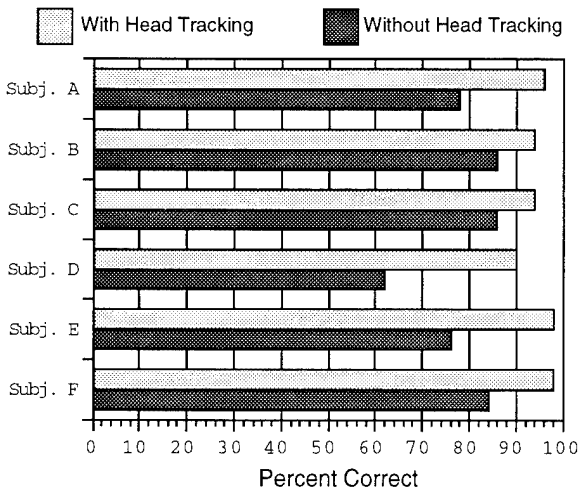



Figure 8: Correct answers ratio for each subject

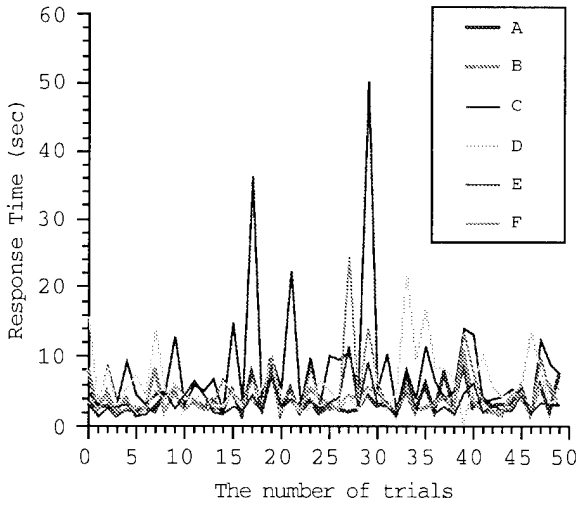



Figure 9: Response time and the number of trials

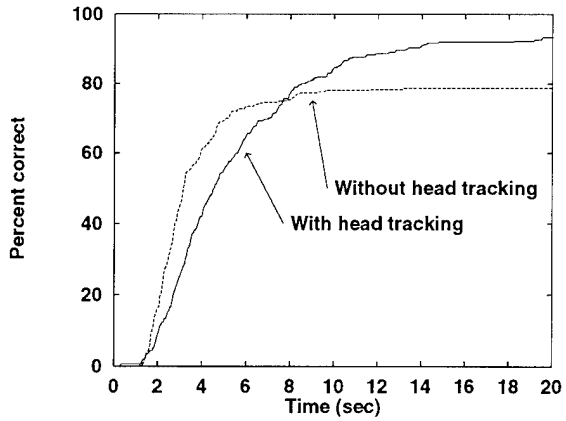



Figure 10: Relationship between response time and the percentage of correct answers

This graph is a relationship between response time and the percentage of correct answers. As illustrated in the graph, the curve of no-head tracking is saturated more rapidly than that of head tracking. This backs up our observation and can help explain why the average response time for head tracking was longer.

Overall, the result implies that a head tracker can improve a user's ability to understand complex 3D structures presented on the screen (although at the expense of time). This offers a strong incentive for incorporating a head tracker into various 3D applications such as engineering CAD systems and scientific visualization systems.

# 4 Related Work

Estimation of a user's head position by using optical techniques is not a new idea. A recent survey on position trackers [9] reports various kinds of optical tracking systems as well as magnetic, ultrasonic, and mechanical position tracking systems. Most of them, however, require the user to wear some kind of active element, ranging from an LED array to a video camera [8, 5, 14, 6]. The Honeywell Videometric System [6] uses a passive element, which is a unique symbology pattern on the user's helmet, instead of an active device. Our system also uses a special pattern, an image of the user's face itself. This feature frees the user from having to wear any tracking gear.

Suenaga et al. implemented a vision-based finger position and orientation tracker which works under rather controlled environments[15]. Their system uses two cameras to determine the position of the finger in a 3D space. Some heuristics are applied on a silhouette image of the hand to detect the orientation of the finger.

# 5 Discussions

## 5.1 Applications

Since this head tracker does not require special equipment except for a camera and capturing hardware, the technique can be applied on a wide range of 3D applications. Notable applications include 3D engineering CAD and scientific visualization systems, because these systems often require correct perception of complex 3D structures. These applications are also suitable for fish tank VR because they do not require immersion, but require high-resolution images that exceed the standard of today's head mounted displays.

In our method, the template used for pattern matching is obtained from the user's face, so it is different for each user. This requirement becomes a problem when applying this method to a system used in a public environment. However, the template's image resolution is not that high, and satisfactory results can be obtained by preparing a small set of templates representing typical patterns of face images. This should be sufficient to cover most unspecified users.

## 5.2 Robustness

Robustness is a very important issue if one applies a technique using computer vision into actual applications. In an office, for example, we can assume neither controlled lighting nor a plain background behind a user in processing incoming images.

Regarding our method, the combination of frame subtraction and pattern matching is a good compromise between robustness and processing cost. In our method, frame subtraction is used only for reducing the search area for template matching (thus increasing performance). It can report accurate results even when someone occasionally goes across the background, a situation in which most of the silhouette-based position tracking methods fail.

The stored background becomes different from the actual background over time, partly because the natural lighting is always changing. The system provides simple commands to retake the background and the template. This can be done by simply pressing a key or clicking the mouse on the captured image, so users can change the background or the template at any time without interrupting their tasks. [1]

Although correlation matching shows slight tolerance for tilted or scaled images, it becomes unstable if the user bends their head too far. A possible solution for this problem is to use two or more templates that are the rotated and scaled versions of the original template, or to incorporate rotation invariant correlation filters such as those described in [11].

## 5.3 Accuracy in tracking

Currently, our tracker does not detect the distance between the camera and the user; it assumes the distance is fixed. This assumption causes the tracker to report inaccurate positions when the user is too close or too far away from the screen. A solution to this is to use two cameras and estimate the distance based on a photographic method. This would require additional hardware and image processing time. An alternative solution I am currently working on is to estimate the distance from the size of the face image. I would also like to mention that omission of distance estimation is less noticeable than other dimensions. When a user is close to the screen, for example, the image on the display looks larger because the physical distance between the user and the display becomes shorter. It has an effect similar to distance tracking.

# 6 Conclusion

In this paper, I described a vision-based head tracker using a video camera and software that performs image processing. The techniques used here are simple, but are robust and useful for adding reality to various 3D applications. The experimental results suggest this head tracker whill help a user to recognize complex 3D structures displayed on the screen.

I believe a video camera mounted on top of the display will soon be the third standard input device (i.e., after the keyboard and mouse) for desktop computers in the near future. In addition, a camera can be useful for multimedia applications such as teleconferencing, and will be vital for human-computer interaction.

[1] Some users preferred the part of their hairline over the area between their eyebrows for its robustness in matching.

## Acknowledgment

I would like to thank Akikazu Takeuchi and other members of the Sony Computer Science Laboratory for their suggestions and comments on this project. Taketo Naito at Keio University provided the source code for a gravity-center-based position detector which eventually became the base of the system described in this paper. Many thanks also go to members in the laboratory who volunteered for the evaluation experiment.

## References

[1] Kevin W. Arthur, Kelogg S. Booth, and Colin Ware. Evaluating 3D task performance for fish tank virtual worlds. *ACM Transactions on Information Systems*, Vol. 11, No. 3, pp. 239–265, 1993.

[2] Steve Aukstakalnis and David Blatner. *Silicon Mirage*. Peachpit Press, 1992.

[3] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In *Computer Graphics Proceedings*, pp. 135–142, 1993.

[4] Michael Deering. High resolution virtual reality. *Computer Graphics*, Vol. 26, No. 2, pp. 195–202, 1992.

[5] Jih fang Wang, Vernon Chi, and Henry Fuchs. A real-time opcital 3D tracker for head-mounted display systems. *Computer Graphics (Special issue on 1990 Symposium on Interactive 3D Graphics)*, Vol. 24, No. 2, pp. 205–215, March 1990.

[6] F. J. Ferrin. Survey of helmet tracking technologies. In *SPIE Proceedings, Large-Screen-Projection, Avionic, and Helment-Mounted Displays*, volume 1456, pp. 86–94, 1991.

[7] Jiandong Liang and Mark Green. Geometric modeling using six degrees of freedom input devices. In *Proc. of the 3rd International Conference on CAD & Computer Graphics (CAD/Graphics '93)*, Beijing, China, 1993.

[8] R. Mann, G. Rowell, F. Conati, A. Tetwsky, D. Ottenheimer, and E. Antonsson. Precise, rapid, automatic 3-d position and orientation tracking of multiple moving bodies. In *Proceedings of the VIII international congress of biomechanics*, pp. 1104–1112, 1981.

[9] Kenneth Meyer and Hugh L. Applewhite. A survey of position trackers. *Presence*, Vol. 1, No. 2, pp. 173–200, 1992.

[10] Neider, Davis, and Woo. *The OpenGL Programming Guide*. Addison-Wesley, 1993.

[11] Gopalan Ravichandran and David Casasent. Advanced in-place rotation-invariant correlation filters. *IEEE trans. on pattern analysis and machine intelligence*, Vol. 16, No. 4, pp. 415–420, 1994.

[12] Silicon Graphics, Inc., Mountain View, California. *Graphic Library Programming Guide*, 1991.

[13] R. L. Sollenberger and P. Milgram. A comparative study of rotational and stereoscopic computer graphics depth cues. In *Proceedings of the Human Factors Society 35th Annual Meeting*, pp. 1452–1456, 1991.

[14] M. Starks. Stereoscopic video and the quest for virtual reality. In *SPIE proceedings, stereoscopic displays and applications II*, volume 1457, pp. 327–342, 1991.

[15] Yasuhito Suenaga, Kenji Mase, Masaaki Fukumoto, and Yasuhiko Watanabe. Human reader: An advanced human machine interface based on human images and speech. *Trans. Inst. Electronics, Inf. & Comm. Eng.*, Vol. J75-D-II, No. 2, pp. 190–202, 1992.

[16] Colin Ware, Kevin Arthur, and Kellogg S. Booth. Fish tank virtual reality. In *INTERCHI'93 Conference Proceedings*, pp. 37–42, 1993.

# Intelligent Assistance for Intravascular Tele-surgery and Experiments on Virtual Simulator

Fumihito Arai*, Masahiro Ito*, Toshio Fukuda*,
Makoto Negoro**, and Toru Naito**

* Nagoya University, Dept. of Mechano-Informatics and Systems
Furo-cho, Chikusa-ku, Nagoya 464-01, JAPAN
E-mail: arai@mein.nagoya-u.ac.jp
** Nagoya University, Dept. of Neurosurgery
65 Tsuruma-cho, Showa-ku, Nagoya 466, JAPAN
*** Kinjo Gakuin University, Omori, Moriyama-ku, Nagoya 463, JAPAN

## *Abstract*

*It is important to assist doctors operating the intravascular surgical tools such as a catheter that is designed for minimum invasive surgery inside complex and narrow brain blood vessels. We propose idea of the intelligent medical assistance system for operation of the intravascular surgical tool that is teleoperated by the doctor seeing 2D X-ray image. We built prototype of a virtual simulator system consist of a joystick and a 3D-Computer Graphics display. The joystick is used for the controller of catheter head direction and the force display. We evaluated effectiveness of the proposed visual and force assistance methods through the extensive experiments.*

## 1. Introduction

Recently, collaborative work with human (surgeon) and robots has grate attention in the medical field. It aims at utilization of both human and machine capabilities to do a task better/1,2,3/. Most of the present works are the basis to realize the actual implementation. There are also some research works on the development of the simulator for the training and planning purpose /4,5/. In recent years, medical diagnostic systems have been developed such as a X-ray CT (Computed Tomography) and a MRI (Magnetic Resonance Imaging) /6/. These medical devices enable us to build 3D models of the human body. Based on the analysis, diagnoses of the patients and planning for the medical operations are conducted. Moreover, minimum invasive surgery using intravascular surgical tools has great attention in the medical field. This technique allows us to reduce physical pain from the patient /7,8,9/.

A catheter is one of the medical tool for endovascular surgery /8,9/. In these days, the catheter is frequently used for the nurosurgical operations. It can be navigated far deep into the brain from the outside through the blood vessel. Most of the cases, the operation of the catheter is conducted based on the 2D image data taken by the real time X ray instrument. The catheter is controlled in the 3D environment, so the operation of the catheter is quite difficult, and it takes time to get accustomed to its operation. The operation of the catheter is quite important, because the surface of the blood vessel is very sensitive. Handling skill of maneuvering the catheter as well as reduction of handling time is quite important to reduce the pain from the patient. So, the medical simulator for the intravascular neurosurgery is very important for training the medical doctors.

We modeled 3D virtual environment of the human blood vessel in the graphics workstation and developed the medical training simulator for the intravascular neurosurgery /10/. We propose the modeling method of the blood vessel and dynamic interaction between the virtual blood vessel and the active catheter. In our future, we can integrate this model with the actual image from the CT or the MRI. This simulator has the force display to send back the local force information to the operator. We designed and built the joystick that has two degrees of freedom. With this system, the operator can feel the force reflection from the wall when the catheter contacts and pushes the wall, and this can contribute augmentation of reality.

Using this prototype simulator, here we propose idea of the intelligent medical assistance methods for operation of the teleoperated intravascular surgical tools, such as the active catheter. Configuration of the actual blood vessel is very complexity, and the easiness of the catheter operation depends on the view point and the coordinate system of the catheter. Based on this fact, we propose how to select the view point and coordinate system of the catheter effectively, and present the experimental results how the operation is improved by the proposed method.

## 2. Assistance System for Catheter Operation



0-8186-7084-3/95 $04.00 

## 2.1 Concept of Assistance System

Figure 1 shows the concept of the assistance system to operate the neurosurgical tools such as the active catheter. The operator handles the joystick and controls the surgical tools on the display. In this case, 3-D image of the catheter and blood vessel is created by computer graphics. The created image data are transferred to the head mount display for augmentation of reality/11/. The operator can navigate direction of the active catheter head and move it back and forth by controlling the joysticks. Restoration force is generated by the actuator at the joystick when the catheter and the virtual vessel wall contact together. Both visual and force information will support the operator to understand intervention at the endovascular surgery. With this assistance system, we can develop a simulator for training purpose and a planning system that is used before the actual operation.

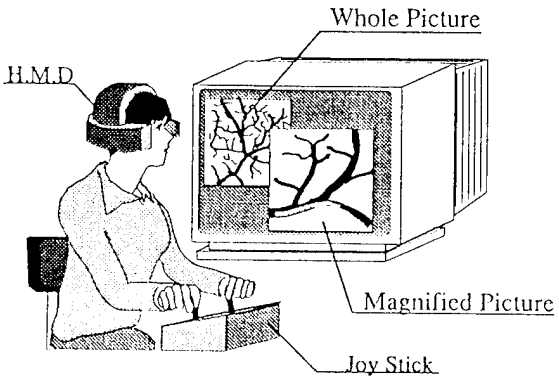



Fig. 1 Medical Assistance System

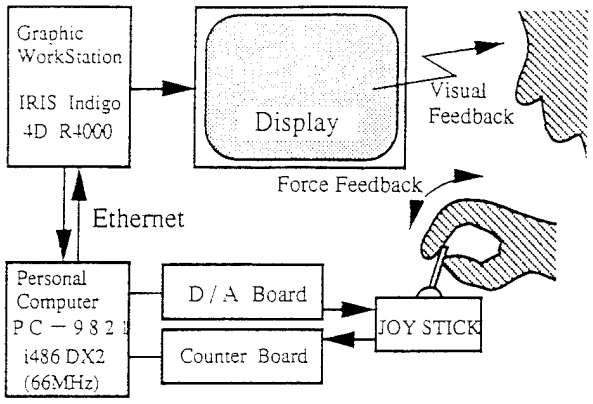



Fig. 2 Configuration of the Assistance System

## 2.2 Some Basic Ideas for the Intelligent Assistance Methods

Operator assistance is roughly classified into information level and motion level /12,13/. For human being, visual sense has very important role for perception. Some 80 to 90 % of the sensual information is coming from the eyes. So, visual assistance is effective for information level assistance. Force information is useful if we deal with physical interaction with the virtual objects. Force assistance is considered to be effective at motion level /12,13/. In case of operating the catheter in the virtual environment, both visual and force assistance will be useful. Here we suppose to develop the active catheter teleoperated with force feedback control/14/, and we propose the following assistance methods.

(1) Visual assistance (Information level) /15/
- Display the total figure from the entrance to the destination and the local extension figure around the catheter.
- Select the view point properly depend on the task automatically. The view point is selected freely and easily by the operator.

(2) Force assistance (Motion level)
- When the catheter contact with the wall, the system sends back restoration force to the operator through the joystick.
- Change rigidity of the blood vessel wall based on its diameter.
- Change the compliance of the joystick depend on the task. When the operator tries to insert the catheter to the narrow hall, the compliance is set small. When the operator tries to insert it to the wide hall, the compliance is set large.

These assistance methods will be useful for the operator at the actual trials.

## 3. Prototype Assistance System

### 3.1 Force Display

To develop the useful medical simulator for the operational training of the active catheter, we need to develop the force display. We have designed a joystick as the controller and the force display of the active catheter /10/. In the previous research works, we have developed an active catheter that has multiple degrees of freedom /9/. The basic component of it has two degrees of freedom. So we designed the joystick that has 2 DOF. This joystick has two DC motors and each motors connects with an optical encoder. The operator can input the polar coordinates' values ($\alpha$, $\theta$) according to the measurement of the encoders. The DC motors apply force to the operator.

### 3.2 System Components

Figure 2 shows the configuration of the prototype assistance system that is used for the intravascular neurosurgical simulator. The system consists of the joystick, personal computer PC9821Ap (i486Dx2-66MHz), graphics work station IRIS Indigo that creates the virtual environment considering interaction with the operator inputs. The personal computer and the graphics work station are connected by the Ether net cable.

## 3.3 Construction of Virtual Environment

### 3.3.1 Modeling of the Catheter

It is recommended to model the catheter dynamics with consideration of the material property, but this leads to increase the calculation time. So, we assume the following conditions to simplify the active catheter model.

(1) The catheter bends with constant curvature.
(2) Length of the active center axis does not change.
(3) Cross section is circle and does not change.
(4) The catheter moves along the curvature when the tip of the catheter contacts the blood vessel wall with the top surface of the catheter and the blood vessel wall being perpendicular.
(5) Except (4), the catheter moves backward and forward constrained along the normal direction of the non-active base section of the active units.

Figure 3 shows the bending model of the catheter under assumptions (1) - (3), and Fig. 4 shows the traveling model of the catheter under the assumptions (4) and (5).

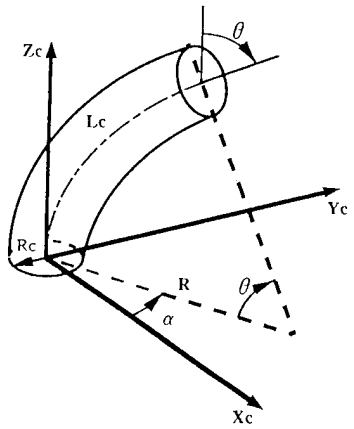



Fig. 3 Bending Model of the Active Catheter

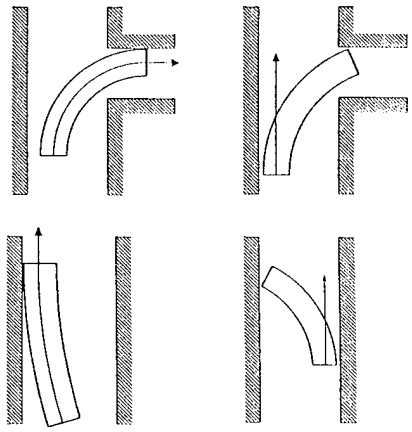


Fig. 4 Traveling Model of the Catheter

### 3.3.2 Modeling of the Blood Vessel

The blood vessel is also modeled simple. The section of the blood vessel is circle. The branch (divergence point) is modeled by the two cylinders. The blood vessel, that is bent or whose section diameter changes, is modeled by connecting the several different kinds of cylinders. An anelisium is modeled as the same way. Figure 5 shows the example of the blood vessel model.

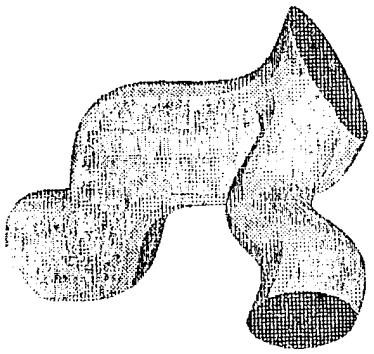


Fig. 5 Blood Vessel Model

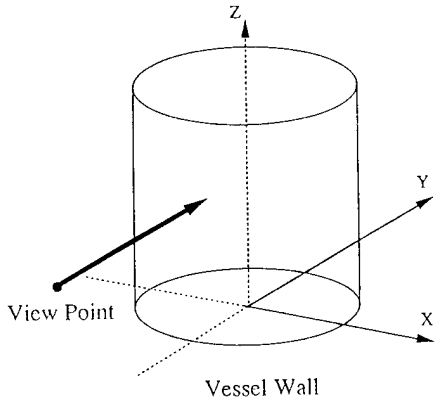



Fig. 6 View Point in the Experiments

## 4. Effectiveness of the Force Feedback System

Force display is designed to simulate restoration force of the catheter and contact force with the blood vessel wall. To check effectiveness of the force feedback system, we conducted simple experiments. The catheter is set at center of the blood vessel, and the operator is ordered to bend the catheter until it touches the blood vessel. The maximum bent position of the catheter at the cross section of the blood vessel is stored for each trial. The operator conducted the trials under the following three different conditions.

(a) Vision information is only given to the operator.
(b) Force information is only given.
(c) Both vision and force information are given.

The view point of the operator is set as shown in Fig. 6, and the experimental results are shown in Table 1. (a) The tip of the active catheter pushes the wall hard so

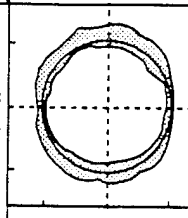
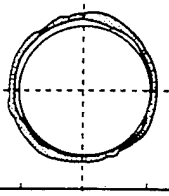
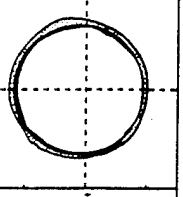
many times. In this case, depth information is not sufficient, and these cause errors in the backward and forward directions more than the right and left directions.

(b) When pushing force is small, restoration force of the blood vessel is small. Moreover, there is time delay in the force response time. From these reasons, the operator tends to push the wall by the tip of the catheter.

(c) We can get better results than (a) and (b). The force applied to the wall is small compared with the other cases.

From these results, we can confirm the effectiveness of the visual and force information to precept the interaction with the virtual environment.

Table 1 Effectiveness of the Force Feedback

| | Visual Feedback | Force Feedback | Visual & Force Feedback |
|---|---|---|---|
| Sketch |  |  |  |
| | -0.5 0.0 0.5 | -0.5 0.0 0.5 | -0.5 0.0 0.5 |
| $\delta x$ | 0.00282 | 0.00943 | 0.00320 |
| $\delta y$ | 0.01077 | 0.00799 | 0.00417 |
| $\delta r$ | 0.01689 | 0.01174 | 0.00737 |

δx: Average position eroor in x-direction
δy: Average position eroor in y-direction
δr: Average psition eroor in radial direction

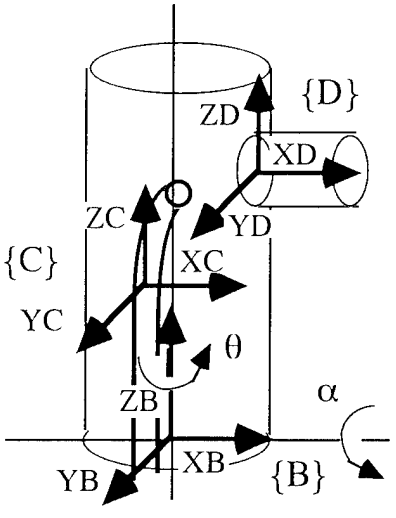



Fig. 7 Coordinate Systems

## 5. Visual Assistance

### 5.1 Coordinate Systems

It is difficult to set a vision sensor at the tip of the catheter, if it becomes extremely thin. Normally, we get the 2D X-ray image for the real time operation. However, the 2D image is one representation of the 3D object, so view point selection becomes very important depending on the task. Moreover, selections of the coordinate systems of the catheter and the blood vessel model are also important. If they are not set properly, the teleoperation task becomes difficult. To discuss these matters, we set the coordinate systems as shown in Fig. 7. {B} is the coordinate system of the blood vessel, {C} is that of the catheter, and {D} is that of the next blood vessel at the branch on the route. Origin of {C} is set at the active basement of the catheter.

### 5.2 Visible Ratio and Average Visible Ratio

Here we define the Visible Ratio(VR) of the surface. VR is the ratio of its area to its area that can be seen from the view point (Fig. 8). An average of the VR for the object's all visible surfaces is defined as the Average Visible Ratio(AVR) /16/. The VR($\omega_i$) and AVR($\omega$) are written as follows.

$$\omega_i = \frac{Sv_i}{S_i} = \cos\theta_i \tag{1}$$

$$\omega = \frac{1}{N}\sum_{i=1}^{n} C_i \omega_i \tag{2}$$

$$N = \sum_{i=1}^{n} C_i \tag{3}$$

$$C_i = \begin{cases} 0 & (\cos\theta_i > 0) \\ 1 & (\cos\theta_i \le 0) \end{cases} \tag{4}$$

where,

$S_i$: Area of the surface
$Sv_i$: Area that can be seen from the view point
$0_i$: Angle between the view line vector and the normal vector of the surface
n: Total number of the object's surfaces
N: The number of the visible surfaces

We can evaluate the visibility of the object by the AVR. The larger the AVR is, the less the number of surfaces that are difficult to see. It seems good to select the view point at which the AVR becomes maximum to understand the configuration of the object.

## 5.3 Extension of the Average Visible Ratio

Here we suppose the blood vessel consists of the symmetric elements, and the blood vessel is modeled by the set of spheres and cylinders. VR and AVR for the visibility measure are essentially defined for the plane. To deal with such symmetric curvature surfaces, we select one divided plane surface as the representation of the curvature and count it as one surface. Then we can extend the Visible Ratio for the symmetric curvature objects.

(1) Sphere

As shown in Fig. 9 (a), the surface plane is selected as the representing surface. Its normal vector is on the line of sphere center and view point. In this case, n=N=1 and the Visible Ratio and Average Visible Ratio are equal (w=1).

(2) Cylinder

As shown in Fig. 9 (b), the surface plane is selected as the representing surface. Its normal vector is parallel with the perpendicular line from the view point to the cylinder center axis

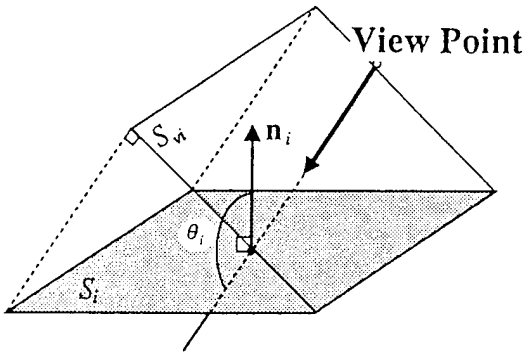
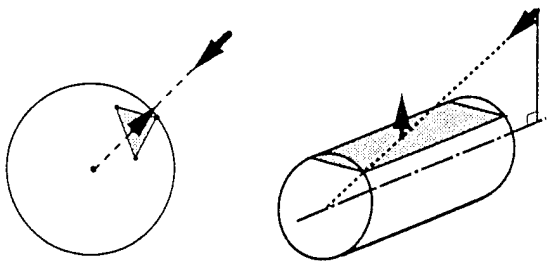



Fig. 8 Visible Ratio



(a) (b)

Fig. 9 Special Cases

## 5.4 View Point Selection based on the Average Visible Ratio

We want to help the operator recognizes the configuration of the object by making the Average Visible Ratio large. The proper number of the visible surface is depend on the operator's purpose. If the operator wants to understand the configuration, the number of the visible surface should be large to get the depth information. On the other hand, if the operator wants to control the tip direction, the number of the visible surface should be limited and proper surface should be seen. At this time, the input direction of the joystick should be selected natural.

In this paper, we propose to select the view point depend on the purpose of the operator. The visual assistance methods are summarized as follows.

(1) To recognize the blood vessel configuration, we select the view point so that the number of visible surface and the Average Visible Ratio becomes maximum.

(2) To make the catheter operation easy, we select the view point so that the number of visible surface becomes minimum and the Average Visible Ratio becomes maximum. In this case, the view point is selected so that it becomes perpendicular to the center axis of the next blood vessel at the branch on the route. Here, the input direction of the joystick and output direction of the catheter should be selected the same. That is, {B}, {C} and {D} in Fig.7 are set all parallel.

If there exist several combinations of (N, ω), the operator should have right to select the view points freely.

# 6. Experiments

## 6.1 Experimental Environment

Figure 10 shows the outline of the experimental system. We use this system to evaluate the proposed visual assistance methods. The blood vessel model for the experiments is shown in Fig. 11. For the catheter operation, steering at the blanch is point of discussion. Single blanch model is used to evaluate basic performance. We use the following counting rules.

(1) Outer surface of the blood vessel is counted one. It is not depend on the number of blanches.

(2) Section of each blood vessel is counted one. If the view point vector and center axis of the blood vessel are perpendicular, we don't count it.

## 6.2 Visual Assistance Based on AVR

Figure 12 shows the calculation results of the Average Visible Ratio for the blood vessel model in Fig. 11. The view point is selected based on the Average Visible Ratio, {C} is selected parallel to {B}, and the results are shown in Fig. 13. Table 2 shows the number of the average collision times when the operator changed direction at the branch. From these results, it is clear to see the number of collision times is small, when the number of the visual surfaces is limited and the Average Visible Ratio is made maximum.

## 6.3 Effect of the Coordinate System of the Catheter and Blood Vessel

Coordinate system of the catheter is important in teleoperating the active catheter. If we select the coordinate systems {B}, {C}, and {D} as shown in Fig. 7, it is easy to direct the catheter. For the expert person in operating this system, we did the experiment of changing {C} as shown in Table 3. Here, we selected the view point as in Fig.13(b). If the Unit vector direction is different, collision times are increased. So, effectiveness of the proposed method is clearly understood.

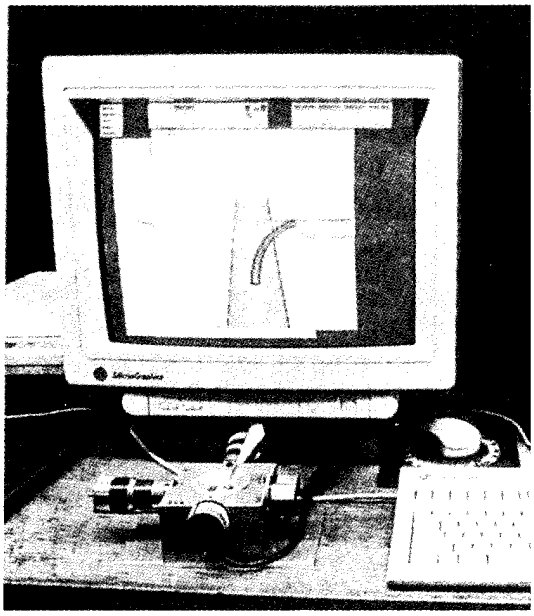


Fig. 10 Experimental System

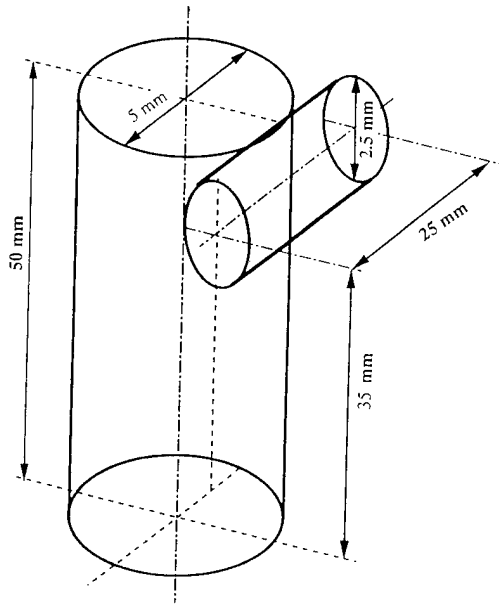



Fig. 11 Virtual Blood Vessel

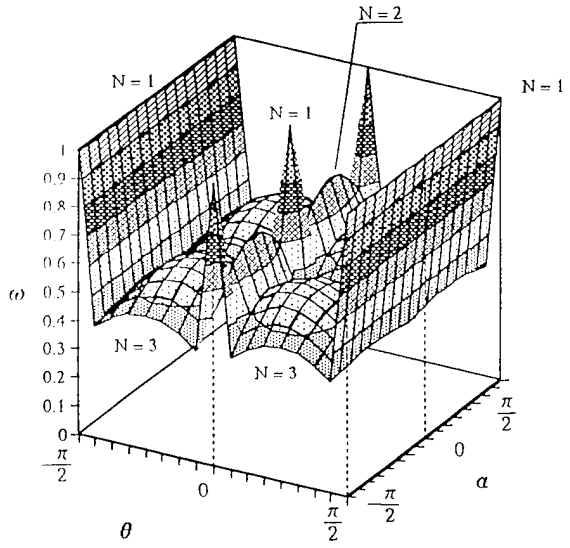
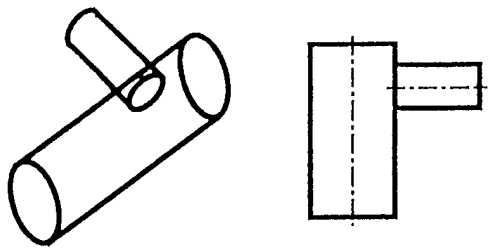



Fig. 12 Average Visible Ratio for the Experimental Model in Fig. 11



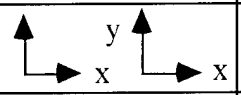
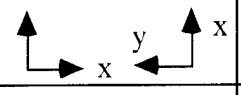
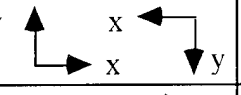
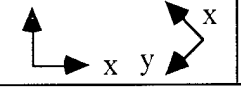
(a) (b)

Fig. 13 Visibility of the Experimental Model

Table 2 Average Collision Times

| Picture | Average Collision Times | $\sigma^2$ |
|---|---|---|
| (a) N=3, $\omega$ =0.569 | 5.833 | 0.567 |
| (b) N=1, $\omega$ =1.0 | 2.875 | 0.411 |

Table 3 Effect of the Coordinate Systems Selection

| Difference of the Coordinate Systems {B} and {C} | Average Operation Time s | Average Collision Times |
|---|---|---|
| 0  | 18.7 | 0 |
| 90  | 24.0 | 1.1 |
| 180  | 15.0 | 0.1 |
| 135  | 20.5 | 0.5 |

## 7. Conclusions

Intelligent assistance for operation of the surgical tools is quite important. The idea of the force assistance methods and visual assistance method is proposed here. The view point plays an important role for the operation. So, we defined Average Visible Ratio for the unified approach of selecting the view points automatically. Coordinate system of the catheter is also important to reduce the collision times. The numbers of visible surface, Average Visible Ratio, and the coordinate system of the catheter are selected properly according to the proposed idea. The operation is improved in the experiments.

## References


[1] K.S.Kwon, et al., "A Robot with Improved Adcolute Positioning Accuracy for CT Guided Stereotactic Suegery", IEEE Trans. on Biomedical Engineering, February 1988, pp.153-161

[2] R.H.Taylor, et al., "An Image-Directed Robotic System for Precise Orthopaedic Surgery", IEEE Trans. on Robotics and Automation, Vol.10, No. 3, June 1994, pp.261-275

[3] S.E. Salcudean and J. Yen, "Towards a Force-reflecting Motion-Scaling System for Microsurgery, Proc. 1994 IEEE Int'l Conf. on Robotics and Automation, Vol.3, 1994, pp.2296-2301

[4] S.K. Singh, et al, "Design of an Interactive Lumbar Puncture Simulator with Tactile Feedback", Proc. IEEE Int'l Workshop on Robot and Human Communication RO-MAN'93, 1993, p.156-159

[5] J.W. Hill, "Telepresence Surgery Demonstration System", Proc. 1994 IEEE Int'l Conf. on Robotics and Automation, Vol.3, 1994, pp.2302-2307

[6] Toshiba Review, Special Issue: Medical Imaging Systems/Advanced Information and Control Systems for Thermal Power Plants, Vol.49, No.2, 1994

[7] K. Ikuta, "The Application of Micro/ Miniature Mechatronics to Medical Robots", Proc. IEEE Intelligent Robotic Systems, 1988, pp9-14

[8] P. Dario, R. Valleggi, M. Pardini, and A. Sabatini, "A Miniature Device for Medical Intracavitary Intervention", Proc. IEEE Micro Electro Mechanical Systems, 1991, pp.171-175

[9] T. Fukuda, S. Guo, K. Kosuge, F. Arai, M. Negoro, and K., Nakabayashi, "Micro Active Catheter System with Multi Degrees of Freedom", Proc. 1994 IEEE Int'l Conf. on Robotics and Automation, Vol.3, 1994, pp.2290-2295

[10] F. Arai, M. Ito, T. Fukuda, M. Negoro, and T. Naito, "Intelligent Assistance in Operation of Active Catheter for Minimun Invasive Surgery", Proc. IEEE Int'l Workshop on Robot and Human Communication RO-MAN'94, 1994, p.192-197

[11] M.F. Deering, "Explorations of Display Interfaces for Virtual Reality", Proc. IEEE Virtual Reality Annual Int'l Symp. VRAIS'93, 1993, p.141-147

[12] F. Arai, T. Fukuda, Y. Yamamoto, T. Naito, and T. Matsui, "Interactive Adaptive Interface Using Recursive Fuzzy Reasoning", Proc. IEEE Virtual Reality Annual Int'l Symp. VRAIS'93, 1993, p.104-110

[13] F. Arai, T. Fukuda, Y. Yamamoto, T. Naito, and T. Matsui, "Interactive Adaption Interface Monitoring and Assisting Operator by Recursive Fuzzy Criterion", Proc. IEEE Int'l Workshop on Robot and Human Communication RO-MAN'93, 1993, p.448-453

[14] S. Maeda, K. Minami, and M. Esashi, "KrF Excimar Laser Induced Selective Non-planer Metalization", Proc. of IEEE MEMS-94, Oiso, 1994

[15] R. Hurteau, S. DeSantis, E. Begin, and M. Gagner, "Laparoscopic Surgery Assisted by a Robotic Cameraman: Concept and Experimental Results, Proc. 1994 IEEE Int'l Conf. on Robotics and Automation, Vol.3, 1994, pp.2286-2289

[16] G. Xue, T. Fukuda, and H. Asama, "Error Recovery in the Assembly of a Self-Organizing Manipulator by Using Active Visual and Force Sensing", JSME Int'l Journal, October, 1994, (to be published).

# Techniques: Animation, Vision, and Collision Detection

# Model Based Vision as feedback for Virtual Reality Robotics Environments

*E. Natonek, Th. Zimmerman, L. Flückiger*
Swiss Federal Institute of Technology
EPFL, IMT-DMT- Vision Laboratory
1015 Lausanne, Switzerland
+41-21-6933826
natonek@dmt.epfl.ch

**ABSTRACT**
Task definition methods for robotic systems are often difficult to use. The "On-line" programming methods are often time expensive or risky for the human operator or the robot itself. On the other hand, "Off-line" techniques are tedious and complex. In addition operator training is costly and time consuming.

In a Virtual Reality Robotics Environment (VRRE), users are not asked to write down complicated functions, but can operate complex robotic systems in an intuitive and cost-effective way. However a VRRE is only effective if all the environment changes and object movements are fed-back to the virtual manipulating system.

This paper describes the use of a VRRE for a semi autonomous robot system comprising an industrial 5-axis robot, its virtual equivalent and a model based vision system used as feed-back. The user is immersed in a 3-D space built out of models of the robot's environment. He directly interacts with the virtual "components", defining tasks and dynamically optimizing them. A model based vision system locates objects in the real workspace to update the VRRE through a bi-directional communication link.

In order to enhance the capabilities of the VRRE, a reflex-type behavior based on vision has been implemented. By locally (independently of the VRRE) controlling the real robot, the operator is discharged of small environmental changes due to transmission delays. Thus once the tasks have been optimized on the VRRE, they are sent to the real robot and a semi autonomous process ensures their correct execution thanks to a camera directly mounted on the robot's end effector. On the other hand if the environmental changes are too important, the robot stops, re-actualizes the VRRE with the new environmental configuration, and waits for task redesign.

Because the operator interacts with the robotic system at a task oriented high level, VRRE systems are easily portable to other robotics environments (mobile robotics and micro assembly).

**KEYWORDS**: Virtual Reality, Model based vision, Robotics task definiton,visual feedback

**INTRODUCTION**
There is no proper definition of Virtual Reality, but we could summarize it by being an extension of a 3-D visualization system with the addition of devices enabling the user to interact with the synthetic world in a natural way. An important aspect of VR is the immersion, i.e. seeing the alternate reality from the inside as opposed to merely observing it through a window. (Recall the movie Tron, where the main character was inside the computer, experiencing the battling game). A typical virtual reality system consists in one or more input devices (joystick, 3-D mouse, data-gloves, ...) and several forms of outputs (such as 3-D images, sound and pressure). Reading the phrase of J.C. Craig [7] « In order to make the description of manipulator motion easy for a human user of a robot system, the user shouldn't be required to write down complicated functions of space and time to specify the task » one can easily understand the huge potential of Virtual Reality in a Robotics Environment (VRRE).

Robotics system task definition methods are often difficult to use. The "On-line" programming methods are often time expensive or risky for the human operator or the robot



0-8186-7084-3/95 $04.00 

itself. On the other hand, "Off-line" techniques are tedious and complex. In addition operator training is costly and time consuming. Several research groups are using Virtual Reality's intuitive man-machine interfaces to guide and program robots.

The Sandia National Laboratories are using VR for the retrieval of hazardous waste from underground storage tanks [11]. Similarly the DLR (German Aerospace Research Establishment) is dealing with Space Robot Technology Experiment (ROTEX) using a VR based user interface [9]. The telerobotic structure for remote control of space robots as developed at the DLR has been inspired by the "learning" by "showing" in a simulated environment. The VR4RobotS (Virtual Reality for Robot System) [15] at the IPA in Stuttgart and the Simulation Animation Visualization and Interactive Control (SAVIC) [4] from Univ. Of Tennessee are a few other research groups using virtual reality for robotic tasks.

But a virtual system is only effective if all the environment changes and object movements in the real world are fed-back to the virtual system. Thus some kind of visual or multi-sensor feedback is required. Vision systems that use 3-D models of the world like Virtual Reality are called 3-D Model Based approaches. Use of model based vision and virtual reality is illustrated in figure 1.

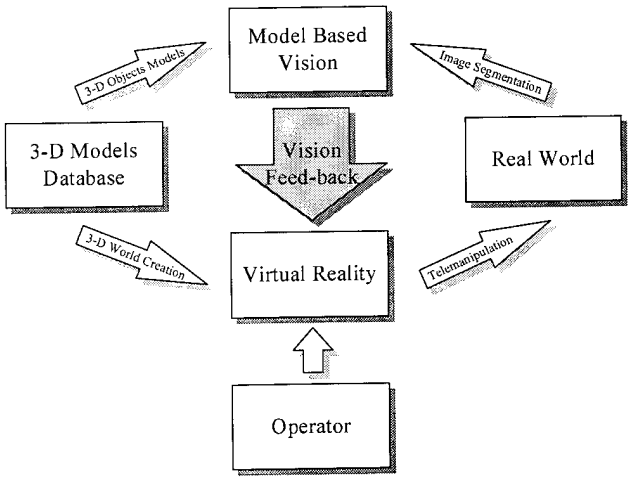



Figure 1: Model based vision as feed-back for Virtual Reality

Models of objets are used to locate them in the real world as well as to represent them to the user through VR. If the vision system continuously updates the VR world the operator doesn't need a camera image of the real world but only needs the VR representation.

It is also true that other outputs like sound and/or force feed-back can be used to give a better *feel* of the real world from the virtual reality environment. For the rest of this paper we will focus only on visual feed-back.

Robots and their environments are essentially 3-D, and the use of 3-D scanners and the according image segmentation and 3-D object recognition will be imperative in the future. In the field of Virtual Reality, vision and robotics, we currently have projects running enabling to program industrial and mobile robots in a VR environment [8] and [13].

In the next section we will emphasize on the necessity of an intuitive Virtual Reality Robotics Environment (VRRE) for complex robotic systems. Further we will describe the architecture of our VRRE system showing the graphical environment and the 3-D model based object recognition used as feed-back, as well as the use of the sensors to enhance the capabilities of a given manipulator. Related VRRE works on mobile robotics and micro-assembly as well as future enhancements will be exposed, and finally we will conclude.

## VIRTUAL REALITY ROBOTICS ENVIRONMENTS

Classical robot programing (point to point for example) is only possible for repetitive tasks like in assembly lines. Trajectory generation should be, with Virtual Reality Robotics Environments (VRRE), absolutely intuitive. In such environments the operator is immersed in a 3-D representation of the world with which he can interact. A typical working session is composed of three main steps: *trajectory creation* in the virtual space, *simulation* of the task and finally *execution* of the task by the real robot if no problem occurred during the simulation step. In VRRE, operators can manipulate the virtual robots as well as move the viewpoint in a very simple and intuitive way. By changing the viewpoint the operator can look at specific regions of the virtual working area.

Not only the manipulators, but also the other elements of a real scene are modeled in the virtual world. Each element has its own properties (shape, position, orientation, behavior laws). Using the data related to the virtual objects, it is possible to automatically generate trajectories to manipulate them (grasp, insert, screw etc...). This way tedious work can be spared to the operator. The created tasks can be simulated at any time with a collision detection algorithm. This is essential to avoid important damage to the real robot.

Working with VRRE has other advantages. First the size of the real robot isn't relevant because, the real robot be huge or very small, the user is brought to the same scale through VR representation. Second the off-line treatment between

the virtual creation and the real execution allows teleoperation without the risks due to direct link. With a direct link, an unexpected or wrong movement of the operator could destroy the system. Moreover, the problems due to transmission time delays are suppressed. A few seconds delay between the operator's action and the visualization of his action's effect on the real robot is really bad for control. It's like adjusting the water temperature in your shower with a two seconds delay. For example, the control of a robot located on Mars (transmission time: over 3 minutes) is impossible with direct link and our method is requested. The last mentioned example implicitly shows an other advantage of VRRE, namely that the distance between the operator and the real system isn't relevant either.

However a VRRE is only effective if all the environment changes and object movements are fed-back to the virtual manipulating system. Some kind of visual or multi-sensor feedback is therefore mandatory.

## VRRE SYSTEM COMPONENTS DESCRIPTION

Virtual Reality systems provide many conveniences: users interface, new tools, possibility of off-line treatment. Our VRRE system comprises an industrial 5-axis Mitsubishi Movemaster RV-M1 robot and its virtual equivalent.

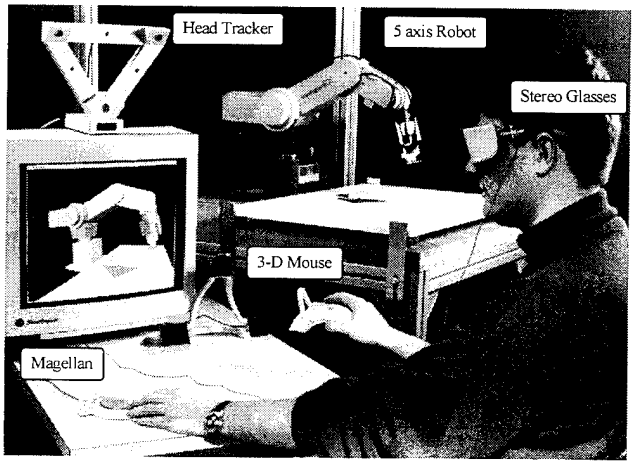



Figure 2: Implemented VRRE

The user is immersed in a 3-D space constituted of models of the robot's environment. He directly interacts with the virtual "components" in an intuitive way. Creating trajectories and tasks, and dynamically optimizing them. A model based vision system updates the virtual world by recognizing objects in the robot's environment. Further, a camera positioned on the robot's end effector is used to collect data during task execution, enabling flexible and efficient object manipulation.

Figure 2 shows, on the left side the VRRE system composed of 3-D input devices that will be described further and on the right side the real robot, its controller and the model based vision system. Our vision system has 2 cameras: one on top of the robot's working area giving a global view of the working space, and the other one on the end effector for close range analysis.

### VR graphical environment

The virtual environment is implemented on a Silicon Graphics (Crimson) workstation. A 3-D object oriented graphical library (from Sense8) is used to manage the virtual world and the input devices (HMD, Magellan, 3-D Mouse). Stereo glasses are used for 3-D visualization.

We currently use two different input devices: a Magellan and a 3-D Mouse ( Magellan and 3-D Mouse are Logitech trademarks). Both have 6 degrees of freedom. The Space Mouse is an optical multiaxis sensor which looks like a joystick. It delivers information on the 3 translations and 3 rotations. The 3-D Mouse is an ultrasonic positioning system. Each of these devices has advantages and drawbacks depending on the application.

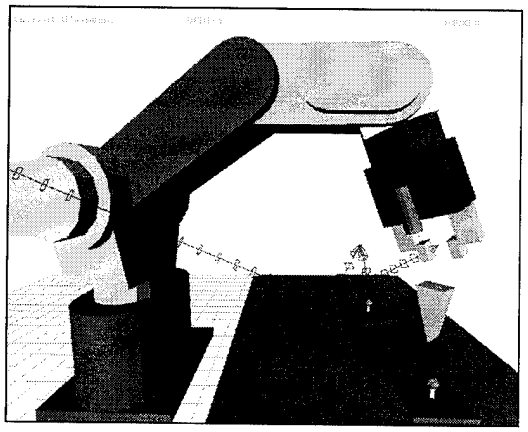


Figure 3: Virtual Reality Robotics Environment for a classical manipulator.

The Magellan behaves like an incremental sensor and is specially suited to move a viewpoint. The 3-D Mouse behaves rather like an absolute position sensor. It is well suited for manipulator control as it always stays in a bounded working area.Position and orientation in space given by the 3-D Mouse is visualized by a 3-D cursor, the "picker" (figure 3).

Using the manipulator's inverse kinematics, the angle of each of the its joints is computed in order for the Tool Center Point to follow the picker as long as it stays within

the robot's working domain. If the picker isn't accessible, the virtual robot stays in the last accessible position until the picker returns within its working domain. The user therefore has a direct feed-back of the robot's capabilities.

This method has the advantage of being independent of any "mechanical puppets" - which have to be specifically designed for each manipulator. It is also independent of the human arm's physionomy which certainly has a different kinematics than the robot's. As proof, we are controlling a 5 axis robot using a 6 axis interface without problems: one of the mouse's orientations isn't considered, but the rotation of the picker around the vertical axis is computed depending on the mouse's other coordinates. A trajectory definition becomes trite: the operator only has to point the position to be reached with the 6-D mouse, and to click in order to insert a control point at this position. The robot's orientation is automatically computed.

### Model Based Vision System

The goal of a computer vision system is to extract information from visual data in order to complete a given task like assembly, inspection, robot navigation, etc. In this section we will describe how our vision system is able to identify and locate specific objects in a scene. Our model based vision system has full a-priori knowledge of the desired object (shape, color, size, etc.). The vision system is implemented on a PC using a high end vision board; the Matrox IMAGE-1280™.

A model based vision system can be divided into the training phase and the classification phase (figure 4).

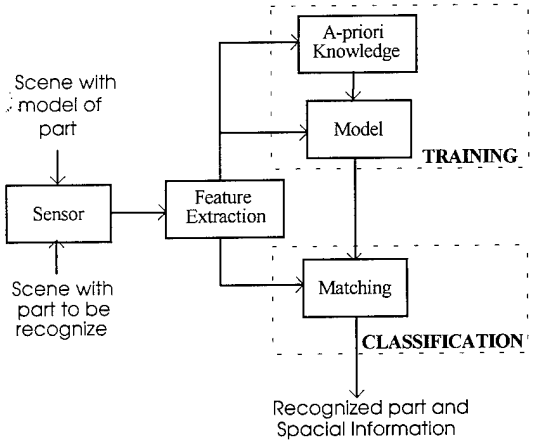



Figure 4: Model based recognition system

Feature extraction and object modeling are the main components of the training phase which is traditionally a bottom-up process.

In typical industrial applications the number of parts is usually small (1-50) and their geometrical properties are exactly specified with known tolerances (their CAD-CAM database is often available). The parts usually have distinctive geometrical features (holes, edges, corners) and the scenes with multiple parts can have different possible configurations with touching or overlapping parts.

Feature extraction and matching are the central aspects of the classification phase.

**Models** in VRRE have two purposes, namely to describe an object in terms of *features recognizable by the user* for visualization (i.e. 3D graphical representation), and in terms of *features extractable with a given sensor* for recognition. These two aspects can more or less overlap but aren't necessarily exactly the same. Color can for instance be used for representation purposes to distinguish small and big objects, whereas area and perimeter will be used for recognition purposes. We will here solely concentrate on the recognition model.

The central issue in model generation is twofold. First we have to describe the physical properties of the object and its spatial relation in the scene. Secondly we have to point out what constitutes an adequate representation for these features. In other words, how should we create a model able to characterize all parts present in the image. So far several research groups have been trying to solve this problem using 2-D, 2½D and 3-D models. We will not examine this in detail but references can be found in [2] , [6] and [16].

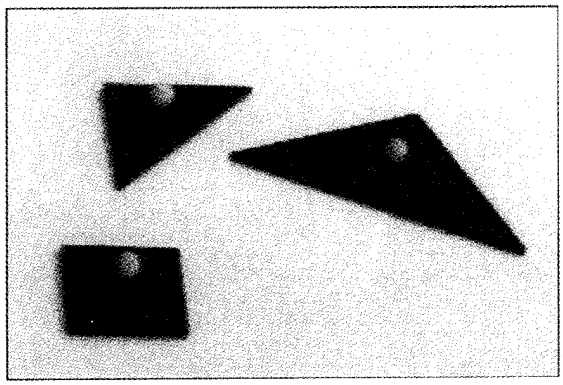


Figure 5: Objects to be recognized

Virtual Reality systems have 3-D models of the world. Each object has its own 3-D database and sometimes its photometric attributes and additional information such as thermal and stress proprieties. The objects our vision system should recognize are squares and triangles of

different sizes as shown in figure 5. For object recognition we are using the top camera.

***Recognition features*** selection is tightly bound to the model space representation (2-D, 2½D or 3-D). Even if our global models are 3-D our sensor is 2D (CCD camera) we therefore will use 2-D features such as area, number of edges, perimeter and moment of inertia. First a gray scale image is binarized using a dynamic threshold. Then the system extracts edges using a conventional gradient operator. Next an edge kernel is used to find connecting edges (corners) which will give us the object type (4 corners for a square, 3 corners for triangle). At the same time a tracking algorithm is applied on each edge to determine object orientation. The moment of inertia is used to locate the object in space. Area and perimeter will discriminate similar objects having different sizes (small or big). The output of the implemented object recognition algorithm is shown in figure 6.

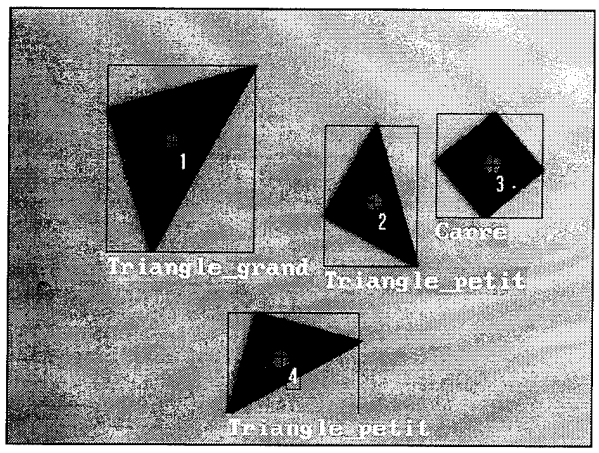



Figure 6: Output of the implemented vision system

Note that bounding boxes surround each detected object, giving indications on the occupied working space. This information will be used for path planing and obstacle avoidance.

***Matching*** means finding a set of salient features in a given image, that match the model's features. Matching 2-D models usually consists in locating parts with a few key features, using global feature or graph-matching techniques.

Our vision system uses a *decision tree* method based on a list of global features described above. A different feature is examined at each level of the tree. The decision tree is built as follows: First, the features are classified for their saliency. A feature is said to be salient if it easily distinguishes different models. The most *salient* feature is examined at the root node of the tree. For each possible feature value a child node is built at the next layer where the next salient feature is examined. The procedure goes on until all features have been examined. The resulting tree has one input node and N output nodes, N being the number of known objects. The main advantage of a decision tree is its speed, the main drawback is the lack of flexibility (i.e. if we want to introduce a new object the previously established saliency classification may not hold any more).

In our case the most salient feature is the number of edges. Two child nodes, one for squares the other for triangles are then built to decide if they are "small" or "big" using area and perimeter.

Once an object has been recognized its high level description (i.e. small triangle, big square etc..), its label, its position, and its orientation are sent to the virtual system. It is important to notice that only *high level information but no image* is sent to the VRRE.

**Task programming through a VRRE**

All the objects present in the real working area are recognized by the vision system and sent to the VRRE through the net via a socket like protocol. Thus enabling the VRRE to be updated. The operator creates a trajectory in the virtual space, simulates the task and if no problem occurred during this step executes it. Task execution is performed by downloading high-level commands to the PC supervising the robot controller via socket like internet communication. These high level commands such as TAKE SQUARE are then converted to native robot language and sent to the robot controller via serial link. The task is then executed by the robot.

**Reflex-type behavior**

As quoted before use of VRRE needs high performance sensors. These being available, it is of interest to use them to enhance the capabilities of a manipulator by locally (independently of the VRRE) processing data to generate a reflex type behavior which can deal with small environmental changes, thereby discharging the operator. The here described reflex is to give the robot capability to fine tune its final approach before grasping an object. For this purpose we use the on arm mounted camera whose signal is processed with the same image processing board previously used for object recognition. This reflex, when activated, tracks the considered object's grasping point, and moves the end effector accordingly until the perfect seizing position is reached.

This way we can successfully avoid position errors and drifts due to transmission delays by local image processing. If the environmental changes are too important (i.e. no grasping point is localized in the vicinity of the position specified by the operator) the robot stops, re-actualizes the VRRE with the new environmental configuration, and waits for task redesign by the VRRE operator.

Figure 7: On board camera and output of the tracking algorithm.

## RELATED VRRE WORKS AND FUTURE EXTENSIONS

The interest in using VRRE for robotic task design lies not only in the intuitive interface, but also in the portability of the environment to other systems. Separating the user interface from the real hardware by using high level commands, ensures great flexibility and portability. Only the appropriate drivers for each specific robot have to be written.

The first implementation of a VRRE was on a conventional 5-axis robot. We next show that this environment was successfully adapted to a mobile robot. For this purpose a tiny mobile robot called Khepera®, conceived and built by the K-Team, Laboratoire de microinformatique of the Swiss Institute of Technology (LAMI), was used. An interesting feature of the Khepera is the on board Motorola 68331 microcontroller (performance similar to the Motorola 68020 running at 16 MHz) with 256 Kbytes of RAM, 256 Kbytes of ROM, six A/D channels with a 10 bits resolution, and a serial link. The base plate consists of two wheels, four rechargeable batteries giving an autonomy of about 30-40 minutes and 8 infra-red proximity and light sensors, enabling obstacle avoidance. For more information see [12].

When the mobile robot "sees" an obstacle it feeds-back to the VRRE in order to construct the 3-D environment in which the robot evolves. Simple rules have been specified like telling the Khepera to stop near an obstacle and to try to completely surround it by following its edges. Finally when enough information is available, the VRRE system produces a bounding box describing the obstacle detected by the Khepera's sensors. No object recognition is performed so far, but the VRRE and the operator, are informed that an obstacle with a known size and position is in the robot's working area and has to be avoided (figure 8).

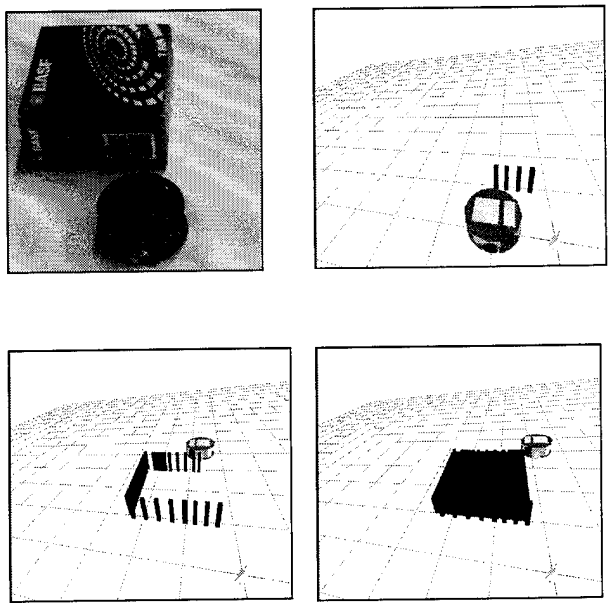


Figure 8: Khepera doing active obstacle detection

An other extension of VRRE is in microsystems. Because microsystems and micro structures become smaller and smaller, it is imperative to build micro-robots capable of precisely manipulating such systems and structures which are in the micron range. Manipulation, programming and visualization of such systems will only be possible through a virtual system or equivalent [17].

A typical microsystem assembly task is shown in figure 9. This figure shows the surface information of an electrostatic micro motor fabricated with the LIGA-process of Karlsruhe [1], scanned by a laser microscope. The motor has an average height of 84 µm, and the rotor a diameter of 120 µm. The performed task is the insertion the shaft on the motor axis that is 40-50µm large.

The idea of this project is to use the laser scanning microscope to perform a real-time positioning control of micro structures to guide a micro robot during assembly using VRRE.

The laser microscope gives a 3-D dense range map of the scanned object. Therefore range image segmentation and

3-D object recognition are essential to ensure proper feedback.

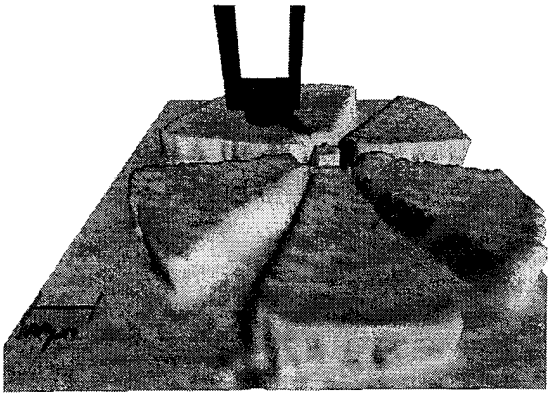


Figure 9: Virtual reality in microsystems. Insertion of the shaft on a micromotor stator.

So far a conventional CCD camera as well as simple range sensors have been used. The future extension of our vision system will be the use of sophisticated laser range scanners coupled with conventional intensity cameras (CCD).

This extension is called Three Dimensional Model Based Approach (3D-MBA) and figure 10 shows the recognition of two 3-D objects. The idea is to use the range image to extract geometric features such as object size, position and orientation. These is done by using an iterative method, conceptually similar to the one described by Besl [3] and Chen[5]. The algorithm starts with partitioning the range image in small equal sized regions. Using least square error, each region is approximated by planar or second order surfaces. We always start with the hypothesis that all points of a region belong to the simplest surface model, i.e. with the smallest number of degrees of freedom. Only regions well described by a planar or second order approximation are kept and called *seeds*., All these seeds will grow until all surfaces in the range image are covered or that the least square error between the surface model and the data is to important, using an modified Leonardis algorithm [10], [18]. These extracted features are then combined with the a priori knowledge of our model based vision, to create a hypothesis.

To achieve the results shown in figure 10, 3D-MBA first uses pyramid of resolution and prediction-verification processes. At the range image stage, the system builds a set of hypotheses about the objects present in the scene and then proceeds by trying to confirm/reject them. Newly developed range image segmentation techniques will extract edges and surfaces for feature checking. Using the a priori knowledge and the depth map it will extract the size and exact positioning of both objects (figure 10). At this step both objects are recognized in a purely geometrical way. For examples see [14]. Next this information is used to construct a more realistic 3-D model (the *virtual image* figure 10d) using the geometrical database and the available photometric attributes. Finally, last recognition is performed by crosscorrelating the virtual image and the real CCD camera image (figure 10a).

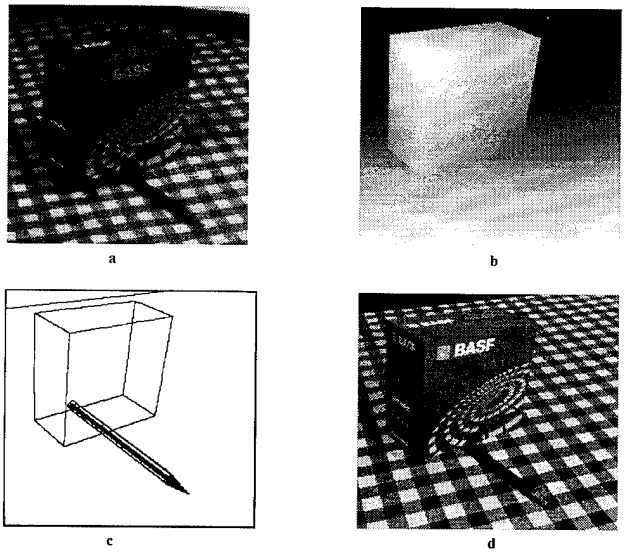



Figure 10: Object recognition example using 3D-MBA and range images. a) conventional CCD camera image, b) dense range map of a), c) geometrical identification of the searched objects, d) reconstructed virtual model.

## CONCLUSION

VR is a potentially very powerful interface for robotic systems control. The very intuitive way in which the operator interacts with the virtual environment, and the simulation capabilities it has, makes it easy and safe (i.e. children have been using our system successfully and without creating any damage). However this is only true if the changes affecting the real world are properly and consistently fed-back to the virtual environment. There is no point in doing multiple collision tests in the virtual world if the description is so far from reality that even a robot with high level of autonomy can't deal with the differences.

Implementation of a reflex-type behavior in robotic systems which make task design easier and task execution

safer, has been shown. Here also the feed-back is of prime importance.

Therefore however you look at it, the more one expects from such systems the better the visual or multi-sensor feedback has to be. Without feed-back the VRRE is very nice to look at, but useless.

We extensively described a VRRE implementation for a classical 5-axis manipulator using a simple but robust vision system. Extension of the vision system to more complex 3-D object recognition has also been exposed, which will further enhance our system's possibilities.

We briefly presented systems being implemented for mobile and micro-robots. Thereby showing the very large scope of possible applications.

**ACKNOWLEDGEMENTS**

This work is supported by the Swiss National Science Foundation under grant N°5003-34336-SSP-IF.

The authors would like to thank all the members of the *Vision Lab* of the Microengineering department (IMT) of the Swiss Federal Institute of Technology.

We also would like to thank E. Zbinden and the LAMI for the work on the Khepera, and A. Sulzmann for his work on microsystem assembly.

**REFERENCES**

1. E.W Becker and all, " Fabrication of microstructures with high ratios and great structural heights by synchrotron radiation lithography, galvanoforming and plastic moulding ", *Microelectronic Engineering 4*, pp 35-56, 1985

2. P.J. Besl and R.C. Jain, "Three-dimensinal object recognition", *ACM Comput. Survey*, vol 17, no1, pp. 75-154, 1985

3. P.J. Besl, Surface in Range Image Understanding, Springler Verlag, 1988

4. C. Chen, M. Trivedi, M. Azam, T. Lassiter,"A Novel Graphical Environment for Virtual and Real World Operations of Tracked Mobile Manipulators", *SPIE Sensor Fusion*, vol 2059, pp. 154-165, 1993

5. D.S. Chen, "A data-driven intermediate level feature extraction algorithm", *IEEE Trans. on Patt. Anal. and Machine Intell.*, vol 11 n 7, pp. 749-758

6. T. Chin Roland and C.R. Dyer ," Model Based Recognition in Robot vision", *Selected Paper on Model-Based Vision* , pp 334-375, 1986.

7. J.C. Craig,"Introduction to Robotics, Mechanics and Control", Addison Wesley, 1986

8. L. Fluckiger," Robotique, Réalité Virtuelle et Vision ", *Technical Report IMT-94-01*, 1994 (In French)

9. G. Hirzinger, "Multisensorry shared autonomy and tele-sensor programmaing - Key issues in space robotics", *Robotics and Autonomous Systems*, vol 11, pp. 141-162,1993

10. A. Leonardis, "Image Analysis Using Parametric Models, Model-recovery and Model-selection Paradigm", *Thesis* University of Ljubljana, 1993

11. N. Miner, S. Stansfield: "An interactive virtual reality simulation system for robot control and operation training", *IEEE Int. Conf. On Robotics and Automation*, pp. 1428-1435, April 1994

12. F. Mondada, F. Franzi and P. Ienne, "Mobile robot miniaturization: a tool for investigation in control algorithns" *ISER 3,* Kyoto Japan, 1993

13. E. Natonek and C. Baur, "Recognition of 3-D objects on complex backgrounds using model based vision and range images: *IEEE Southwest Symposium on Image Analysis and Interpretation,* Dallas, 1994

14. E. Natonek, C. Baur, "Model based 3-D Object Recognition using Intensity and Range image: *SPIE Int. Symposium on Optical Eng. In Aerospace Sensing,* vol 2234-01, Orlando, 1994

15. R.D. Schraft, W.M. Strommer, J.G. Neugebauer: "A Virtual reality testbed for robot application ", *ISIR 92*, pp. 105-110, 1994

16. P. Suetens, P. Fua and A.J. Hanson: "Computational Strategies for Object Recognition", *ACM Computing Survey*, vol 24 n°1, 1992

17. A. Sulzmann, C. Baur: "A Virtual reality based interface for micro-telemanipulation", *ICAT'94*, pp. 255-262, Tokyo 14-15 july 1994

18. D. Sun," Vision 3-D utilisant des cartes de profoneurs", *Technical Report IMT-94-02*, 1994 (In French)

# Human Figure Synthesis and Animation for Virtual Space Teleconferencing

Karansher SINGH*,† Jun OHYA* Richard PARENT†

* ATR Communication Systems Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02, Japan

† Dept. of Computer & Information Science, The Ohio State University
2036 Neil Avenue, Columbus, OH 43210, USA

## Abstract

*Human figure animation is a widely researched area with many applications. This paper addresses specific issues that deal with the synthesis, animation and environmental interaction of human figures within a virtual space teleconferencing system. A layered representation of the human figure is adopted. Skeletal posture is determined from magnetic sensors on the body, using heuristics and inverse kinematics. This paper describes the use of implicit function techniques in the synthesis and animation of a polymesh geometric skin over the skeletal structure. Implicit functions perform detection and handling of collisions with an optimal worst case time complexity that is linear in the number polymesh vertices. Body deformations resulting from auto-collisions are handled elegantly and homogeneously as part of the environment. Further, implicit functions generate precise collision contact surfaces and have the capability to model the physical characteristics of muscles in systems that employ force feedback. The real time implementation within a virtual space teleconferencing system, illustrates this new approach, coupling polymesh and implicit surface based modeling and animation techniques.*

## 1 Introduction

An important application of human figure animation is the evolving area of virtual space teleconferencing [8]. At ATR CSRL, the authors are building a **VI**rtual **S**pace **TEL**econferencing (*VISTEL*) system [8], aimed at an environment where teleconference participants at different sites can feel as if they are all at one site, allowing them to hold meetings and work cooperatively. In *VISTEL*, 3D models of the participants at different sites are combined into 3D virtual space images, giving participants the sensation of meeting each other in a common space (See Figure 7). Real time, realistic reproduction of motion of participants on their 3D models is important for good communication. Additionally, cooperative work requires efficient and robust handling of interaction of the human models with the virtual environment.

A layered approach [2] to the modeling and animation of articulated figures is a widely adopted methodology. With respect to human figures, the layers may be broadly classified into skeletal, muscle and skin, and clothing layers. Layers are often omitted, collapsed or further subdivided, depending on the sophistication of the model.

The **skeletal layer** is usually approximated as an articulated rigid body. Despite the problems arising from the approximations [5], realistic results may be obtained with such a model. In *VISTEL*, an articulated rigid skeleton is used. The skeleton is animated based on the posture of the teleconference participant obtained from magnetic sensors on the body.

Various techniques have been proposed for the modeling and animation of the **muscle, skin layer** in real time [2], [6],[16]. Deformations during animation to a geometric (typically polymesh) skin are specified empirically or based on an underlying muscle model.

The deformable nature of human muscle, fatty tissue and skin is described in [16], with respect to facial animation. A spring and damper mesh muscle, tissue model, attaches skin to the underlying skeleton and iteratively applied forces shape the skin. In *VISTEL*, real time facial animation is realized by visually tracking tape marks on the faces of the participants [8]. The tape mark positions drive deformations of the facial polymesh.

Free form deformations are used to empirically deform the skin layer in [2] based on joint angle values. Position of the skin polymesh around joints in terms



0-8186-7084-3/95 $04.00 

of a specific function local to the skeletal joint area is presented in [6]. There is no explicit muscle model in either case.

The muscle and skin layer is very important for visual realism and its effect on subsequent layers. Further, tight fitting apparel may be modeled as textures applied to the geometric skin. This is the approach adopted in *VISTEL*. The specific requirements virtual space teleconferencing imposes on this layer are the following:

1. Realistic appearance and efficient display of geometric skin for realistic skeletal postures.

2. Efficient collision detection and deformation of the skin with the environment (and rest of the body), the importance of which is increased by tolerable inaccuracy in posture computation.

3. Efficient modeling and computation of forces and contact areas for systems with force feedback.

The paper thus focuses on addressing the above requirements for this layer by combining implicit function and polymesh based techniques.

**Implicit functions** are a popular approach to the modeling and animation of physically deformable objects [18],[1], [10],[7],[3]. An implicit surface is defined as the set of points $P$ satisfying an equation $F(P) = 0$. Complex shapes may be intuitively built and animated using simple implicit primitive shapes [18],[14]. Superquadric primitives [10] and blobs [7], can be fitted to real 3D data. Implicit functions are used to model exact collision contact surfaces in [3]. Implicit functions can replace discrete spring models with continuous stiffness fields [3]. Efficient implicit function evaluation facilitates efficient collision detection between objects [11].

An implicit function based model for a geometric skin with a physical interpretation for a muscle model [12], shows the effective use of implicit surfaces in modeling and animating human muscle and skin. Implicit surface representations, unfortunately, have inefficient display characteristics. To realize a teleconferencing system of the complexity and realism desired, with current computing resources, it is thus imperative that we make use of polygon based representations, which have extensive hardware support.

This paper presents an implicit function based muscle model that deforms a polymesh geometric skin. The model possesses the advantages of implicit functions [12] as well as the display efficiency of a polygon based structure. Its modular nature makes it simple to integrate into existing systems and unify with other polymesh based techniques [2],[6].

The physical characteristics of objects are separated into **rigid** and **deformable** components. The implicit function based deformable component performs all physically based deformations including efficient collision detection and handling. For human figures, a geometric polymesh skin is first synthesized using digitized limbs blended together using implicit functions. The skin is then embedded in a hierarchy of implicit functions that model muscles. These implicit functions interact with one another and with other implicit functions in the environment, to model blends and collision deformations. Human figures (and other objects) are animated by rigid body transformation of the polymesh model on a skeletal structure and subsequent deformation based on contributing implicit functions (See Figure 4).

Section 2 describes the synthesis of the polymesh skin model and its animation as an articulated rigid body. A muscle model embedding the polymesh skin in a hierarchy of implicit functions is proposed in Section 3. Section 4 provides the working details of the deformable component. The interaction of the implicit functions for blending and to efficiently detect and model collision deformations on the underlying polymesh model is presented. Section 5 describes the implementation of the model within *VISTEL*. Section 6 presents conclusions and scope for future work.

## 2 Human Polymesh Model : Synthesis and Animation

**Synthesis** of polymesh models of real humans is a nontrivial problem. Various reconstruction methods, using sculpted models, range data, photographic images have been proposed [9]. In our approach a number of polymesh parts corresponding to various limbs are obtained, conveniently sought using a Cyberware digitizer [8]. These parts are then fitted together, which may be done by lofting between the end contours of segments. We choose, however, to blend the parts, juxtaposed in space [12]. This better preserves the overall length and shape of the limbs. Further, control over the region of the blend can help automatically attenuate glitches and noise in the scanned data that often results at the fringes.

Results are shown on a polygonized elbow in Figure 3. The upper and lower digitized limbs are treated as implicit surface primitives and blended together as described in [12]. A user controllable region is then defined within which the blended data is polygonized and seamlessly connected to the existing polymesh struc-

ture of the limbs. Color textures (skin and clothing) for the polygonized region maybe obtained by blending the two limb textures as shown in Figure 3. This provides a polymesh prototype representing the human in some prespecified skeletal posture.

**Animation** of the polymesh model based on motion of the underlying skeletal structure must then be specified. Posture computation for teleconferencing is in itself a nontrivial problem [8]. For this paper we assume skeletal posture computation to some reasonable level of accuracy.

Rigid bodies are well represented by polymesh models and can be animated efficiently and robustly. Various techniques, such as springs and dampers [15], free form deformations [2], and implicit primitives [10], [3] have been used to model deformable objects. This paper separates the physical characteristics of objects into **rigid** and **deformable** components that may be applied successively [3]. Specifically, the rigid component is determined by a skeletal structure; the deformable component by the interaction with other objects in the environment. This model is particularly well suited to human figure animation. We wish to model the deformable component using implicit functions, which facilitate an efficient computation of interaction of the deformable body with the environment.

For the rigid component it suffices to subject the polymesh model to the translations and rotations specified by the underlying skeleton. Care is taken to preserve connectivity and continuity around joints. This may be done using free form deformations [2]. Alternatively the quarternion based rotation around a joint is interpolated for vertices in the polygonized region (See Figure 3) around a joint [13]. This maintains smooth connectivity around joints but the animated model resembles a flexible pipe (See Figure 8).

The next two sections of the paper develop a model for the deformable component, that is applied to the polymesh model after the rigid component transformation described above. Figure 4 shows the overall perspective of such a system.

# 3 Deformable Model : Synthesis

The deformable model immerses the polymesh model in a hierarchy of functions, specified by a number of implicit primitives [18]. The vertices of the polymesh prototype are then calibrated based on their implicit function values, so that they all lie on some convenient implicit surface. During animation the vertices are transformed appropriately to stay on that implicit surface. The next subsection describes implicit function concepts crucial to the understanding of the proposed model.

## 3.1 Implicit Function primitives

A useful set of implicit surfaces can be generated as an algebraic combination of polynomial functions each of which is defined over a finite volume. For a summation (smooth blend) $F(P) = \sum F_i(P) - T$, where $i$ runs over the primitive polynomial functions $F_i$ and $T$ is a threshold value $\in [0,1]$. A subset of these surfaces are **distance surfaces** and **convolution surfaces** [1]. Each primitive is defined by a finite volume $V$ (typically spherical), a skeleton within the volume $S$ (typically the center of the sphere) and a function $f$ (typically a polynomial as in Figure 1). The primitive only contributes within $V$. An example of $f : [0,1] \rightarrow [0,1]$, also called a density function, with the desired properties [18] is shown in Figure 1. For a point $P$ within $V$ the function value is determined by first computing a value $\in [0,1]$ called a distance ratio of $P$. $F(P) = f(distanceratio(P))$ is the function value for the primitive. The distance ratio is computed by taking the ratio of the shortest euclidean distance from $P$ to a point $Q$ on $S$, and a value determined by the shape of $V$ (usually a constant radius value $R$). Such a primitive (See Figure 1) defines an **offset surface** [1].

Different objects have their own implicit functions that determine the object surface. These objects may interact by imparting additional implicit functions to one another [3]. The functions imparted can model interactions such as collision deformations as described in Section 4.1 (See Figure 6,8,9), fusion and fission [12].

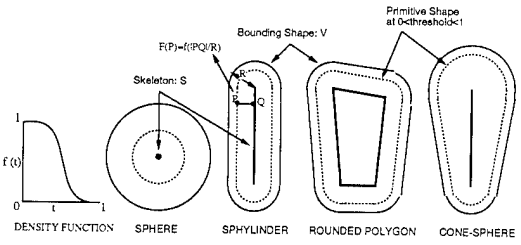



Figure 1: Implicit Primitive Shapes

## 3.2 Implicit Primitive Model

An implicit primitive is attached to every skeletal limb and embeds the corresponding polymesh model around it. A convenient threshold value $T$ (See Section 3.1) is chosen at which the implicit surface deforming the polymesh model will be sought. The choice of $T$ is influenced by the deformable characteristics of the

object being modeled.

The shape of the implicit primitive [18] provides a bounded volume for the realm of influence of the associated function, as well as a mapping from a point within the volume to a value within the domain of the density function (to calculate the function at that point) [13].

The choice of the implicit primitive shape for an object (limb) is influenced by the following requirements :

1. Implicit function computation should be efficient.

2. The primitive shape at some threshold value should fit the embedded region of the polymesh model well [7]. The behavior of the polymesh region then closely follows that of the surface defined by the implicit primitive at that threshold.

3. The primitive bounding volume around the embedded polymesh region should be reasonably tight, so as to avoid wasted polymesh deformation computation on interaction with the environment.

4. Bounding volume intersection computation between the primitives used should be efficient, so as to determine primitives in the environment with which a given primitive interacts.

Primitive volumes satisfying these requirements well for different human parts are offset surfaces (spheres, sphylinders, offset polygons) and conespheres [1],[14] (See Figure 1). Efficient computation methods for these primitives are described in detail in [14]. The choice of primitive shapes for various parts of the body is fairly intuitive and shown in Figure 2.

Figure 2 also shows the hierarchical implicit primitive structure used for the human figure. The implicit primitives are combined selectively as specified in [12] using a combination graph. Primitives that do not blend mutually, are treated as different implicit objects in the environment, that interact with each other for collision detection and deformation. Thus auto-collisions between different parts of the body are homogeneously handled along with any other implicit objects in the environment. Joint regions such as the elbow where both blending and collision deformations occur, are handled by the introduction of a dummy primitive at the joint that blends the two arm primitives together (See Figure 8).

Human bones on animation, often come close to the skin surface and define the shape of the skin [5]. This is a major shortcoming of virtual point-linked skeletons

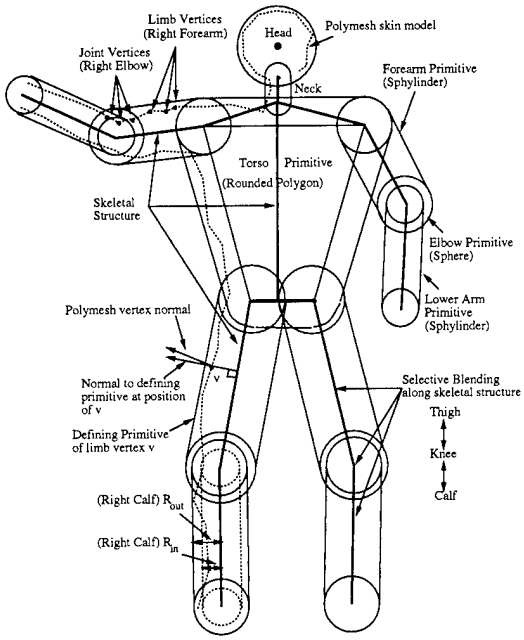



Figure 2: Implicit Primitive Hierarchy

and is handled elegantly by the implicit model [12]. Skeletal implicit primitives are specified and blended with the geometric skin primitives. On animation, the skeletal primitives contribute to the shape of the skin automatically, only when the primitive is close to the geometric surface primitive (See Figure 8). Other features like veins and wrinkles may be similarly incorporated [12].

### 3.3 Implicit Primitive Synthesis

Once an appropriate primitive shape for a limb or joint is selected, it must be fitted to the underlying polymesh data. The polymesh model synthesis, described in Section 2, partitions vertices into those belonging to different limbs and polygonized regions corresponding to joints. Vertices are henceforth referred to as **limb vertices** or **joint vertices**. Limb vertices contribute to the fitting of the corresponding limb primitive. Joint vertices contribute to the joint primitive and the primitives of the limbs joined. The primitives which a vertex contributes to are also called the **defining primitives** for the vertex and are largely responsible for its deformation on animation.

For offset surface primitives, skeletal joint centers are used to position the primitive skeleton $S$, as shown in Figure 2. Two radii $R_{out}$ and $R_{in}$ are then calculated for each offset primitive. These are based on the distances of contributing polymesh ver-

tices from the primitive skeleton $S$. $R_{out}$ is the maximum and $R_{in}$ the average of distances of contributing vertices from $S$. $R_{out}$ is the bounding radius of the primitive and $R_{in}$ the primitive of best fit. $W = T/f(R_{in}/R_{out})$ is the shape weight for the primitive, where $f$ is the associated density function. The function value for primitive $i$ at a point $P$ is then $F_i(P) = W_i * f_i(distanceratio(P))$. Thus the shape at threshold $T$ of the primitive is the offset surface with radius $R_{in}$. Figure 5 shows primitives fitted to a human polymesh model. Bounding volumes for the right half of the body and primitives of best fit for the left half are shown. Complex primitives such as conespheres or superquadrics may be fitted as described in [7],[10],[13].

Once the implicit primitives have been synthesized, the polymesh model needs to be calibrated. Here a weight attribute for each vertex is calculated, ensuring that the implicit function model produces no deformation to the polymesh prototype for the prespecified posture, in the absence of any environmental interaction.

### 3.4 Polymesh Model Calibration

For a vertex $v$ on the prototype polymesh, let $D_v(P)$ be the implicit function computed based on its defining primitive(s) at a point $P$ in space. $v$ is calibrated by assigning a weight $w_v = T/D_v(P)$, where $P$ is the spatial position of the vertex on the prototype polymesh. In the absence of environmental interaction the implicit function for the vertex at $P$ is, $F(P) = w_v * D_v(P) = T$, ensuring that the polymesh prototype for the prespecified posture lies on the implicit surface determined by the implicit model at threshold $T$.

This completes the construction of the implicit model and polymesh calibration for the human figure model. A similar approach may be taken for any polymesh object, articulated or otherwise, as long as implicit primitive(s), with the required attributes, can be sought.

## 4 Deformable Model: Animation

The result of the rigid component transformations to the polymesh prototype and implicit primitives based on the skeletal posture, described in Section 2, forms the input to the deformable component model (See Figure 4).

The individually manipulated implicit primitives automatically maintain a smoothly blended body as well as collision deformations [12]. These functions then appropriately shape the embedded polymesh model. Deformation of the polymesh models is carried out by deforming the position of each vertex of the model from its current position $P$ to a point $P'$, such that $F(P') = T$.

### 4.1 Implicit Function Computation at Polymesh Vertices

The implicit function for a vertex $v$ at point $P$, based on its defining primitive(s), $D_v(P)$ is computed as follows :

**Limb Vertex**: $D_v(P) = F_i(P)$, where $i$ is the defining limb primitive.

**Joint Vertex**: $D_v(P) = F_i(P) + DIFF(F_j(P), F_k(P))$, where $i$ is the joint primitive and $j$, $k$ the limb primitives. The $DIFF$ function models the formation of creases at joints (See Figure 8). An example of $DIFF$ is $DIFF(a,b) = |a^n - b^n|^{1/n}$.

The implicit function for $v$ at $P$ is then $F(P) = w_v * D_v(P) + \sum_i COLL_{i,v}(P)$, where $i$ runs over all interacting primitives (where $F_i(P) > 0$).

$COLL_{i,v}(P)$ is the collision deformation function [3] imparted by primitive $i$ with a minor difference:
**Penetration zone**: $COLL_{i,v}(P) = T - w_v * F_i(P)$.
**Propagation zone**: $COLL_{i,v}(P) = w_v * Prop_i(P)$, where $Prop_i(P)$ is the propagation function in [3].

Both $DIFF$ and $COLL$ are derived from collision deformation concepts presented in [3]. Precise mathematical examples and details on the behavior of these functions may be found in [13]. A modification to the function handles degrees of relative deformability of colliding objects [13]. Temporal modification of the functions handle various elasticity characteristics [13] (See Figure 9). Multiple objects mutually colliding are handled homogeneously [3].

The vertex $v$ is constrained to a point $P$, where $F(P) = T$, or for example in the presence of a colliding primitive $i$, the common collision contact surface where $D_v(P) = F_i(P)$. Proof of correctness of the generated collision contact surfaces may be found in [3],[13].

### 4.2 Polymesh Deformation

Having laid the theoretical foundation that deforms the polymesh model, we address the algorithmic aspects for a practical implementation. The algorithm for deformable component transformation of objects is carried out in 3 steps as follows:

1. A list of interacting objects is constructed for each object in the environment.

2. The vertices of the polymesh model of each object are then deformed based on function values

computed using its defining primitive(s) and all objects the defining primitive(s) interact with.

3. Further processing using the deformed polymeshes.

*Step 1* benefits from efficient intersection computation of the bounding volumes of primitive shapes. It is an optimization step based on spatial coherence that does not affect the polymesh deformation in *Step 2*. A simple analytic solution [14] exists for intersection testing between simple shapes like offset surfaces, which may be exhaustively intersected with each other. Spatial subdivision techniques such as octrees may also be used.

*Step 2* is the crux of the algorithm. Every vertex of a polymesh object must now be deformed based on an implicit function $F$. This function is specific to the vertex and is calculated as described in Section 4.1. The position of this vertex must be deformed from its position $P$ computed by the rigid component transformation, to a point $P'$, such that $F(P') = T$. As the implicit function defines a continuous implicit surface in the neighbourhood of $P$, the deformation mapping of $P$ to $P'$ for the vertex is ill-defined. This is an artifact of using a discrete polymesh representation to follow the deformations of a continuous implicit representation. The solution proposed is to deform $P$ along its vertex normal. Alternatively, $P$ maybe deformed along $\nabla F(P)$ or the normal to the implicit function at that point (See Figure 2).

For offset primitives, the distance ratio function for any point along a ray can be represented in terms of its parametric distance along the ray [14]. Thus $P'$ may be obtained analytically along any ray, by solving a sequence of quartic equations [14]. Alternatively, a hybrid Newton- Raphson, Regula Falsi iterative technique [14] may be used which is efficient in this case, as the deformations, being incremental are small.

*Step 3* deals with the computation of surface normals or other spatial attributes of the polymesh or any desired parameters based on the deformed polymesh. Additionally, for systems with force feedback, integration of forces at individual vertices of the object and collision contact area computation may be done in this step.

The worst case complexity of the above algorithm is $O(m^2+mn)$, where $m$ is the number of implicit primitives and $n$ the number of polymesh vertices in the environment ($n >> m$). Despite fast octree based techniques for coarse collision detection, precise detection and handling using conventional polygon methods is $O(n^2)$ in the worst case [11],[4], making our approach superior as the complexity of virtual worlds increase.

### 4.3 Dynamics

A physical muscle interpretation, which in conjunction with the skeleton controls the animation of the geometric model, is presented in [12]. Variations in tissue characteristics [16] are modeled by piecewise smooth density polynomials whose gradient reflects the change in stiffness, making reaction force computation for a vertex simple [13]. Area computation of polymesh faces whose vertices are deformed to lie on a collision contact surface [3] may be done in *Step 3* of the algorithm in Section 4.2. These areas can then be used in the computation of area dependent reaction and friction forces completing a force feedback loop.

## 5 Implementation

Figure 7 shows the *VISTEL* system in operation. Posture computation employs 4 Fastrak magnetic sensors (head, chest and wrists) and cyber gloves on each participant. SGI (Onyx, Reality Engine) machines perform graphics processing and display at each site. Real time 3D display of the synthesized virtual space is then projected on a 70-inch stereoscopic display.

Human figure data is obtained as a number of digitized parts using a Cyberware Color 3D digitizer. The polymesh parts are registered with color textures and fitted together on a virtual skeleton, using implicit function blending (See Figure 3) [12] or other techniques.

The implicit primitives are then constructed on the figure as shown in Figure 5. The human figure polymesh ($\approx$ 7500 vertices) is embedded in 23 implicit primitives. The implicit model is constructed hierarchically, in an object oriented fashion, making the fitting of other polymesh objects as well as the introduction of new implicit primitive shapes, a simple task.

Figure 8 shows the elbow region after rigid component transformation and its subsequent deformable component transformation. Spherical primitives modeling skeletal elbows cause the elbow to protrude in the bent arm and precise crease formation may be seen. The deformable component computation using a Regula Falsi-Newton Raphson approach typically takes 2-3 iterations per vertex.

Figure 6 shows deformations of an arm and a ball on collision. Figure 9 shows a ball bouncing off a head. The head is given a plastic attribute and the ball a viscoelastic one so as to clearly see the deformations that result.

Existing polymesh based muscle and skin modeling techniques may easily be integrated with the rigid component in our implementation. As an example

FFDs [2] on the spine, animate the torso (See Figure 5). Functional calculations for torso vertices, with respect to the torso implicit primitive, employ the undeformed spatial position in the calculations.

## 6 Conclusion

To summarize, this paper describes a model for the synthesis and animation of objects with a polymesh representation. The physical characteristics of the object are separated into rigid and deformable components. The implicit function based deformable component performs optimal collision detection and handling. The model is catered to address issues involved in the modeling and animation of the human muscle and skin layer within a virtual space teleconferencing system.

The implementation of the presented concepts shows their effectiveness both in terms of computation speed and the degree of realism obtained. The separation of the physical characteristics of objects into rigid and deformable components, works particularly well for human figures. The model handles auto-collisions of the body and skeletally based deformations (bone, crease in Figure 8) elegantly. Attribution of physical characteristics can be overlaid on the model for dynamic simulations [13].

The ability to apply implicit function techniques in general to existing polygon based data is an important advantage of our approach. It can unify and be integrated with existing polygon based or implicit surface based modeling and animation systems.

The approach achieves linear time complexity in terms of number of object vertices for collision detection and handling, which is important when dealing with complex virtual worlds [11].

The underconstrained nature of the deformable component mapping of polymesh vertices may cause surface consistency problems. A dense human figure polymesh and the radial nature of the limbs and primitives, causes simple displacements along vertex normals to give good results without vertices bunching together or diverging abnormally. Incorporation of techniques such as [17], that adaptively subdivide and coalesce the polymesh in real time are subject to current research.

There is scope for future work on construction and fitting of primitives to the polymesh. Poor fitting primitives may result in very close objects being deformed to abut at the implicit model contact surface. For *VISTEL* and other applications where visual realism dominates over spatial accuracy (we assume a tolerable inaccuracy in the transition from real to virtual space), the above artifact does not pose a problem. Using a greater number and more complex primitives improves the fit but degrades implicit function and bounding volume intersection computation efficiency. An empirical tradeoff between a better fit and computation efficiency should therefore, be taken into consideration.

## References


[1] J. Bloomenthal and K. Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, 1991.

[2] J. Chadwick, D. Haumann and R. Parent. Layered construction for deformable animated characters. *Computer Graphics*, 23(3):234–243, 1989.

[3] M. Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Proc. of SIGGRAPH*, 313–320, 1993.

[4] J. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, 1988.

[5] N. Magnetat-Thalmann and D. Thalmann. Complex models for visualising humans, *SIGGRAPH Course Notes C20*, 3–10, 1991.

[6] N. Magnetat-Thalmann and D. Thalmann. Human body deformations using Joint Dependent Local Operators and Finite Element Theory. *Making Them Move*, Morgan Kaufmann, 243–262.

[7] S. Muraki. Volumetric shape description of range data using the blobby model. *Computer Graphics*, 23(3):234–243, 1991.

[8] J. Ohya, Y. Kitamura, H. Takemura, F. Kishino and N. Terashima. Real-time reproduction of 3D human images in Virtual Space Teleconferencing. *VRAIS*, 408–414, 1993.

[9] A. Paouri, N. Magnetat-Thalmann and D. Thalmann. Creating realistic 3D human shape characters for computer generated films. *SIGGRAPH Course Notes C20*, 18–29, 1991.

[10] S. Sclaroff and A. Pentland. Generalized implicit functions for computer graphics. *Computer Graphics*, 25(4):247–250, 1991.

[11] A. Pentland. Computational complexity versus virtual worlds. *Computer Graphics*, 24(2):185–192, 1990.

[12] K. Singh, J. Ohya and F. Kishino. Realistic modeling and animation of a muscle and skin layer for human figures using implicit function techniques. *IPS Graphics and CAD Conference*, 49–56, 1994.

[13] K. Singh. Human Figure Synthesis and Animation for Virtual Space Teleconferencing. *ATR Technical Report*, 1994.

[14] K. Singh and R. Parent. Fast scanline processing of useful implicitly-defined shapes: Sphylinders, cone-spheres and rounded polygons. *Ohio State Univ. Tech. Rep. OSU-CISRC-5/94-TR26*, 1994.

[15] D. Terzopoulos and K. Fleischer. Deformable Models. *Visual Computer*, 4:306–331, 1988.

[16] D. Terzopoulos and K. Waters. Physically based facial modeling, analysis and animation. *Journal of Visualization and Computer Animation*, Vol.1, No.2, 73–79, 1990

[17] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. *Proc. of SIGGRAPH*, 269–278, 1994.

[18] G. Wyvill C. McPheeters and B. Wyvill. Data structures for soft objects. *Visual Computer*, 2:227–234, 1986.

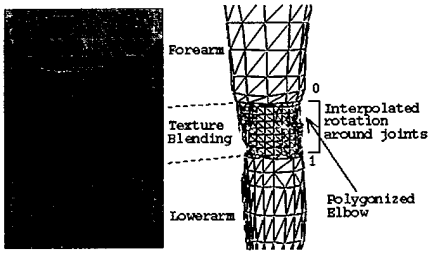



Figure 3: Human Figure Polymesh Synthesis

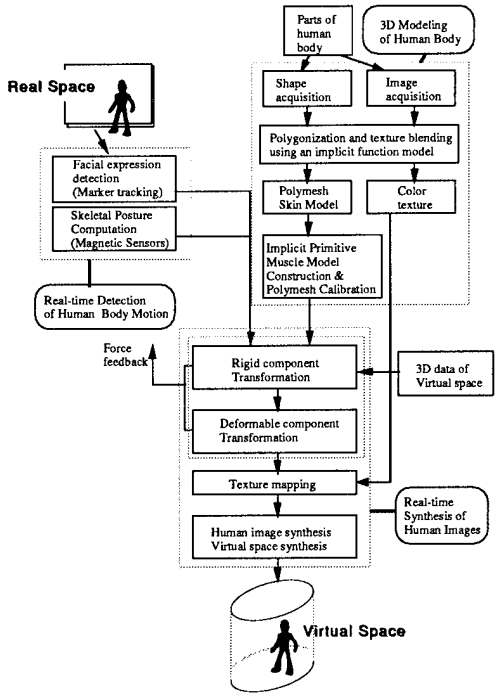



Figure 4: VISTEL Framework

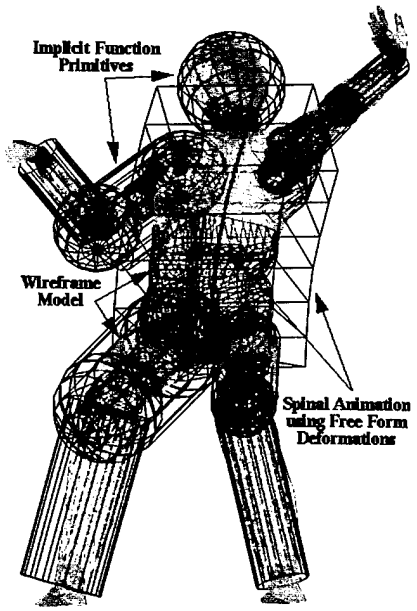



Figure 5: Implicit Primitive Synthesis

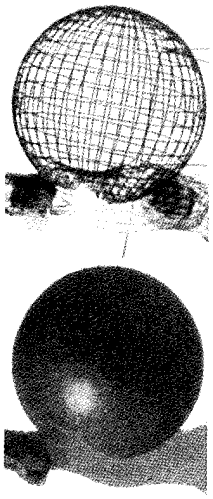


Figure 6: Collision Deformation

Figure 7: VISTEL System

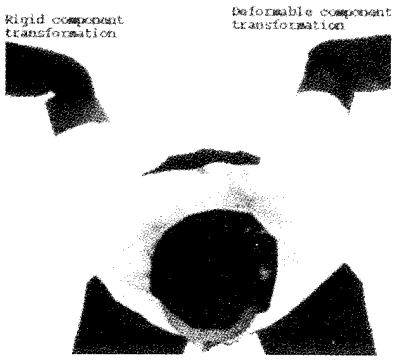



Figure 8: Rigid, Deformable Component Transformation

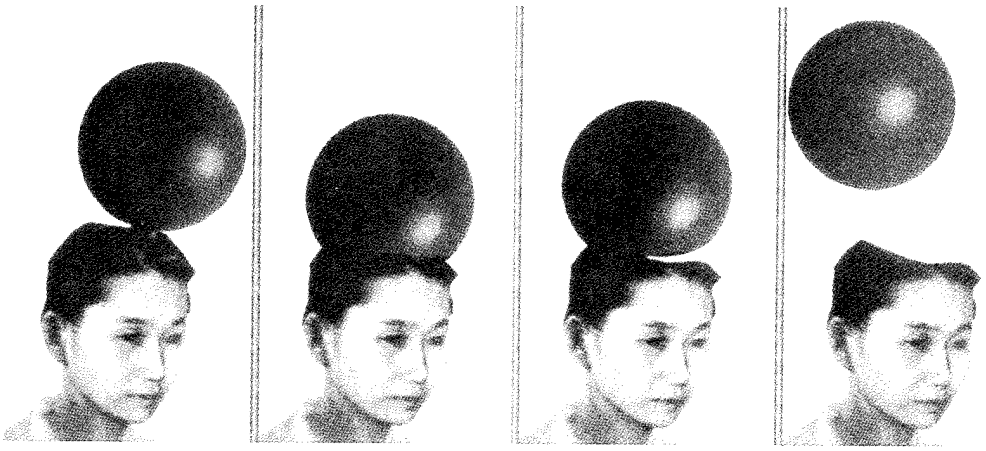


Figure 9: Temporal Collision Deformations (Head: plastic, Ball: viscoelastic)

# Production and Playback of Human Figure Motion for 3D Virtual Environments

John P. Granieri, Jonathan Crabtree, Norman I. Badler

Center for Human Modeling and Simulation
University of Pennsylvania
Philadelphia, PA 19104-6389, USA
granieri|crabtree|badler@graphics.cis.upenn.edu

## Abstract

*We describe a system for off-line production and real-time playback of motion for articulated human figures in 3D virtual environments. The key notions are (1) the logical storage of full-body motion in posture graphs, which provides a simple motion access method for playback, and (2) mapping the motions of higher DOF figures to lower DOF figures using slaving to provide human models at several levels of detail, both in geometry and articulation, for later playback. We present our system in the context of a simple problem: Animating human figures in a distributed simulation, using DIS protocols for communicating the human state information. We also discuss several related techniques for real-time animation of articulated figures in visual simulation.*

## 1 Introduction

The ability to render realistic motion is an essential part of many virtual environment applications. Nowhere is this more true than in virtual worlds containing simulated humans. Whether these human figures represent the users' virtual personae (avatars) or computer-controlled characters, people's innate sensitivity as to what looks "natural" with respect to human motion demands, at the very least, that moving characters be updated with each new frame that the image generator produces.

We first discuss a topical problem which requires the real-time rendering of realistic human motion, and then describe our system for authoring the motion offline, and playing back that motion in real time. We also address some of the issues in real-time image generation of highly-articulated figures, as well as compare several other methods used for real-time animation.

## 2 Human motion in DIS

The problem we are interested in is generating and displaying motion for human figures, in particular soldiers, in distributed virtual environments. Parts of the general problem and the need for representing simulated soldiers (referred to as Dismounted Infantry, or DIs), are covered in [15, 5]. Although primarily driven by military requirements today, the general technologies for projecting real humans into, and representing simulated humans within, virtual environments, should be widely applicable in industry, entertainment and commerce in the near future.

The Distributed Interactive Simulation (DIS) [7] protocol is used for defining and communicating human state information in the distributed virtual environment. The DIS protocol, at least the part relating to human entities, is in its early stages of development, and fairly limited in what it can specify about a human figure [11], but is a good baseline to start with. Our purpose here is not to engage in a discussion of the intricacies (nor worth) of the DIS protocol, but merely to use it as an example of a distributed simulation protocol which can communicate state information on a simulated human entity between simulation nodes in a network.

The information representing a human entity is currently defined by several discrete enumerations in the appearance field of an Entity State Protocol Data Unit (PDU) in the DIS protocol [8]. The relevant information we are interested in from the Entity State PDU is shown in Fig. 1. The human is always in one of the four postures, along with a weapon state. The heading defines the forward direction. Although there are enumerations for walking and crawling, we use combinations, such as (posture=*standing*)+(velocity>0) to be equivalent to walking or running. Although the protocol allows for up to three weapons of different types on a soldier, we only modeled one, a rifle.

If the human can be in any of $n$ possible postures, there are potentially $n^2$ transitions between the postures. Rather than create $n^2$ posture transitions, we encode the postures and transitions into a *posture graph* [1]. The graph defines the motion path to traverse to move the human figure from any one posture to another. These graphs are directed and may include cycles. It also provides the logical structure for the run-time motion database.

When the velocity of the human is zero, the possible transitions between *static* (for lack of a better term) postures are encoded in the posture graph of Fig. 2. A few of the actual postures are shown in Fig. 3. In



0-8186-7084-3/95 $04.00 

| Field | Value | Units |
|---|---|---|
| Posture | *Standing*<br>*Kneeling*<br>*Prone*<br>*Dead* | n/a |
| Weapon | *Deployed*<br>*Firing* | n/a |
| Position | $P_x$, $P_y$, $P_z$ | meters |
| Velocity | $V_x$, $V_y$, $V_z$ | meters/second |
| Heading | *theta* | degrees |

Figure 1: Essential human state information in a DIS Entity State PDU

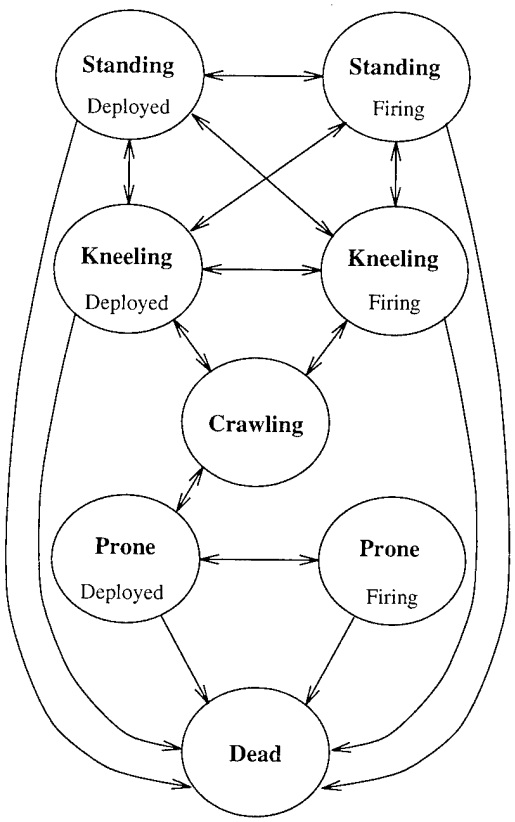



Figure 2: The *static* posture graph

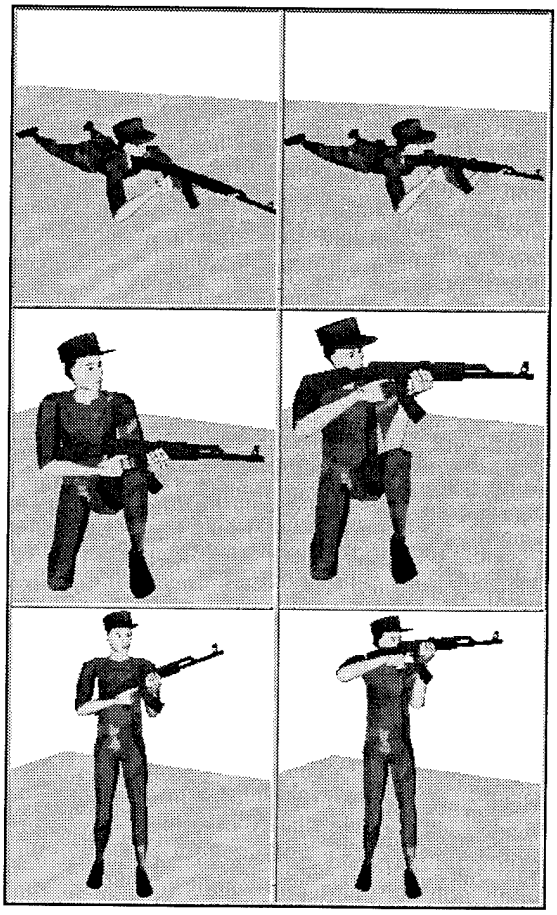


Figure 3: Some of the static postures a soldier can take in DIS

the posture graph, the nodes represent static postures, and the directed arcs represent the animated full-body transitions, or movements, from posture to posture. Each arc has an associated time for traversal, which is used to find the shortest path, in time, if more than one path exists between a starting posture and a goal posture.

When the velocity of the figure is non-zero, the possible transitions between locomotion postures are shown in the posture graph of Fig. 4. In this graph, the nodes are static postures, but the figure would never be in the posture for more than one frame.

The system we built consists of two distinct parts: 1) the off-line motion data generator, and 2) the on-line real-time playback mechanism, running in a high-performance IRIS Performer-based [12] image generator application.

## 3 Off-line motion production

Motion production involves three steps: 1) creating postures and motion for each node and arc in a posture graph, for one human model, 2) mapping the resulting motion onto human models with lower degrees-of-freedom (DOF) and lower resolution geometry, and

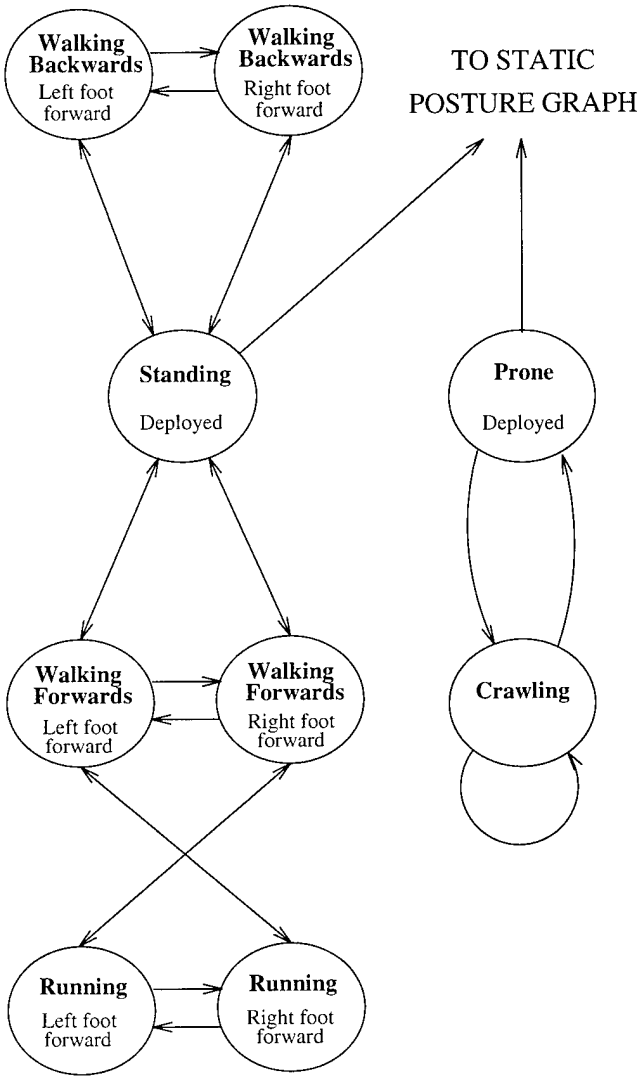



Figure 4: The *locomotion* posture graph

finally 3) recording the results and storing in a format for easy retrieval during playback.

### 3.1 Authoring the motion

The first step in producing motion for real-time playback is to create postures representing the nodes in the posture graphs, as well as the corresponding motions between them, represented as the directed arcs in the graphs. We used a slightly modified version of the *Jack* human modeling and animation system [2] for this purpose. *Jack* provides a nice constraint-based, goal-driven system (relying heavily on inverse-kinematics and primitive "behavioral" controls) for animating human figures, as well as facilities for organizing motions for general posture interpolation [1]. It is important to note that the posture graphs presented in this paper differ from the *posture transition graphs* presented in [1]. In the latter, the posture transition graphs are used to organize motion primitives for general posture interpolation with collision avoidance. In the former application (this paper) the posture graphs are a logical mechanism for organizing a database of pre-recorded motion, and determining motion sequences as paths between nodes of the graph. An underlying assumption of the posture graphs is that the articulated human figure's motion is continuous, and therefore can be organized into a connected graph.

Each directed transition in the static posture graph typically was produced from 10 to 15 motion primitives (e.g. `move_arm`, `bend_torso`, etc). Many of the directed motions from a posture node A to a posture node B are simply run in reverse to get the corresponding motion from posture B to posture A. In several cases, the reverse motion was scripted explicitly for more natural results.

The human figure can also move (either forwards or backwards, depending on the difference between the heading and the direction of the velocity vector) by either locomoting (if posture is standing) or crawling (if posture is prone). The locomotion posture graph transitions of Fig. 4 were generated by Hyeongseok Ko's walking system [9]. Six strides for each type of walking transition were generated (forward walking, backward walking, running): left and right starting steps, left and right ending steps, and left and right cyclic steps. The crawling animation was generated manually, and is based on two animations - one that goes from the prone posture to the cyclic state, and one complete cyclic motion. Note that only straight line locomotion of fixed stride is modeled. We are currently working on extending the system to handle variable stride length and curved path locomotion, as possible in the system of [9].

### 3.2 Slaving

The second step in the production process is concerned with preparing the motion for the real-time playback system. We wish to have tens, and potentially hundreds, of simulated humans in a virtual environment. This neccesitates having multiple level-of-detail (LOD) models, where the higher resolution models can be rendered when close to the viewpoint, and lower resolution models can be used when farther

| | human–1 | human–2 | human–3 |
|---|---|---|---|
| polygons | 2400 | 500 | 120 |
| rigid segments | 69 | 19 | 12 |
| joints | 73 | 17 | 11 |
| DOFs | 134 | 50 | 21 |
| motion | 60Hz | 30Hz | 15Hz |

Figure 5: The different levels of detail for the human models

away. We reduce the level of detail in the geometry and articulation by creating lower-resolution (both in geometry and articulation) human figures, with the characteristics listed in the table of Fig. 5.

All the motions and postures of the first step are authored on a (relatively) high resolution human body model which includes fully articulated hands and spine. This model is referred to as "human–1" in the above table. We manually created the two lower-resolution models, human–2 and human–3. Because of the difference in internal joint structure between human–1 and the lower LOD models, their motions cannot be controlled by the available human control routines in *Jack* (which all make assumptions about the structure of the human figure - they assume a structure similar to human–1). Instead of controlling their motion directly, we use the motion scripts generated in the first step to control the motion of a human–1, and then map the motion onto the lower resolution human–2 and human–3. We call this process *slaving*, because the high resolution figure acts as the *master*, and the low resolution figure acts as the *slave*.

Even though the different human models have different internal joint structures and segment shapes, their gross dimensions (e.g., length of arms, torso, etc.) are similar. The slaving process consists of interpolating the motions for the full human figure, generating all the in-between frames, and simultaneously having a lower LOD human model (human–2 or human–3) slaved, and then saving the in-between frames for the soldier. We will describe the process used for slaving from human–1 to human–2; the case with human–3 is similar.

For each frame of an animation, we first compute the position and joint angles for human–1. Then, an approximation of the joint angles for human–2 are computed. This is straightforward, as certain joints are the same (the elbow, for example, is only one DOF on both human models), and others can be approximated by linear combinations (for example, the 35 DOFs of the spine on human–1 can be summed and mapped directly onto the 7 DOF torso of human–2). This gives a good first approximation of the posture mapping, and provides an initial configuration for the final mapping. For the resulting motion of human–2 to look correct, we need to have certain landmark sites of the two bodies match exactly (the hands must be on the rifle). The final mapping step involves solving a set of constraints (point-to-point and orientation), to bring the key landmark sites into alignment. The

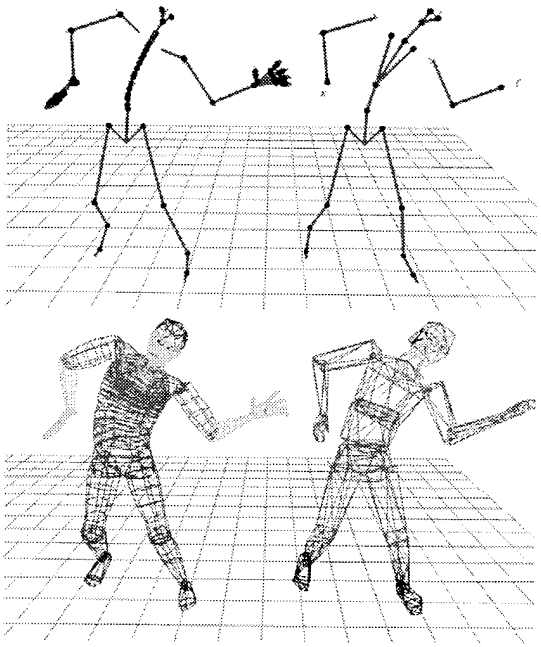


Figure 6: human–1 and human–2 models during slaving. human–1 is the master. Upper window is the skeletal articulation. Models are offset for illustrative purposes.

constraints are solved using an iterative inverse kinematics routine [17] to move the body parts into alignment.

Because of differences in geometry between the master and slave, in general we cannot expect all the landmark sites to match exactly. For the problem domain of this paper, animating the DIS protocol, the hands are always holding a rifle, so matching the hand positions accurately from the master is very important (otherwise the slave's hands may penetrate the rifle). Using a priority scheme in evaluating constraints, we assign higher priority to the hand-matching constraints than others, to account for this fact. If the slaving procedure cannot fit the master and slave within a certain tolerance, it will generate a warning for the animator.

### 3.3 Recording

The final step in the motion production process is to record the resulting motions of the human figures. The recorded motion for one transition is referred to as a *channel set* (where each joint or figure position is referred to as a *channel*; the channel is indexed by time). For each LOD human figure, a homogeneous transform is recorded, representing figure position relative to a fixed point, and for each joint, the joint angles are recorded (one angle per DOF). Also for joints, the composite joint transform is pre-computed and stored as a 4x4 matrix (which can be plugged directly into the parenting hierarchy of the visual database of the run-time system). Each channel set has an associated

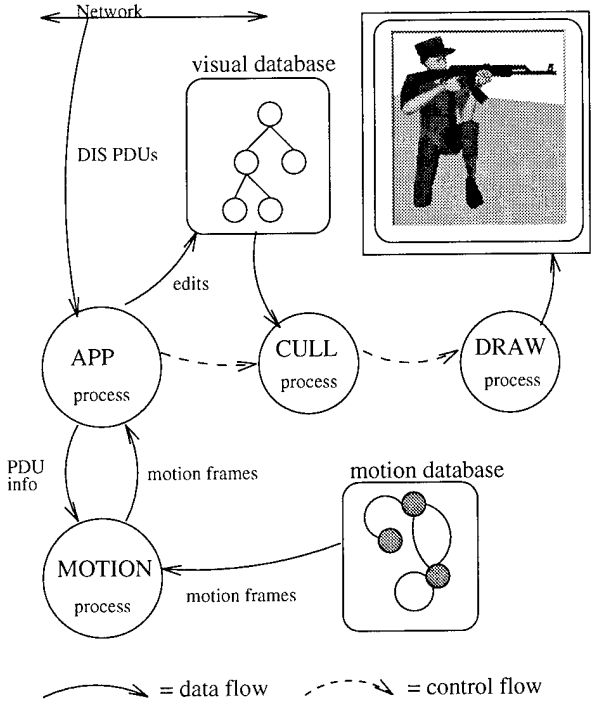



Figure 7: Overview of multi-processing framework for run-time system.

transition time. The channels of human–1 are interpolated and stored at 60Hz, human–2 at 30Hz, and human–3 at 15Hz. These rates correspond to the motion sampling during playback (see below).

## 4 Real-time motion playback

The real-time playback functions are packaged as a single linkable library, intended to be embedded in a host IRIS Performer-based visual simulation application. The library loads the posture graphs shown in Fig. 3 and 4, as well as the associated channel set motion files. Only one set of motions are loaded, and shared amongst any number of soldier figures being managed by the library. The articulated soldier figures themselves are loaded into the Performer runtime visual database. The library runs as a separate process, the MOTION process, serving motion data to the APP process (the APP, CULL and DRAW process are defined in the Performer multiprocessing framework). See Fig. 7 for a schematic overview of the runtime system.

An update function in the APP process is provided which maps joint angle values into the joint transforms of the soldier figures in the Performer visual database.

The APP process sends requests to the MOTION process, and receives joint angle packets back from the library. The content of the request to the library is simply the state information extracted from a DIS Entity State PDU, as shown in Figure 1. A simple control function translates these requests into playbacks of channel sets (the traversal of arcs of the posture graphs).

In the case of a static posture change (a motion from the static posture graph of Figure 2), the system will find the shortest path (as defined by traversal time) between the current and goal postures in the graph, and execute the sequence of transitions. For example, if the posture graph is currently at Standing Deployed, and Prone Firing is requested, it will transition from Stand Deployed to Crawl to Prone Deployed, and finally to Prone Firing.

The same shortest-path traversal method is used for executing posture changes in the locomotion posture graph of Fig. 4. It is important to realize that the only difference between the "static" and "locomotion" posture graphs is conceptual; the data structures involved are identical, and the distinction merely has to do with the conditions under which posture transitions are made. A posture change is made with a node of the static graph as a destination only upon receipt of a DIS Entity State PDU indicating that the agent is in such a posture. In the absence of further information, the agent **remains** in that posture. Conversely, when a posture change is made with a node of the locomotion graph as the destination, something that will occur if a PDU indicates the agent now has a nonzero speed, the agent does **not** remain in that posture once it is reached; absence of further information in this case means that the agent's speed is still nonzero, and hence the agent must take another step, or crawl another meter forwards, or whatever is appropriate for the current mode of locomotion. This continued motion requires that another posture change be made immediately.

One may think of labeling the transition arcs between posture graph nodes with conditions, as in a finite state machine. For instance, the transition from Standing Deployed to Walking Forwards (left foot forward) is taken whenever the agent's speed becomes non-zero and the agent's heading vector agrees with the velocity vector. On the other hand, if the vectors are not pointing in approximately the same direction, a transition is instead made to one of the Walking Backwards states. While the agent's speed remains nonzero (as it is assumed to in the absence of PDU updates), the system continually makes transitions back and forth between, for example, the Walking Forwards (left foot forward) and Walking Forwards (right foot forward) nodes. This cycle of transitions creates a smooth walking motion by concatenating successive left and right steps. Note that since we currently have no cycles of more than two nodes, finding the shortest path between postures in a cycle is a trivial matter!

Crawling is handled similarly, though it is a simpler case; there is no need for separate "left foot forward" and "right foot forward" states.

The system samples all the pre-recorded motion using elapsed time, so it is guaranteed to always play back in real time. For a 2 second posture transition recorded at 60fps, and a current frame rate of the image generator of 20fps, the playback system would play frames $0, 3, 6, ..., 120$. It recomputes the elapsed transition time on every frame, in case the frame rate of the image generator is not uniform.

The motion frame update packets sent from the

MOTION process back to the APP process are packaged to only include those joint angles which have changed from the last update. This is one way we can minimize joint angle updates, and take advantage of frame-to-frame coherence in the stored motions [1]. A full update (all joint angles and figure positions) is about 400 bytes.

### 4.1 Motion level-of-detail

It is recognized that maintaining a constant frame rate is essential to the believability of a simulation, even if it means accepting an update speed bounded by the most complex scene to be rendered. Automatic geometric level-of-detail selection, such as that supported by the IRIS Performer toolkit, is a well-known technique for dynamically responding to graphics load by selecting the version of a model most appropriate to the current viewing context [4, 6, 14].

The LOD selection within the visual database seeks to minimize polygon flow to the rendering pipeline (both in the software CULL and DRAW components of the software pipeline, as well as to the transformation engines within the hardware pipeline).

Given our representation, which enforces the separation of geometry and motion, it is possible to expand level of detail selection into the temporal domain, through *motion level-of-detail* selection. In addition to reducing polygon flow, via selecting lower LOD geometric models, we also are selecting lower LOD articulation models, with fewer articulation matrices, as well as sampling motion at lower frequencies. This reduces the flow of motion updates to the articulation matrices in the visual database. The models we are using are listed in Fig. 3.2.

In the playback system, we simultaneously transition to a different geometric representation with a simpler articulation structure, and switch between stored motions for each articulation model. We gain performance in the image generator, while consuming more run-time storage space for the motions. Our metric for LOD selection is simply the distance to the virtual camera. This appears to work satisfactorily for our current application domain, but further evaluation of the technique, as well as more sophisticated selection metrics (e.g. the metrics described in [6, 4]) need to be explored.

## 5 Example implementations and uses

The real-time playback system is currently being used in two DIS-based applications to create motion for simulated soldiers in virtual environments.

The Team Tactical Engagement Simulator [15] projects one or more soldiers into a virtual environment, where they may engage hostile forces and practice coordinated team activities. See Fig. 8 for a sample view into the training environment. The soldier stands in front of a large projection screen, which is his view into the environment. He has a sensor on his head and one on his weapon. He locomotes through the environment by stepping on a resistive pad and controls direction of movement and field of gaze by turning his head. The soldier may also move off the movement pad, and the view frustum is updated accordingly based on his eye position (head-coupled display). This allows the soldier, for example, to crouch down to see under a parked vehicle, or to peek around the corner of a building while still affording himself the protection of the building. TTES also creates the necessary DIS Entity State PDUs to represent the real soldier (mapping from sensor values into the small set of postures in the Entity State PDU), and sends them out over the net to other TTES stations that are participating in the exercise.

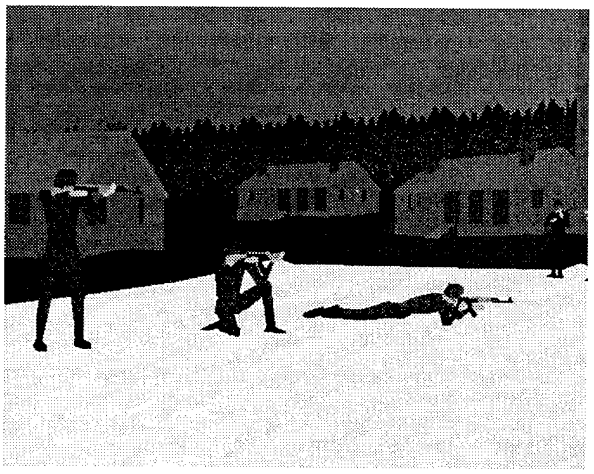


Figure 8: A View of Battle Town with several soldiers in different postures

The playback system is also used in a version of the NPSNET-IV [5] system, for generating motion of SIMNET- and DIS-controlled soldier entities.

Motion level-of-detail selection is of particular relevance to the example projects described above, because in the situation where a hostile agent enters the field of view of a soldier (one of the real human participants) and brings his weapon into the deployed position, the hostile's actions will probably be noted only in the participant's peripheral vision. It is well-known that humans can detect the presence of motion in their peripheral vision very easily, but that resolution of detail is very low. When head-tracking data is available or a head-mounted display is in use it is possible to designate areas of the viewing frustum as peripheral and reduce geometric and motion detail accordingly (not just based on linear distance to the camera, but angular offsets also). In the TTES environment this "focus of attention" information can be obtained from the aim of the real soldier's rifle when it is in the raised position, as the real soldier will almost certainly be sighting in this situation.

[1] An initial implementation of the playback library was run as an independent process, on another machine, from the host image generator, and joint angle packets were sent over TCP/IP stream sockets, hence the desire to minimize net traffic.

## 6 Comparison of production/playback methods

One of the most obvious criteria for evaluating a given motion representation is size; there is a clear progression in the methods used to animate humans (or any entity whose geometric representation varies over time) based on the amount of space required to store a given motion. We look at three methods.

The first method, requiring the most storage, involves generating and rendering the movements of characters in an off-line fashion. Frame-by-frame, a sequence of two-dimensional snapshots is captured and saved for later playback. The image generator then displays the bit-mapped frames in sequence, possibly as texture maps on simple rectangular polygons. Hardware support for texture mapping and alpha blending (for transparent background areas in the texture bitmaps) make this an attractive and fast playback scheme. Furthermore, mip-mapping takes care of level-of-detail management that must be programmed explicitly in other representations. Since the stored images are two-dimensional, it is frequently the case that artists will draw each frame by hand. In fact, this is precisely the approach utilized in most video games for many years. It is clear that very little computation is required at run-time, and that altering the motions incurs a high cost and cannot be done in real time. In fact, modifying almost any parameter except playback speed must be done off-line, and even playback speed adjustments are limited by the recording frequency. However, one real problem with using two-dimensional recording for playback in a three-dimensional scene is that non-symmetric characters will require the generation of several or many sets of frames, one for each possible viewing angle, increasing storage requirements still further. The authors of the popular game DOOM [13] record eight views of each animated character (for each frame) by digitizing pictures of movable models, and at run time the appropriate frames for the current viewing angle (approximately) are pasted onto a polygon. These eight views give a limited number of realistic viewing angles; it is impossible, for instance, to view a DOOM creature from directly above or below. Interestingly enough, an article on plans for a follow-up to DOOM reveals that the authors intend to switch to one of the two remaining representations we describe here:

> Unlike the previous games, the graphic representation of characters will be polygon models with very coarse texture mapping. This will make it hard to emulate natural locomotion, so they'll stay away from creating too many biped characters.[16]

Making the move to the second method involves a relatively slight conceptual change, namely taking 3-dimensional snapshots instead of 2-dimensional snapshots. This means storing each frame of a figure's motion as a full three-dimensional model. Doing so obviates the need for multiple data sets corresponding to multiple viewing positions and shifts slightly more of the computational burden over to the image generator. Instead of drawing pixels on a polygon the run-time system sends three-dimensional polygonal information to the graphics subsystem. It is still an inflexible approach because the figures are stored as solid "lumps" of geometry (albeit textured), from which it is extremely difficult, if not impossible, to extract the articulated parts of which the original model is comprised. Modifications must still be effected off-line, although rendering is done in real time. This is essentially the approach used by the SIMNET image generators to display soldiers on a simulated battlefield [3].

The final method is the one implemented by the system described in this paper, in which we record not the **results** of the motions, but the motions themselves. We store a single **articulated** three-dimensional model of each agent, and from frame to frame record only the joint angles between articulated segments. Modern rendering toolkits such as the IRIS Performer system used in this project increasingly allow support for storing coordinate transformations within a visual database, with relatively little cost associated with updating the transformation matrices in real time. As a result of adopting this approach, storage space is reduced and it is far easier to accurately perform interpolation between key frames because articulation information is not lost during motion recording. It also allows for virtual agents with some motions replayed strictly "as-is" and some motions which may be modified or generated entirely in real time. For instance, the slight changes in shoulder and elbow joint orientation required to alter the aim of a weapon held by a virtual soldier could be generated on demand.

We believe that the smallest representation presented in our size hierarchy, the third method, actually retains the **most** useful information and affords the most flexibility, while placing an acceptable amount of additional computational burden on the run-time display system.

## 7 Extensions & future work

We are currently exploring several extensions to the techniques described above, to add more expressive power to the tool bag of the real-time animator.

**Key-framing and interpolation** The use of the pre-recorded motions in the above posture graphs trades time for space. We do not compute joint angles on the fly, but merely sample stored motions. As the motions become more complex, it becomes very time-consuming to produce all the motions in the off-line phase, so we only produce key frames in a transition, every 5 to 10 frames, and then use simple interpolations to generate the inbetweens during real-time playback. This technique can't be extended much beyond that, as full-body human motion does not interpolate well beyond that many frames. This also reduces the amount of stored motions by a factor proportional to the spacing of the key frames, but increases computation time when a playback frame lands between two key frames.

**Partitioning full-body motion**
In the posture graphs described previously, each motion transition included all the joint angles for the whole body. A technique to reduce motion storage, while increasing playback flexibility, is to partition the body into several regions, and record motion independently for each region. For example, the lower body can be treated separately during locomotion, and the upper body can have a variety of different animations played on it. Also, to support the mapping of motion from partially sensed real humans (i.e. sensors on the hands) onto the animated human figures, we want to animate the lower body and torso separately, then place the hands and arms using a fast inverse kinematics solution.

**Articulation level-of-detail** The various LOD models we used for the human figures were all built manually. Techniques for synthesizing lower LOD geometric models are known, but they don't apply to building lower articulation LOD models. Some techniques for automatically synthesizing the lower articulation skeletal models, given a high resolution skeleton and a set of motions to render, would be very useful.

**Other dynamic properties** A limitation is currently imposed by the fact that the segments of our articulated figures must be rigid. However, this is more an implementation detail than a conceptual problem, since with sufficient computational power in the run-time system real-time segment deformation will become possible. In general it seems likely that the limiting factor in visual simulation systems will continue to be the speed at which the graphics subsystem can actually render geometry. The adoption of coarse-grained multiprocessing techniques [12] will allow such operations as rigid or elastic body deformations to be carried out in parallel as another part of the rendering pipeline. The bottom line is that greater realism in VR environments will not be obtained by pouring off-line CPU time and run-time space into rendering and recording characters in exacting detail; the visual effect of even the most perfectly animated figure is significantly reduced once the viewer recognizes that its movements are **exactly** the same each and every time it does something. We seek to capitalize on the intrinsically dynamic nature of interacting with and in a virtual world by recording less information and allowing motions to be modified on the fly to match the context in which they are replayed. Beginning efforts in this direction may be found in [10].

## 8 Conclusions

We have described a system for off-line production and on-line playback of human figure motion for 3D virtual environments. The techniques employed are straightforward, and build upon several well known software systems and capabilities. As the number of possible states for a simulated human increases, the posture graphs will need to be replaced with a more procedural approach to changing posture. For applications built today on current workstations, the current technique is a balance between performance and realism.

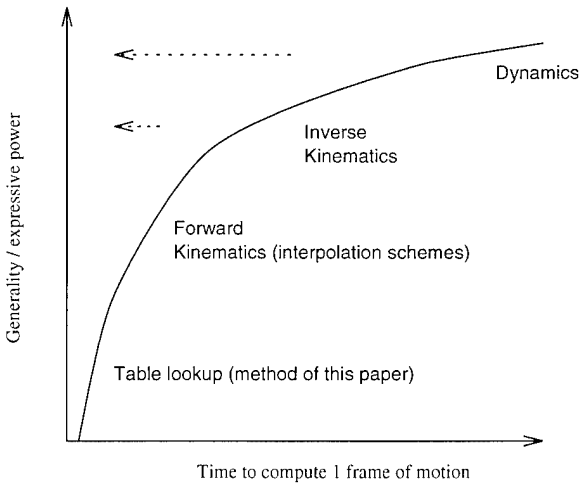



Figure 9: Trade-off between time and generality for motion generation techniques

Figure 9 shows a very coarse, and albeit intuitive, plot of the trade-offs between generality and computation time for several motion generation techniques. For realistic agent animation in virtual environments, the research community will be trying to push this curve to the left, making the more powerful techniques run faster. The curve has been drifting to the left in recent years mainly on the progress made in rendering hardware and overall workstation compute performance.

We chose humans for animating, as they are what we are interested in, but the techniques described in this paper could be applied to other complex articulated figures, whose states can be characterized by postures, and whose motions between postures can be organized into posture graphs.

## Acknowledgments

This research is partially supported by ARO DAAL03-89-C-0031 including U.S. Army Research Laboratory (Aberdeen), Natick Laboratory, and NASA Ames Research Center; U.S. Air Force DEPTH through Hughes Missile Systems F33615-91-C-0001; Naval Training Systems Center N61339-93-M-0843; Sandia Labs AG-6076; DMSO through the University of Iowa; NASA KSC NAG10-0122; MOCO, Inc.; NSF IRI91-17110, CISE CDA88-22719, and Instrumentation and Laboratory Improvement Program #USE-9152503.

## References

[1] Norman I. Badler, Rama Bindiganavale, John Granieri, Susanna Wei, and Xinmin Zhao. Posture interpolation with collision avoidance. In *Proceedings of Computer Animation '94*, Geneva,

Switzerland, May 1994. IEEE Computer Society Press.

[2] Norman I. Badler, Cary B. Phillips, and Bonnie L. Webber. *Simulating Humans: Computer Graphics, Animation, and Control.* Oxford University Press, June 1993.

[3] Jay Banchero. Results to be published on system for dismounted infantry motion in a SIMNET image generator. Topographical Engineering Center, US Army.

[4] Edwin H. Blake. A metric for computing adaptive detail in animated scenes using object-oriented programming. In G. Marechal, editor, *Eurographics '87*, pages 295–307. North-Holland, August 1987.

[5] David R. Pratt et al. Insertion of an Articulated Human into a Networked Virtual Environment. In *Proceedings of the 1994 AI, Simulation and Planning in High Autonomy Systems Conference*, University of Florida, Gainesville, 7–9 December 1994.

[6] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, August 1993.

[7] Institute for Simulation and Training, Orlando, FL. *Standard for Distributed Interactive Simulation - Application Protocols (v 2.0, 4th draft, revised)*, 1993.

[8] Institute for Simulation and Training, Orlando, FL. *Enumeration and Bit-encoded Values for use with IEEE 1278.1 DIS - 1994*, ist-cr-93-46 edition, 1994.

[9] Hyeongseok Ko. *Kinematic and Dynamic Techniques for Analyzing, Predicting, and Animating Human Locomotion.* PhD thesis, University of Pennsylvania, 1994.

[10] Ken Perlin. Danse interactif. SIGGRAPH Video Review, Vol. 101 1994.

[11] Douglas A. Reece. Extending DIS for Individual Combatants. In *Proceedings of the 1994 AI, Simulation and Planning in High Autonomy Systems Conference*, University of Florida, Gainesville, 7–9 December 1994.

[12] John Rohlf and James Helman. IRIS Performer: A High Performance Multiprocessing Toolkit for Real–Time 3D Graphics. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, pages 381–395, July 1994.

[13] Neil J. Rubenking. The DOOM Phenomenon. *PC Magazine*, 13(19):314–318, 1994.

[14] Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

[15] Frank Wysocki and David Fowlkes. Team Target Engagement Simulator Advanced Technology Demonstration. In *Proceedings of the Individual Combatant Modeling and Simulation Symposium*, pages 144–190, 15–17 February 1994. Held in Fort Benning, GA.

[16] Jeffrey Adam Young. Doom's Day Afternoon. *Computer Player*, pages 20–28, October 1994.

[17] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics, to appear*, 1995.

# A Simple and Efficient Method for Accurate Collision Detection Among Deformable Polyhedral Objects in Arbitrary Motion

Andrew Smith Yoshifumi Kitamura Haruo Takemura* Fumio Kishino

ATR Communication Systems Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02, Japan
<asmith, kitamura, kishino>@atr-sw.atr.co.jp

## Abstract

*We propose an accurate collision detection algorithm for use in virtual reality applications. The algorithm works for three-dimensional graphical environments where multiple objects, represented as polyhedra (boundary representation), are undergoing arbitrary motion (translation and rotation). The algorithm can be used directly for both convex and concave objects and objects can be deformed (non-rigid) during motion. The algorithm works efficiently by first reducing the number of face pairs that need to be checked accurately for interference by first localizing possible collision regions using bounding box and spatial subdivision techniques; face pairs that remain after this pruning stage are then accurately checked for interference. The algorithm is efficient, simple to implement, and does not require any memory intensive auxiliary data structures to be precomputed and updated. Since polyhedral shape representation is one of the most common shape representation schemes, this algorithm should be useful to a wide audience. Performance results are given to show the efficiency of the proposed method.*

## 1 Introduction

In a virtual environment, we can simulate various kinds of physical phenomena. An important example of this is being able to determine when moving objects collide; this is called the "collision detection problem." It is vitally important to be able to update a virtual environment at real-time rates to engender realism for a user. Unfortunately, current collision detection algorithms, if used, are an enormous bottleneck and make real-time update impossible [1, 2]. The difficulty of collision detection for polyhedral objects can be seen by examining the basic, naive way of performing it. The basic method works by performing static intersection tests at discrete time instants; the time interval between tests is assumed small enough so that collisions are not missed. Then, interference among polyhedral objects at a time instant is detected by testing all combinations of faces and edges for the presence of an edge of one object piercing the face of another object; if such an edge-face pair exists then there is a collision [3]. The average time complexity for this test (for $n$ objects) is $O(n^2EF)$, where $E$ and $F$ are the number of edges and faces in the average object. As can be seen from this complexity figure, the problem lies in the necessity of having to perform such a large number of computationally expensive intersection tests at every time instant, where the number of such tests increases quadratically as the number and complexity of objects increase. For anything more than a simple world with a few objects of a few hundred faces each, this method is untenable in terms of maintaining real-time performance.

The main problem with the basic, naive collision detection method is that it requires such a large number of computationally expensive edge-face intersection checks. In an actual virtual world, the number of edge-face pairs that intersect at any time instant is a small percentage of the total number of possible pairs (in fact, much of the time there are no intersections). Thus, it is desirable to have a collision detection algorithm which checks a number of edge-face pairs proportional to the number that actually intersect. In this paper, we present an algorithm that realizes this and can be used for general (i.e., the environment can contain both convex and concave objects), deformable polyhedral objects undergoing arbitrary motion.

The remainder of the paper is organized as follows. The next section discusses other research efforts towards efficient collision detection. After that, the details of our collision detection algorithm are described. Next, experiments carried out using our algorithm are described, and performance results, showing the efficiency of the approach, are given. Finally, the last section concludes the paper.

## 2 Efficient Collision Detection Approaches

There is much literature devoted to efficient collision detection approaches and this section discusses this research. The first subsection simply describes other approaches to efficient collision detection. Then, the last two subsections evaluate these other approaches, describing the problems with them which make them not entirely suitable for practical, large-scale virtual environments and how our algorithm addresses these problems.

*Currently with the Nara Institute of Science and Technology, Japan (takemura@is.aist-nara.ac.jp)



0-8186-7084-3/95 $04.00 © 1995 IEEE

## 2.1 Related Collision Detection Research

Much research on collision detection for polyhedra aims to drastically reduce the number of edge-face pairs that need to be checked for intersection. A common first step in many collision detection routines is an approximate bounding region (usually an axis-aligned box or a sphere) overlap test to quickly eliminate many objects as not interfering. An extension of this idea is to use a hierarchy of bounding regions to localize collision regions quickly [2]. Related methods use octrees and voxel sets. [4] stores a voxel data structure with each object, with pointers from voxels to polyhedra faces that intersect them. Collision is localized by testing for intersection of voxels between two objects. [5] stores an octree for each object and, at each time instant, checks the interference of objects' updated octrees; face pairs from inside of interfering octree nodes are then checked for collision. Other voxel and octree methods include [6–10].

Another method for collision detection involves keeping track of the distance between each pair of objects in the world; if the distance between a pair goes below some small threshold then the pair has collided. A noteworthy use of this idea for collision detection of rigid, convex objects is [11], where coherence of objects between time instants (i.e., object positions change only slightly) and the property of convex polyhedra are used to detect collisions among objects in roughly constant time per object pair. Other research which uses this distance based approach include [12, 13].

Briefly, some other approaches to collision detection are as follows. [14] uses a data structure called a "BRep-Index" (an extension of the well-known BSP tree) for quick spatial access of a polyhedron in order to localize contact regions between two objects. [15] finds separating planes for pairs of objects; using object coherence, these separating planes are cached and then checked at succeeding time instants to yield a quick reply of non-collision most of the time. [16] uses ideas from the z-buffer visible surface algorithm to perform interference detection through rasterization. [17] uses back-face culling to remove roughly half of the faces of objects from being checked for detailed interference; the basic idea is that polygons of a moving object which do not face in the general direction of motion cannot possibly collide. [18] uses a scheduling scheme, whereby object pairs are sorted by distance and only close objects are checked at each time instance. [19] uses four dimensional space-time bounds to determine the earliest time that each pair of objects could collide and does not check the pair until then. [1] models objects as superquadrics and shows how collision detection can be done efficiently using the inside/outside function of a superquadric. For coarse collision detection, [20] stores bounding regions of objects in a stack of 2D structures similar to quadtrees (to reduce memory use) and uses only bit manipulations to add or delete objects to this (to reduce computation).

Finally, our algorithm uses ideas from methods for localized set operations on polyhedra [21, 22]. These methods attempt to perform efficiently set operations, such as intersection, union, etc., on polyhedra by localizing the regions where faces are using spatial subdivision techniques; a set operation for a face then only needs to be done against the other faces inside the region that the face is in. As a particular example, the idea of intersecting faces with overlap regions of bounding boxes in order to localize the interference region of two objects was first described in [23] and we use this idea effectively in our algorithm.

## 2.2 Evaluation

We evaluate the above algorithms on the basis of four properties of a collision detection algorithm necessary for effective use in a practical, large-scale virtual environment inhabited by humans. These are the ability to handle deformable (non-rigid) objects, the ability to handle concave objects, not using excessive amounts of memory for storing auxiliary data structures, and having better than $O(n^2)$ complexity for $n$ objects in the world. None of the algorithms surveyed in the previous subsection has all four properties and some do not even have one of them. Our algorithm can satisfy all four of these properties.

### 2.2.1 Deformable Objects

In a virtual environment inhabited by humans, it is very important to be able to perform collision detection for objects which deform during motion. For example, in physical-based simulations forces between colliding objects are determined and the colliding objects are then deformed based on these forces. In general, a user should be allowed to deform objects in a virtual environment, which necessitates collision detection for deformable objects. Many of the above algorithms require precomputation and computationally expensive updating of auxiliary data structures (e.g., octrees, voxel sets, BRep-indices, etc.) for each object. This limits their usefulness because it means that objects are essentially limited to being rigid; this is because when an object deforms, its auxiliary data structures must be recomputed and this is usually an expensive operation. Our collision detection algorithm handles deformable objects.

### 2.2.2 Auxiliary Data Structures

In addition to being expensive to recompute, storing auxiliary data structures for each object can take up considerable memory. This limits the number of objects for which such algorithms can be effectively used. Our algorithm does not require any auxiliary data structures beyond simple bounding boxes and arrays.

### 2.2.3 Concave Objects

Another problem is that some of the above collision detection algorithms require objects to be convex [11–13, 15]. However, it is clear that most objects of interest in the real-world are concave and a virtual environment, to be useful, should allow concave objects. To solve this problem, the above authors argue that a

concave object can be modeled as a collection of convex pieces. While this can in fact be done for any concave object, it adds many fictitious elements (i.e., vertices, edges, etc.) to an object. In addition, breaking a concave object up into convex pieces means that the one object becomes many objects; unfortunately, this greatly worsens the complexity problem described in the next section (because each convex piece of the concave object must be treated as a separate object for the purposes of collision detection). Most importantly, however, any algorithm that requires objects to be convex or unions of convex pieces cannot be used to detect collisions for deformable objects; this is because, in general, deformations of an object easily lead to concavities. Our algorithm deals directly with concave objects in the same way as convex ones, with no extra computational overhead.

#### 2.2.4 Complexity

The $O(n^2)$ complexity problem becomes apparent for large-scale virtual environments. [1] discusses problems due to computational complexity in computer-simulated graphical environments and notes that collision detection is one such problem for which, in order to simulate realistically complex worlds, algorithms which scale linearly or better with problem size are needed. To understand the problem concretely, consider a collision detection algorithm that takes 1 millisecond per pair of objects. While for very small environments this algorithm is extremely fast, the algorithm is impractical for large-scale environments. For example, for an environment with just 50 objects 1225 pairwise checks between objects must be done, taking more than a second of computation; in this example, real-time performance cannot be maintained for environments with more than 14 objects (being able to compute something in 100 milliseconds or less is considered to be real-time performance [24]). All of the distance based approaches [11–13] and many of the others [1, 4, 15] suffer from this complexity problem. In our experiments, we did use a bounding box test among objects which is $O(n^2)$ for $n$ objects. However, the bounding box test between two objects is extremely fast and thus should not become a bottleneck unless there are many objects in the environment; for such an environment, however, the problem can be easily solved by using a bounding box check with better complexity ([25] describes such a method) or by skipping the bounding box stage altogether and going directly to the face octree spatial subdivision stage which is $O(n)$ for $n$ objects.

#### 2.2.5 Other

A few other minor problems with the surveyed algorithms are as follows. Some of these algorithms [1, 7] cannot be used for polyhedra, which limits their usefulness for current graphical applications where polyhedra dominate as the object representation. Some of the algorithms do not provide accurate collision detection (i.e., identify exactly which objects are interfering and which faces of the objects interfere— [5] describes how this is useful for operator assistance) among objects [16, 18–20]. While most of the algorithms described above are clearly improvements over the basic, naive collision detection algorithm, none of them provide a solution to the problem that is as general, efficient, and simple as ours. The details of our collision detection algorithm are presented next.

### 2.3 Proposed Algorithm

Our proposed algorithm is an extension of the methods for localized set operations for use in collision detection. In particular, we extend the ideas in [21–23] to a world with multiple objects; these papers describe algorithms for 2 objects but never precisely explain how to extend their algorithms efficiently to handle multiple objects (thus, direct use of these algorithms requires $O(n^2)$ complexity for $n$ objects). In addition, these algorithms, in testing for intersection between a face and an axis-aligned box (while performing spatial subdivision), advocate using an approximate test between the bounding box of the face and the axis-aligned box; however, in developing our algorithm we found that using the exact intersection test described in a later section (section 3.7) gave better performance (because it reduces the number of edge-face pairs even more, without much of an added computational cost). Also, these algorithms are used for performing static set operations; we show how they can be used for collision detection in a dynamic environment with multiple moving objects. Finally, we provide empirical evidence to show the efficiency of the proposed algorithm. The next section describes the details of our proposed collision detection algorithm.

## 3 Collision Detection Algorithm

### 3.1 Assumptions

All objects in the world are modeled as polyhedron (boundary representation). The faces of a polyhedron are assumed to be triangular patches without any loss of generality of range of representation. Objects can be concave or convex. The objects are undergoing motion which is not predetermined (e.g., a user can move his graphical hand in a sequence of non-predetermined, jerky motions); object motion can be both translation and rotation. Objects can be deformed during motion. Given this kind of environment, the goal is to be able to detect the colliding objects in the world and, in particular, the face pairs, between objects that are interfering. Collision will be checked for all objects at discrete time instants (i.e., at each time instant the new positions of objects will be determined, and collision will be checked for at that time instant *before* the computer graphic images of the objects are drawn to the screen). It is assumed that the speeds of objects are sufficiently slow compared with the sampling interval so that collisions are not missed. Finally, it is assumed that there is a large cube which completely bounds the world (i.e., that all objects will stay inside of this cube); let the side length of this cube be $L$.

### 3.2 Outline of the Method

Figure 1 shows the control flow of our method. Suppose there are $n$ objects in the workspace. The bounding boxes for each object are updated periodically (at discrete time instants $\cdots$, $t_{i-1}$, $t_i$, $t_{i+1}$, $\cdots$) using observed object motion parameters. Updated bounding boxes are checked for interference. For each object with an interfering bounding box all overlap regions of the object's bounding box with other objects' bounding boxes are determined. Next, for each such object all faces of the object are checked for intersection with the overlap regions of the object; a list of the object's faces which intersect one or more of the overlap regions is stored. Then, if there is a list of faces for more than one object, a face octree (i.e., an octree where a node is black if and only if it intersects faces) is built for the remaining faces (for all objects' face lists, together), where the root node is the world cube of side length $L$ and the face octree is built to some user specified resolution. Finally, for each pair of faces which are from separate objects and which intersect the same face octree voxel (i.e., smallest resolution cube) it is determined whether the faces intersect each other in three-dimensional space. In this way, all interfering face pairs are found. Note that the intersection of faces with overlap regions and face octree stages repeatedly test for intersection of a face with an axis-aligned box; thus, we describe an efficient algorithm for testing this intersection.

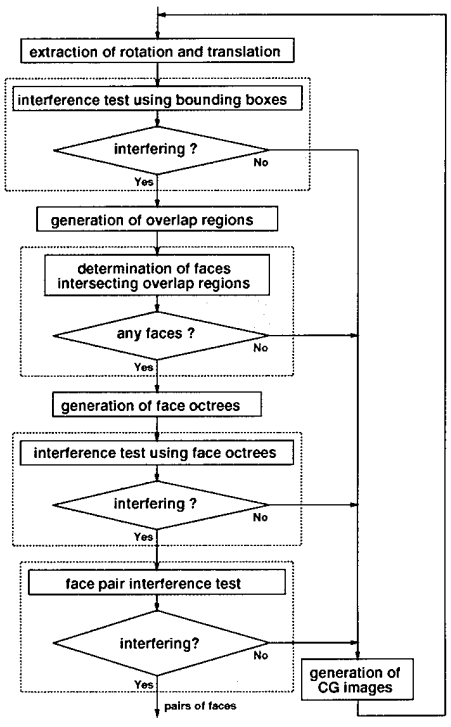



Figure 1: Control flow of collision detection.

### 3.3 Approximate Interference Detection Using Bounding Boxes

At every time instant, axis-aligned bounding boxes are computed for all objects and all pairs of objects are compared for overlap of their bounding boxes. For each pair of objects whose bounding boxes overlap, the intersection between the two bounding boxes is determined (called an overlap region as shown in Figure 2) and put into a list of overlap regions for each of the two objects. The overlap regions are passed to the next step.

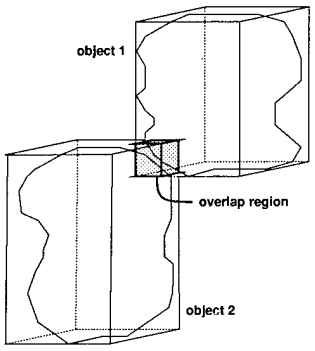



Figure 2: An overlap region

### 3.4 Determination of Faces Intersecting Overlap Regions

For every object which has a list of overlap regions, all faces of the object are compared for intersection with the overlap regions. Once a face of an object is determined to be intersecting with at least one overlap region it is placed in a face check list for the object. If there are face check lists for two or more objects then these are passed on to the next stage. Figure 3 shows an example of faces intersecting an overlap region.

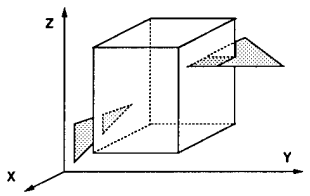



Figure 3: Faces intersecting the overlap region

### 3.5 Face Octree Spatial Subdivision Stage

A face octree is built down to a user specified resolution for the remaining faces starting from the world cube of side length $L$ as the root. To minimize computation, only as much of the face octree as is necessary for collision detection is built; in particular, a parent node is subdivided into its 8 children only if it contains faces from two or more objects, and only

the faces which were found to intersect the parent node are tested for intersection with the children nodes. Also, there is no condensation of the face octree (i.e., 8 black child nodes are not erased and replaced by their single, black parent node). If there are voxels in the face octree, then in each voxel there are faces from two or more objects. For each voxel, all possible pairs of faces, where the faces are from different objects, are determined and put into a face pair checklist. However, a face pair is only put into this face pair checklist if it was not previously put there by examination of another voxel. The face pair checklist is then passed to the next stage. Note that it is not necessary to allocate memory and actually build a face octree; faces can simply be checked for intersection with the standard cubes of an octree and checked recursively for lower-level cubes (thus no memory, beyond the small amount used by the stack during recursion, for storing octrees is necessary). Also note that a face octree is built for only a very small portion of all the faces; the previous stage eliminates most faces as not interfering.

### 3.6 Face Pair Interference Check

A pair of faces is checked for intersection at a time instant as follows. First, the bounding boxes of the faces are computed and checked for overlap; if there is no overlap in the bounding boxes then the faces do not intersect. Otherwise, the plane equation of the face plane of the first face is computed and the vertices of the second face are evaluated in this equation; if all vertices lead to the same sign (+ or -) then the second face is completely on one side of the face plane of the first face and thus there is no intersection. The plane equation of the face plane of the second face is then determined and the vertices of the first face are evaluated in it in the same way. If neither face is found to be completely on one side of the face plane of the other face, then more detailed checks are done as follows. For each edge, in turn, of face 1 the intersection point of it with the face plane of face 2 is found and checked to see if it is inside face 2 (i.e., three-dimensional point-in-polygon check—the method used is described in [26]); if the point is inside the face then the two faces intersect. The case when an edge and face plane are coplaner is handled by projecting the edge and face onto the two-dimensional coordinate axis most parallel to the face plane and performing a two-dimensional intersection check between the projected face and edge. In the same way, the edges of face 2 are checked for intersection with face 1. If no edges of either face are found to intersect the other face, then the two faces do not intersect.

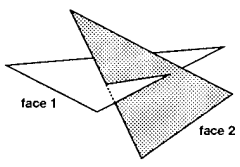



Figure 4: Face pair intersection test.

### 3.7 Efficient Triangular Patch and Axis-Aligned Box Intersection Determination

To determine whether or not a triangular patch intersects with an axis-aligned box, we perform clipping against 4 of the face planes of the faces that comprise the box; the 4 face planes are the maximum and minimum extents of two of the three x,y,z dimensions (e.g., in our implementation we arbitrarily chose to clip against the maximum and minimum x extents and the maximum and minimum y extents). For the final dimension, it is only necessary to check whether or not the remaining vertices of the clipped triangular patch are either all greater than the maximum extent or all less than the minimum extent; if either case is true then there is no intersection, otherwise there is intersection. In addition, it is often not even necessary to clip against four planes. During clipping, whenever the intersection point of a segment with the current face plane is calculated this point can be quickly checked to see if it is inside of the face of the face plane; if it is inside, then the triangular patch and box intersect and no more computation needs to be done. Finally, before performing any clipping at all, two quick tests are done. As a first step, a quick overlap check between the bounding box of the triangular patch and the axis-aligned box can be done to quickly determine non-intersection in many cases. Second, the three vertices of the triangular patch can be checked to see if one of them is inside of the axis-aligned box; if so, then the triangular patch and axis-aligned box intersect.

## 4 Experiments

The algorithm and an experimental environment were implemented and run on a Silicon Graphics Indigo$^2$ (this has an R4400/150 MHz processor); experiments were done to determine the efficiency of the proposed algorithm. In all experiments described in this section, face octrees were built to resolution level 6.

### 4.1 Standardized Objects

For performance evaluation, sphere-like objects approximated by differing numbers of triangular patches were used; spheres were selected for testing because of their orientation invariance. Figure 5 shows some of the spheres which were used in the experiments. The basic experiment done was to have two identical sphere objects start at different (non-penetrating) positions and have them move towards each other (with both translation and rotation motion) until they interfere. This basic experiment was done with sphere objects having respectively 8, 10, 24, 48, 54, 80, 120, 168, 224, 360, 528, 728, 960, and 3968 triangular patches. Figure 6 shows the computation time required at each processing cycle from $t = 1(cycle)$, when there is no interference, until $t = 72(cycle)$, when faces from the two sphere objects are found to be intersecting, for four of the experimental sphere objects; at the last cycle, 70, 24, 16, and 11 milliseconds of computation are required to determine the colliding faces for the

spheres with 3968, 960, 528, and 168 faces. Finally, figure 7 shows the computation required at the last stage (i.e., when faces from the two objects are found to be interfering—this requires maximum computation and is the true measure of the efficiency of a collision detection algorithm) of the proposed collision detection algorithm between two sphere objects against the number of triangular patches of the sphere objects.

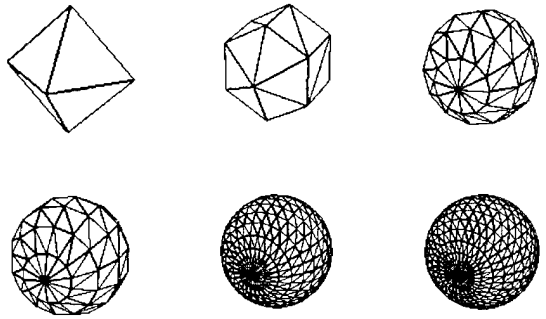


Figure 5: Examples of experimental objects (standardized spheres with different numbers of faces)

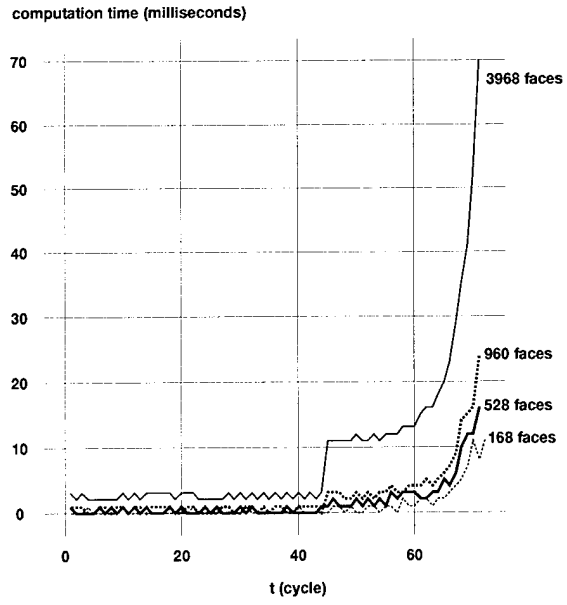



Figure 6: Computation time for each processing cycle for two identical sphere objects with 168, 528, 960 and 3968 faces.

## 4.2 Multiple General Objects

An experiment was also done with multiple general (i.e., concave—a real-world type of object) objects. Specifically, 15 identical objects (space shuttles with 528 triangular patches—see figure 9) were moved (both translation and rotation) in the test environment for many processing cycles and the computation time required at each cycle to perform collision detection was measured. At every cycle, many objects' bounding boxes were overlapping; thus, many triangular patches had to be tested for intersection with overlap regions at every cycle. At the last cycle of the test, faces from two objects were found to be interfering, taking 31 milliseconds of computation. Figure 8 shows the results of this experiment. Also in this figure, in order to provide a basis for comparison, are the results for when only the two interfering space shuttle objects are in the test environment; here, the last step, where faces are determined to be colliding, required 16 milliseconds of computation.

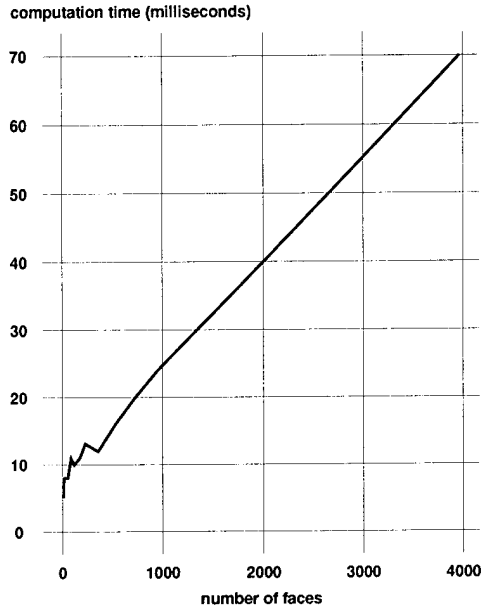



Figure 7: Computation time at the last stage of the proposed collision detection between two identical sphere objects against the number of planar patches of the objects.

## 4.3 Comparison Against Competing Algorithms

In order to show that our algorithm is truly efficient, we directly compared the performance of our algorithm against two other competing algorithms. In general, it is difficult to make such direct comparisons because authors of collision detection papers do not normally give out the code that they used to get experimental results. Fortunately, however, we found the C language code for the first competing collision detection algorithm in [27], and the second competing algorithm is a slight modification of the algorithm proposed in this paper.

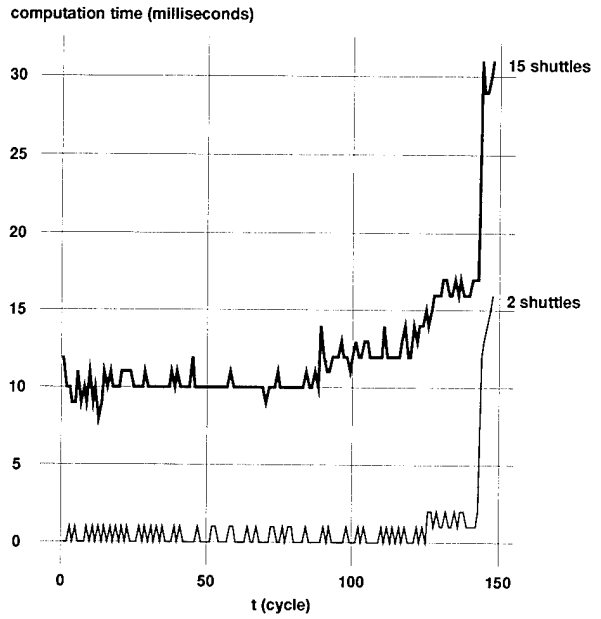



Figure 8: Computation time at each processing cycle for 15 space shuttle objects-collision between two objects is detected at the last cycle.

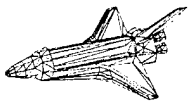


Figure 9: The space shuttle experimental object (528 triangular patches).

### 4.3.1 Separating Plane Algorithm

The first competing algorithm is based on ideas from [12] and [15]. This algorithm can only be used for convex, rigid objects and it does not return the list of face pairs that are interfering, as ours does. Thus, it is not completely fair to compare our algorithm against this algorithm because our algorithm is more general and gives more complete collision analysis. Even so, however, our algorithm gives better performance for non-trivial virtual environments.

The details of this competing algorithm are given in [27]. However, briefly, the algorithm works by initially finding a separating plane between each pair of objects. A separating plane is found for two objects by finding the two closest vertices on the two objects (using the method in [12]); the vector between these two points is the normal vector of the plane and the plane passes through one of the two points. Separating planes are cached between time instants and the previous time instant's separating plane is checked at the current time instant to see if it still separates the two objects; if it no longer separates then an attempt is made to find a new separating plane, which is then cached. If no new separating plane can be found then there is collision. Note that the complexity for this test (for $n$ objects) is $O(n^2)$.

We compared our algorithm against this competing algorithm for environments containing differing numbers of same sphere objects (528 triangular patches). In particular, we tested both algorithms in environments with 10, 20, 30, and 40 moving sphere objects; at the last cycle of the tests two of the sphere objects were interfering. For 10 sphere objects, our algorithm performed roughly the same as the competing algorithm; in particular, our algorithm required 16 milliseconds of computation at the last cycle, while the competing algorithm required approximately 10 milliseconds per cycle. However, for 20, 30, and 40 objects our algorithm performed better. In particular, for 20, 30, and 40 objects, our algorithm required 21, 22, and 41 milliseconds at the last cycle; against this, the competing algorithm required approximately 35, 76, and 140 milliseconds per cycle. The results of these experiments can be seen in figure 10 (the competing algorithm's times are drawn with dotted lines, while the proposed algorithm's are drawn with solid lines).

### 4.3.2 Octree Update Algorithm

The second competing algorithm [28] is a slight modification of the algorithm proposed in this paper, and is representative of the bounding region hierarchy, octree and voxel approaches described in the section on related work (section 2). Essentially, the modification is to precompute complete face octrees for all of the polyhedral objects, and to store a list for each black node of the faces which intersect that black node. Then, the proposed collision detection algorithm is modified as follows. Instead of determining the polyhedral faces which intersect with overlap regions, the octree update algorithm determines the black nodes from the precomputed face octrees which intersect with the overlap regions; these intersecting black nodes are then put into a "node check list" (as opposed to a "face check list"). Then, in the next stage (face octree spatial subdivision stage), instead of creating a face octree by testing for intersections between the polyhedral faces in the face check list and the standard octree nodes, the octree update algorithm builds an octree by testing for intersections between the transformed (i.e., using the same transformation matrix as for the polyhedral objects) black nodes of the node check list and the standard octree nodes. Finally, for each standard octree voxel which was found to contain transformed black nodes from more than one object, all unique pairs of faces, where the faces are inside a precomputed face list of one of the transformed black nodes and the faces are from different objects, are enumerated and checked for intersection (using the method described in section 3.6). Basically, the octree update algorithm substitutes precomputed face octree black nodes for faces in checking for intersection with overlap regions and standard octree nodes. Note that this algorithm can be used for concave objects, but that objects must be rigid; thus, it is not as general

as the proposed algorithm.

We tested the proposed algorithm against this competing algorithm for the environment of figure 12; this environment contains a sphere (120 faces), a space shuttle (528 faces), a chair (146 faces), and a Venus head (1816 faces). The experiment that was performed was to move the venus head and the space shuttle towards each other (with translation and rotation) until they collided at the last cycle; the other two objects also translated and rotated slightly (without any collision). The proposed algorithm performed much better than the competing algorithm for all cycles after cycle 17 (when bounding boxes first overlapped); in particular, at the last cycle the competing algorithm required 161 milliseconds of computation, while the proposed algorithm required only 11 milliseconds (roughly 16 times better performance). Figure 11 shows the results of this experiment.

## 5 Discussion

As can be seen from the various graphs given, our collision detection algorithm is quite efficient. A common definition for "real-time" performance of a computer graphics application is being able to render 10 frames per second [24]. Using this definition, our algorithm is able to perform real-time collision detection for objects having up to approximately 5936 faces (extrapolated from figure 7). Also important is the fact that the algorithm takes negligible compute time (rarely more than 10 milliseconds) when no objects in the environment are interfering. In addition, adding many objects to the environment increases computation time only slightly (i.e., for the case that only two objects at a time interfere—if more objects interfere at the same time then computation time will increase, but not greatly).

We did not implement the basic, naive collision detection algorithm in order to compare it to our algorithm (because our algorithm is clearly better—see [5] and [16] to see how ludicrously long the naive algorithm can take for even very simple environments). The important basis of comparison should be with other authors accurate collision detection algorithms for general, deformable polyhedra; as shown in the section on related work (section 2), there are very few collision detection algorithms which providerevised this generality. We were not able to compare directly our algorithm against another competing algorithm which is as general as ours; however, even against the more restrictive algorithms of the previous section our algorithm gives better performance.

Based on these experiments, it seems reasonable to conclude that our algorithm would perform quite well in many applications. Unfortunately, however, we cannot assert, based solely on these experiments, that our algorithm is the fastest for all possible applications. There has already been much research into efficient collision detection, and many different efficient approaches have been proposed. We feel that, in addition to exploring new collision detection approaches, "comparative collision detection" would be a worthy new research topic. We feel that our proposed algorithm would fare well in such a comparative study, and we have made a start towards such research with our comparisons against two competing algorithms. However, more comprehensive research, which does more complete comparisons and which tests variations and combinations of the various algorithms in situations that mimic real applications, is necessary. For the time being, however, we feel that, considering the generality of our algorithm, its ease of implementation, its small memory requirements, and its proven efficiency, we have provided a practical solution to the problem of real-time collision detection.

## 6 Conclusion

In this paper, we have presented an efficient algorithm for accurate collision detection among polyhedral objects. The algorithm can be used for both convex and concave objects; both types of objects are dealt with in the same way and there is no performance penalty for concave objects. The algorithm can be used for objects whose motion is not prespecified, and both translation and rotation motion are allowed. The algorithm can also be used for objects that deform during motion. Thus, the algorithm is very general. The algorithm is fairly straightforward and should be easy to implement. The algorithm does not require the precomputation and update of memory intensive auxiliary data structures, which some collision detection algorithms require and which can sap the memory resources of an application, making it impossible to perform collision detection for a large number of objects. And finally and most importantly, even though the algorithm is very general it is extremely fast; Adding many objects to the environment does not require much more computation and the algorithm can run in real-time on a graphics workstation for polyhedra containing several thousands of faces.

We are currently exploring various optimizations to this algorithm, such as using face bintrees instead of face octrees, using a more efficient bounding box check (to reduce the $O(n^2)$ complexity for n objects), and determining the optimal level for face octree subdivision (the PM-octree [29] might be useful for this). In addition, we are implementing a parallel version of the algorithm, which should be quite effective because of the many independent intersection calculations done by the algorithm. The algorithm is already sufficiently fast for most applications. However, with anticipated speedups from optimization and parallelization, our algorithm should be suitable for very large, practical virtual environments.

## References


[1] Pentland, Alex P. Computational complexity versus simulated environments. *Computer Graphics*, Vol. 24, No. 2, pp. 185–192, 1990.

[2] Hahn, James K. Realistic animation of rigid bodies. *Computer Graphics*, Vol. 22, No. 4, pp. 299–308, 1988.

[3] Boyse, John W. Interference decision among solids and surfaces. *Communications of the ACM*, Vol. 22, No. 1, pp. 3–9, 1979.

[4] Garcia-Alonso, A., Serrano, N., and Flaquer, J. Solving the collision detection problem. *Computer Graphics and Applications*, Vol. 14, No. 3, pp. 36–43, May 1994.

[5] Kitamura, Y., Takemura, H., and Kishino, F. Coarse-to-fine collision detection for real-time applications in virtual workspace. In *International Conference on Artificial Reality and Tele-Existence*, pp. 147–157, July 1994.

[6] Moore, M. and Wilhelms, J. Collision detection and response for computer animation. *Computer Graphics*, Vol. 22, No. 4, pp. 289–298, 1988.

[7] Turk, Greg. Interactive collision detection for molecular graphics. M.sc. thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 1989.

[8] Zyda, M. J., Pratt, D. R., Osborne, W. D., and Monahan, J. G. NPSNET: Real-time collision detection and response. *The Journal of Visualization and Computer Animation*, Vol. 4, No. 1, pp. 13–24, 1993.

[9] Shaffer, C. A. and Herb, G. M. A real-time robot arm collision avoidance system. *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 2, pp. 149–160, 1992.

[10] Hayward, V. Fast collision detection scheme by recursive decomposition of a manipulator workspace. In *International Conference on Robotics and Automation*, pp. 1044–1049. IEEE, 1986.

[11] Lin, M. C., Manocha, D., and Canny J. F. Fast contact determination in dynamic environments. In *International Conference on Robotics and Automation*, pp. 602–608. IEEE, 1994.

[12] Gilbert, Elmer G., Johnson, Daniel W., and Keerth, S. Sathiya. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2, pp. 193–203, 1988.

[13] Quinlan, Sean. Efficient distance computation between non-convex objects. In *International Conference on Robotics and Automation*, pp. 3324–3329. IEEE, 1994.

[14] Bouma, W. and Vanecek, G. Collision detection and analysis in a physical based simulation. In *Eurographics Workshop on Animation and Simulation*, pp. 191–203, September 1991.

[15] Baraff, David. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics*, Vol. 24, No. 4, pp. 19–28, 1990.

[16] Shinya, M. and Forgue, M. Interference detection through rasterization. *The Journal of Visualization and Computer Animation*, Vol. 2, pp. 132–134, 1991.

[17] Vanecek, George. Back-face culling applied to collision detection of polyhedra. Technical report, Purdue University Department of Computer Science, 1994.

[18] Foisy, A., Hayward, V., and Aubry, S. The use of awareness in collision prediction. In *International Conference on Robotics and Automation*, pp. 338–343. IEEE, 1990.

[19] Hubbard, Philip M. Interactive collision decision. In *Symposium on Research Frontiers in Virtual Reality*, pp. 24–31. IEEE, 1993.

[20] Fairchild, K. M., Poston, Timothy, and Bricken, William. Efficient virtual collision detection for multiple users in large virtual spaces. In *Virtual Reality Software and Technology*, 1994.

[21] Mantyla, M. and Tamminen, M. Localized set operations for solid modeling. *Computer Graphics*, Vol. 17, No. 3, pp. 279–288, July 1983.

[22] Fujimura, K. and Kunii, T. A hierarchical space indexing method. In *Visual Technology and Art (Computer Graphics Tokyo)*, pp. 21–33, 1985.

[23] Maruyama, K. A procedure to determine intersections between polyhedral objects. *International Journal of Computer and Information Sciences*, Vol. 1, No. 3, pp. 255–266, 1972.

[24] Card, S. K., Moran, T. P., and Newell, A. *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

[25] Kirk, David, editor. *Graphics Gems III.* Academic Press Professional, 1992.

[26] Arvo, James, editor. *Graphics Gems II.* Academic Press Professional, 1991.

[27] Heckbert, Paul, editor. *Graphics Gems IV.* Academic Press Professional, 1994.

[28] Kitamura, Y., Smith, A., Takemura, H., and Kishino, F. Optimization and parallelization of octree-based collision detection for real-time performance. In *IEICE Conference 1994 Autumn.* D-323, 1994. (in Japanese).

[29] Hanan Samet. *The design and analysis of spatial data structures.* Addison-Wesley, 1990.

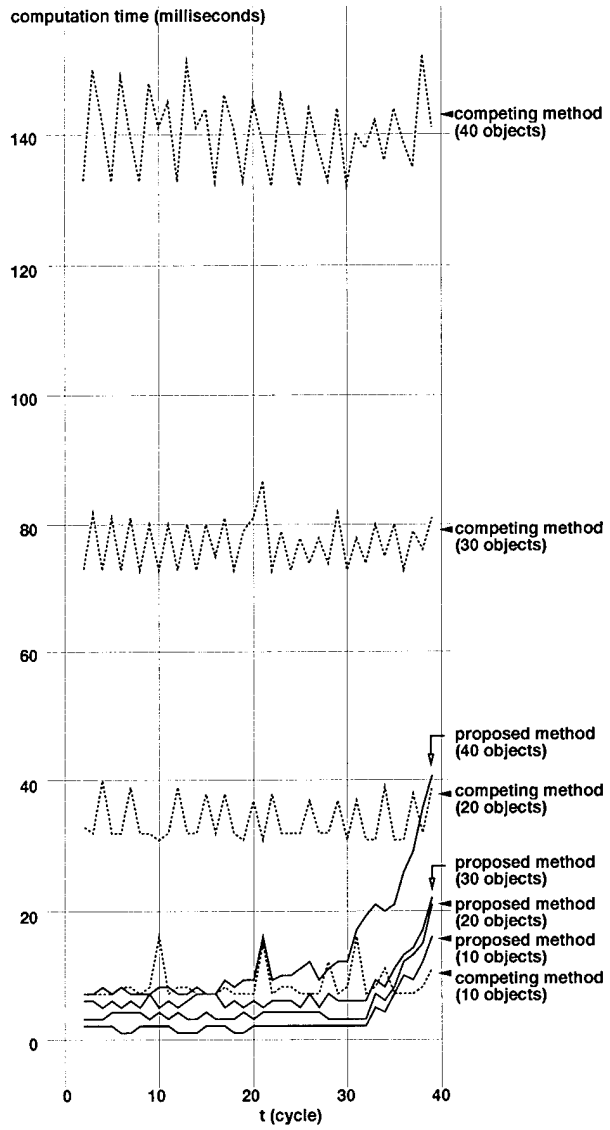



Figure 10: Computation time for each processing cycle for the proposed algorithm (solid lines) and the separating plane competing algorithm (dotted lines) for 10, 20, 30, and 40 identical sphere objects (528 triangular patches each).

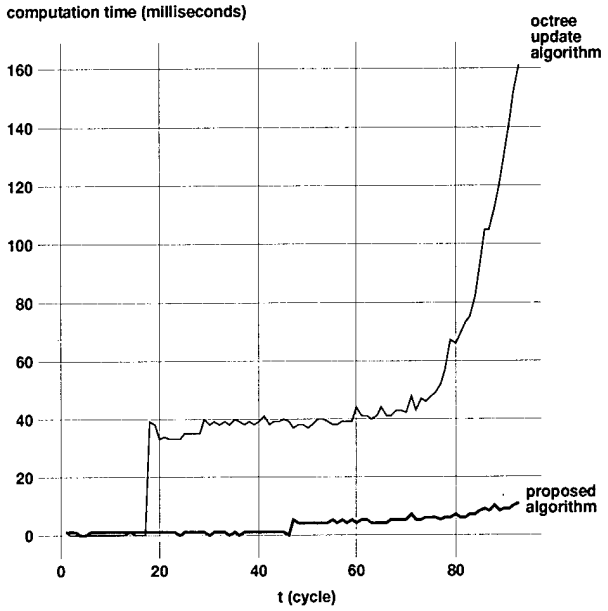



Figure 11: Computation time for each processing cycle for the proposed algorithm and the octree update competing algorithm for the environment of figure 12.

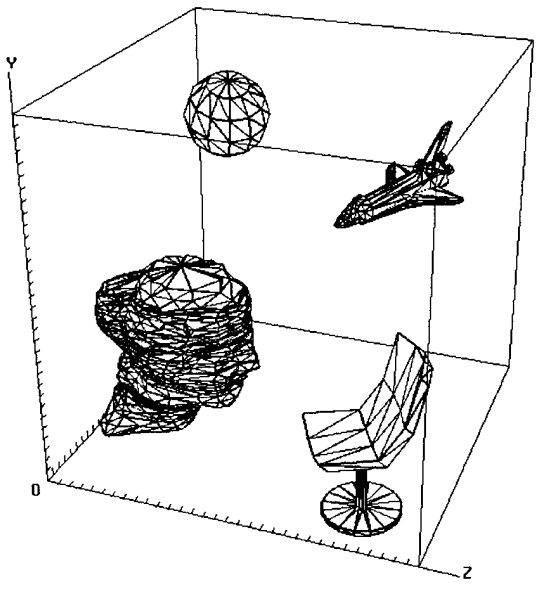



Figure 12: The experimental environment used to obtain the data for figure 11.

# Distributed Virtual Reality Applications

# Interacting in Distributed Collaborative Virtual Environments

*Wolfgang Broll*
Institute for Applied Information Technology
German National Research Center for Computer Science (GMD)
Schloß Birlinghoven
53757 St. Augustin, GERMANY
+49-2241-14-2715
broll@gmd.de

**ABSTRACT**

Virtual reality toolkits and systems for computer supported cooperative work are often treated separately. However, combining them offers new possibilities for remote cooperative (or collaborative) group working. In this paper we review existing distribution models of virtual environments and propose a new method of concurrent interaction management. We will examine the different types of communication layers, which are needed by collaborative virtual reality (VR) applications to achieve complex user interaction. Finally we propose a model for handling the different requirements of such applications, depending on the connection strategies used within a distributed VR system.

**KEYWORDS**

Human computer interaction, computer supported cooperative work, virtual reality.

**INTRODUCTION**

With the increasing performance of modern workstations the acceptance and distribution of 3D applications has become more common. The best known and probably most spectacular development in this area is that of virtual reality (VR) and virtual environments (VE) [15]. Until recently VR was the domain of a small number of research institutes. Today several applications and research projects exist. So now we can begin to examine the usefulness of VR.

Initial skepticism has been replaced by a search for new and suitable applications. The first VR systems were single user systems, but now we can see that multi-user VR [7] can open new areas of application. Connecting several people at different places, i.e. the world wide distribution of virtual environments, is the current focus of our research. There are several possibilities for working together within distributed virtual worlds (VW). Some promising examples already exist in the areas of remote manipulations and cooperative engineering and modeling [12].

The problems of multiple users sharing the same workspace are already known from the field of computer supported cooperative work (CSCW). Some of the major problems are: the distribution of objects and information [11] as well as the delegation of rights and the representation of group structures [14].

In contrast to most CSCW systems, direct (real-time) cooperation leads to completely new interaction possibilities, especially concurrent interaction of several users with one or more objects. This raises serious problems for distributed systems, which so far have had to be solved for each application. There is a lack of application-independent support for most VR systems in this field. Figure [1] gives a simple example: Two different persons might lift a table one after another, but they are not able to carry it individually. Both of them would have to lift the table at the same time, but this example could not be replicated with existing VR systems.

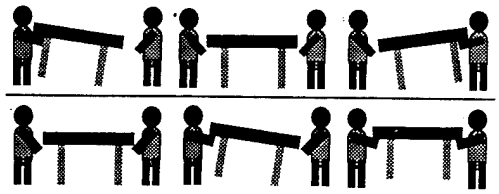


Figure 1: Cooperative vs. collaborative work.

When discussing distributed, multi-user VR systems, the terms 'cooperative' and 'collaborative' are used interchangeably. For us, 'cooperative' implies joint editing of shared objects, while 'collaborative' additionally allows truly concurrent editing.

Therefore the change from cooperative virtual environments to collaborative virtual environments is mostly continuous. Nevertheless real concurrent editing (collaborative work) makes more strenuous requirements on a VR system. It is necessary that services such as concurrent access, or the handling of distributed interactions, are adequately supported.

The intention of this paper is to give a short overview of distribution models already used within different VR systems. The implementations of these models in some existing VE will be presented. Since concurrent interaction is not adequately provided by these systems, we will then introduce our approach to handling concurrent interaction. This leads directly to the next section which gives an overview of a universal distribution model, consisting of distribution ser-

vices required by collaborative applications. We also look at the communications strategies required by a multi-user VE.

## OBJECT DISTRIBUTION IN VIRTUAL ENVIRONMENTS

Cooperative and collaborative virtual environments are mainly used in distributed VR systems. The distribution of the virtual environment objects complicates the interactions between objects. Thus it is very important to take a look at the different distribution models and assess their advantages and disadvantages with respect to VR.

### Distribution Models

The distribution models widely used in VR systems today are:

- active replication
- replication on demand
- migration combined with partial replication

Centralized systems or system with some main centralized components (especially client-server approaches) are not considered in this paper.

Using *active replication* a copy of each object is distributed among all processes or process groups after creating a new object. Since usually only one VE is running on each site (machine) at the same time, in the term *site* is used for a process or process group sharing the same data on a single workstation connected to other locations of the VE by a network. Usually each site provides a database of VW objects or at least a part of the database. Any new site gets all objects of the existing sites. New objects, or changes to existing objects at any site, are distributed among all sites. Thereby it has to be guaranteed that different orders of object changes at different sites (due to signal delays) are resolved into a definite state within the whole system, i.e. at all sites.

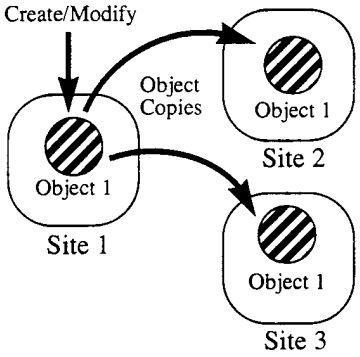



Figure 2: Object distribution by active replication.

Using the *replication on demand* method, objects, or parts of objects, are only distributed, if a process, i.e. a VR application, expresses some interest in that object. After changing an object, a copy of it is distributed among all the processes, which requested it. There are two approaches to managing the copies: Either the objects manage themselves or a special manager for all objects of a site (or process) has to be set up. This manager is then responsible for the object copies, i.e. the manager knows where to send object changes. This manager, or the object, has to be informed if no more updates are needed.

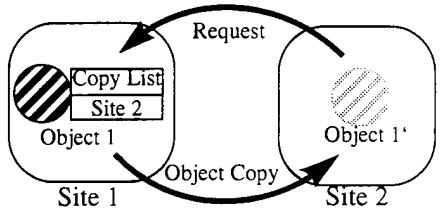



Figure 3: Object distribution by replication on demand.

In contrast to the two other methods *migration* does not distribute copies – objects are distributed but not duplicated. The distribution is done according to the load at each site (or processors) and the frequency of object access. Nevertheless some duplicated information among the sites is required, e.g. geometry for object visualization. Otherwise the connecting network would become a bottleneck of the system. For the distribution of duplicated information one of the two previous distribution methods has to be used.

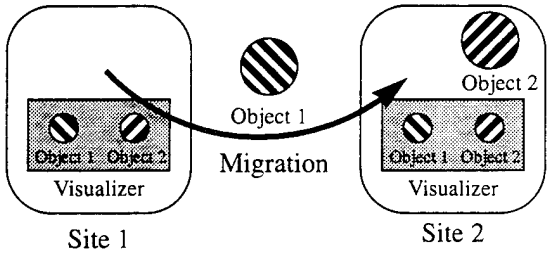



Figure 4: Object migration with partial replication.

Beside the difficulties above some other issues arise. Active replication has some advantages in that all information is always available at every site: there is no need for an application to be aware of distribution. However this advantage may turn into a disadvantage since it might cause a high network load. This causes serious problems in large (complex) virtual worlds which have a large number of sites and users, particularly across wide area networks with slow connections. Usually, however, most of the information in a VW is rarely, or never, needed by each site. For example a site which allows a flight through the world without interacting with the objects, only the rendering data is needed. Replication on demand avoids this overhead, but each application has to take care to get the information it needs. If migration is used without any replication it leads to very high network load. Combining partial replication with one of the two other models looks very similar to replication on demand. Therefore, to avoid the disadvantages of a single model,

many existing systems actually use a combination of them.

### Distributed Object Access

Since virtual environments are highly interactive and dynamic, objects, or parts of them within the VW, have to be modified quite frequently. Concurrent object access may be resolved, using one of the methods realized in cooperative (CSCW) systems [8]:

- locking
- transaction mechanisms
- turn-taking protocols (e.g. floor control)
- centralized controllers
- dependency-detection
- reversible execution
- master entities

Actually most VR systems use simple locking or master entities. These two methods are described in more detail in the following.

*Locking* demands a lock on an object and all its copies at each site or process for access. After modifying the object and distributing the changes the lock can be released. The most important disadvantage of locking is, that depending on which object is actually locked, the access to other objects is blocked and hence the performance of the whole application is reduced. One example of this is the rendering process which has to access the data of all objects periodically. For that reason, many systems allow service processes to access (read) locked data.

The use of *master entities* implies that the right to change (write) object data is held by only one site or process at any time. Since writing may occur concurrently to reading at other sites, an algorithm for restoring a consistent state (e.g. by logging and reversible execution) has to be used. Usually master entities are located at the site where the object is created. Other sites can change object data either by sending messages to the master site, or by migrating the master entity to the site.

Because only one instance of each object exists at any time, with migration no special method for modifications is needed. However since one of the other distribution methods has to be used for replicated object data, the same problems remain.

Finally, there is another important difference between the two methods of object access. While both methods transfer a consistent system state into another consistent system state, locking leads to deterministic results whereas master entities produce non-deterministic results. Although this can be rectified by methods such as logging, it fails for VR systems, since interactions of users are highly dependent on the consistent behavior of VW objects and real-time applications are not suitable for reversible execution.

To show which combinations of distribution models and object access are used in existing systems, we will give some examples in the next section before dealing with the specific requirements of collaborative VW.

## OBJECT DISTRIBUTION IN EXISTING VR SYSTEMS

Beside a large number of single-user and local VR systems some distributed VR systems already exist. These support several active users at the same time, as well as concurrent applications. The distribution models used in some of these VR systems are shown below.

### The DIVE System

The distribution within the DIVE VR toolkit [6] directly depends on the use of a separate distribution package. Nevertheless DIVE always uses active replication in combination with locking. At the moment the distribution package ISIS [2] is used to implement the concept of process groups [1]. A new distribution package called SID is still under development and should be available soon.

At each site, which starts a DIVE application, the distribution package has to be set up for site and network distribution. On initializing the first DIVE application at any connected site, the corresponding object files of the selected world will be read once. Several worlds can exist within the system concurrently. When starting any other DIVE application at another site – within the same VW – this site also gets all its object data via the distribution package. This also allows all application processes on each site to exchange data. Changes to objects can only be done by locking the object at each site.

### The MR Toolkit

The MR Toolkit [15] supports distributed multi-user virtual environments by its peers package [16]. In contrast to most other systems, distribution is usually achieved by connecting devices across the network and processing the input events in local systems. However the package also supports sharing of application-specific data between independent applications using an active replication mechanism.

### The GIVEN Toolkit

The GIVEN (GIVEN++) [3] toolkit uses replication on demand in combination with master entities, which cannot been moved. Applications, other than the one which created an object, can get information about an object only on demand and cannot manipulate the object directly. Changes are distributed among the holders of copies by a special manager. It is on the responsibility of each application to resolve objects into a consistent state when receiving an up-to-date copy of an object.

### NPSNET Network Simulator

Like DIVE, the NPSNET [13,18] network simulator uses active replication for 'small' systems, i.e. systems with less than 500 users (one user for each site). The network standard used is DIS [10]. For large systems (more than 10,000 users) a new mechanism is under development, which will use the "mirror world" approach [9].

### The AVIARY System

The AVIARY System [17] pursues a strong object-oriented approach and therefore uses migration. Objects communicate with each other or with the enclosing 'worlds' by messages. The distribution of the objects among the sites as well as the message handling (sending the message to the site where the object currently is) is done by special service processes.

## CONCURRENT INTERACTION

While concurrent interaction is already realized in some CSCW systems, such as GROVE [8], very few existing virtual environments permit both distributed and local concurrent interactions between several users and one or more objects. One of the reasons is that, until now there was no requirement to implement concurrent access. Today the availability of more sophisticated interaction techniques [4], with new input devices and applications, such as collaborative engineering or collaborative design, offers new fields to researchers as well as to users. Another reason can be found in the structure of most existing VR systems: interaction is implemented by directly manipulating the corresponding object data, which implies an explicit permission to write.

In addition, in distributed systems, the problem of signal propagation time delays occurs. This has an adverse effect not only on the real-time performance of the system but also on the ability to realize concurrent interaction.

Distributed virtual environments providing concurrent interactions have to deal with two different kinds of problems: First the concurrent interaction requests have to be detected and second a 'good' mechanism to resolve these requests is necessary.

### Resolving Concurrent Interaction Requests

Concurrent interaction requests, might, if they are not resolved, mutually exclude one another, cancel each other out or lead to inconsistent states. Some interaction requests have to occur concurrently, since this a requirement of the interaction (see the example of carrying a table). How different interaction requests are processed, as well as how many participants may interact with an object concurrently, is highly dependent on the object itself. Beside other more specialized methods, the following alternatives for processing concurrent interaction seemed practicable:

- priority based interaction request resolving
- request time dependent interaction sequencing
- constraint based interaction request resolving
- combining interaction requests

These four possibilities will be discussed below.

*Priority based interaction request resolving* leads to a new sequence of the requests, which will be processed step by step. Thus the request with the highest priority will always be processed while other requests can be postponed or refused. Usually lower priority requests are refused as this ensures that such requests cannot reverse any preceding interaction. Another option is to combine the interactions as long as requests of higher priority are not affected by those of lower priority.

Actually the *request time dependent interaction sequencing* is a special case of the priority based interaction request resolving. Sequencing of concurrent interaction requests is done by sorting them by criteria such as the request time at the individual sites or the request time at the executing site (actually these are FIFO strategies). This works since not even really concurrent requests will really arrive at the same time at the interaction manager. Sequencing may cause the refusal of some requests, since requests postponed will not be processed, depending on the preceding interactions. The interaction sequence can also lead to unintentional results not expected by the senders of the requests.

Resolving interaction requests by the use of *constraints* seems to be a good approach when a few interaction requests are combined with a high degree of freedom. A short example demonstrates this: An object in 3D space has six degrees of freedom: three for the position and three for the direction. One user can fix an object's position by grabbing it. A second user may now turn the object to change its direction, but he or she cannot change its position. The use of constraints leads very quickly to a situation where none of the interaction requests can be satisfied, because the underlying constraints cannot been resolved. Another disadvantage lies in the complex nature of constraint resolution systems. However this possibility also includes the separation of independent interaction requests for the same object. This can be rather simple, e.g. one request is for changing the object's position and one the object's color. In other cases this separation cannot be provided by a general mechanism at all, e.g. if object behavior such as rotation speed can be changed, while the object is grabbed.

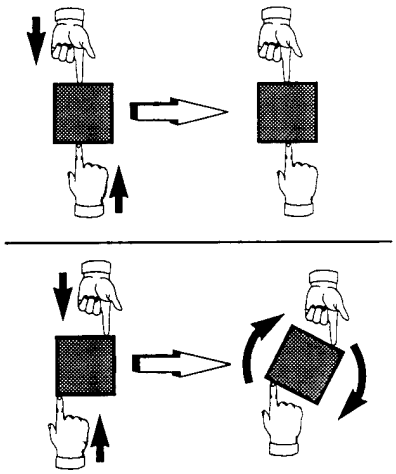


Figure 5: Combining interaction requests.

*Combining interaction requests* involves calculating a new total request from a series of single requests. How this cal-

culation is done depends on the type of interaction as well as on the individual receiving object. An example for this is given in figure 5. As mentioned above this may also be used in combination with priority based request resolving.

Obviously only the last two methods really process concurrent interaction requests. The first two methods only cause a sequencing of the requests. This does not seem to be an advantage compared to systems without support of concurrent interactions, but we have to take into account, that in analogy to multi-tasking operating system, no fixed object connections are established. This means that other, or even new, users can participate in the interaction process. In general a VR toolkit can only support such methods for arbitrary interactions. The last two more complex methods depend too much on each interaction and the involved objects. The VR toolkit has to provide suitable interface functions for concurrent interaction request recognition and parameter transfer. Over and above this it may provide some additional functions, e.g. for the resolving of certain constraints, etc.

### Detecting and Synchronizing Concurrent Interactions

Another important question is: "What is a concurrent interaction request?" or "How can concurrent interaction requests be recognised?". The detection of concurrent interaction requests is difficult for two reasons: First the term *concurrent* has to be defined mathematically. Second it has to be clarified where (at which site) the resulting interaction will take place and which object copy will be modified.

The recognition and processing of the interaction request takes place at exactly one site. Which site this is depends considerably on the distribution method. For example, with identical object copies at all site, it is the site where the interaction with the object started first. When using master entities or migration, the site where the object is located will be chosen. Two interaction requests can be considered as concurrent if the period of time between them is less or equal $\Delta\varepsilon$.

Actually interacting with an object is only possible if some kind of connection has been established. This may be done by selecting the object or by an application process. The maximal number of participants for the interaction can be determined by the object or the application. If two or more interaction connections are established at the same time (from different users), interaction requests are no longer processed immediately. Interaction requests are temporarily stored and a timer is started. After the requests of all participants have arrived, or at the end of the timer period, the interaction is processed. Of course if local request time dependent sequencing is used, the timer is not needed at all and the requests can be processed immediately. There are several options for handling more than one request of one site during this period:

- further requests will be ignored
- new requests will be used instead of preceding ones
- requests are stored and processed after the current interaction
- new requests are combined with (added to) preceding requests

Using the algorithm above, the loss of time $\Delta t_{ia}$ compared to an interaction with only one participant is:

$$\Delta t_{ia} = min\left( \Delta t_{max}, \max_{i=1, i\neq j}^{i=n} (t_{init_i} - t_{init_j}) \right) \qquad \text{(EQ 1)}$$

Where $\Delta t_{max}$ is the time period of the timer, $t_{init}$ is the time when the interaction request arrives at the processing site, j is the index of the participant with the first request of the cycle (starting the timer) and i=1...n are the indices of all participants. The total loss of time at site k (related to the interaction request at site k) is:

$$\begin{aligned} \Delta t_{iarq_k} &= \Delta t_{ia} + \underbrace{t_{init_k} - t_{start_k}} - \left( \underbrace{t_{init_k} - t_{init_j}} \right) \\ &= \Delta t_{ia} + \quad \Delta t_{sig_k} \quad - \quad \Delta t_{init_k} \end{aligned} \qquad \text{(EQ 2)}$$

Where $t_{start}$ is the time of the request at each site, $\Delta t_{sig}$ is the signal delay between the site of the request and the processing site, $\Delta t_{init}$ is the time period between the arrival of the request from site k and the arrival of the first request (site j) at the processing site. Under these conditions (and with a constant signal delay) the total loss of time for the interaction feedback (communication lag) at site k is:

$$\Delta t_{resp_k} = \Delta t_{iarq_k} + \Delta t_{sig_k} \qquad \text{(EQ 3)}$$

To get an adequate feedback it is very important to use as short a timer interval as possible[1]. By determining the individual signal propagation time delays, periodically, for all sites, this interval can be updated depending on the current interaction participants. To avoid large delays, sites with a very large signal propagation time can be suspended from concurrent interactions.

One example for the case of a concurrent interaction with three participating sites (where the object is located at site 1) is shown in figure 6. Intersections are not shown in this figure, although in particular interaction request and their feedback may overlap.

1. A lag of 100 milliseconds or more is considered to be significant for real-time interactions [16].

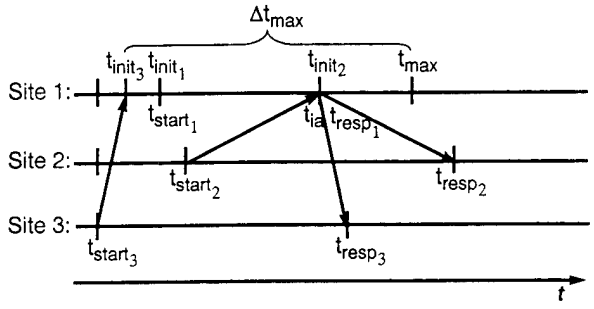



Figure 6: Detection of concurrent interaction requests.

As the example shows, the algorithm used for detection is not correct using the above mathematical definition. A more exact detection can be achieved by considering the network propagation time to calculate the request time at the individual sites. Thus (without the consideration of short-term changes in propagation time) only requests within the interval $[t_{start_i}, t_{start_i} + \Delta\varepsilon]$ would be recognized as concurrent. This would require several different queues for the requests at the processing site, each with its individual timer. The reason for that is, that $t_{init_j} < t_{init_k}$ as well as $t_{start_k} < t_{start_j}$, if $\Delta t_{sig_k} > \Delta t_{sig_j}$.

## DISTRIBUTION IN COLLABORATIVE VE

To facilitate complex services such as concurrent interaction, including an adequate interaction feedback and distribution services for shared databases, different types of communication have to be provided by a VR system.

### Distribution Services for Collaborative VE

Existing distributed VE's usually support one or two different distribution layers: One layer for passing whole objects and one for passing (short) messages. In our opinion at least four different layers are needed for adequate service distribution in collaborative VE. Since no special services for concurrent interactions and interaction feedback exist, these are distributed using a secure but often not fast enough service. Thus, new faster, but less secure distribution services have to be added.

| distribution services | real-time ability | security level |
|---|---|---|
| concurrent interaction ability | high | low |
| feedback of remote object interaction | | |
| messages from and to distributed objects | | |
| creating, migrating and destroying objects | low | high |

Table 1. Distribution services for collaborative VE.

On the top layer (ability for concurrent interactions) real-time ability takes priority over security. Thus fast (mostly insecure) connection strategies are used for that layer. For the application security is not that important, since not all participants of an interaction (see above) have to provide input. In the worst case the interaction has to be repeated.

At the second layer, feedback for interactions with distributed or remote objects is distributed. In this layer (particularly when interacting with very complex objects) it cannot be guaranteed that the feedback of the interaction shows the current state of the object. Nevertheless an almost real-time interaction feedback is extremely important to allow users a more engaging interaction. The user should always be able to have an overview, and see if the current interaction corresponds to the interaction they intended. Thus in this level fast transmission also has precedence over secure transmission. The main difference from the first layer is the use of higher tolerance values until a site is suspended from interaction feedback or direct interaction. If an application needs real-time interaction, and feedback, for concurrent interactions, e.g. for remote manipulations, special network connections and new special protocols will have to be used.

The third layer passes messages to and from distributed objects, e.g. changes of object parameter, as well as selection of an object for interaction or collision propagation. Since the correct transmission of these messages is very important for the consistency of the system, more secure protocols have to be applied. In this layer it may be useful not to send the same data again after an incorrect transmission, but to send the current value only. While this is useful for data such as object color, it may cause serious loss of information for collision signals. One of the major differences between this layer and the fourth layer is the size of the transmitted data. Since messages are usually rather short they can be transmitted much faster (they do not need to be split into several small packages).

The last layer finally refers to the creation, migration and destruction of objects. Secure transmission is the major goal of this layer so that each site has the same view of the virtual world. High security comes along with slow transmission rates, in addition the distributed packages are rather large (whole objects can be very complex). On the other hand the restrictive security priority may slow down the whole system because of one bad (defective) connection. This has to be avoided, e.g. by methods for disconnecting and setting up those sites.

### Application-Dependent Use of Connection Strategies

It does not seem very useful to hard code the use of concrete transfer and connection strategies for the different layers. Each application should rather have the option to change the strategy for each communication layer, depending on its individual requirements (Figure 7). However the phrase: "the more important the real-time performance so the more insecure the used protocol" will always be true. Each application must be able to decide how to handle insufficient hardware and network capabilities. Table 2 shows some pos-

| connection type / strategy type | transmission security | handling temporal connection delays | handling sudden disconnection | connecting to sites with small bandwidth |
|---|---|---|---|---|
| real-time always | low | time-out: temporal site suspend | site suspend reconnect | service refused |
| real-time | low | not detected | site suspend, reconnect | time-out: site suspend |
| real-time if possible | low | not detected | site suspend, reconnect | slow down transmission rate |
| fast always | low | not detected | reconnect, service resume | not considered |
| fast | high | time-out: update, resend | reconnect, update, resend | time-out: disconnect, reconnect, resume |
| fast and secure | high | time-out: update, resend | reconnect, update, resend | scalable time-out values |
| secure | high | time-out: resend | reconnect, resend | time-out: disconnect, reconnect, resume |
| secure always | high | time-out: resend | reconnect, resend | transmit always, slow down system |

Table 2. Possible transfer and connection strategies to fit application dependent requirements

sible connection strategies which could be used to supply the required distribution services.

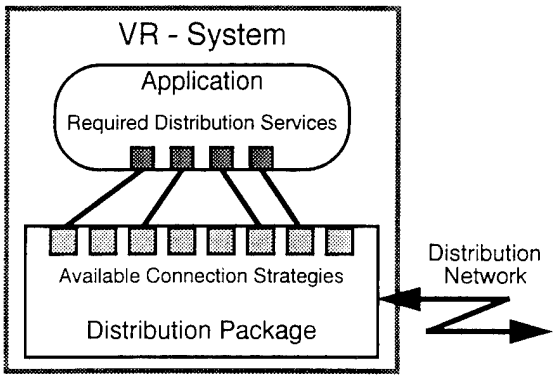



Figure 7: Application dependent use of connection strategies.

The strategies differ in four major ways: The security level of the transmission, the way they handle temporal connection delays and sudden disconnections as well as the treatment of bad (low bandwidth) connections. The strategy type gives an intension of the possible use of the transfer and connection strategies. Secure transmission will probably imply acknowledged transmission and might be extended to identical event orders at all sites (e.g. using the two-phase commit protocol).

Temporal connection delays may not be detected at all, leading to a suspension of the site, or attract a resend after a site or application-dependent time-out. Sudden disconnections can be handled by temporary suspending the site from a certain service, or by resending the information after reconnection. Connections to sites with low bandwidth (slow connections) may be refused, serviced with slower transmission rates, or temporary suspended (as long as high rates are recommended), unless we wish to slow down the whole system.

## CONCLUSIONS AND FUTURE WORK

As we can see in existing VR systems, it is not possible to provide a distributed virtual environment which can satisfy the requirements of all applications for speed (real-time ability) and transmission security. The special needs of collaborative environments, including support for concurrent interactions, have not been considered fully. This paper shows the basic ideas of our approach in dealing with distributed concurrent interactions, as well as the possibilities of selecting suitable types of distribution strategies, for distributed collaborative VE's. The implementation of these concepts is part of the development of a distributed virtual environment toolkit for collaborative applications. One of our goals is to find the criteria which will allow applications to select the best fitting transfer and connection strategy without having to know any details about the hardware or network itself.

## REFERENCES


1. Birman, K.P., Cooper, R., and Glesson, B. Programming with process groups: Group and multicast semantics. Technical Report TR-91-1185, Dept. of Computer Science, University of Cornell, (January 1991).
2. Birman, K.P., and Cooper, R. The ISIS project: Real experience with fault tolerant programming system. In *European SIGOPS Workshop*, (September 1990), and *Operating Systems Review*, (April 1991).

3. Böhm, K., Sokolowicz, M., and Zedler, J. GIVEN++: A Toolkit for Advanced 3D User Interface Construction. In *Virtual Reality Vienna '93 Proceedings*, Vienna, Austria, December 1993.
4. Böhm, K., Broll, W., and Sokolowicz, M. Dynamic Gesture Recognition using Neural Networks: A Fundament for Advanced Interaction Construction. In *Proceedings of the IS&T: SPIE's Symposium on Imaging Science & Technology*, San Jose, (Februar1994), pp. 336-346.
5. Calvin, J., Dickens, A., Gaines, B., Metzger, P., Miller, D. and Owen, D. The SIMNET Virtual World Architecture. In *Proceedings of the IEEE Virtual Reality Annual International Symposium 1993*, IEEE, NJ, 1993, pp. 450-455.
6. Carlsson, C., and Hagsand, O. DIVE–A Platform for Multi-User Virtual Environments. *Computer & Graphics*, Vol.17, No. 6 (1993), pp. 663–669.
7. Codella, C.F., Jalili, R., Koved, L., Lewis, J.B., A Toolkit for Developing Multi-User, Distributed Virtual Environments. In *Proceedings of the IEEE Virtual Reality Annual International Symposium 1993*, IEEE, NJ, 1993, pp. 401-407.
8. Ellis, C.A., Gibbs, S.J., and Rein, G.L. Groupware - Some Issues and Experiences. *Communications of the ACM*, Vol. 34, No. 1, pp. 38–58, January 1991.
9. Gelernter, D., Mirror Worlds: or the Day Software Puts the Universe in Showbox ... How It Will Happen and What It Will Mean, *Oxford University Press*, 1991.
10. Institute for Simulation and Training, Protocol Data Units for Entity Information and Entity Interaction in Distributed Interactive Simulation, Military Standard (DRAFT), IST-PD-90-2, Orlando, FL, (September 1991).
11. Kreifelts, T., Hinrichs, E., and Woetzel, G. Sharing ToDo List with a distributed Task Manager. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work – ECSCW '93*, G.d. Michelis, C. Simone, and K. Schmidt (eds.), Kluwer, Dordrecht, 1993, pp. 31-46.
12. Liang, J., Green, M. Geometric Modeling Using Six Degrees of Freedom Input Devices. In *3rd International Conference on CAD and Computer Graphics Proceedings*, Beijing, China, (August 1993), pp. 217–222.
13. Pratt, D.R. A Software Architecture for the Construction and Management of Real-Time Virtual Worlds. Dissertation, Naval Postgraduate School, Monterey, California, Jan. 1993.
14. Prinz, W., TOSCA: Providing Organisational Information to CSCW applications. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work – ECSCW '93*, G.d. Michelis, C. Simone, and K. Schmidt (eds.), Kluwer, Dordrecht, 1993, pp. 139–154.
15. Shaw, C., Green, M., Liang, J., and Sun, Y. Decoupled Simulation in Virtual Reality with The MR Toolkit. In *ACM Transactions on Information Systems*, Vol. 11, No. 3, (July 1993), pp. 287-317.
16. Shaw, C., and Green, M. The MR Tookit Peers Package and Experiment. In *Proceedings of the IEEE Virtual Reality Annual International Symposium 1993*, IEEE, NJ, 1993, pp. 463-469.
17. Snowdon, D.N., and West, A.J. The AVIARY VR-System. A Prototype Implementation, *6th ERCIM workshop*, Stockholm, Sweden, 1994.
18. Zyda, M.J., Pratt, D.R., Falby, J.S., Lombardo, C., and Kelleher, K.M., The Software Required for the Computer Generation of Virtual Environments. *Presence 2*, 2 (1994).

# An Application of Shared Virtual Reality to Situational Training

Sharon Stansfield
Nadine Miner

Sandia National Laboratories
Albuquerque, NM 87185

Dan Shawver
Dave Rogers

University of New Mexico
Albuquerque, NM 87106

## Abstract

*This paper presents current research being undertaken at Sandia National Laboratories to develop a distributed, shared virtual reality simulation system. The architecture of the system is presented within the framework of an initial application: situational training of inspectors and escorts under programs to verify compliance with nuclear non-proliferation treaties.*

## 1 Introduction

A primary goal of the research being carried out by our group is the development of a virtual reality (VR) platform to support generic applications in situational training. In situational training, a student is taught to handle a variety of different situations, or scenarios. These scenarios may vary from routine events that the student will encounter constantly, to emergency situations that may arise only rarely in the actual job. Toward this end, we are currently carrying out research in several areas of multi-participant, distributed virtual reality and intelligent simulation. Our main emphasis is on applications that require "close quarters" situational training. Close quarters situations involve personnel and action at the individual or team level. Participant's perceive and interact with each other as people, rather than as machines, such as aircraft, or logical groupings, such as battalions. Therefore, it is critical that participants be visible to one another as complete human figures. Also important is the granularity of their movements. For example, participants might need not only to see one another, but also to determine what is being done with arms, hands, or heads. Therefore a VR platform for situational training must provide fully articulated graphical representations of each participant. It must also allow interactions with both the simulation and with other participants which are of a finer detail and greater complexity than do current simulations. SIMNET, for example, allows only gross actions such as maneuvering a vehicle through the environment or pointing and firing a weapon.

The remainder of this paper presents our preliminary system architecture and implementation using a current project in non-proliferation training as an example application. In this application, an instructor and a trainee share a common virtual environment. The system automatically monitors participant's actions and reports when a security infraction, which should have been prevented by the trainee, has occurred. The system allows participants to manipulate objects in the environment, with actions such as grasping, lifting, placing, and dropping. Object behaviors, such as falling, are then invoked.

Related work includes SIMNET [8] and NPSNET [10], both of which are distributed, heterogeneous simulation systems for large-scale battlefield training; the extensive body of work related to the development and use of vehicle simulators, especially flight simulators [7], and NASA's work in using VR for training astronauts to handle repair of the Hubble space telescope [5].

## 2 Application Description

Nuclear non-proliferation treaties often provide for facility inspection to allow verification of compliance. For each participating nation, this requires two types of personnel, facility inspectors and personnel to escort these inspectors. U. S. inspectors must be trained to inspect foreign facilities to determine whether the hosting nation is complying with the terms of the treaty. Foreign inspectors do the same for U. S. facilities subject to such treaties. The time which an inspector is allowed within any foreign facility is limited and, therefore, training the inspector to become familiar with the facility ahead of time would make him or her more effective. The responsibilities of the escort are twofold. U. S. escorts of foreign inspectors (inspecting U. S. sites) must be familiar with the facilities in order to comply with treaty regulations concerning the foreign inspector's right of access. In addition, they must prevent any attempts at espionage by the foreign inspector as the inspection proceeds.

We are exploring virtual reality as a training tool for both inspectors and escorts. To address the problem of limited access to foreign sites, virtual models of foreign facilities can be created to allow an inspector to become familiar with the site before the actual inspection. Availability of information concerning foreign sites is the primary limiting factor.

Access to U. S. sites by escort/trainees is also limited for many reasons. These facilities are geographically scattered, they are secure facilities, and they are



0-8186-7084-3/95 $04.00 

often hazardous environments, due to the operations carried out within them. Virtual models of these U. S. sites can be used to familiarize the escort with a facility before the arrival of an inspector. This is especially important given the fact that short notification of an upcoming inspection is possible.

Both of the above applications rely heavily on the use of virtual environments for architectural walkthrough – allowing a participant to visualize a facility without being physically present in that facility. The training of escorts, however, provides an additional problem domain for the application of VR. The escorts must not only be familiar with the facility, but they must also be trained to detect and prevent security infractions by the foreign inspector. It is this component of the training system on which we are now concentrating.

The escort training system is being developed concurrently with the generic VR platform for situational training applications. This application provides a test example during the implementation and verification of various platform components and provides a set of "real world" guidelines as to what features might or might not be useful in such a platform. Below, we describe the components of the platform in more detail.

# 3 The VR Platform for Situational Training

## 3.1 Geometric Modeling

The virtual environment (VE) used for the escort training work is a model of the hot cell laboratory, which is part of Sandia National Laboratories' Reactor Engineering Center. The facility consists of two laboratories, one containing *glove boxes* and the other containing shielded *hot cells*. Glove boxes and hot cells allow experimenters to handle radioactive materials safely. Figure 1 shows the hot cell VE. This environment was created in three stages. First, the basic architecture was modeled from facility blueprints using Alias Designer$^{TM}$. A visit to the facility was then conducted and a video was made providing comprehensive coverage of the areas being modeled. In stage two, this video was used to model and place other structural details, such as overhead piping, as well as furnishings, such as instrument racks, lab tables, etc. In stage three, the same video was used to create texture maps for walls, floors, signs, instrument dials, and so on. These textures were taken from the digitized video, manually processed using a paint program, and then applied to the hot cell model. Other objects relating to the application were also modeled and placed in the virtual environment, including books, cups, and other everyday objects found in labs, as well as *shrouds* which cover classified items, and unclassified representations of such items.

Modeling of the environment is an independent component of the system. As each virtual environment is built, its model components are added to a reusable object library. This allows us to reuse objects and object components in building other VEs. Architectural items which can be reused are stored in a separate architectural library. The VR platform can import geometric models in several different "standard" file formats, such as DXF. A more comprehensive internal format is used that includes texturing and other surface properties, as well as more complex model definitions.

## 3.2 VR Hardware

A participant may interact with the virtual environment in one of two modes. *Active* participants are entities present in and known to the simulation. They have an associated *avatar*, or graphical body, whose movements are slaved to their own. Active participants interact with the simulation and may cause changes in the state of the virtual world by carrying out actions such as handling objects. Currently, VR gear for active participants consists of a Virtual Research EyeGen3$^{TM}$ headmounted display for immersive viewing, four Polhemus FASTRAK$^{TM}$ magnetic trackers for position and posture tracking, and hand-mounted switches for indicating locomotion and the state of the hand (opened or closed.) In the escort training application, there are two active participants: the escort/trainee and the inspector/adversary. *Transparent* participants may view and move through the virtual environment, but they are not visible to the simulation or active participants. A transparent participant might be a visitor watching the training session, or an instructor controlling the scenario from "behind the scenes." We currently provide transparent participants with an immersive viewing capability via the Fakespace BOOM3C$^{TM}$ which is a full-color, stereo viewer with mechanical tracking of the head and buttons to control motion through the virtual environment. A flatscreen monitor and mouse may also be used by the transparent participant. Figure 2 shows two active participants suited up in the VR gear. The platform also provides audio, in the form of simulation-related sounds and vocal feedback of simulation status, using the sound capabilities of the Silicon Graphics, Inc. (SGI) Indigo$^{TM}$. We refer to the system component which updates and renders a participant's view, including independent changes in world objects and other participants, as the *VR Station*. We currently use SGI platforms with RealityEngine$^{TM}$ graphics to drive our VR Stations. This allows us to utilize texture mapping to increase the realism of our virtual environments.

## 3.3 Representing Participants Within the Virtual Environment

We call the graphical representation of an active participant his or her avatar. Avatars are modeled and controlled using a version of the *Jack*® software developed by the University of Pennsylvania [2]. The *Jack*® software is a constraint-based graphical human figure controller originally developed for ergonomic analysis. It provides a complete and highly articulated graphical human model. The human figure software used for this work is a version of the *Jack*® software that separates the underlying human simulation and modeling component from the general-purpose display and control. This separation allows us to import the human figure model into our software environment

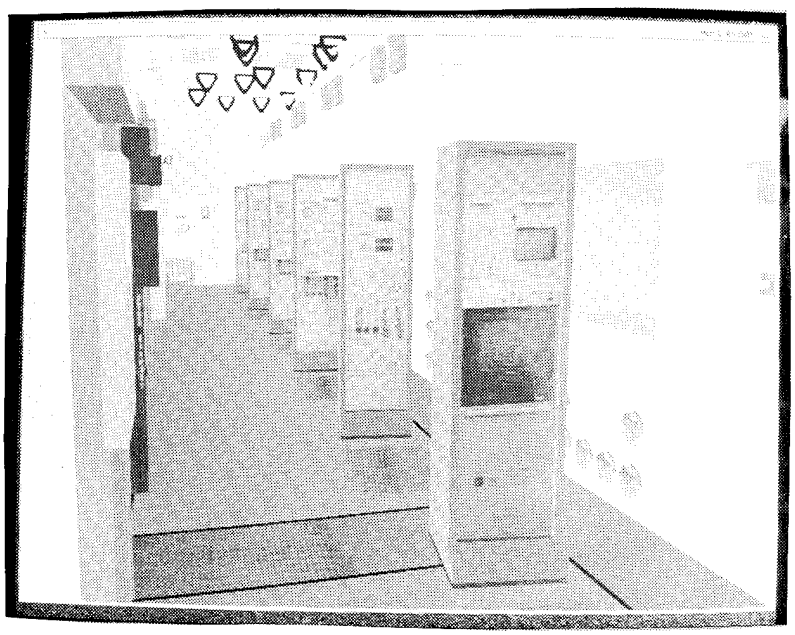


Figure 1: Hot cell lab virtual environment.

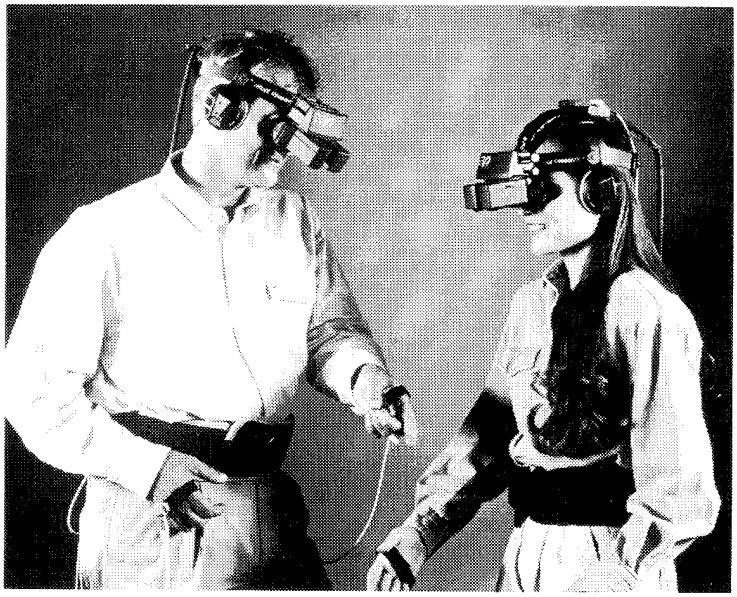


Figure 2: VR gear for active participants.

and to drive this figure using the *Jack*® controller as a server, residing on a separate processor. The posture and position of an avatar is controlled using the input from four Polhemus trackers mounted on the participant's head, each hand, and lower back. As a participant moves, changes in position are measured by the trackers. The *Jack*® server then generates the transforms for the avatar's new posture [1]. Figure 3 shows an avatar within the hot cell virtual environment. The escort training task requires two *Jack*® servers, one for the escort/trainee and one for the inspector/adversary.

### 3.4 Object Behaviors and Simulation Intelligence

Simulation and system intelligence within the generic VR platform may be distributed across any number of heterogeneous processes. The most important of these processes is the *world engine*. At a minimum, the world engine is responsible for consistency and coordination of object interactions, for updating the state of the virtual world at each simulation time-step and communicating these changes to other processes, and for coordinating execution of other simulation components. The architecture of the generic VR platform allows the world engine to be any software module capable of carrying out the indicated tasks. We are currently implementing a world engine using the BE Software Co.'s Behavior Engine™ (BE) software [4]. The BE is an extended, object-oriented software platform that allows behaviors to be defined, stored, and instantiated in the same way as objects in languages such as C++. Our intention is to build a library of reusable behaviors within the BE which may be applied to any application domain. These behaviors include, for example, *grasped* (the behavior of an object when held by a manipulator) and *falling* (the behavior of an object when not supported.) In addition, the BE is used to create application-specific behaviors. One example, in the escort training application, is the *security violation* behavior, which is attached to all classified objects. This behavior logs illegal contact with the object for end-of-session performance review, and vocally informs the participants of the infraction. In our current implementation, the BE-based world engine is also responsible for collision detection and interpretation of some incoming sensor data.

The world engine drives the actions and events within the virtual world. The intelligence required to reason about these actions and events and to invoke the proper responses may reside wholly within the world engine, or it may be distributed among any number of other modules, with the world engine serving as command and control. Within the escort training system, we have chosen to add an additional reasoning module to the system. This reasoner is written in CLIPS (C Language Integrated Production System), a rule-based expert system shell [3]. When events, such as collisions between objects, cause the state of the virtual world to change, the world engine updates its state information and communicates the new states to the reasoner. The reasoner uses this new information to determine the consequences of these events and communicates the resultant state changes back to the world engine, along with requests for execution of any associated behaviors. The world engine updates its own states, carries out the requested behaviors, and communicates the results to all other simulation modules.

Take, for example, the action of a participant grasping an object: the world engine reports to the reasoner that the hand is in contact with the object and that the hand is closed. The reasoner checks for conditions such as the object being liftable and not being held by another participant, etc. It then determines the new state of the hand and object and communicates this back to the world engine along with a request to invoke the grasp behavior (essentially affixing the hand and object.) The world engine also determines the new position of the object as the hand moves and communicates this to all VR Stations.

### 3.5 Communication Within the Distributed System

Figure 4 shows the configuration of the training platform. The platform components are distributed across multiple processors and computers. Many of these components, such as the VR Station and the *Jack*® servers, may have multiple instantiations for a particular application. For example, the escort training application has a VR Station and a *Jack*® server for each of the two participants, along with the world engine and reasoner, for a total of six processes. It may also have one or more transparent viewer VR Stations.

Each instance of a simulation component (eg. world engine and *Jack*® servers) must cooperate with all other components to "cover" its own part of the simulated environment for output to the VR Station. This is accomplished by having an indexed space defined by the world description (currently loaded from a common file by each process at start-up.) Each simulation component is given exclusive use of a portion of the index space. This is done by providing, in a standard order, a common set of shared, unique names for transforms which describe the position of an object in the world. Ethernet multicasting of datagram packets is used to communicate these transforms, in the shared index order, to the VR Stations. This index scheme allows all instances of the VR Station to display a local view of the same virtual world. In addition, it allows a process to communicate only those transforms which have changed since its last communicated update. This minimizes network traffic and processing by any processes receiving this information.

Similarly, sensors are grouped into logical sources, one for each active participant. Each such source has a standard set of potential sensors, with a standard naming order scheme, providing each source with its own index space for multicasting. A simulation process receives sensor input from the source or sources providing its required data (i.e. the *Jack*® server controlling avatar1 receives data from the position trackers attached to participant1.)

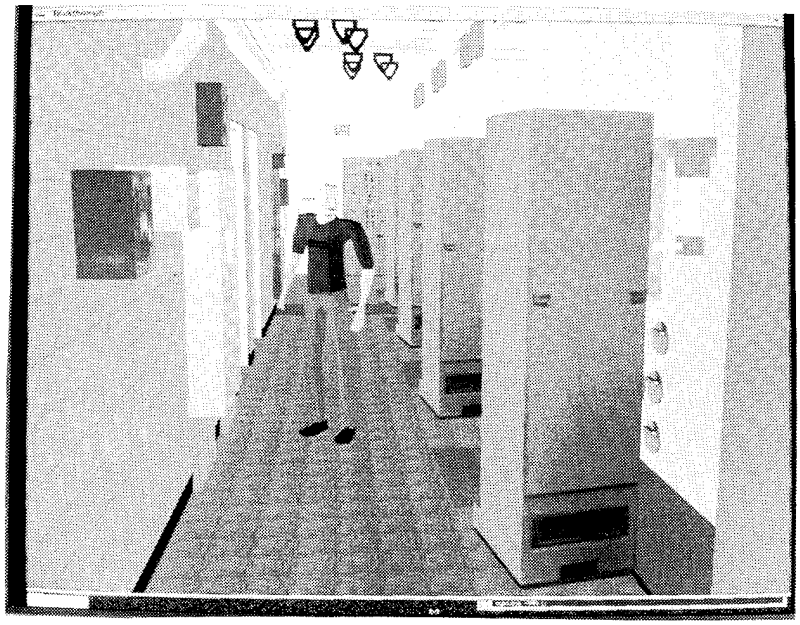


Figure 3: Avatar in hot cell VE.

## 4 Summary and Future Work

This paper has presented preliminary work in developing a generic VR platform for situational training applications. The primary goals of this work are development of a networked VR system that will handle close quarters situational training and the integration of intelligence and behavior-oriented design into the simulation process. We have illustrated the components of this system through an example application: a system to train escorts of foreign inspectors under non-proliferation treaty compliance. This application allows an instructor and a trainee to share a common virtual environment. The system automatically monitors the actions of the participants and reports when a security infraction, which should have been prevented by the trainee, has occurred. The system allows participants to manipulate objects in the environment, for example grasping, lifting, placing, and dropping a cup or book. The proper object behaviors, such as falling, are then invoked. This current implementation is both primitive and limited. It provides enough utility to demonstrate the use of VR for close quarters situational training and to show how it would be applied to a real world application, but it is by no means complete. The number of states and objects is limited, as is the simulation intelligence; behaviors of objects are primitive (eg. no acceleration of a falling object due to gravity); and the simulation cycle and event handling are simple. Developing both the capability of the platform and the complexity of the escort training application which utilizes it is our primary research agenda for the future. In addition, we will be exploring other applications for the platform, such as the training of battlefield medics on the synthetic battlefield and the redesign of an earlier training system for robot operators which uses SILMA's CimStation™ platform as the world engine [6]. A related goal is the integration of a hypermedia component to allow trainees access to relevant information, such as texts and videos, while using the VR platform for training [9].


### Acknowledgements

This work was performed at Sandia National Laboratories and was supported by the U.S. Department of Energy under Contract DE-AC04-94AL85000.

We are pleased to acknowledge the valuable contributions made to this project by the following people: Jim Pinkerton, Lydia Tapia, Meisha Collins and Sofia Pastoriza-Nunez (Sandia National Laboratories); Norman Badler and Mike Hollick (University of Pennsylvania); Chris Goad (BE Software Co.)


## References


[1] "Real-Time Control of a Virtual Human Using Minimal Sensors," N.I. Badler, M.J. Hollick, and J.P. Granieri, *Presence,* Vol. 2, No. 1, Winter 1993.

[2] "Simulating Humans: Computer Graphics, Animation and Control," N.I. Badler, C.B. Phillips, and B.L. Webber, Oxford University Press, 1993.

[3] "CLIPS Reference Manual, Version 6.0" *Technical Report,* Number JSC-25012, Software Technology Branch, Lyndon B. Johnson Space Center, Houston, TX, 1994.

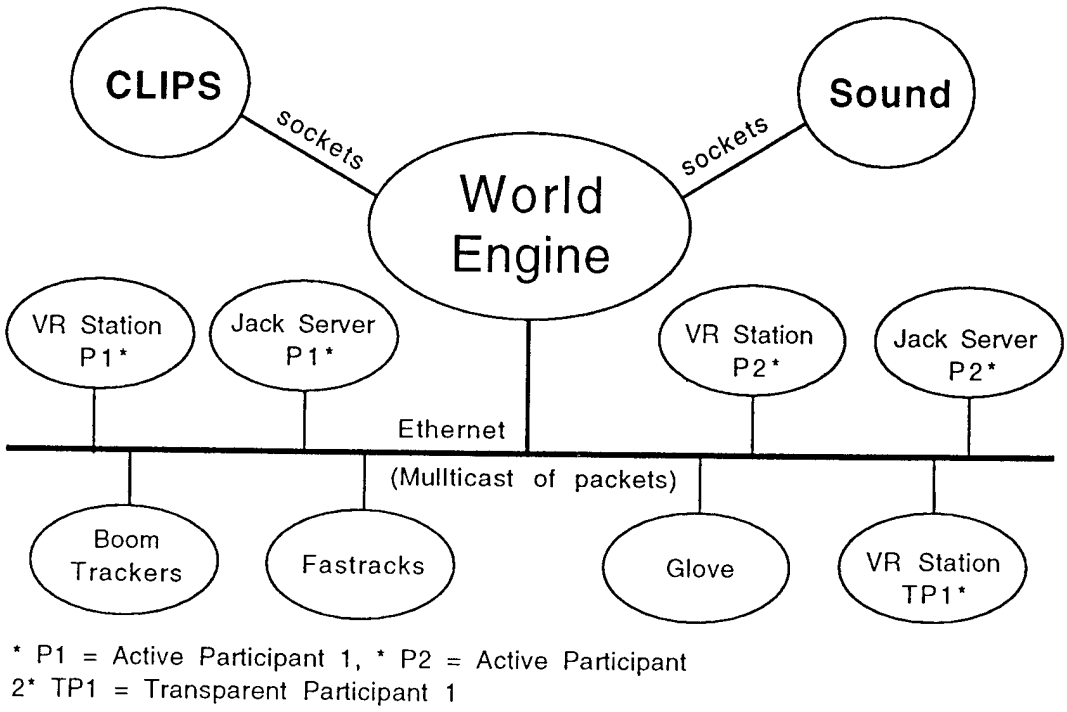



Figure 4: Configuration of the VR platform.

[4] "The Behavior Engine and BEF: A Technical Overview," C. Goad, *Technical Report,* BE Software Co., 1631 NW Johnson St., Portland, OR, 1994.

[5] "Virtual Environments in Training: NASA's Hubble Space Telescope Mission," R. Bowen Loftin and Patrick J. Kenney, Robin Benedetti, Chris Culbert, Mark Engelberg, Robert Jones, Paige Lucas, Mason Menniger, John Muratore, Lac Nguyen, Tim Saito, Robert T. Savely, and Mark Voss, to appear in *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference,* Orlando, FLA, 1994.

[6] "An Interactive Virtual Reality Simulation System for Robot Control and Operator Training," N. E. Miner and S. A. Stansfield, *Proceedings of the IEEE Robotics and Automation Conference,* San Diego, CA, May, 1994.

[7] "A Literature Survey for Virtual Environments: Military Flight Simulator Visual Systems and Simulator Sickness," R. Pausch, T. Crea and M. Conway, *Presence,* Vol. 1, No. 3, Summer 1992.

[8] "The SIMNET Network and Protocols," A. Pope, *BBN Report No. 7102,* BBN Systems and Technologies, Cambridge, MA, 1989.

[9] "A Computer-based Training System Combining Virtual Reality and Multimedia," S. A. Stansfield, Proceedings of the NASA Conference on Intelligent Computer-aided Training and Virtual Environment Technologies, Houston, TX, May 5-7, 1993.

[10] "The Software Required for the Computer Generation of Virtual Environments," M. J. Zyda, D. R. Pratt, J. S. Falby, C. Lombardo, and K. M. Kelleher, *Presence,* Vol. 2, No. 2, Spring 1994.

# A Distributed Virtual Environment for Concurrent Engineering

*John Maxfield, Terrence Fernando and Peter Dew*
The Keyworth Institute of Manufacturing and Information Systems Engineering,
Virtual Working Environments Laboratory, School of Computer Studies,
The University of Leeds, Leeds LS2 9JT,
United Kingdom.
email : {max,ltpf,dew}@uk.ac.leeds.scs

**ABSTRACT**
This paper presents a distributed virtual environment that supports collaboration among members of a geographically dispersed multidisciplinary team engaged in concurrent product development. The distributed virtual environment maintains a shared information space that contains product data in a standard ISO STEP compliant format. It supports a user configurable virtual environment and the integration of different CAE applications to support different engineering perspectives. The realistic manipulation of assembly models within the distributed virtual environment is supported by constraint-based 3D manipulation techniques developed at Leeds. The initial implementation of this architecture supports accurate assembly modelling and kinematic simulation for virtual prototypes and runs on a network of SGI Indy workstations over an ATM network.

**KEYWORDS**: Concurrent Engineering, Distributed Virtual Environments, STEP, Shared Objects, Computer Supported Collaborative Working.

## 1. INTRODUCTION

The pressures for modern design and manufacturing companies to remain competitive in today's world markets has led many to investigate the adoption of concurrent engineering to reduce the lead time for new products and improve their quality. Concurrent engineering is a systematic approach to the integrated concurrent design of products and related processes, including manufacture and support. When using concurrent engineering, specialist knowledge and expertise from downstream tasks of a sequential design process, such as manufacturing and maintenance, are introduced during the early design phases. The largest percentage of design and manufacturing costs are allocated during the first stages of a project. As a result decisions made during these early stages are the most difficult and expensive to correct at a later time. The basic philosophy behind concurrent engineering is to encourage the consideration of as many product development issues as possible, during this crucial early phase. Such considerations should result in fewer unexpected problems during subsequent development and consequently fewer design iterations, reduced development time and costs, and a better quality design.

Given sufficient resources, experts from several product development stages can be introduced by simply making the individuals available for consultation during the design process. An efficient method of managing such interaction is through the creation of multidisciplinary teams. These teams can then have regular meetings to review progress and make important decisions. Such meetings will normally be arranged in advance to give the individuals involved time to prepare necessary documentation and travel to the location of the meeting. During the meetings the experts from different areas of product development will be able to offer advice and suggestions from their own perspectives and can ensure that important issues are not overlooked. However, the physical co-location of a team is no longer a trivial issue. This is because many companies are now exploiting the opportunity to trade in the global world market and consequently are becoming more decentralised in their activities. The travel, time and expense lost as a direct consequence, can inhibit the regularity and spontaneity of team interactions and act as a direct barrier to the successful implementation of concurrent engineering techniques.

One solution to this problem is to develop a real-time collaborative working environment to support such meetings over computer networks. Technology advances such as ATM and advanced workstations have made it feasible to build such real-time collaborative working environments that integrate multimedia and virtual



0-8186-7084-3/95 $04.00 

environments. Such collaborative working environments should support more dynamic and synchronous communications between distributed users, making the geographical dispersion transparent. A team that conducts its work in such a way is *virtually co-located* and thus called a *virtual team*, since interactions only require the participants to be available at the same time, but not necessarily at the same place.

This paper presents a distributed virtual environment that supports collaboration among members of a geographically dispersed multidisciplinary team, who are engaged in concurrent product development. We call such a system a *Distributed Virtual Engineering (DVE) Environment*. In particular the environment allows the team to interact and make decisions from multiple perspectives in a shared information space over accurate virtual prototypes of mechanical components and assemblies. The system uses a user configurable virtual environment and has been designed to allow the integration of different CAE applications for supporting different engineering perspectives. The applications interoperate using shared objects that encapsulate product information in a standard format based on the a standard called STEP (the Standard for The Exchange of Product data, ISO-10303). The management of geometric constraints within the shared information space is supported by a Constraint Manager. This Constraint Manager enables the users to directly manipulate assembly models in the shared information space to carry out interactive assembly modelling operations.

## 2. BACKGROUND

Concurrent engineering has received a great deal of attention in the engineering and management research communities since its introduction over a decade ago and much work has been published on its advantages over more traditional sequential design processes. Major research projects such as DARPA DICE [1], SHARED [2] and PACT [3] are all addressing the issues involved in computer support for managing and co-ordinating multifunctional, cross-disciplinary teams in this context. However these projects have mainly concentrated on the asynchronous activities of such teams. Although this is important, the teams must have regular meetings. This is not a trivial issue if the team is geographically dispersed. The issues that still need to be addressed for synchronous working in concurrent engineering have been indicated in several papers [4,5]. In summary these are:

- The establishment of virtually co-located multidisciplinary teams with integration of and mapping between individual view points.
- The ability to share and exchange standard product data and tools.
- The ability to make collaborative decisions in a single trade-off space with a common understanding of the problems.

Some concurrent engineering projects are beginning to address these issues. The SHARE [6] project is addressing negotiation and trade-off in real-time through video conferencing. Support for synchronous collaboration in virtual teams has also been explored within the DICE project through the use of MONET, a teleconferencing system, and COMIX, a system for transparently sharing X-Windows applications [1]. However, these projects are not investigating the use of distributed virtual environments for supporting collaboration over a virtual prototype of a product and the collaborative tools that are used do not tackle the problems of real-time multiperspective meetings.

There now exist a large number of commercial and non-commercial toolkits for the creation of distributed virtual environments, for example dVS (Division), World Toolkit (Sense8), MR-Toolkit (University of Alberta), and DIVE (Swedish Institute of Computer Science). Many research projects are using such toolkits to develop distributed or single user virtual environments for specific engineering applications [7,8,9]. However, the effective integration of these engineering applications in a multiperspective distributed virtual environment for supporting concurrent engineering has not been addressed. The integration of international product data standards such as STEP in distributed or even single user virtual environments has not yet progressed beyond the ability to access IGES or DXF (AutoCAD) geometric definition files.

Another limitation of current virtual environments is the lack of efficient geometric constraint management facilities. Run-time constraint detection and the maintenance of constraint consistencies for 3D manipulations have not been integrated and, as noted by several researchers [10,12,13], this lack of support for constraints makes it difficult to achieve the accurate 3D positioning of solid models in a 3D environment. Engineering applications, such as 3D solid modelling and assembly modelling, that demand accurate positioning and manipulation are impractical in the current virtual environments. Therefore, techniques such as those developed previously by the authors [14,15] are essential to support such realistic manipulation of solid models within virtual environments.

## 3. THE DVE ENVIRONMENT

The Distributed Virtual Engineering environment has been developed to satisfy a number of requirements that will be discussed in section 3.1. The environment is based on a number of concepts that are outlined in section 3.2. A description of the detailed architecture and its current implementation is presented in section 3.3.

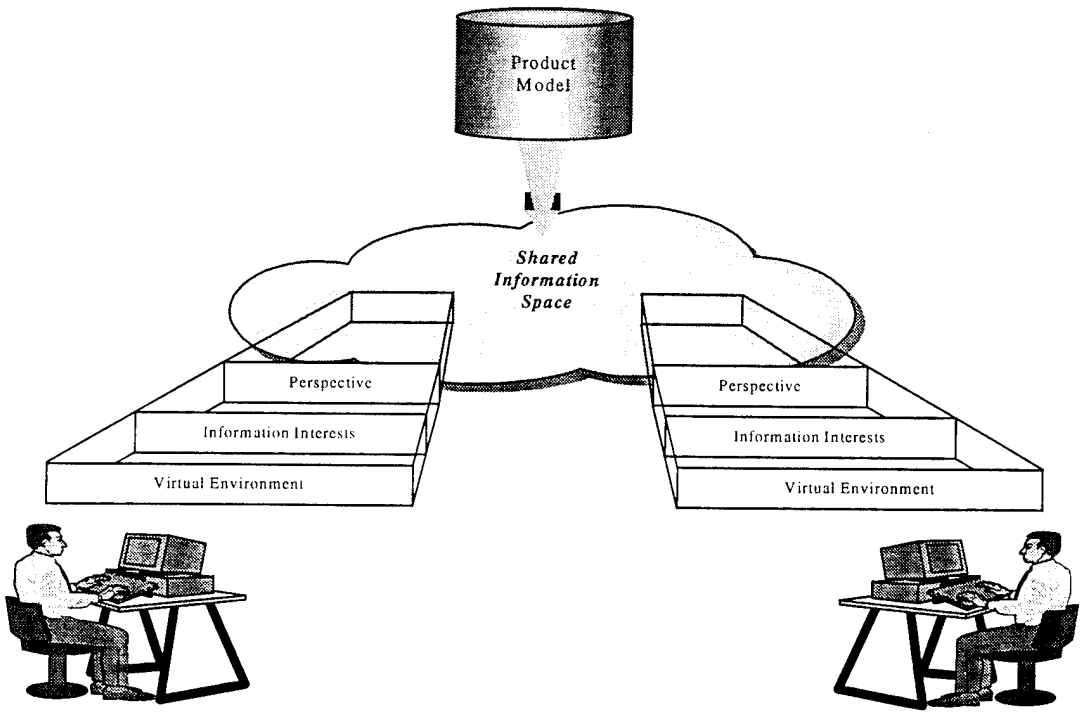



Figure 1: A Conceptual View of the DVE Architecture

### 3.1. Requirements

A fundamental requirement for synchronous collaboration between participants of any distributed meeting, is the ability to share information in *real time*. This shared information will then be the basis for discussion within the meeting. A second fundamental requirement that is specific to collaboration in multidisciplinary teams is the ability to make collaborative decisions in a single trade-off space with a common understanding of the problems and with integration of and mapping between individual perspectives. A *perspective* defines a context within which the shared information can be manipulated in a meaningful way by an individual. In addition, several requirements have been considered during the implementation of the distributed virtual engineering environment. These are, that the architecture should:

- be open and extensible so that different perspectives and engineering applications can be integrated easily.
- be built on emerging standards where possible, including the product modelling standard STEP.
- have the potential to be scaleable for large meetings with a dynamic number of participants.
- provide support for quality of service over high speed networks such as ATM, for real time interaction.

### 3.2. A Conceptual View Of The Architecture

This section introduces a number of fundamental concepts that are used in the DVE environment to satisfy some of the requirements outlined above. For each concept, a high level description of the mechanisms employed by the DVE environment is also presented. A conceptual view of the DVE architecture is illustrated in Figure 1.

The user's *perspective* defines what information is meaningful to them and how they will interact with that information. The DVE environment uses the concept of an *information mask* to define what information is meaningful to a user with a given perspective. The mask acts conceptually as a filter that defines what subset of the information in the shared space a user can actually access. The mask can also be used to filter an entire product model to define what information the user may add to the shared information space. The perspective also defines how the user will manipulate the information they access. To support this, each user of the DVE environment has a user interface that they may *configure* with the operations they wish to perform and the visualisation style they wish to use. These operations are actually performed on the shared information by a set of distributed engineering applications that are invoked and controlled in the background automatically by the DVE environment. The users may modify their perspectives at any time during a meeting by changing the configuration of their interface. This will imply a change in the users' information requirements and consequently a change to the users' information mask. Such changes may also affect the set of background engineering applications that support their perspective.

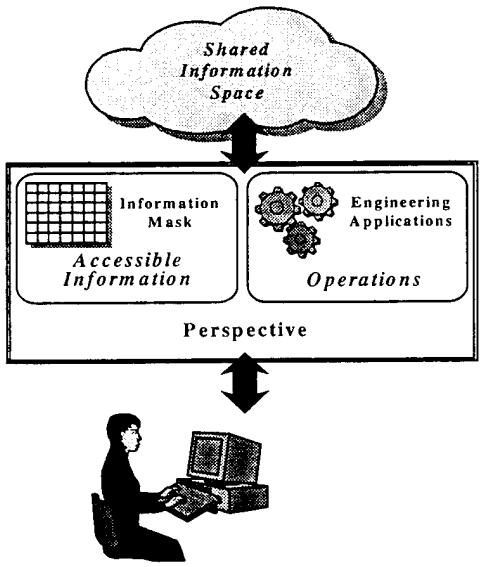
The concept of the perspective and how it is supported within the DVE environment is illustrated in Figure 2(a).

As a meeting progresses the discussion may change focus and more information may be added to the shared information space from the product model. The shared information space can potentially contain a huge amount of information that may be meaningful to a user, and therefore satisfy their information mask, but may not all be of interest. The DVE environment allows a user to choose which subsets of the available information they are interested in and *register* this interest with the environment. Users may change their interest as the meeting progresses by registering an interest in further sets of information or discarding an interest in a particular set of information. Collaboration can occur when users register an interest in the same information, i.e., their particular interests intersect. Internally the DVE environment maintains either a *passive* or *interactive* interest in a particular area of the shared information space for each user. A *passive interest* is automatically registered in all information that the user accesses. An *interactive interest* is registered in any information that the user attempts to modify or manipulate in any way. By distinguishing between passive and interactive interests in this way, locking techniques can be employed by the environment to eliminate any chance of inconsistency in the shared information space. The concepts and mechanisms of interest registration are illustrated in Figure 2(b).

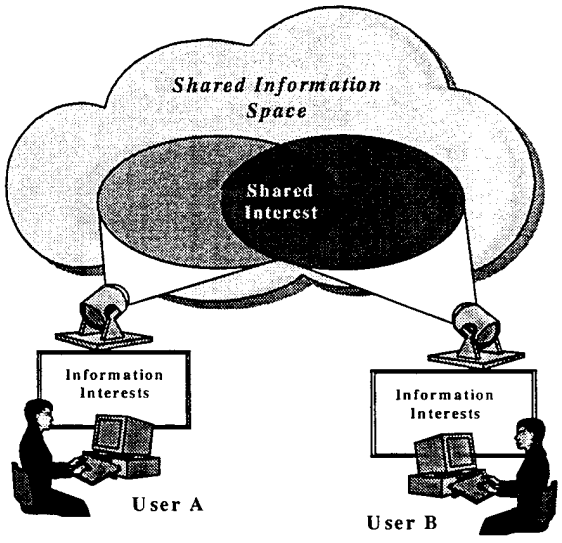
### 3.3. A Detailed View Of The Architecture

A layered view of the detailed DVE architecture is illustrated in Figure 3. At the heart of this architecture is the shared information space that is referred to as the *Product Data Sharing System* (PDSS). This system makes extensive use of a library of sharable objects that can be instantiated and populated with product information. The users access the shared information through a graphical user interface that visualises the information in a virtual environment. They may then interact with the information using operations supported by a set of engineering applications.

*The Product Data Sharing System (PDSS).* This section discusses the structure of the sharable objects and the sharing mechanisms used within PDSS for collaborating over product data. The shared product information must be accessible by many different applications and therefore a neutral, usage independent representation for the information is important. A *product data model* is an integrated set of data schemata that describe such a standard format and content for storing product data. An instance of the product data model will contain data regarding a specific product and is called a *product model.*




a) The Perspective is composed of accessible information and operations.




b) Users register an interest in information held in the shared space. Users dynamically share information that they share an interest in.

Figure 2: The Fundamental Concepts Employed by the DVE Environment

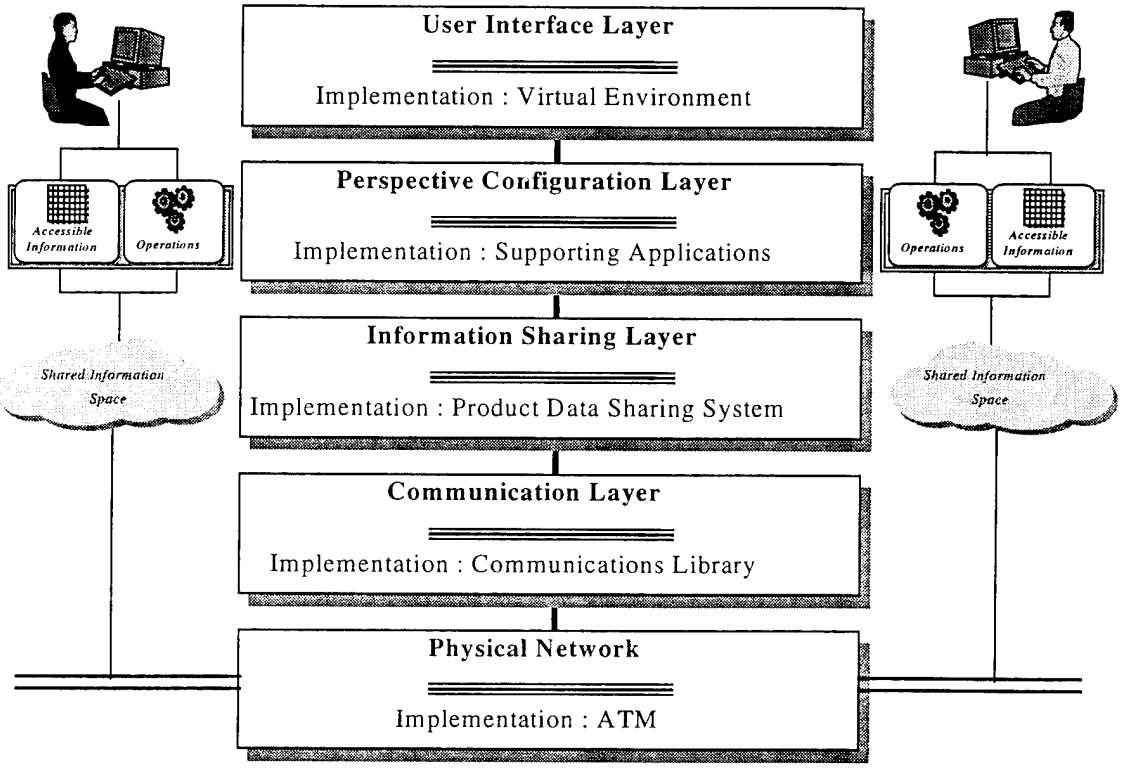



Figure 3: A Layered View of the DVE Architecture

The objects generated and used by the DVE system, encapsulate product information and methods for validating the correctness of that information. Product information is held in an area of the object called the *payload*. Each object in the library consists of a payload (the information that will be shared) and useful methods for encoding and decoding the payload, to assist in sharing the object, that are derived from a basic object type by all of the objects.

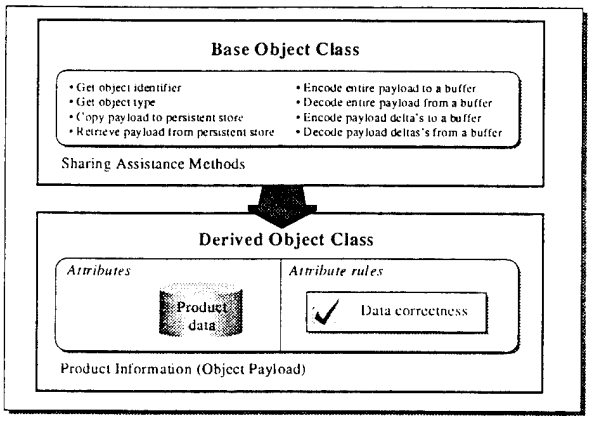



Figure 4: The Shared Object Structure

This object structure is illustrated in Figure 4. Given objects that encapsulate product information structured this way, it is feasible for this information to be retrieved and stored, without any translation, from and to a product model database, if a common product data model is used to structure the information in both. Therefore DVE environment users can add information to the environment from a persistent product model database during the meeting.

At the heart of the PDSS is an *Object Manager*. The manager controls the instantiation, population, shared access and destruction of the objects. It does not distinguish between user interfaces and engineering applications but treats all entities that wish to share product information as *clients*. The object manager will distribute copies of the objects to its client on request. Each client has a *PDSS wrapper* that allows the client to register an interest in some product information using registration services and then view and edit the information using a standard set of *enquiry* and *modification* services. The wrappers will also inform the interface or application of any changes to its local information made by another client, through a set of *notification* services. The PDSS wrappers effectively hide the sharing mechanisms and communicate with the object manager through a standard PDSS protocol that is

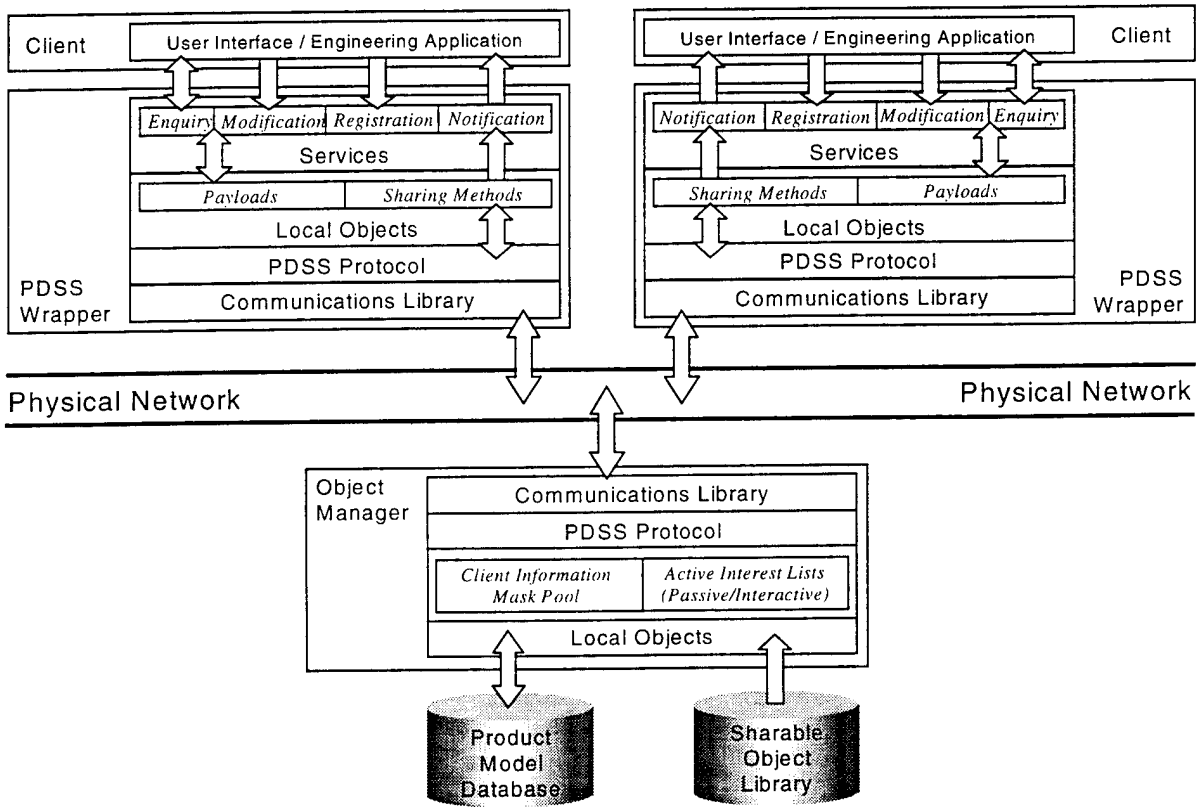



Figure 5: Product Data Sharing Mechanisms

currently built on top of a network and platform independent communications library.

The internal components of the PDSS are illustrated in Figure 5. The object manager and PDSS wrappers maintain local copies of the objects. The responsibility for maintaining consistency between the local copies is shared by the wrappers and the object manager. A PDSS protocol implemented in both the wrappers and object manager defines the interaction that is required to retrieve objects and maintain consistency in them.

*The Configurable Virtual Environment and Supporting Applications.* The DVE architecture supports collaborative working between its users through virtual environment interfaces. Each interface provides a view into the shared product information space managed by the PDSS. The interface visualises components and assemblies as solid 3D models and allows the user to interact with them. Each user can construct a different perspective or personal interface for the virtual environment by configuring the basic interface to support a variety of engineering applications. The user configures the virtual environment by choosing from a list of applications those that they wish to be supported by their virtual environment interface. Each application chosen by a user relates to an engineering application that is integrated into the DVE architecture and invoked as a background process by the DVE environment. The user interface provides an application toolbar from which the user may select and adjust *operations, modes* and *ranged values* for each application selected. Figure 6 illustrates the user interfaces for the toolbar (on the right) and the virtual environment (on the left) in the current implementation.

The user interface and applications exist as different clients of the PDSS and use a special set of shared objects to communicate with each other. These objects are called *transient objects* because the lifetime of the information they contain is only as long as the time the user or application exists within the DVE environment. The application toolbar informs each selected application of the user's interests and request's operations, mode changes, and adjustments to ranged values using the application's transient object. Using this technique, an application can be shared between many users by simply allows many users to register an interest in the applications transient object.

### 3.4. Constraint Management

The management of geometric constraints within the shared information space is supported by a *Constraint Manager*. Constraint-based 3D manipulation techniques,

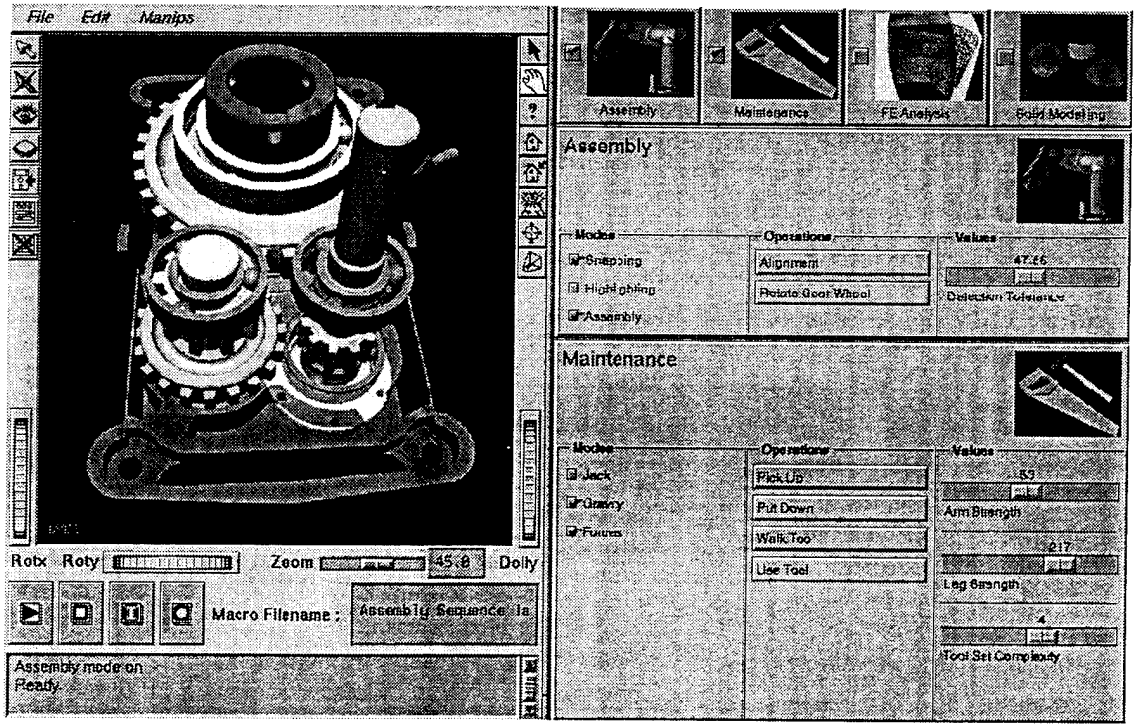



Figure 6: The Virtual Environment Interface and the Gearbox Case Study

developed previously at Leeds [14,15], have been employed within this constraint manager. The constraint manager automatically registers an interest in the information that the users are accessing and monitors the relative positions of the geometric solids as they are manipulated within the virtual environment.

*Automatic constraint recognition techniques* are used within the Constraint Manager to recognise constraints between geometric entities from the user's 3D manipulations. Currently constraints such as against, coincidence, cylindrical fit, gear fit and screw fit can be automatically recognised. Once the a geometric constraints are recognised, the allowable motion of the solid is derived and used to interpret subsequent manipulation the constrained solid without invalidating the previously satisfied constraints. For example, when the user positions a block on top of a larger block using an against constraint, the allowable motion of the block is derived and subsequent manipulations are interpreted as either translations or rotations with respect to the bottom block. In the case of a gear fit between two gears, the rotation of one gear is transmitted to the second gear through a coupled rotation.

The combination of automatic constraint recognition and allowable motion techniques support the accurate positioning of solid models in 3D space using 3D input devices, such as a dataglove or spaceball, during the assembly of complex solid models. Once the objects are assembled, the kinematic behaviour of entire assembly is automatically simulated using the allowable motion of the individual solids. The constraint manager therefore supports a highly interactive virtual environment in which the users can carry out assembly modelling and kinematic simulation of virtual prototypes in a realistic manner.

## 4. IMPLEMENTATION AND RESULTS

The current prototype of the DVE architecture runs on Silicon Graphics Indy workstations over an ATM network. The architecture has been implemented using C and C++ and the current virtual environment interface uses Silicon Graphics Iris Inventor V2.0. The shared object library used in the PDSS is generated by a compiler that has been developed locally by the authors to convert a STEP Express schema into a set of C++ classes. The initial implementation uses part of a product data model that has been developed within a project called MOSES, by the Department of Mechanical Engineering at Leeds University [16].

The current prototype of the DVE architecture supports an assembly modelling perspective through the constraint manager. It enables engineers to register an interest in assemblies within a product model and perform assembly and disassembly operations. A video conferencing system is run along-side the virtual environment to support communication between members of the product development team. Several case studies including a complex gear box model are currently being used to

demonstrate the potential of this environment for collaboration among team members. The current interface with the gear box case study is illustrated in figure 6.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper we have discussed an architecture to support multiperspective collaboration over complex interrelated product information for geographically dispersed virtual teams in concurrent engineering. The current implementation of the DVE environment supports assembly modelling and kinematic simulation of virtual prototypes. When used in conjunction with other collaborative tools, the DVE environment allows geometric and assembly problems to be explained clearly and assists a virtual team in reaching a common understanding of problems quickly. The environment will then also allow the users to discuss and test precise solutions in real time. The architecture is currently being evaluated and refined using case studies.

Work is now underway within the Keyworth Institute to integrate further engineering applications into the system to allow support a wider range of perspectives. These applications include the JACK [11] human factor modeller and a solid modelling kernel based on Parasolid. JACK will allow us to demonstrate a maintenance engineer's perspective more effectively, allowing a user to consider the ergonomic and reachability issues in more detail. The integration of a solid modelling kernel will allow parametric variation of the virtual prototypes in the DVE environment.

A more flexible and powerful communication architecture is also being developed for the DVE architecture to support Quality of Service over ATM, synchronisation and a directory service (based on X.500) to support access to distributed data and users. Experiments involving the scalability of the architecture will study the replication of components of the PDSS and the management of multiple, but related meetings. A trial involving engineers within an real distributed manufacturing company is also being planned in which the DVE system will be used by a distributed team of designers, manufacturers, maintenance engineers and analyst to collaborate during the development of a product.

## ACKNOWLEDGEMENTS

A large number of people have contributed to this work, in particular, special thanks go to Mingxian Fa, Neil Hunter, Jim Baxter, Neal Juster, Mark Pearson and Professor Alan de Pennington of the DVE project within the Keyworth Institute. Thanks also go to Brian Henson from the Department of Mechanical Engineering and Richard Drew and David Morris of the Virtual Science Park Project within the School of Computer Studies.

## REFERENCES

1. K.J.Kleetus, The Virtual Team Framework, *Concurrent Engineering : Tool and Technologies for Mechanical Systems Design*, NATO ASI Series, Springer-Verlag, 1993.

2. A.Wong and D.Sriram, SHARED : An Information Model for Co-operative Product Development, *Technical report*, Intelligent Engineering Systems Laboratory, MIT, Department of Civil Engineering, MIT, Cambridge, MA, USA, July 1993.

3. M.R.Cutkosky et. al., PACT : An Experiment in Integrating Concurrent Engineering Systems, *IEEE COMPUTER*, pages 28-37, January 1993.

4. B.Prasad, R.S.Morenc, and R.M.Rangan. Information Management for Concurrent Engineering : Research Issues, *ISPE Concurrent Engineering : Research and Applications*, pages 3-20, 1993.

5. D.P.Clausing, World Class Concurrent Engineering, *Concurrent Engineering : Tool and Technologies for Mechanical Systems Design*, NATO ASI Series, Springer-Verlag, 1993.

6. G.Toye *et. al.*, SHARE : A Methodology and Environment for Collaborative Product Development, *Workshop on Enabling Technologies : Infrastructure for Collaborative Engineering*, IEEE, 1993.

7. G.M.Bayliss, A.Bowyer, R.Taylor and P.Willis, Virtual Manufacturing, *CSG94 : Set-theoretic Solid Modelling Techniques and Applications*, pages 353 - 365, April 1994.

8. S.Benford *et al*, The Virtuosi project, *VR'94*, February 1994.

9. D.Hancock, Prototyping the Hubbe Fix, *IEEE Spectrum, Special Issue on Virtual Reality : tools, trends and applications*, pages 34-39, October 1993.

10. N.I.Badler and K.H.Manoochehri and D.Baraff, Multi-dimensional Input Techniques and Articulated Figure Positioning by Multiple Constraints, *Workshop on Interactive 3D Graphics*, pages 151-169, 1986.

11. N.I.Badler, C.B.Philips, and B.L.Webber, *Simulating Humans : Computer Graphics, Aniation and Control*, Oxford University Press, June 1993.

12. M.Figueiredo, K.Bohm and J.Teixeira, Precise Object Interactions using Solid Modelling Techniques, Modelling in *Computer Graphics: Methods and Applications*, pages 156-176, June 1993.

13. M.J.Papper and M.A.Gigante, Using Physical Constraints in a Virtual Environment, *Virtual Reality Systems*, Ed. by R. A. Earnshaw, M. A. Gigante and H. Jones, Academic Press Ltd, pages 107-118, 1993.

14. M.Fa, T.Fernando, P.M.Dew, Direct 3D Manipulation Techniques for Interactive Constraint-based Solid Modelling, Computer Graphics Forum, *Proc. of EuroGraphics'93*, Vol.12(3), pages 237-248, September 1993.

15. M.Fa, T.Fernando and P.M.Dew, Interactive Constraint-based Solid Modelling using Allowable Motion, *Proc. of ACM/SIGGRAPH Symposium on Solid Modelling and Applications*, pages 243-252, May 1993.

16. B.Henson, An Assembly Data Model. *Technical Report*, moses-misc-10, MOSES Group, Department of Mechanical Engineering, University of Leeds, U.K, April 1994.

# Calibration and Registration

# Using Texture Maps to Correct for Optical Distortion in Head-Mounted Displays

*Benjamin A. Watson*
*Larry F. Hodges*
Graphics, Visualization & Usability Center
College of Computing, Georgia Inst. Technology
Atlanta, GA 30332 USA
tel: +1-404-894-8787 fax: +1-404-853-0673
watsonb@cc.gatech.edu

**ABSTRACT**

This paper describes a fast method of correcting for optical distortion in head-mounted displays (HMDs). Since the distorted display surface in an HMD is not rectilinear, the shape and location of the graphics window used with the display must be chosen carefully, and some corrections made to the predistortion model. A distortion correction might be performed with optics that reverse the distortion caused by HMD lenses, but such optics can be expensive and offer a correction for only one specific HMD. Integer incremental methods or a lookup table might be used to calculate the correction, but an I/O bottleneck makes this impractical in software. Instead, a texture map may be defined that approximates the required optical correction. Recent equipment advances allow undistorted images to be input into texture mapping hardware at interactive rates. Built in filtering handles predistortion aliasing artifacts.

## 1. INTRODUCTION

The display screens in head-mounted-displays (HMDs) are placed within inches of a wearer's eyes. Between the wearer's eyes and the screens is an optical lens system. The purpose of the lens system is to provide an image to the user that is located at a comfortable distance for accommodation and that is magnified to provide a reasonable field-of view. Unfortunately, the optics also cause nonlinear distortions in the image so that straight lines in the model appear curved in the visual image. The most significant component of this distortion in the widely used LEEP optics is radial distortion [7, 9], which stretches the image away from the lens center. The farther an image element from this center, the more it is stretched. Image elements at the center remain unaffected. Figures 1 and 2 illustrate the effect of radial distortion.

Distortion introduces other complications as well. Because the edges of the HMD raster itself are distorted, using all of the available display space would require clipping to a curved graphics window. A slight misregistration is introduced into stereo pairs [11]. Also, since distortion changes the perceived shape and size of raster pixels, distortion can introduce a perceived blur, and complicate antialiasing. Distortion also has some beneficial side effects. Distorting the image increases field-of-view (FOV) slightly. Furthermore, because the optics do not distort the center of the image, this FOV increase does not come at the expense of central image detail, where viewer attention is usually focused. However, most of an HMD's optical FOV increase is attributable to optical magnification, not distortion.

Robinett and Rolland [14] have described an optical model for HMDs. Using this model, they were able to suggest a method of correcting for HMD distortion with a predistortion. Later, their colleague Gary Bishop implemented this method on Pixel Planes hardware [6]. Predistorting a stereo image frame took roughly 1/20 of a second Essentially, their suggested approach began with a normally generated raster image, and relocated the individual

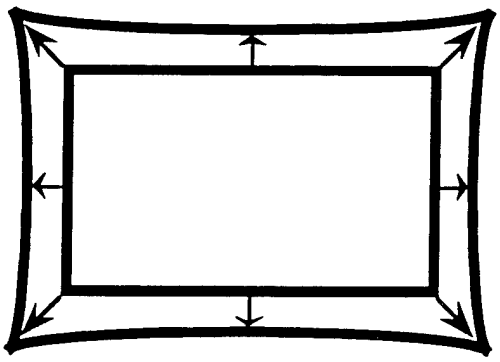


Figure 1: Radial distortion expands the extent of the display as well as the image it contains.

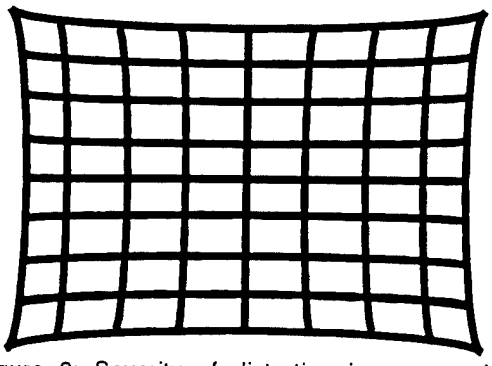


Figure 2: Severity of distortion increases with distance from the optical axis. This example shows a distorted grid.



0-8186-7084-3/95 $04.00 © 1995 IEEE

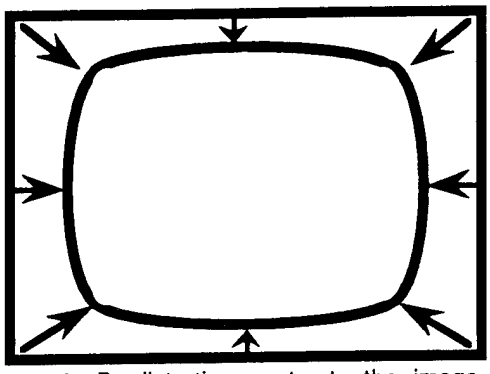


Figure 3: Predistortion contracts the image in proportion to distance from the optical axis.

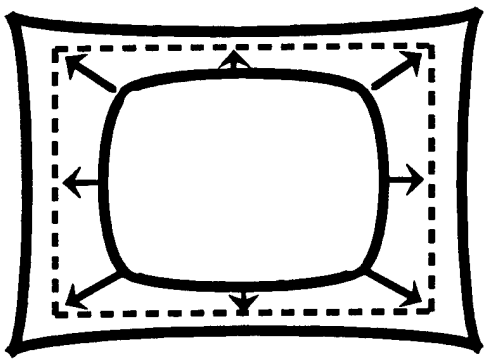


Figure 4: The predistorted image is expanded by the distorting lenses, and fills the undistorted original image's extent.

pixels in the raster*. The effect of such a predistortion is illustrated in Figure 3. When this predistorted image is viewed through HMD optics, as illustrated in Figure 4, it appears to be undistorted. Note that the raster itself is still distorted.

In this paper, we will describe a real-time method of predistortion that uses the model described by Robinett & Rolland. Our goal with this research was to find a real time method for predistortion that works on standard graphics hardware.

## 2. WHY BOTHER CORRECTING DISTORTION?

In Sutherland's early HMD work [16, 17], distortion was not considered a pressing issue. More important was the realization of any sort of device that approximated "the ultimate display." Even later researchers such as Bryson [3], after some experiments, have concluded that compensating for distortion takes too much time, and have moved on to other research issues. Psychophysical studies have shown that the human visual system can correct for displayed distortion [20]. Yet it may not be capable of this correction when the distorting display, like most HMDs, is only intermittently used. In addition, it may be that distortion has a significant effect on the performance of HMD users, especially for tasks that require heavy use of the peripheral areas of the FOV, where distortion is most severe. Efficient correction for distortion is an important step in achieving the proper experimental control for investigating these and other questions.

Early HMDs had horizontal binocular FOVs ranging to well over 100 degrees. But they paid a price for this immersive width: severe distortion and poor resolution. Indeed, the level of visual acuity provided by many of these HMDs fits the legal definition of blindness [3]. Recently, commercial HMD manufacturers have begun to sacrifice FOV in order to achieve greater resolutions and reduced distortion. Some new HMDs have FOVs in the range of 40 degrees. Manufacturers may now have crossed a perceptual breaking point, one at which the perceptual gain of increasing resolution is offset by the loss in FOV [1, 21]. If an efficient algorithmic method of eliminating distortion could be found, HMD designers could be freed from concern about the distortion that accompanies wide FOV, and could concentrate on optimizing other lens parameters. Sharp in Japan is planning to develop a wide FOV HMD, and has shown interest in just such an approach [19].

Perhaps the most important use for an efficient distortion correction would be in the arena of augmented reality, in which virtual objects are superimposed onto a real world view [2, 5]. Many feel that this largely untapped research area could give rise to extremely promising training and visualization applications. One of the most difficult challenges facing augmented reality researchers is the accurate, real-time placement of virtual objects onto the real world view, commonly called the registration problem. A significant component of this problem is the static misregistration introduced by optical distortion. Techniques for eliminating this distortion could have a powerful impact in this field.

## 3. MAKING MAXIMUM USE OF DISPLAY AREA

Before an image can be predistorted, a graphics window must be defined, and the view of the modelled world through this window generated. To maintain psychophysical continuity and make maximum use of available display area, the boundary of this graphics window should match the boundary of the display viewport. However, because distorted HMD screens are not rectangular, achieving this continuity with HMDs is more complicated than with traditional CRT displays. We defined the graphics window as the largest rectangle that fit inside the outline of the distorted screen.

In order to fill this graphics window, it was necessary to make adjustments to the distortion correction proposed by Robinett and Rolland. That correction is an exact reversal of the modelled distortion, resulting in a graphics window that has the same bounds as the undistorted display. However, since the undistorted display is smaller than the distorted display, this correction does not make good use of the available display space (see again Figure 4) or fill the graphics window we defined. Our adjustments increased the size of the predistorted image so that, when viewed through

* This information was obtained through the author's professional correspondence with Gary Bishop of the University of North Carolina.

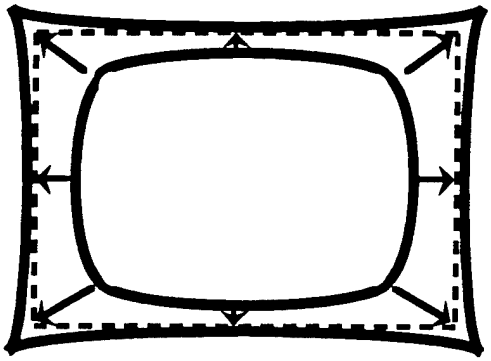


Figure 5: Adjusting the predistorting correction allows the entire distorted display extent to be utilized.

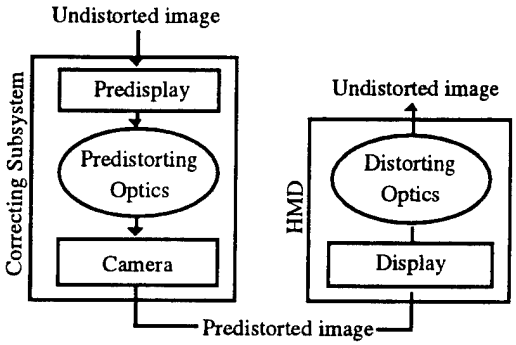



Figure 6: A hypothetical correcting subsystem would pass the undistorted image through a correcting lens before display on an HMD.

the HMD lenses, it filled our graphics window (Figure 5). We outline these adjustments below.

The boundaries of our graphics window depend on the HMD's distortion, modelled by these equations:

$$x_{vn} = x_{sn} + k_{vs}x_{sn}(x_{sn}^2 + y_{sn}^2)$$
$$y_{vn} = y_{sn} + k_{vs}y_{sn}(x_{sn}^2 + y_{sn}^2).$$

In the above notation, the distortion constant $k_{vs}$ is a parameter of the optics being used. $(x_{sn},y_{sn})$ locates the pixel on the HMD raster, while $(x_{vn},y_{vn})$ locates the pixel location after optical distortion. For both of these coordinate systems, the origin is located at the optical axis of the HMD. If the normalized right boundary of the undistorted HMD screen is $R/w_s$, then the right edge of our graphics window has the distorted boundary

$$R_d = R/w_s + k_{vs}(R/w_s)^3.$$

Calculations of the distorted boundaries $L_d$, $T_d$ and $B_d$ of the other sides of the graphics window are similar. The resulting window is larger than the undistorted HMD screen, and may have a different aspect ratio from the undistorted screen. To take account of this new aspect ratio in our correction, we renormalized the graphics window boundaries by dividing each of $L_d$, $R_d$, $T_d$ and $B_d$ by $\max(L_d,R_d,T_d,B_d)$, and substituting them into the original model in place of the undistorted window boundaries $L/w_s$, $R/w_s$, $T/w_s$ and $B/w_s$.

The severity of radial distortion increases with distance from the optical axis. Because our graphics window's boundaries are farther from the optical axis of the HMD than the boundaries of the undistorted HMD screen, distortion within our graphics window is more severe than distortion within the undistorted display screen. To take account of this increased distortion, we increased the magnitudes of the distortion constant $k_{vs}$ and the predistortion constant $k_{sv}$ slightly.

## 4. APPROACHES TO DISTORTION CORRECTION

One very straightforward approach to correcting for optical distortion is optical rather than digital. Before a normally generated image is presented on a distorting display, it can be passed through a set of lenses that reverses the distortion introduced by the display. One could, for example, place a video camera with correcting lenses in front of a computer display, and send the output of the camera to an HMD. This method does not introduce any latency into the rendering pipeline* and skirts many of the filtering problems introduced by digital predistortion. However, this approach does have its drawbacks. New correcting lenses are required for each new distorting lens. The camera setup described is not very stable, and if correcting lenses were incorporated into HMD displays, they could increase the bulkiness and expense of a display that already has questionable ergonomics. Nevertheless, an HMD system with a separate predisplay and correction component (see Figure 6) might prove useful. See [4] for a description of a similar system.

A digital predistortion of an image may be performed with the use of the following equations, which approximate the continuous relocation performed by optical predistortion:

$$x_{sn} = x_{vn} + k_{sv}x_{vn}(x_{vn}^2 + y_{vn}^2) \quad (1)$$
$$y_{sn} = y_{vn} + k_{sv}y_{vn}(x_{vn}^2 + y_{vn}^2), \quad (2)$$

where $k_{sv}$ is the predistortion constant.

Since the equations (1) and (2) are essentially parametric cubics, the predistorting relocations might be calculated dynamically and efficiently using well-known curve rasterization methods [10, 18]. For example, after perspective projection, polygon vertices could be predistorted and their edges rendered as predistorted curves. If the equation for the line defined by such an edge is

$$y_{vn} = cx_{vn} + d, \quad (3)$$

the parametric equation that defines its predistorted equivalent is obtained by substituting equation 3 into equations 1 and 2:

*Note that most modern cameras integrate light for 1/30 of a second, acting as a frame store.

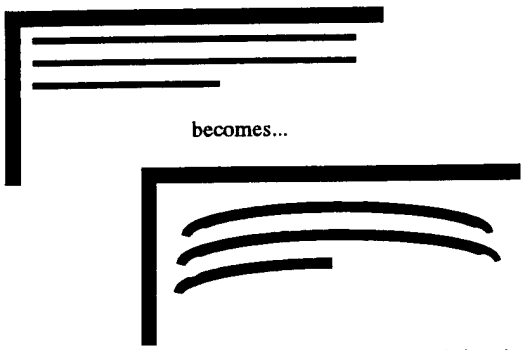



Figure 7: Undistorted scan lines might be predistorted using integer-only rasterization methods.

$$x_{sn} = (k_{sv}c^2+k_{sv})x_{vn}^3 + (2k_{sv}cd)x_{vn}^2 + (k_{sv}d^2+1)x_{vn}. \quad (4)$$

$$y_{sn} = (k_{sv}c^3+k_{sv}c)x_{vn}^3 + (3k_{sv}c^2d+k_{sv}d)x_{vn}^2 + (3ck_{sv}d^2+c)x_{vn} + (k_{sv}d^3+d). \quad (5)$$

For accuracy, polygon fill and shading routines used with this method would have to be adjusted to account for predistortion. Because this approach would have a variable complexity dependent on world geometry, we call it the geometry-dependent approach.

Rolland and Hopkins [13] are experimenting with a algorithm related to the geometry-dependent approach that uses a lookup table to predistort vertices. Rather than rendering predistorted edges, however, they minimize the edge distortion by subdividing large screen polygons and predistorting the new vertices.

Alternatively, after an undistorted image was rendered, each image scan line could be rendered as a predistorted curve (Figure 7). Because this method would have a fixed complexity dependent on image size, we call it the image-dependent approach. The predistortion of the zero-slope scan lines corrected in this approach is an especially simple case, making this image-dependent method easy to implement in hardware. By setting the slope c to 0 in equations 4 and 5, we arrive at these much simpler equations:

$$x_{sn} = k_{sv}x_{vn}^3 + (k_{sv}d^2+1)x_{vn} \quad (6)$$

$$y_{sn} = k_{sv}dx_{vn}^2 + (k_{sv}d^3+d). \quad (7)$$

Predistortion for both the geometry- and image-dependent approaches could be adjusted for different HMDs by simply passing new screen boundary and distortion severity parameters to the algorithm.

Finally, since the predistortion for the HMD being used is usually well-known, it might be more efficient to precalculate the relocation of each undistorted pixel into a table, if sufficient memory is available. Regan and Pose have implemented such an approach in hardware [12]. Their specialized architecture incorporates a pixel relocation lookup table that allows any raster image to be predistorted to correct for distortion in any sort of lens.

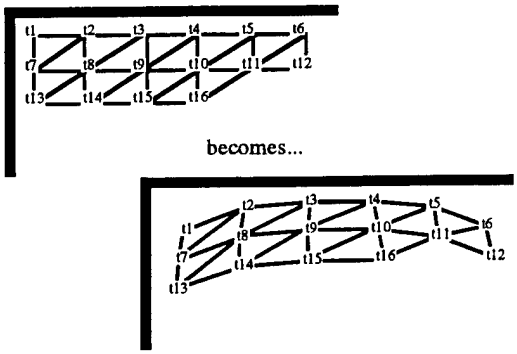



Figure 8: The vertices in a rectangular mesh textured with an undistorted image may be predistorted with equations 1 and 2. Since the texture indexes are unchanged, this predistorts the undistorted image.

## 5. EXPERIMENTS WITH TWO OF THESE APPROACHES

We implemented both the table-based approach and the dynamic image-dependent approach in software on a Silicon Graphics Crimson Reality Engine, and tested them with 320 x 240 images. The predistorted scan lines generated by the image-dependent method had very small slopes over most of their length, and as a result the predistorting relocation of successive scan line pixels exhibited a significant amount of coherency. We were able to take advantage of this fact to reduce the amount of memory used to store relocations in our implementation of the table-based method to 25K.

Surprisingly, the speed of these two approaches did not differ greatly. With each of these approaches we were able to predistort six graphics window frames per second, clearly inadequate for real time graphics applications. Testing showed that most CPU time was spent transferring frames between frame buffer and main memory. This explained the lack of any significant difference in speed between the two approaches, and suggested that hardware implementations of these algorithms could be quite effective.

All of the digital predistortion approaches discussed here introduce image filtering complications that worsened with the severity of the modelled distortion. We experimented with pixel by pixel blending on the predistorted image as a solution and found it inadequate. The resulting images had a snowy, speckled look. We obtained better results when we used the Z-buffer value of the undistorted image pixels to resolve relocation pixel conflicts. While we did not encounter this problem in our experimentation, this Z-buffering approach could conceivably result in binocular rivalry (right/left eye conflicts) in stereoscopic displays. Both of these filtering solutions were simple software approaches dictated by our real-time demands. A more complete solution was implemented by Regan and Pose in their specialized graphics architecture, which passes the image through a linear filter after pixel relocation.

## 6. PREDISTORTING WITH TEXTURE MAPS

Predistortion can be viewed as a special case of texture mapping [8]. If the undistorted image is a texture, and the

TABLE 1:
*Predistorted Performance*

Performance of predistorting algorithm with textured and untextured world views, and with differing undistorted source image resolutions. Predistorted views were always rendered monoscopically at 640 x 480 resolution. Reported predistortions per second are not geometry or world dependent and are highly variable. Reported frame rates include time required for rendering of undistorted source view, and so are dependent on the size of the rendered world, which contained 8211 flat shaded polygons. When texturing was used, 1588 of these polygons were mapped with 128 x 128 textures. The world contained five different textures. The predistorted view was a textured mesh with 400 vertices, filtered bilinearly. All textures were stored in a two byte internal format.

| Textures? | Yes | | | No | | |
|---|---|---|---|---|---|---|
| Source Res | 640x480 | 512x256 | 256x256 | 640x480 | 512x256 | 256x256 |
| Frames/Sec | 10 | 14 | 19 | 11 | 14 | 19 |
| Predist/Sec | 17 | 20 | 40 | 19 | 23 | 77 |

TABLE 2:
*Normal Performance*

Rendering performance for an undistorted virtual world rendered monoscopically at 640 x 480 resolution. The textured and untextured worlds used for these tests were the same worlds used for the tests reported in table 1.

| Textures? | Yes | No |
|---|---|---|
| Frames/Sec | 20 | 25 |

display surface a polygon, predistortion is simply a mapping of a texture onto a polygon. One salient difference between predistortion and traditional texture mapping is that unlike most textured polygons, the display surface in predistortion must have a new undistorted image mapped onto it for each new image frame. Nevertheless, this analogy provides a useful perspective on several of the problems we discussed earlier, and suggests a new approach to performing predistortion.

We can model the undistorted image space with a flat, two-dimensional polygonal mesh. Paired with each vertex in the mesh is an index into the corresponding point in the undistorted image (the texture) itself . We then apply the predistorting equations (1) and (2) to the vertices of the mesh, leaving the texture indexes unchanged. Figure 8 illustrates this process. If the mesh is then rendered onto the display screen, a predistorted image will appear.

Until recently, it was not possible to load new textures into texturing hardware at interactive rates, and as a result, undistorted images could not be predistorted with this technique in real time. But a new and largely undocumented feature on high-end Silicon Graphics machines [15] has made this predistortion approach practical. Undistorted images can be rendered to the frame buffer and addressed directly by texturing hardware.

We have implemented this technique on a Silicon Graphics Onyx Reality Engine II, and tested it on a virtual world containing 8211 flat shaded polygons. When a textured view of the world was rendered, five 128 x 128 textures were mapped onto 1588 of these polygons. The predistorted mesh contained 400 vertices, which in our experience struck a good balance between over-sampling and over-interpolating. All textures were stored with a two byte internal format. Our results are displayed in tables 1 and 2. Our scan converter required 640 x 480 image input, and so the undistorted world was always rendered at 640 x 480 resolution. Figure 9 shows a textured and undistorted world view. While the predistorted view of the world rendered with predistortion was always rendered with 640 x 480 resolution, we varied the resolution of the source undistorted image at three levels: 640 x 480, 512 x 256, and 256 x 256. Figure 10 shows a predistorted image rendered with Figure 9 as its source texture. Figures 11 and 12 show an undistorted image rendered at 256 x 256 resolution and a matching predistorted view.

Not surprisingly, rendering an undistorted view of the world was faster than rendering a predistorted view of the world. Note that adding textures to the predistorted world had little impact on frame rate. Adding textures to the undistorted world, however, significantly impacted performance. This indicates that a large part of predistortion's impact on performance is due to its addition of texturing to an otherwise untextured world. By reducing the size of source textures, one may trade image fidelity for significant improvements in performance. This would be particularly appropriate when the resolution of the displays in the HMD being used is less than NTSC.

Filtering has long been an important concern of texture mapping researchers, and accordingly most texturing systems provide many different filtering options, at various levels of image quality and speed. We found that use of larger and more complex filters improved predistorted image quality without significantly impacting overall performance.

When considering the quality and accuracy of the images produced with this approach, it is important to realize that the equations 1 and 2, which form the basis for all the techniques discussed in this paper, are themselves only approximations of the necessary optical predistortion. The predistortion techniques discussed in sections 4 and 5 digitally sample these approximating functions. Texture-based predistortion samples the approximating functions much more sparsely than the techniques in sections 4 and 5.

## 7. CONCLUSION

This paper has described a method of correcting for optical HMD distortion in real time. The method makes use of widely distributed texture mapping hardware. As texture

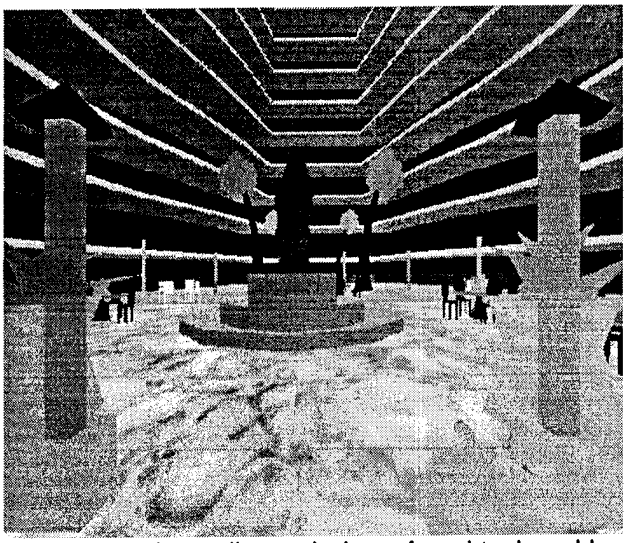


Figure 9: An undistorted view of a virtual world rendered at 640 x 480 resolution.

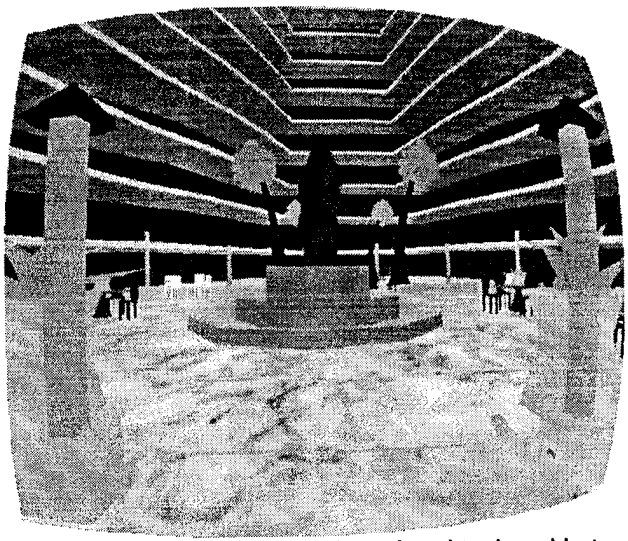


Figure 10: A predistorted view of a virtual world at 640 x 480 resolution, using the image in figure 9 as its texture.

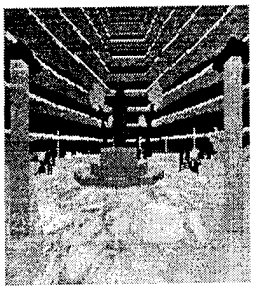


Figure 11: An undistorted view of a virtual world rendered at 256 x 256 resolution.

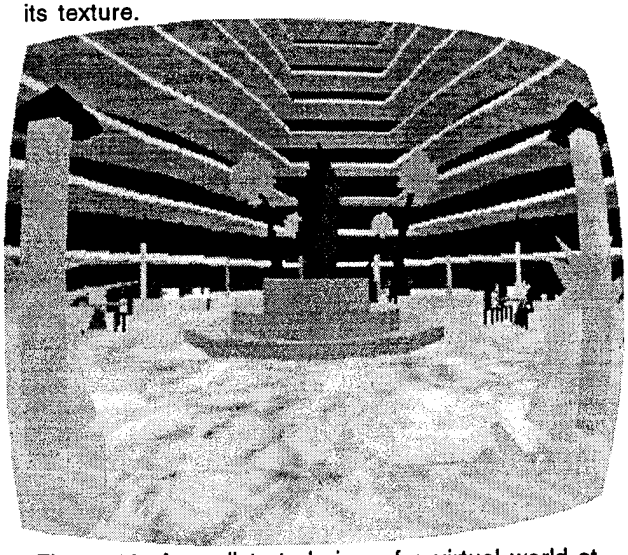


Figure 12: A predistorted view of a virtual world at 640 x 480 resolution, using the image in figure 11 as its texture.

mapping hardware improves, this technique will gain speed and utility.

## 8. ACKNOWLEDGMENTS

I am indebted to Gentaro Hirota of IBM Japan for discussions suggesting the possible application of texture maps to this problem. My reviewers made several valuable suggestions. This work was supported in part by the NSF Summer Institute in Japan program, a Link Foundation Fellowship for Advanced Simulation and Training and a Georgia Tech Foundation equipment grant.

## 9. REFERENCES

1. Alfano, P. and Michel, G. Restricting the field of view: perceptual and performance effects. *Perceptual and Motor Skills 70* (1990), 35-45.
2. Azuma, R. and Bishop, G. Improving static and dynamic registration in an optical see-through HMD. *Proc.Computer Graphics '94* (July 24-29, Orlando, FL), 197-204.
3. Bryson, S. Implementing virtual reality. *SIGGRAPH 1993 Course 43 Notes*, 1.1.1-1.3.26.
4. Edwards, E.K., Rolland, J.P. and Keller, K.P. Video see-through design for merging of real and virtual environments. *Proc. IEEE VRAIS '93* (September 18-22, Seattle, WA), 223-233.
5. Feiner, S., MacIntyre, B. and Seligman D. Annotating the real world with knowledge-based graphics on a see-through head-mounted display. *Proc. Graphics Interface '92* (May 11-15, Vancouver, BC), 78-85.
6. Fuchs, H. et al. Pixel Planes 5: a heterogeneous multiprocessing graphics system using processor enhanced memories. *Computer Graphics 23*, 3, (July '89) 79-88.
7. Hecht, E. and Zajac, A. *Optics*, Addison-Wesley, Reading, MA, 1974.

8. Heckbert, P. Survey of texture mapping. *IEEE Computer Graphics and Applications 6*, 10 (November '86), 56-67.
9. Howlett, E. Wide angle color photography method and system. U.S. Patent No. 4406532, 1983.
10. Klassen, R. V. Integer forward differencing of cubic polynomials. *ACM Trans. Graph. 10*, 2 (April 1991), 152-181.
11. Min, P. and Jense, H. Interactive stereoscopy optimization for head-mounted displays. *Proc. SPIE, Stereoscopic Displays and Virtual Reality Systems 2177* (1994).
12. Regan, M. and Pose, R. Priority rendering with a virtual reality address recalculation pipeline. *Proc.Computer Graphics '94* (July 24-29, Orlando, FL), 155-162.
13. Rolland, J. and Hopkins, H. A method of computational correction for optical distortion in head-mounted displays. *Univ. N. Carolina at Chapel Hill Tech Rpt. 93-045*. Available on the web at site http://www.cs.unc.edu.
14. Robinett, W. and Rolland, J. P. A computational model for the stereoscopic optics of a head-mounted display. *Presence 1*, 1 (Winter 1992), 45-62.
15. Silicon Graphics, on-line man pages on commands texdef2d and fbsubtexload on systems with IRIX 5.1.1.1 or later.
16. Sutherland, I. E. The ultimate display. *Proc. IFIPS Congress* (1965), 2, 506-508.
17. Sutherland, I. E. A head-mounted three-dimensional display. *Fall Joint Computer Conf., AFIPS Conf. Proc. 33* (1968), 757-764.
18. Watson, B. and Hodges L.F. Fast algorithms for rendering cubic curves. *Proc. Graphics Interface '92* (May 11-15 1992, Vancouver, BC), 19-28 .
19. Watson, B. A survey of virtual reality in Japan. *Presence 3*, 1 (Spring 1994), 1-18.
20. Welch, Robert B. Adaptation of space perception. In Boff, K.R., Kaufman, L. and Thomas, J.P. (1986). *Handbook of Perception and Human Performance*, ch. 24. Wiley Interscience, Chichester, UK, 1986.
21. Wells, M. and Venturino, M. Performance and head movements using a helmet-mounted display with different sized field-of-view. *Optical Engineering 29* (1989), 8, 870-877.

# Ultrasonic Calibration of a Magnetic Tracker in a Virtual Reality Space

Morteza Ghazisaedy
David Adamczyk
Daniel J. Sandin
Robert V. Kenyon
Thomas A. DeFanti

Electronic Visualization Laboratory (EVL)
Department of Electrical Engineering and Computer Science
and School of Art and Design
The University of Illinois at Chicago
851 S. Morgan, Room 1120 SEO (M/C 154)
Chicago, IL 60607-7053
(312)996-3002
Email: sandin@eecs.uic.edu

## ABSTRACT

*This paper describes a system for calibrating the position component of a 6-degree-of-freedom magnetic tracker by comparing the output with a custom-built ultrasonic measuring system. A look-up table, created from the collected difference data, is used to interpolate for corrected values. The error of the resulting corrected magnetic tracker position is measured to be less than 5% over the calibrated range.*

**Keywords:** Virtual reality, CAVE, magnetic tracker, ultrasonic tracker

## I. INTRODUCTION

### A. Purpose and Motivation

A goal of any virtual reality (VR) system is to make user control of the environment as natural as possible. Accurate tracking is needed for VR systems to generate correctly sized and oriented perspective views, to allow user picking of objects, and to facilitate navigation.

### B. Background

There are 3 major types of tracking devices that detect both position and angular orientation.

**1 - Mechanical Linkages.** Mechanical linkage systems use an arm-like structure composed of several joints with one end fixed and the other end free to move with the user. These devices measure the position and angular orientation of the free end by measuring the angles at each joint of the structure, factoring in the length of each segment. The BOOM by Fake Space Labs uses such a linkage setup well.

Advantages include low latency and the potential of high positional accuracy. Disadvantages derive from the limited extent of movement determined by the total length of the arm, and the inertia of the structure (especially with a BOOM monitor attached) [3]. In addition, using a second mechanical linkage system to capture the user's hand information is highly tangle-prone.

**2 - Ultrasonic Systems.** Ultrasonic systems have two major components, a transmitter generating an ultrasound signal and a receiver detecting the signal. The distance is calculated by measuring the time-of-flight of the ultrasonic pulse. Three transmitters and receivers are needed to calculate a full 3D position and orientation [2]. A major

disadvantage is that an unobscured path from the transmitter to the receiver needs to be maintained. Two systems that use ultrasonic tracking are the Power Glove manufactured by Mattel and the 3D Mouse by Logitech [3].

**3 - Orthogonal Electromagnetic Field.** Orthogonal field systems use magnetic fields to determine position and orientation. A transmitter generates electromagnetic signals which are received by a sensor. The strength of the electromagnetic signals are used to determine the absolute position and orientation of the receiver relative to the transmitter. The advantage is that this type of tracker allows arbitrary movement in a relatively large (8 ft. radius) space. On the other hand, such trackers exhibit substantial delay and increased inaccuracy with distance from the transmitter. Two well known versions are the Polhemus 3-Space and the Ascension Flock of Birds.

## II. CAVE VR SYSTEM

### A. CAVE Overview

The CAVE is a Virtual Reality system developed at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago. The current CAVE is ten feet on a side [4] where two or three walls are rear-projected and the floor is projected down from above. The Ascension Extended Range Transmitter Flock of Birds (ERTFOB) magnetic tracker is used to measure the position of the user's head and a hand-held position device called the "wand."

### B. Tracking System

The electromagnetic transmitter has 3 orthogonal coils which are pulsed in sequence. The receiver also contains 3 orthogonal coils which measure the components of the electromagnetic signal. The strength of the 3 components of the received pulse are compared to the strength of the transmitted pulse to determine the position. The strength of the 3 received signals are compared to each other to determine the orientation (thus the receiver coil most parallel to the transmitter coil will give the highest value and the one most orthogonal will give the lowest). For each position, the transmitter sends three pulses, one for each of its coils. The three receiver coils each get 3 pulses, for a total of 9 signals. The range is claimed to be up to an 8 ft. radius from the transmitter. Unfortunately, the accuracy of the system decreases markedly as distance from the sensor to the transmitter increases [4].

Metal structures near the tracker distort the magnetic field, so the CAVE screen frame is made of austenetic stainless steel which is non-magnetic and has a low conductivity. However, other components needed for the CAVE to function such as projectors and mirrors significantly distort the field [4].

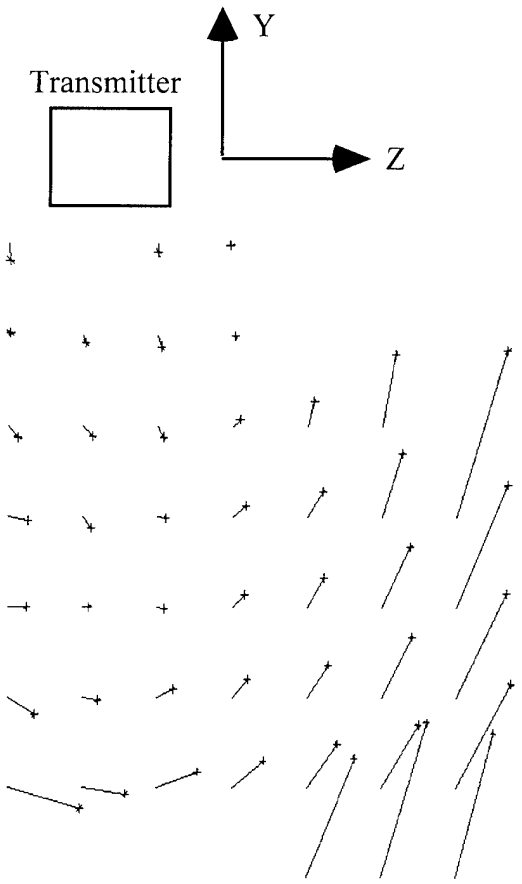



Figure 1. ERTFOB errors in one plane of the CAVE, viewed from the left side. Transmitter is one foot in front of this plane.

## III. CALIBRATION METHOD

### A. Overview

The goal of the calibration system is to correct for ERTFOB static position errors. For each position reported by the magnetic tracker, the physical position of the sensor is measured using our more accurate ultrasonic measuring device (UMD). A table is built containing positions of the magnetic sensor reported by the ERTFOB and their corresponding positions reported by the UMD. Using this

table, any point within the range can be corrected by interpolation between the corrected points in the calibrated area. The table must be rebuilt whenever the tracking system or the CAVE is moved.

Figure 1 shows the position errors for a plane placed one foot right of center, vertically oriented and perpendicular to the front wall of the CAVE. The positions as measured by the UMD are shown by the x's at the end of the lines whose other ends are the positions as measured by the ERTFOB. Figures 2 and 3 show in 3D all the planes at once. Each figure has two stereograms: the left pair for cross-eyed viewing and the right pair for wall-eyed viewing.

## B. The Ultrasonic Measuring Device

The UMD generates an ultrasonic sound signal using a transducer and sends it toward an object. The sound reflected from the object, or echo, is also detected by the transducer. Distance is obtained by measuring the time interval between the moment the sound is transmitted and the echo is received. The elapsed time between the transmission and echo signal is a linear function of the distance [5].

To measure position in all 3 dimensions, 4 Polaroid ultrasonic transducers are used, one to measure the distance to each wall and the floor of the CAVE. The distance to the left and right walls is measured by two transducers and gives the X coordinate, the distance to the floor gives the Y coordinate, and the distance to the front wall gives the Z coordinate. Two transducers are used redundantly for the X coordinate to detect yaw error by checking that the sum of the two distances (left and right) are equal to the distance across the CAVE (10 ft.). If the sum is the greater than 10 ft., the left and right transducers are not perpendicular to

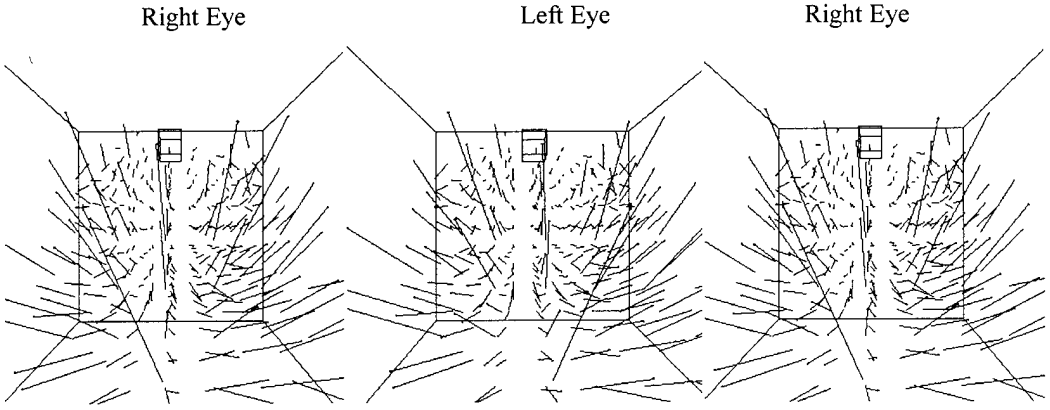



Figure 2. Stereogram of ERTFOB errors viewed from back of CAVE.

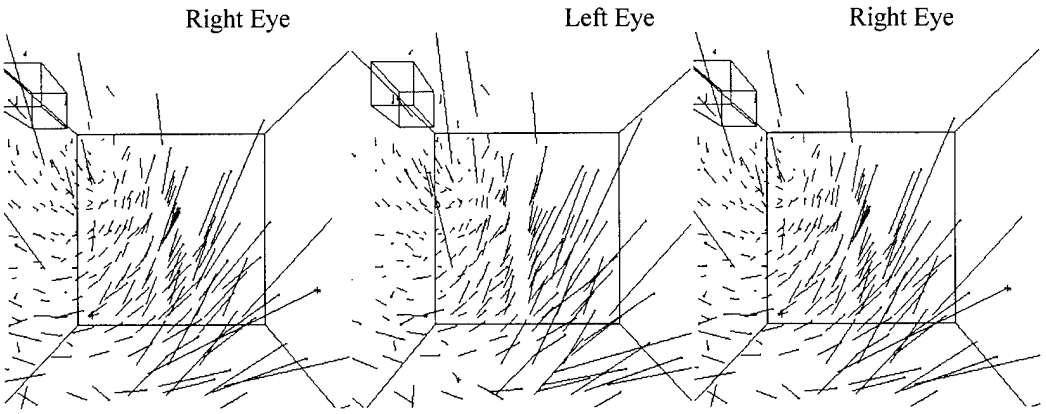



Figure 3. Stereogram of ERTFOB errors viewed from left side of CAVE.

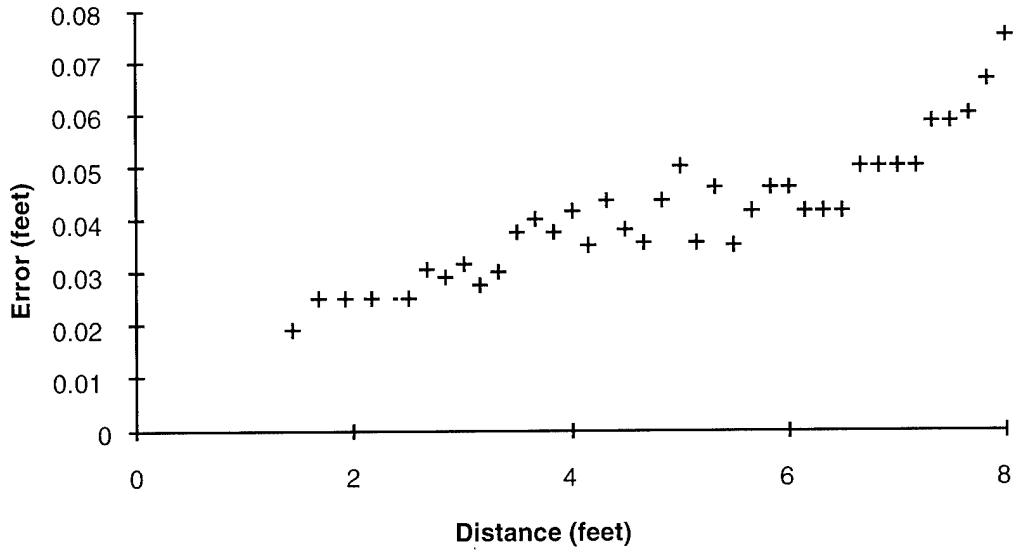



Figure 4: UMD error versus distance.

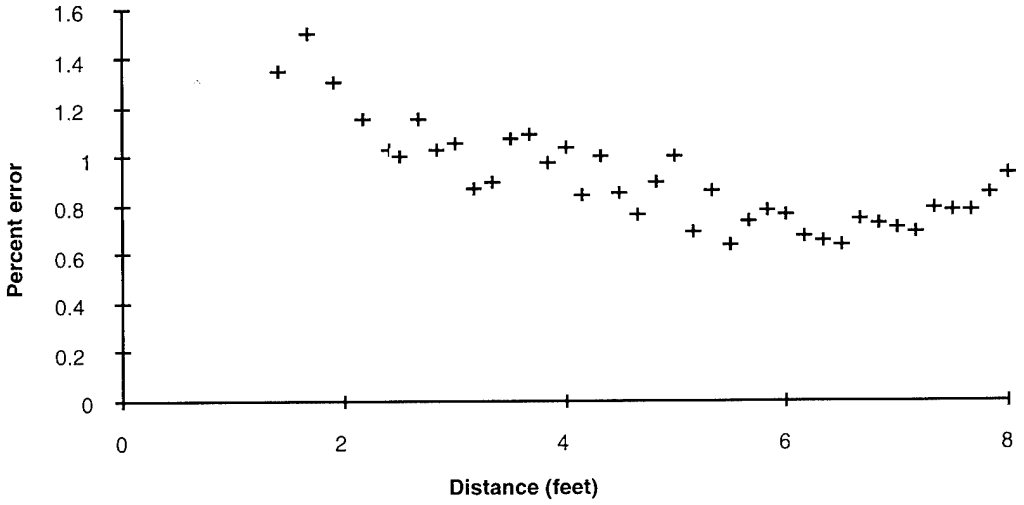



Figure 5: Percent UMD error versus distance.

the CAVE walls. In addition, 4 mercury switches are used for each side of the transducer box as a level. Whenever the transducers are pitched or rolled more than 10 degrees, the system will not record data. The Polaroid transducers report the shortest distance within ±10 degrees of perpendicular to the target [8].

To calibrate the UMD, we use an optical bench and a target approximately 4 feet square. Using a velocity value of 347 m/s[1], the actual distance and the readings given by the UMD were compared. The overall error of the UMD is less than 1.5% (Figure 5), so in the 10 ft. CAVE, the maximum error computes to be 1.8 inches.

[1] The speed of sound at 0° C is 331m/s. As the temperature increases so does the speed of sound. The relationship between temperature and speed of sound is given by: $V = 20.034\sqrt{273 + t}$ where V is the speed of sound in meters/second at a temperature t in centigrade [9]. Since the temperature at EVL was 27° C we used the value 347m/s for the speed of sound. If more accuracy is desired, one can measure the temperature and adjust the measured distances accordingly [11].

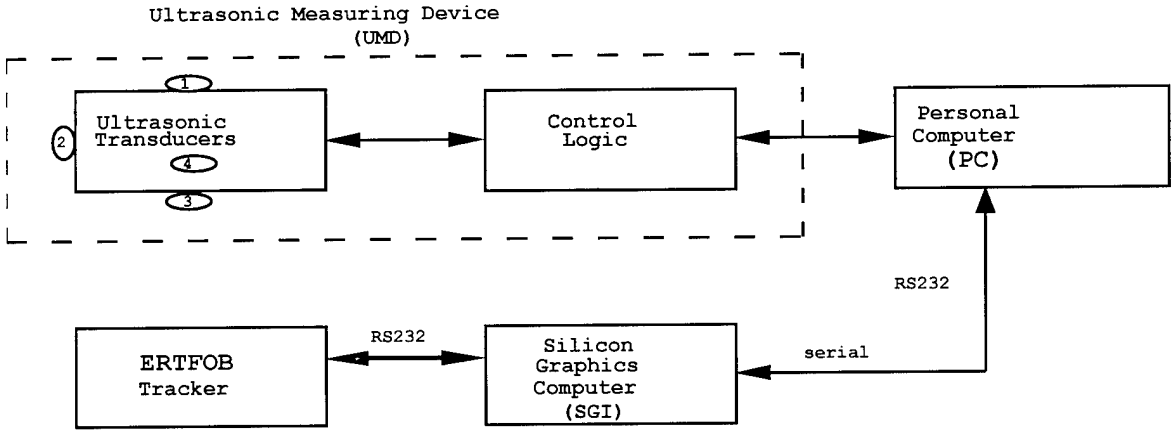



Figure 6: UMD block diagram

Figure 7: Photo of UMD .

The advantages of this ultrasonic measurement system are good linearity over large distances, insensitivity to magnetic fields, more accuracy than the magnetic tracker, and its relatively low cost. Disadvantages include the lack of angular orientation information and the need to keep the transducers parallel to the walls and level (which prevents its use as a primary tracking mechanism, but works well for calibration purposes). In addition, the signal path must not be physically blocked.

Figure 6 shows a block diagram of the entire system [5]. The transducers are controlled by a circuit which uses the Texas Instruments ultrasonic ranging module TI2728 [10]. This circuit, controlled by the PC, sends pulses to each ultrasonic transducer in sequence to eliminate false echoes from the other transducers. It measures the elapsed time between transmitted and received pulses, and then transfers this data to the PC. The PC sends the data to the Onyx CAVE computer through an RS-232 line. The UMD circuitry itself does not affect the magnetic field, because it is located well outside the tracker range and the small UMD box upon which the ERTFOB sits is plastic and contains only the ultrasonic transducers. Empirical observations bear this out as well.

## C. Calibration Procedure

The CAVE is first filled by a 3D stereo graphic image of 1-inch boxes on 1-foot intervals (Figure 8). A 1-inch cursor shows the position of the magnetic sensor which is placed atop the ultrasonic transducer housing. A person wearing 3D glasses holds the UMD reasonably straight and moves it until the displayed cursor is inside of each box. The program records the position given by the ERTFOB and the Onyx sends a signal to the PC to get the position measured by the UMD. This procedure continues until all the boxes in the tracker range inside the CAVE are thus sampled. In practice less than 400 points are collected, essentially all points in the center of the CAVE. The collected points are not exactly at one foot intervals as measured by the ERTFOB, but lie somewhere inside the 1 inch box at that point, since trying to get the cursor on the exact point is nearly impossible. As we show below, this error is largely taken into account.

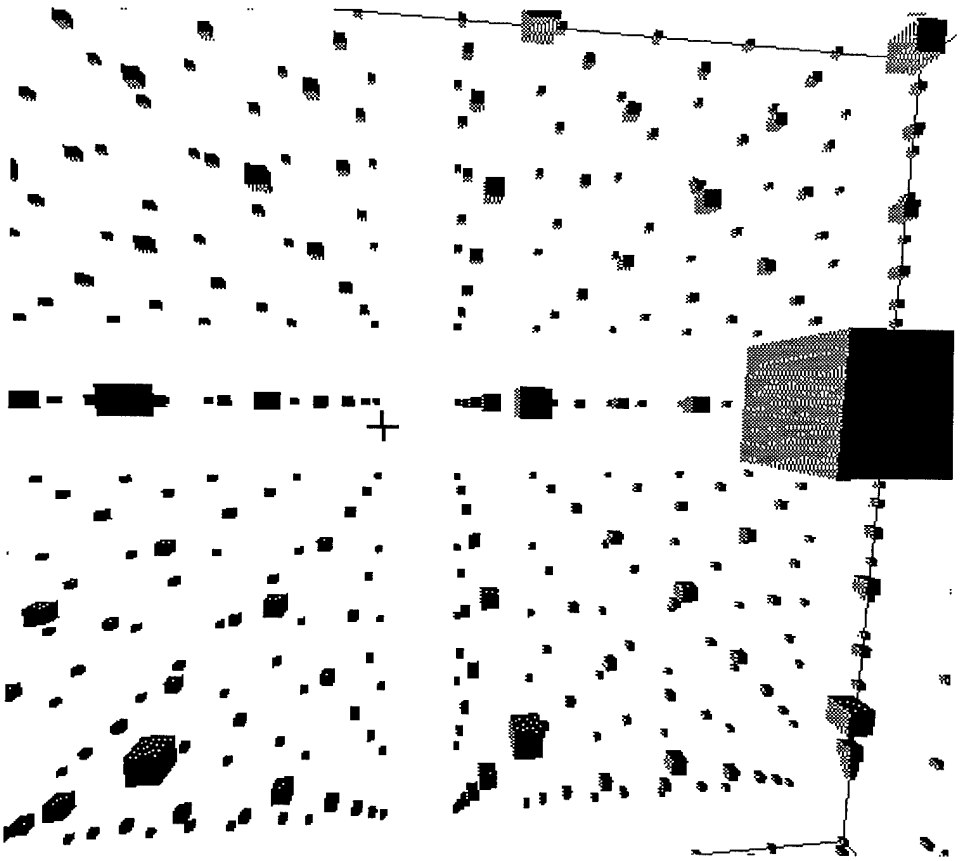


Figure 8: Display in the CAVE of boxes to collect.
Cursor (crosshairs) is moved inside boxes to collect point.

## IV. CORRECTION METHOD FOR TRACKER

### A. Look-Up Table Generation

In previous work [2], the magnetic tracker was moved by constant physical steps and the tracker output was recorded. To create a look-up table of corrections, this data matrix must be inverted. We, instead, create a look-up table of corrections directly by collecting simultaneous tracker output and the actual values as read by the UMD at constant steps. To simplify human performance requirements in the data collection phase, we record data as soon as the tracker is within one inch of the ideal position. Of course, the tracker is measuring how far it is off the mark, so assuming that the tracker is differentially correct for distances of less than an inch, that amount is subtracted from the distances from both the UMD and the ERTFOB. The maximum distance from a collected point to the calibration point is 0.866 inches. We subtract this distance vector from both the UMD and ERTFOB position. Figure 9 shows the 2D case.

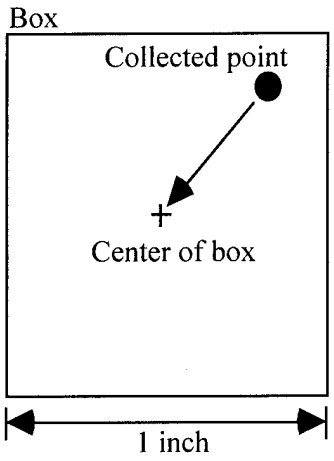



Figure 9: 2D shift. Collected point is shifted by the Cartesian distance to the center of the box

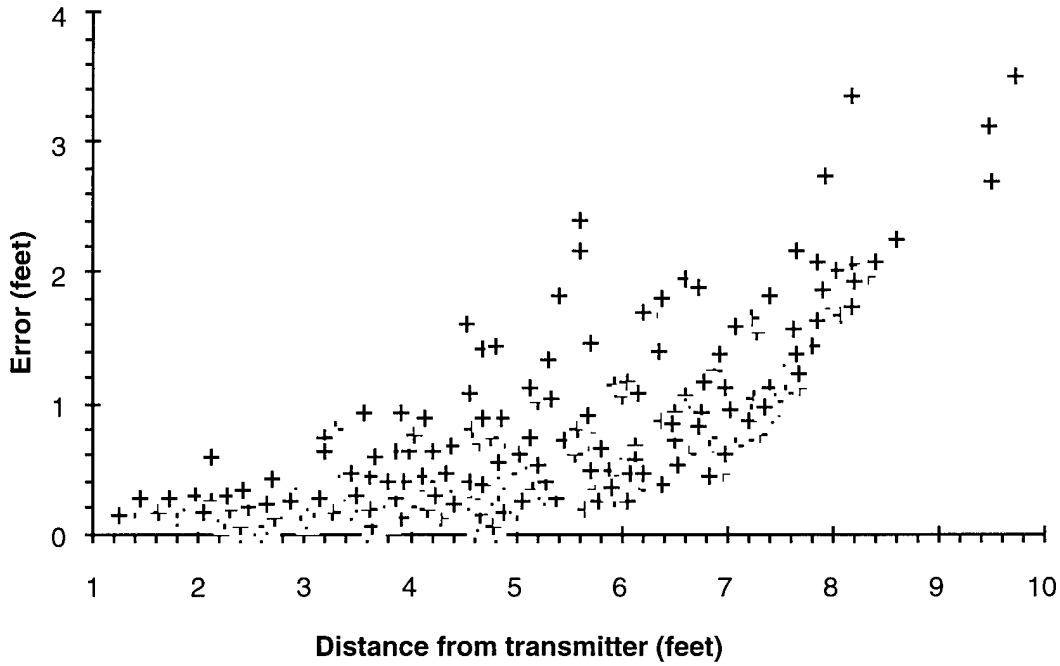



Figure 10 : ERTFOB errors measured

## B. Linear Interpolation

The collected points are thus exactly one foot apart; in practice, of course, a continuum of points must be corrected. We assume that the magnetic tracker is linear within one foot intervals. We use trilinear interpolation to calculate corrected values. This procedure is thoroughly described in [9].

## C. Results

To measure residual errors after calibration we collected data at one foot intervals on half-foot centers instead of one foot interval on one foot centers. Therefore we measured residual errors half way between the calibration points. These results are shown in Figures 11 and 12 as compared with Figure 10 before correction. These measurements of course depend on the accuracy of the UMD (less than 1.5% over 10 ft.). The maximum error before calibration is seen to be 4 ft. over a 10 ft. range (40%) (Figure 10). The error after calibrating is 0.27 ft. in the same 10 ft. range (2.7%) (Figure 11). Similarly, the maximum error before calibration is 0.6 ft. in a 3 ft. range (20%) (Figure 10). The error after calibrating is 0.13 ft. over the same range (4.3%) (Figure 11). Clearly, this procedure is better at correcting larger errors than smaller ones, why this is true is not well understood at this point. Minimizing tracker latency is desirable in VR systems, so it is important that the correction computation does not substantially increase existing tracker latency. This linear interpolation method needs 30 additions and 72 multiplications for each correction. On the CAVE Onyx R4400 processor, the above calculation takes less than 10 microseconds [7]. Since the theoretical minimum tracker latency is 21 milliseconds, adding 10 microseconds of delay is negligible.

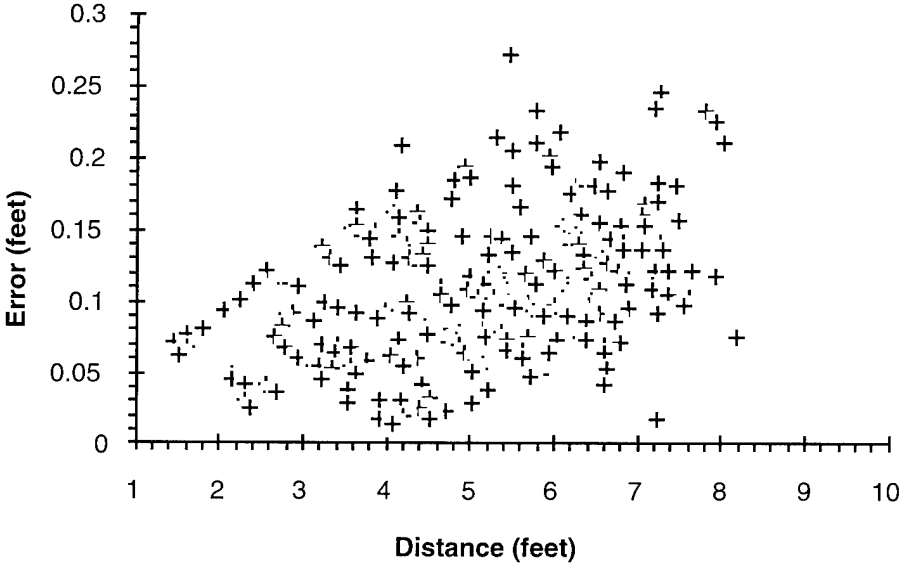



Figure 11: Errors of ERTFOB after correction

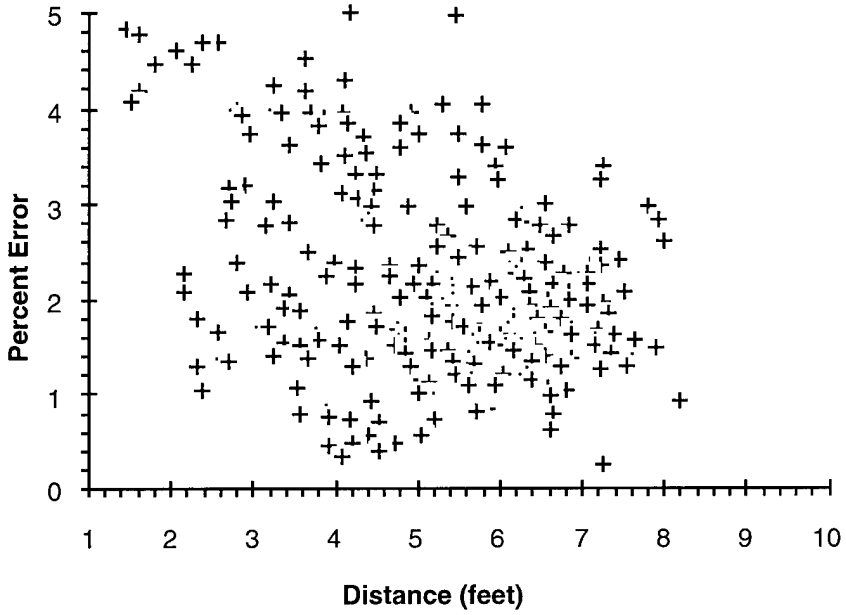



Figure 12: Percentage errors of ERTFOB after correction

## V. CONCLUSION

A relatively simple static calibration procedure as outlined above can make a significant improvement in the static accuracy of a magnetic tracking system. This is particularly true when the magnetic fields are distorted by metal in the environment, such as pipes and ducts in the ceiling, and metal reinforcing rods in the floor. For example, a copy of the CAVE installed at Argonne National Laboratories had a somewhat less uncorrected long range error of 2.5 ft. For the installation discussed in this paper, long range errors were reduced from 4 feet to 3.25 inches.

In the CAVE, a physical wand with a ERTFOB receiver attached is used to interact with objects in space, e.g. to pick up or select a virtual object. Often a cursor or graphic extension to this physical wand is used to indicate state of activity or to point to parts of the scene. If there are errors in either the head tract position or the wand position the physical world and the virtual world will not align properly. For instance, objects in the virtual world which should be fixed and stable will move or change size in incorrect ways, as the viewer moves around.

The quality of experience in a virtual reality system is quite dependent on the accuracy of the tracking subsystem. To gain qualitative information about the improvement of performance with position correction, the right half of the CAVE was uncorrected while the left half of the cave was corrected. There was a very significant improvement in the size and position stability of objects in the virtual scene and extensions to the wand appeared to stay attached to the wand.

The general improvement is very much worth the effort. The hardware for the UMD itself (not counting the PC) is less than $300.00 in parts. Detailed documentation is available [1, 5]. The calibration procedure takes a person approximately 2 hours to collect 400 points.

### A. Application to Other Systems.

The UMD is particularly well suited to the CAVE in that the projection screens form natural reflectors for the sound. In other systems such as Head Mounted Displays, existing walls in a physical room could be used, or temporary walls could be constructed out of any sound reflecting materials.

## VI. FUTURE WORK

The assumption of linearity of the ERTBOF for small distances is used in two places in the calibration procedure. Most importantly, linear interpolation is used on one foot centers. We propose to use splines passing through the correction points to generate a larger lookup table to better model the nonlinear magnetic fields. As mentioned in Section IV, we assume that the ERTFOB is differentially correct to make a small correction from 1 ft centers to the recording position of the UMD. As a post process, one

could use the created lookup table to better estimate this differential and create a more accurate table.

We currently are adding inclinometers to measure roll and pitch. It is our intention to use these readings to create a table of corrections for angle as a function of angle and position.

## VI. REFERENCES


1. Adamczyk, D., M.S. Project, University of Illinois at Chicago, September, 1994.
2. Bryson, S., *Measurement and calibration of static distortion of position data from 3D trackers*, Computer Graphics Course #43 Notes, SIGGRAPH Conference, 1993.
3. Bryson, S., *Body tracking*, Computer Graphics Course #43 Notes, SIGGRAPH Conference, California, 1993.
4. Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., "Surround-Screen Projection-Based Virtual Reality: The design and Implementation of the CAVE", *Computer Graphics (Proceedings of SIGGRAPH '93)*, ACM SIGGRAPH, August 1993.
5. Ghazisaedy, M., M.S. Project, University of Illinois at Chicago, July, 1994.
6. Gottschalk, S., Hughes, F.J., "Auto calibration for Virtual environments Tracking Hardware," *Computer Graphics (Proceedings of SIGGRAPH '93)*, ACM SIGGRAPH, August 1993.
7. Graphics Library Programming Guide, Silicon Graphics 1993.
8. Polaroid Ultrasonic Transducer Data Sheet, Model No. 607281, Polaroid Corp, Cambridge, MA.
9. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes in C*, Cambridge University Press, 1990.
10. *Texas Instruments Databook Linear Circuits, Amplifiers, Comparators, and Special Functions*, Vol. 1, 1989.
11. Wilson, J.D., *Practical Physics*, Sanders College Publisher, 1986.

# Dynamic Registration Correction in Augmented-Reality Systems

*Michael Bajura*
Department of Computer Science
UNC Chapel Hill
Chapel Hill, NC 27599-3175
bajura@cs.unc.edu

*Ulrich Neumann*
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
uneumann@cs.usc.edu

**ABSTRACT**
This paper addresses the problem of correcting visual registration errors in video-based augmented-reality systems. Accurate visual registration between real and computer-generated objects in combined images is critically important for conveying the perception that both types of object occupy the same 3-dimensional (3D) space. To date, augmented-reality systems have concentrated on simply improving 3D coordinate system registration in order to improve apparent (image) registration error. This paper introduces the the idea of dynamically measuring registration error in combined images (2D error) and using that information to correct 3D coordinate system registration error which in turn improves registration in the combined images. Registration can be made exact in every combined image if a small video delay can be tolerated. Our experimental augmented-reality system achieves improved image registration, stability, and error tolerance from tracking system drift and jitter over current augmented-reality systems. No additional tracking hardware or other devices are needed on the user's head-mounted display. Computer-generated objects can be "nailed" to real-world reference points in every image the user sees with an easily-implemented algorithm. Dynamic error correction as demonstrated here will likely be a key component of future augmented-reality systems.

**KEYWORDS:** Augmented Reality, Virtual Reality, Registration.

## 1. INTRODUCTION

Augmented-reality (AR) systems allow users to interact with real and computer-generated objects by displaying 3D virtual objects registered in a user's natural environment. Figure 1 illustrates an application of this powerful visualization tool where a user can visualize an as-yet unbuilt building in its proposed natural setting. Other applications include interactive 3D illustrations for constructing and for maintaining complex machinery [Feiner, MacIntyre, Seligmann 92, 93] [Caudell, Mizell 92] and in-patient visualization of medical data, e.g., ultrasound [Bajura, Fuchs, Ohbuchi 92]. In all these applications it is vitally necessary for computer-generated objects and real-world objects to be visually registered with respect to each other in every image the user sees. If accurate registration is not maintained, the computer-generated objects appear to float around in the user's natural environment without having a specific 3D spatial position.

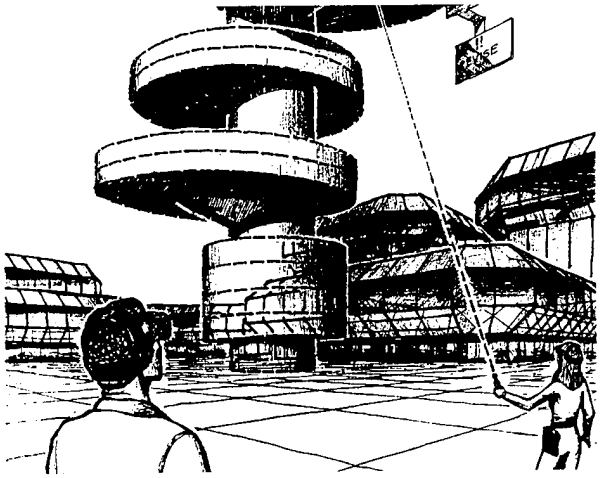


Figure 1: Example augmented-reality application: Visualization of a proposed building design.

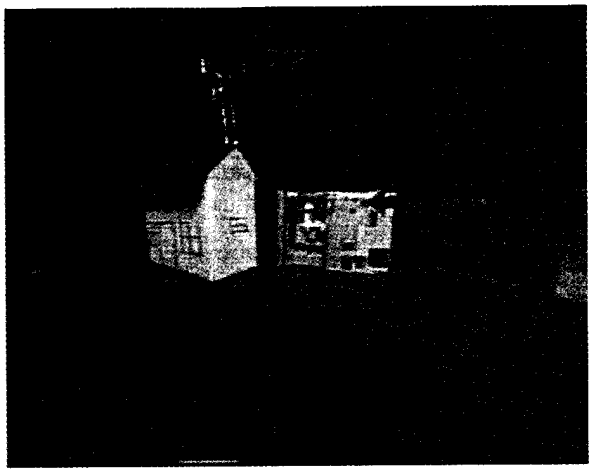


Figure 2: Experimental augmented-reality system showing dynamic registration of a virtual antenna and an annotation arrow which appear to be "nailed" in place.

Figure 2 is an image from our experimental AR system which dynamically corrects image registration on a frame-by-frame basis. It shows a computer-generated television antenna registered correctly on a toy house and a direction



0-8186-7084-3/95 $04.00 © 1995 IEEE

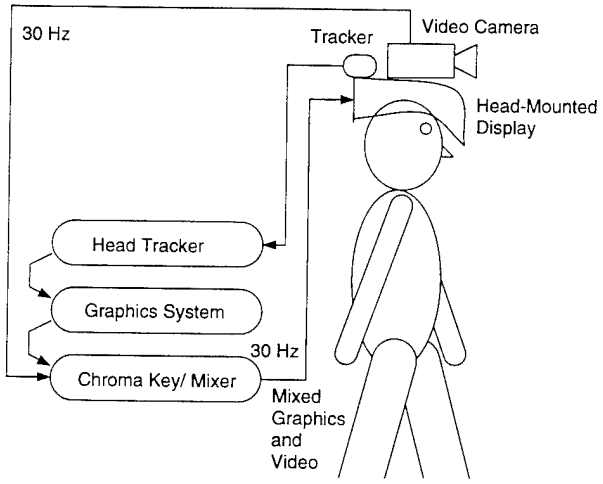



Figure 3: Typical video-based augmented-reality system components.

arrow registered correctly on a disk drive. The antenna and the arrow maintain correct registration in every image the user sees, including when the user is moving. The rest of this paper explains how this result is achieved and suggests future directions for dynamic registration correction. Section 2 describes the current model for augmented-reality systems and the sources of error in them. Section 3 explains a method for dynamically correcting registration error. Section 4 describes the implementation and results of our experimental AR system which dynamically corrects registration error. Conclusions and future directions follow in section 5.

## 2. CURRENT MODEL FOR AR SYSTEMS

Video-based augmented-reality systems are currently based on the model shown in figure 3. The user wears a head-mounted display (HMD) which presents combined images of both real and virtual (computer-generated) objects. Images of real objects are obtained from a video camera mounted on the user's display helmet. Images of virtual objects are generated by a graphics system. A tracking system reports the user's head position to the graphics system so it can render images with a virtual camera model of the video (real) camera's view of the world. The real and virtual images are typically merged with a chroma-key or video mixer for display in the HMD. For clarity in this paper, we discuss only the monocular case with one video camera mounted on the HMD. A stereo system would add a second video camera which would be treated as a second independent monocular system. Constructing a binocular HMD which presents correct stereopsis is problem addressed in [Edwards, Rolland, Keller 93].

The apparent registration between real and virtual objects depends on how accurately the virtual camera models the real one. Figure 4 shows a detailed transformation model for the virtual camera. Virtual objects are positioned by an *Origin-to-Object* transformation which specifies the position and orientation of a virtual object relative to a coordinate system origin. The virtual camera is positioned relative to the coordinate system origin by the composition of two transformations: *Origin-to-Head*, and *Head-to-Camera*. The *Origin-to-Head* transformation is reported by the tracking system and specifies the location of a tracking element's position on the user's HMD. The fixed transformation from this tracking element's position to the effective viewpoint of the real camera is the *Head-to-Camera* transformation. Virtual camera images are produced by a perspective projection onto a virtual image plane. A non-linear *Camera-to-Image* mapping is then applied which matches the field of view and lens distortion of the real camera. Typically the *Head-to-Camera* transformation and the *Camera-to-Image* mapping are determined by a calibration procedure such as the one described in section 4.2. It should be noted that this model does not address the problem of correcting for distortion in the HMD optics which is a separate problem from generating correctly registered images [Edwards, Rolland, Keller 93].

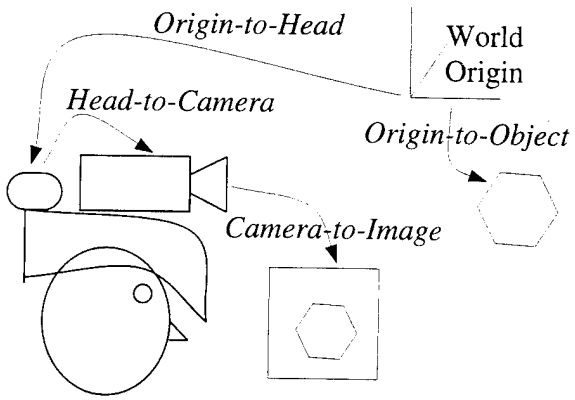



Figure 4: Transformations from object to image.

Image registration error in combined real and virtual images is caused by the following types of errors:

1) The tracking system's origin is not aligned with the world coordinate system origin. This error causes all virtual objects to appear to be displaced from their proper positions.

2) The virtual *Origin-to-Object* transformation is not the same as the real *Origin-to-Object* transformation for a particular object. This error causes individual objects to appear out of position.

3) The virtual camera position is not the same as the real camera position. This can be caused by errors in either the static *Head-to-Camera* transformation or the dynamic *Origin-to-Head* transformation reported by the tracking system. The tracking system exhibits two types of error: temporal error, and position and orientation error. Position and orientation errors cause misregistration in all cases, while temporal errors cause misregistration only during user movement. Temporal errors are caused by a delay in sensing and reporting tracking information to the computer graphics system and the computer graphics system's delay in generating the appropriate virtual images [Adelstein, Johnston, Ellis 92].

4) The virtual *Camera-to-Image* mapping doesn't accurately model the real camera. The *Camera-to-Image* mapping abstraction is that any real camera can be modelled by an idealized pinhole camera with a particular center of projection, viewing direction, field of view, and distortion function. The distortion function is a 2D warp which accounts for the non-linearities found in lens-based projection systems. Errors in the *Camera-to-Image* mapping cause misregistration to vary with screen position.

## 3. CORRECTING REGISTRATION ERROR

The new idea presented here is to dynamically measure the registration error, or misregistration, in each combined image and use that information to correct the system errors that caused the misregistration. In the above model (figure 3), absolute correctness of all the transformations shown in figure 4 is needed for absolute image registration. This situation is much like the design of an "open loop" system from systems theory. An input generates an output which has errors. The only way to improve the system is to make each system component more accurate. Another idea from systems theory is a "closed loop" system where a system's error output is used again at the input to improve the system's output. The model advanced here resembles the "closed loop" design where the image registration error, or misregistration, is used to correct the transformation parameters which caused it. The type of correction which can be performed depends on two main factors: 1) the method used for detecting and measuring image misregistration, and 2) the uncertainty and image-space sensitivity of the different parameters to be adjusted. Both are described below.

Different methods for measuring image misregistration dictate what kinds of correction can be performed. One way to measure image misregistration is to identify a recognizable point on each object to be registered. The image coordinates of each point are located in both the real and uncorrected virtual images. The difference between each point's position in each real image and corresponding uncorrected virtual image is the registration error, or misregistration, for the object corresponding with that point. This measure of misregistration can correct for errors such as camera orientation and sometimes camera position. A drawback with this measure is that neither the distance between an object and the camera nor an object's orientation can be estimated. Another way to measure image misregistration is to attempt to recognize an object's position, size, and orientation in each real image. This information can correct camera to object distance as well as relative object orientation.

Even if a particular misregistration measure does not allow estimation of a particular transformation parameter, misregistration can still be reduced. In many cases parameters which cannot be estimated can be assumed to be correct and the image registration error can be reduced by adjusting the remaining parameters. In other cases there is no way to separate the error contributions from different parameters and one or more must be adjusted depending on their relative uncertainty. The important point is that image registration error can be reduced even if some approximations are made.

The selection of which parameters to adjust depends on both their uncertainty and how sensitive image-space errors are to that uncertainty. For example, if the positions of objects are well known but the camera position and orientation are relatively uncertain, the camera position and orientation should be adjusted instead of object positions. Another example is camera position versus camera orientation. When an object is relatively close to a camera, its projection in image coordinates is more sensitive to the camera's position and less so to its orientation. For objects relatively far from a camera, the camera's orientation most strongly influences where an object's image appears.

This approach to correcting registration error can also be used to correct temporal errors. If the tracking system delay is longer than the delay in measuring image misregistration, the misregistration can be used to improve the most recent tracking system estimate. In video-based AR systems it is possible to effectively reduce the delay in measuring image registration to zero by delaying the the real video image stream by the time it takes to measure image registration and generate corrected virtual images. This makes it possible to correct temporal and spatial image registration exactly in every image the AR user sees. If there is registration error, it is only because the error compensation algorithm failed. For applications which can tolerate minimal delays, potentially perfect registration can be achieved. This trade-off is not possible with optically based AR systems which allow the user to see his surroundings directly.

Some success at improving registration error has been achieved with autocalibration approaches [Gottschalk, Hughes 93] and predictive tracking techniques [Azuma 94; List 84] which use a state estimate to help predict current measurements. However these approaches still suffer from the "open loop" requirement for perfect tracking and calibration.

## 4. THE EXPERIMENTAL SYSTEM

This section describes an experimental AR system which corrects image registration error on a frame-by-frame basis. Section 4.1 describes the functional components of the system and what hardware is used for them. System calibration is discussed in section 4.2. Section 4.3 describes how registration error is corrected in the experimental system. Section 4.4 shows the results of operating the system both with and without dynamic registration correction.

### 4.1 System Components

Figure 5 is a schematic for the experimental AR system. It is similar to the system described in figure 3 except for the addition of a real-time video delay and unwarp pipeline and a real-time image feature tracker. The delay and unwarp pipeline delays video by a constant number of frames and optionally applies an inverse distortion function which converts the incoming signal into an equivalent pinhole camera image. With our hardware, it is more practical to undistort the real camera video images to match the undistorted virtual images instead of distorting the virtual images to match the real camera ones. The pipeline delay is adjustable but constant during operation. The pipeline delay is set to match the delay in generating the correct virtual image to mix with the corresponding real camera video frame.

The image feature tracker recognizes features in the real video stream and passes their image coordinates to the graphics system. The features to be detected are red LEDs driven by a 9V power supply. The LEDs are significantly brighter than other objects in the environment. The LEDs are detected by applying a brightness and image area threshold to each image. Correspondence between LEDs and the particular features they represent is established by matching detected LED positions with the nearest estimated feature positions in

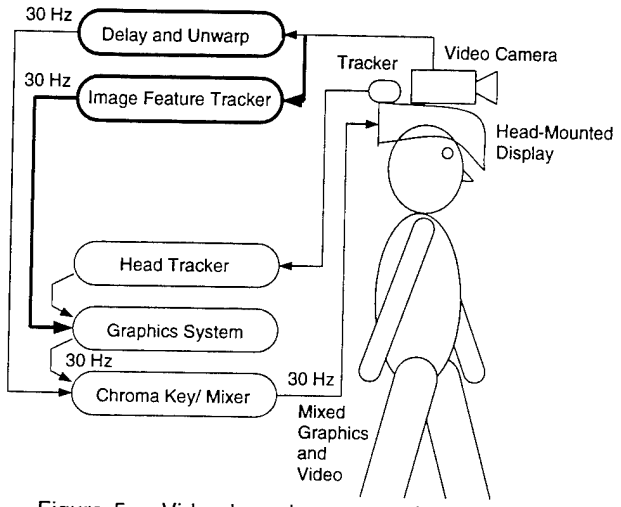
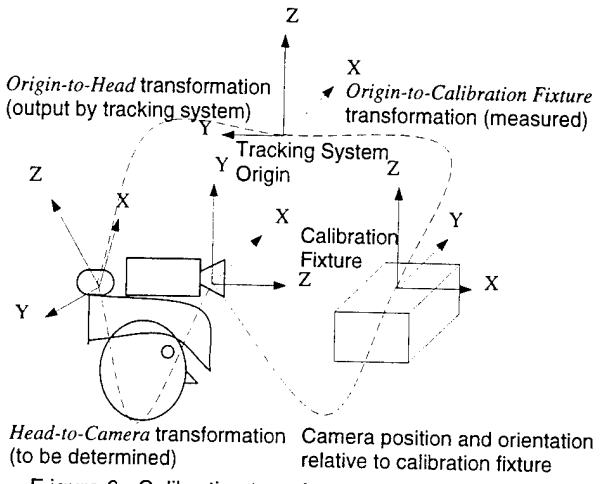



Figure 5: Video-based augmented-reality system model with registration correction.




F igure 6: Calibration transformations.

each corresponding uncorrected virtual image. To do this it is not necessary to render uncorrected virtual images. Only the feature positions must be computed. Once feature correspondence is established, the difference between each feature's position in each real image and it's estimated position in each corresponding virtual image can be used to render virtual images which are better registered with the real video images. The methods used for correcting registration are explained in section 4.3.

The camera used this experiment is a *Panasonic GP-KS102* color CCD camera with a highly distorting 110 degree wide angle lens. The head tracking system is an Ascension *A Flock of Birds* magnetic tracking system. The delay and unwarp, image feature tracker, and graphics system are different software modules which utilize separate portions of the *Pixel-Planes 5* graphics multicomputer at UNC [Fuchs 89]. Video is input to the *Pixel-Planes 5* system via a real-time video digitizer and output via a standard double-buffered frame buffer. Although it is desirable to mix the real camera and virtual camera video signals digitally, the bandwidth requirements of 30 Hz operation require the use of an analog Sony *CRK-2000 Universal Chroma Keyer* video mixer.

The AR world of the experimental system consists of a virtual TV antenna positioned atop a real model house (where an LED is located) and a virtual arrow which indicates an adjustment screw on a real disk drive (also where an LED is located).

### 4.2 Calibration

Before the system can be operated, the *Head-to-Camera* transformation and the *Camera-to-Image* mapping must be estimated. This is done by operating the AR system and using manual feedback to converge on a solution. Optional compensation for non-linear lens distortion in the *Camera-to-Image* mapping is measured by examining a distorted camera image and finding a 2D warp function which converts that image into an undistorted one [Bajura 93]. If non-linear lens distortion is not considered, a best-fit calibration solution by matching field of view is possible even for distorting lenses.

Figure 6 shows how the *Head-to-Camera* transformation is initially estimated. A calibration fixture is used to represent a fixed position and orientation which are measured relative to the tracking system origin. When the camera is placed in a specific position and orientation relative to the calibration fixture, the position and orientation of the head tracking element is recorded. The *Head-to-Camera* transformation is the difference between the head tracking element's position and orientation and the camera's position and orientation. A calibration fixture is needed because the tracking system reports positions relative to a fixed but not precisely known origin. If the tracking system reported coordinates in a known coordinate system relative to itself a calibration fixture wouldn't be necessary.

The calibration fixture is located by using the head tracking element to perform rigid body rotations about each of the calibration fixture's coordinate axes. As rotations about each axis are performed, the tracking element's position and orientation are recorded and used to compute each axis of rotation. Because rigid body rotations are used it isn't necessary to know the offset of the tracking element from each axis of rotation beforehand. The position and orientation of the calibration fixture's coordinate system are determined once rotations about two axes are performed. By taking enough careful measurements it is possible to locate the calibration fixture to nearly the precision of the tracking system itself.

Ideally, only one measurement of the head tracking element is needed to estimate the *Head-to-Camera* transformation while the camera is simultaneously placed in both a specific position and a specific orientation relative to the calibration fixture. Because it is difficult to accurately both position and orient the camera at the same time, separate measurements are made to estimate the position and orientation components of the *Head-to-Camera* transformation.

The *Head-to-Camera* estimation is refined by making further measurements while running the AR system. Virtual 3D coordinate axes are placed at the same position as the calibra-

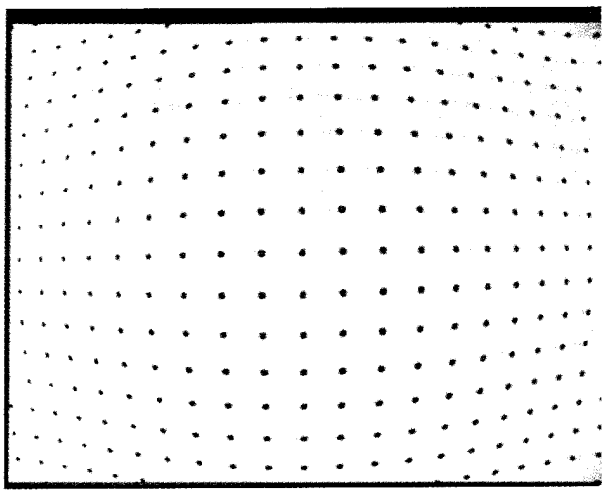


Figure 7: Distorted image of calibration pattern.

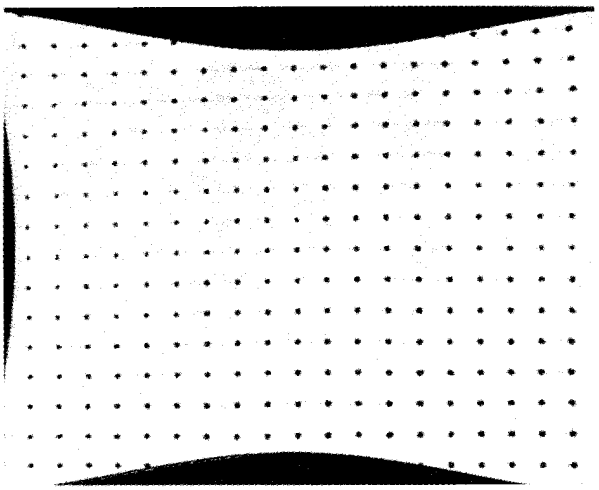


Figure 8: Corrected image of calibration pattern.

tion fixture's coordinate axes determined above. If the *Head-to-Camera* transformation is correct, the virtual and real coordinate axes will appear in the composite images to be aligned in both position and orientation when viewed along the optical axis of the camera. If the coordinate axes are not aligned, the *Head-to-Camera* estimate can be improved with the following heuristics. Only rough estimates for the *Camera-to-Image* mapping are needed at this point because alignment along the optical axis isn't affected by either the field of view or the lens distortion model.

• If the camera is positioned relatively far from the coordinate axes, misregistration is primarily due to *orientation* errors in the *Head-to-Camera* transformation. Rotations about the X and Y axes of the camera orientation should be made to align the axis positions. Rotations about the camera Z axis should be made to align the axis orientations.

• If the camera is positioned relatively near to the coordinate axes, misregistration is primarily due to *position* errors in the *Head-to-Camera* transformation. Translations along the camera X and Y axes should be made to align the axis positions. Translations along the camera Z axis will move the camera viewpoint either in front of or behind a virtual point, for example the coordinate axis origin, when the camera is very close to that point.

Once the *Head-to-Camera* transformation has been adjusted so that the virtual and real coordinate axes appear to be aligned when viewed along the camera's optical axis, the field of view component of the *Camera-to-Image* mapping can be adjusted. This is done by viewing the coordinate axes at angles off the camera's optical axis and separately adjusting the camera's X and Y fields of view until the coordinate axes are realigned. Without lens distortion correction, alignment will not be possible for all off-axis viewing angles. However, misalignment may be minimal with lower distortion lenses.

Non-linear lens distortion in the *Camera-to-Image* mapping is calibrated by imaging a test pattern and finding a distortion function which undistorts the test pattern image. This is done by appealing to a basic rule of (linear) projective geometry: straight lines remain straight under projection. Scales may change and parallel lines may intersect, but the image of a straight line is always straight. If there is a mapping which converts images from a distorting camera into ones where all straight lines appear to be straight, then the distorting camera can be modelled by a composition of this mapping and a pinhole camera model.

Figures 7 and 8 are images of a test pattern imaged with the 110 degree wide angle camera lens. Figure 7 is the distorted image output from the camera. Figure 8 is a corrected version of the same image. The correction is a radial distortion at the image center which accounts for most of the image distortion [Weng, Cohen, Herniou 92].

The important point about calibration is that it is difficult to do accurately, particularly when the tracking system used has noticeable tracking error throughout its working volume. Using a more accurate measuring device to measure the *Head-to-Camera* transformation would not eliminate errors in the AR system because camera position would still be a function of the tracking system which reports the *Origin-to-Head* transformation [Janin, Mizell, Caudell 93].

### 4.3 Correcting Registration Error

The image registration model of matching a point on each object makes it difficult to determine which particular errors are causing misregistration. One way to think about this is to consider the misregistration as a function of the camera position and orientation error (a composition of errors in the *Origin-to-Head* and *Head-to-Camera* transformations), *Camera-to-Image* mapping error, and *Origin-to-Object* transformation error:

Misregistration = $f$(camera position and orientation error,
*Camera-to-Image* mapping error,
*Origin-to-Object* transformation error)

Misregistration can be reduced by modifying one or more of the parameters which might be causing it. Two approaches to reducing registration error are studied in this experiment.

One approach assumes that the camera position and orientation are absolutely correct and that misregistration is due to errors in the *Camera-to-Image* mapping and *Origin-to-Object* transformation. The second approach assumes that the *Camera-to-Image* mapping and *Origin-to-Object* transformations are correct and that the camera position and orientation are in error. Neither of these approaches is optimal in the sense of minimizing error by smoothly adjusting all the possible parameters according to parameter certainty and registration sensitivity, e.g. optimal filtering. Such an analysis is difficult to make and may not be any better than making a few reasonable assumptions. Both of the approaches tried here are relatively easy to implement and are sensible in certain situations.

In the first correction approach, if the reported position and orientation of the virtual camera are assumed to be correct, there is no way to tell whether registration errors were caused by incorrect *Camera-to-Image* mapping, incorrect *Origin-to-Object* transformations, or both. By making a further assumption that the *Camera-to-Image* mapping is also correct, object positions alone can be adjusted to account for any registration error. To render a corrected image, each misregistered object is temporarily displaced to a position where it will appear to be registered correctly. This correction produces combined images with no measured registration error. Since the registration metric gives no estimate of distance between each object and the camera, virtual objects are displaced on a constant radius (rotated) from the virtual camera viewpoint. This maintains the best estimate of distance between the camera and each object so that objects don't grow and shrink unnaturally.

In the second correction approach, the virtual camera viewpoint is corrected to reduce registration error while object position and camera distortion are assumed to be correct. If enough features are visible it is theoretically possible to compute both camera position and orientation from the 3D (X,Y,Z) feature positions and their corresponding (U,V) image locations. If the feature positions aren't degenerate, the camera position and orientation can be recovered by nonlinear methods with a minimum of 4 points and by linear methods with a minimum of 6 points [Horaud, Conio, Leboulleux 89] [Ganapathy 84]. Trying to correct the camera position this way isn't practical for at least three reasons. First, there is no way to guarantee enough features will be visible in every image. Second, these solution methods are highly sensitive to noise and spatial feature distribution. Third, a good estimate of the virtual camera position is already available.

The easiest simplification to make is that the virtual camera position is correct as reported by the *Origin-to-Head* and *Head-to-Camera* transformations and that the registration error is entirely due to camera orientation error. This is a good assumption for three reasons. First, orientation corrections can be made when only one feature is visible. If more than one feature is visible a best-fit solution can be found. Second, under the assumption that objects are relatively far from the camera, which is true in most AR applications, registration errors are much more sensitive to errors in camera orientation than camera position. This means that solving for camera position is unstable (sensitive to errors) and that solving for camera orientation (when camera position is fixed) is well-behaved (relatively insensitive to errors). Third, tracking system data has more error in rotation than in translation. This is because HMD wearers typically rotate their heads faster than they move them and the head tracking system used incurs significant delays in reporting measurements (temporal error) [Liang, Shaw, Green 91]. In the experimental system, camera orientation error is adjusted by considering only one "reference" feature position and rotating the virtual camera to align that position. This is only an approximation which can correct the alignment of a particular point but not an orientation about that point.

### 4.4 Registration Results

The experimental system (figure 6) can be operated in nine different modes by different selections of the two parameters *real-video-delay* and *registration-correction-method*. *Real-video-delay* is one of:

1) no delay or distortion correction
2) delay without distortion correction
3) delay with distortion correction.

*Registration-correction-method* is one of:

A) none
B) correction by adjusting *Camera-to-Image* mapping and/or *Origin-to-Object* transformations (move the object)
C) correction by adjusting camera orientation (rotate the camera).

The results of different combinations of these parameters are described below:

(1,A): This "open loop" mode is equivalent to the "current model" shown in figure 3. Figure 9 shows the result: the virtual objects are not aligned with their proper positions and lag noticeably behind during user movement in spite of careful calibration and system tuning.

(1,B): This option has good registration at the object feature positions except during user motion when the registration still lags noticeably. It appears to be possible to shake the virtual objects from their proper positions, but they always return. This case shows the simple power of the "closed loop" system model over the "open loop" system model in figures 3 and 9. Despite the lack of lens distortion correction, noticeable lag, and various other errors, the virtual objects still appear to belong in specific spatial positions, a result not easily achieved without dynamic registration correction.

(2,A), (2,B): These combinations have the same static results as (1,A) and (1,B) above. However the registration error during motion (temporal registration error) is extremely small because the real video delay is the same as the tracking and image generation delay – the dynamic registration appears to be the same as the static registration. The reduction in the "swimming" of the virtual objects during motion makes them appear much more stationary and solid, even in the case of (2,A) where the registration is poor.

(3,A): The addition of lens distortion correction without registration correction produces the best "open loop" operation

Figure 9: "Open loop" mode without dynamic registration correction or distortion correction. Virtual objects "swim" around and are poorly registered.

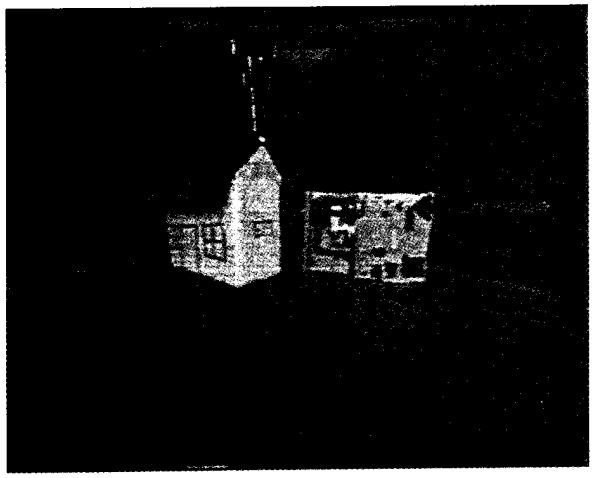


Figure 10: "Open loop" mode with video delay and optical distortion correction. Virtual objects do not swim as much and registration is somewhat improved.

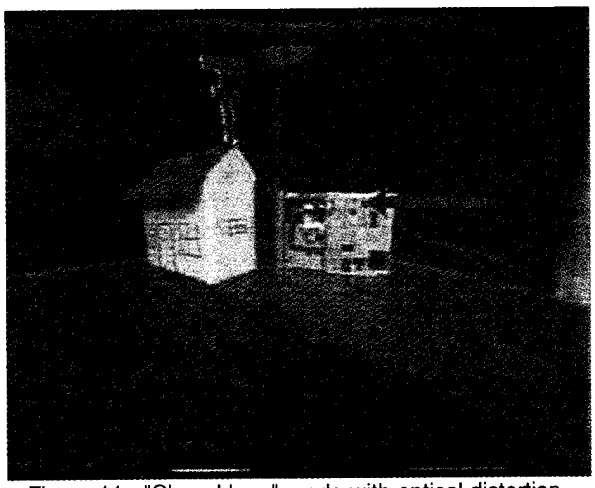


Figure 11: "Closed loop" mode with optical distortion correction, video delay, and dynamic correction of object position and lens distortion. Virtual objects appear to be "nailed" to their reference positions.

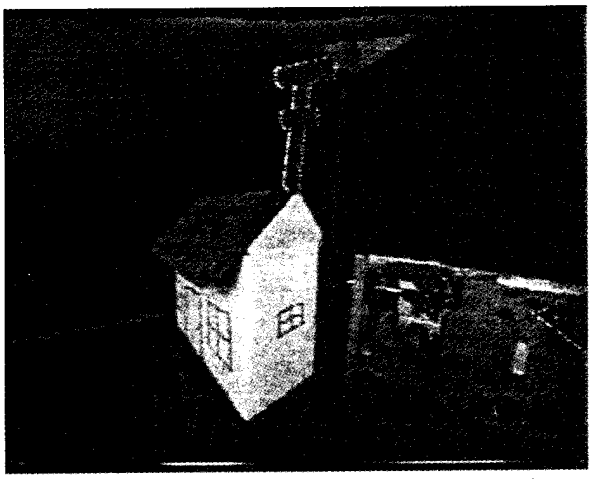


Figure 12: "Closed loop" mode with optical distortion correction, video delay, and camera orientation correction to align base of antenna. Adjustment arrow is steady but slightly misregistered.

possible with the experimental system (figure 10). The lens distortion correction improves registration considerably but the virtual objects still wander slightly during movement and appear in different positions as the tracking system exhibits errors within its working volume.

(3,B): This combination of distortion correction, delay, and registration correction by displacing objects produces the best registration in the experimental system (figure 11). In all cases, during both static and dynamic viewing, the virtual objects appear to be registered correctly with respect to their reference positions. They appear to be "nailed" in place.

(3,C): Here only the reference position for the TV antenna is used to adjust the virtual camera orientation while the real video is corrected for distortion and delayed(figure 12). No registration correction is made for lens distortion or object position errors. This combination produces the 2nd best registration after combination (3,B). The base of the antenna appears to be registered correctly on the house, but the arrow on the disk drive adjustment screw consistently appears to be just a bit low. This misregistration could be caused by errors in the *Origin-to-Object* transformations for the TV antenna and disk drive screw or by errors in the initial camera orientation which aren't completely corrected with this method.

(1,C), (2,C): These combinations did not make sense. Without lens distortion correction it is not possible to modify the camera position to improve registration for more than one object.

## 5. CONCLUSIONS AND FUTURE DIRECTIONS

• Building augmented-reality systems with accurate registration is difficult. The visual registration requirement between real objects and virtual objects exposes any measurement or calibration error in an AR system. The strongest argument in favor of dynamic compensation is that no matter how much measurement and calibration are performed, there may (will) still be errors in the composite images. What the real camera sees must be taken as the ground truth and the registration error between the real and virtual images must be corrected by compensating for errors in calibration, tracking, and/or distortion correction. It is more practical to measure and correct errors using a "closed loop" design than to avoid making them in the first place with an "open loop" design.

• The experiment described here demonstrates the importance and feasibility of dynamically measuring and correcting image space registration error. The experimental system is more stable and better aligned than systems without registration correction.

• The idea of measuring and correcting image registration error has implications for the design of future augmented-reality systems. Since feedback can compensate for tracking errors, in essence becoming part of the tracking system itself, less accurate and less expensive tracking systems may be feasible. Optical tracking systems [Azuma 94] could be designed to use stationary cameras to track a user's position while cameras on the user's head could look outward to determine the user's orientation. Feedback also reduces the accuracy requirements for lens distortion correction and system calibration.

• The success of registration correction depends on the ability to accurately measure registration in the first place. This is not a simple task in general. The experiment described here uses an oversimplified method for measuring registration which may not be practical in many environments. A large amount of work in this area has already been done by the computer vision community. Hopefully some of their results can be applied to AR systems.

• Correct occlusion cues are still needed for augmented-reality systems to be truly believable. This method of registration only works for virtual objects which are completely in front of real ones. What is really needed is a way to sense positions and depths in the environment from the real camera. With such information, the reference positions could be used to position virtual objects which could be hidden properly if they were obscured.

## 6. ACKNOWLEDGMENTS

Support for this research was provided by The Link Foundation and the NSF/ARPA Science and Technology Center for Computer Graphics and Scientific Visualization (NSF Cooperative Agreement #ASC8920219). Thanks to: Andrei State for Figure 1, Henry Fuchs for writing suggestions.

## 7. REFERENCES

[Adelstein, Johnston, Ellis 92] Adelstein, B., Johnston, E., Ellis, S. "A testbed for Characterizing Dynamic Response of Virtual Environment Spatial Sensors," Proceedings 5th Annual ACM Symposium on User Interface Software and Technology (UIST) 1992, ACM SIGGRAPH/SIGCHI, Monterey, California, Nov. 1992, pp.15-22.

[Azuma 94] Azuma, R., Bishop, G. "Improving Static and Dynamic Registration in an Optical See-through HMD," Computer Graphics (Proceedings of Siggraph 1994), pp.197-204.

[Bajura 93] Bajura, M. "Camera Calibration for Video See-Through Head-Mounted Display," TR93-048 Computer Science Technical Report UNC Chapel Hill, July 1993

[Bajura, Fuchs, Ohbuchi 92] Bajura, M., Fuchs, H., Ohbuchi, R. "Merging Virtual Reality with the Real World: Seeing Ultrasound Imagery within the Patient," Computer Graphics (Proceedings of Siggraph 1992), pp.203-210.

[Caudell, Mizell 92] Caudell, T.P., Mizell, D.W. "Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes," Proceedings Hawaii Intl. Conf. on System Sciences, Jan 1992, vol. 2, pp.659-669.

[Edwards, Rolland, Keller 93] Edwards, E.K., Rolland, J.P., Keller, K.P. "Video See-through Design for Merging of Real and Virtual Environments," Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS) 1993, Seattle, WA.

[Feiner, MacIntyre, Seligmann 92] Feiner, S., MacIntyre, B., Seligmann, D. "Annotating the Real World with Knowledge-Based Graphics on a See-Through Head-Mounted Display," Proceedings Graphics Interface 1992, Canadian Information Proc. Soc., pp.78-85.

[Feiner, MacIntyre, Seligmann 93] Feiner, S., MacIntyre, B., Seligmann, D. "Knowledge-Based Augmented Reality," Communications of the ACM, July 1993, Vol. 30, No. 7, pp.53-62.

[Ganapathy 84] Ganapathy "Real-Time Motion Tracking Using a Single Camera," AT&T Bell Labs Tech Report 11358-841105-21-TM.

[Gottschalk, Hughes 93] Gottschalk, S., Hughes, J. "Autocalibration for Virtual Environments Tracking Hardware," Computer Graphics (Proceedings of Siggraph 1993), pp.65-72.

[Horaud, Conio, Leboulleux 89] Horaud, R., Conio, B., Leboulleux, O. "An Analytic Solution for the Perspective 4-Point Problem," Computer Vision, Graphics, and Image Processing, Vol 47, No. 33-34 (1989), pp.33-43.

[Janin, Mizell, Caudell 93] Janin, A.L., Mizell, D.W., Caudell, T.P. "Calibration of Head-Mounted Displays for Augmented Reality Applications," Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS) 1993, Seattle, WA.

[List 84] List "Nonlinear Prediction of Head Movements for Helmet-Mounted Displays," Tech. report AFHRL-TP-83-45, Williams AFB, AZ.

[Liang, Shaw, Green 91] Liang, J., Shaw, C., Green, M. "On Temporal-Spatial Realism in the Virtual Reality Environment," Proceedings 4th Annual ACM Symposium on User Interface Software and Technology (UIST) 1991, ACM SIGGRAPH/SIGCHI, Hilton Head, South Carolina, Nov. 1991, pp.19-25.

[Weng, Cohen, Herniou 92] Weng, J., Cohen, P., Herniou, M. "Camera Calibration with Distortion Models and Accuracy Evaluation," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 14, No. 10, October 1992, pp.965-980.

# Haptic Interfaces

# Integration of the Rutgers Master II in a Virtual Reality Simulation

*Daniel Gomez, Grigore Burdea*
*and Noshir Langrana*
Human-Machine Interface Laboratory
Caip Center - Rutgers University
P.O. Box 1390
Piscatway NJ 00859
burdea@caip.rutgers.edu

**ABSTRACT**
A novel compact hand master device with force feedback is presented. The Second Generation Rutgers Master (RM-II) integrates position-sensing and force-feedback to multiple fingers in a single structure, without the use of sensing gloves. The paper first discusses the kinematics and calibration followed by the integration of the device into a single-user, ethernet-distributed, virtual reality (VR) environment. The VR simulation features: visual feedback, force feedback, interactive sound and object interaction.

**KEYWORDS:** VR, force feedback, master, calibration.

**INTRODUCTION**
Virtual reality is a computer generated immersive environment with which users have real-time, multisensorial interactions [6]. In general, these interactions involve all human senses through visual feedback [19], 3-D sound [7], force and touch feedback [24], and even smell and taste [25]. Key to immersion realism is the capacity of the user to use his/her hand to interactively manipulate virtual objects. Unfortunately, the majority of today's commercial VR systems use hand-sensing devices that provide no haptic feedback [26][8][6]. The lack of this sensorial information represents a significant limitation to the realism of present VR applications.

Earlier research work on integrating force feedback to virtual reality simulations was done by the robotics community which adapted existing teleoperation servo-arms [14] [1] [13]. These systems have high bandwidth, but are complex, expensive and hard to maintain. Steps towards less complex systems were taken by integrating force feedback to joysticks [18][22][12]. Unfortunately, force feedback joysticks limit the hand motion to a small volume and can not provide force feedback to independent fingers. EXOS Inc. introduced recently the first commercial force feedback masters [9][16]. This was followed by the Phanton master [17] produced by SenSable Device Inc., a small manipulator with 6 degrees of freedom, three of which have force feedback.

These new products have either force or touch feedback (but not both), large weights and high costs. It is therefore necessary to develop new tools that are more compact, easy to maintain, safe and inexpensive. Work towards improved master portability with more freedom of motion was done by Burdea et al. [1992]. This resulted in a very light feedback structure (about 70 grams) called the Rutgers Portable Force Feedback Master ("Rutgers Master I" for short). This master was designed to retrofit open-loop sensing gloves such as the DataGlove.

Initial human-factor studies done on six groups of 14 volunteers (42 males and 42 females) showed that force feedback using the Rutgers Master I reduced the task (average) error rate and learning time by 52.3% as compare to no feedback at all [20]. The task performed was to grab and relocate a virtual ball while causing the smallest deformation possible (less than 10%). Four force feedback modalities including auditive, visual (bargraph), graphics (object graphical deformation) and haptic where studied. The Rutgers Master I system was still complex as it used a sensing glove for position measurements. Burdea and Gomez, [1994] proposed a unified position/force feedback master, which eliminates the need for an additional sensing glove.

**THE RUTGERS MASTER II (RM-II)**
The Second Generation Rutgers Master (RM-II) is an improved design of the Rutgers Portable Force Feedback Master I. The RM-II is worn on the user's hand like a glove. The main structure consists of a small platform on which are mounted four custom-made pneumatic cylinders. The platform is made of carbon-epoxy composite to reduce weight while maintaining structural strength. The top part of each cylinder's shaft is connected to its corresponding fingertip. The coupling between the shaft and fingertip is performed by means of a "Y" shaped attachment. The platform is attached to the glove by means of a velcro belt allowing adjustments for different users. A simple thin leather glove is used as a support structure for the sensor/feedback system. The prototype of RM-II is shown in Figure 1.

Each pneumatic cylinder is attached to the platform by means of two-degrees of freedom mounts. The degrees of freedom are the *yaw* $\theta_y$ and *pitch* $\theta_p$ angles of the cylinder with respect to the platform. These angles are measured by means of angular position Hall effect sensors embedded in the mounts. The linear displacement $d$ of the cylinder's shaft is measured by means of a novel non-contact position sensor using an IR LED-phototransistor pair. The sensor pair is mounted coaxially *inside* the pneumatic cylinder. The phototransistor is



0-8186-7084-3/95 $04.00 

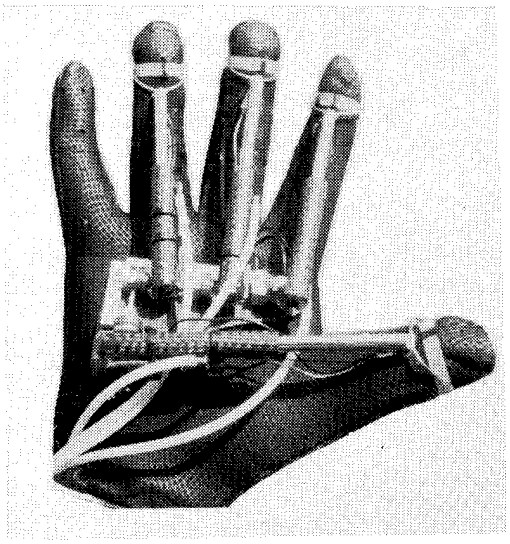


Figure 1: The Rutgers Master II prototype

fixed to the bottom of the cylinders while the LED is embedded in the piston. The LED is driven with constant current to keep a constant intensity. The light intensity measured by the phototransistor is a function of the distance to the LED which moves together with the piston. Thus, both, angular and linear displacements are measured with non-contact sensors. This is important in order not to introduce additional friction forces in the process of measuring the hand gesture. The details in sensor design and calibration tests were reported preliminarily in [10].

To calculate the fingertip position (but not its orientation) relative to the palm three variables yaw $\theta_y$, pitch $\theta_p$ and piston-displacement $d$ are sufficient. A third Hall-effect sensor is in the process of being mounted at the fingertip attachment to measure the relative orientation $\beta$ of the fingertip with respect to the cylinder axis. This measurement, in addition to the previous three variables, represents the complete information required to determine the fingertip configuration. A Polhemus sensor mounted on the back of the hand is used to determine wrist position/orientation. Thus RM-II has 22 potential degrees of freedom.

The force feedback actuators are pneumatic cylinders made of Pyrex glass with 5 mm internal diameter. The cylinders are protected by an external layer of carbon composite material. The moving piston is made of Lexan plastic which reduces frictional forces to improve feedback control. The air input is located on the side of the piston bottom in order to keep a path free for the IR sensing system. The pressure in the cylinder chamber determines the force applied by the piston directly to the fingertip (through the finger attachment). A maximum force of 16.38 N with a resolution of 0.05 N was obtained. The corresponding dynamic range is then 0.05/16.38 = 0.003. Since the force feedback actuators were custom-made, it was possible to increase the range of motion, resulting in a larger finger work envelope compared with that of RM-I. The piston linear motion increased by 20% for the index, middle and ring fingers, and by 50% for the thumb. The cylinder mounts (with two degrees of freedom each) also present a larger angular range of motion (adduction/abduction and flexion/extension).

The RM-II interface unit contains four voltage-controlled pressure regulators which drive the feedback actuators, a compact $\frac{1}{4}$HP-60psi compressor which provides the necessary air pressure for the regulators, a main pressure gauge showing the regulator input pressure and four LED bar-graphs on the front panel to visualize each of the outputs air pressures. The interface also includes signal conditioning electronics for all the position sensors and a power supply. The interface I/O are analog voltages which are write/read by the data acquisition board installed in the host computer. The force control loop of RM-II is shown in Figure 2.

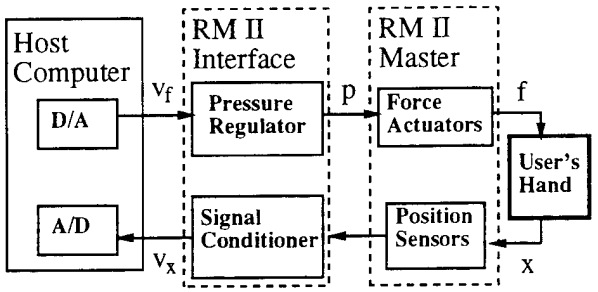



Figure 2: RM-II force control loop

## KINEMATICS AND CALIBRATION

RM-II measures the fingertip position-orientation of distal phalanx with respect to the palm of the hand. This measurement is sufficient to provide the corresponding force feedback due to fingertip virtual interaction. However, it is not enough to graphically represent the finger configuration during the simulation. That is because the phalanx flexion angles are unknown. By solving the kinematics problem it is possible to infer the finger configuration from the known information.

Figures 3 presents a 2D model of the joint-link structure associated to the kinematics and static force problem. The finger phalanges are assumed to be rigid links with lengths $\{a_1, a_2, a_3\}$ while the joints of the phalanges are assumed to be 1 DOF revolute joints with angles $\{\theta_1, \theta_2, \theta_3\}$ located in a sagittal plane. The sensing/force feedback structure is modeled as consisting of one revolute joint with pitch angle $\theta_p$ and one prismatic joint with displacement $d$. $\beta$ is the angle formed by the distal phalanx and the feedback actuator.

The kinematic problem is defined as: Given the variables $\{\theta_p, \beta, d\}$ and the parameters $\{h, l, a_1, a_2, a_3\}$ find the phalanx flexion angles $\{\theta_1, \theta_2, \theta_3\}$. Here, we assume that the finger link lengths $\{a_1, a_2, a_3\}$ are known. However, in practice, average values or direct measurement should be used.

Coordinates for points $Q$ and $P$ can be obtained from the geometric solution of the direct kinematics problem associated with the feedback master structure. Here the coordinates are measured with respect to the hand reference frame $H$ (which

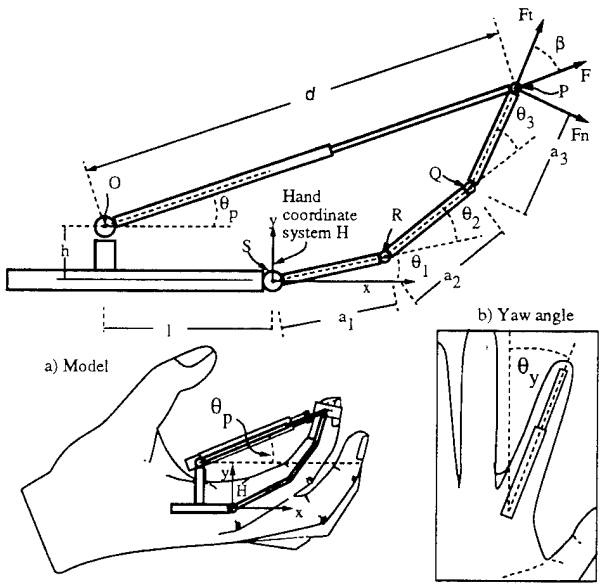



Figure 3: Kinematic and Static Force model

has origin at point $S$):

$$P^H = \begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} d\ sin(\theta_p) - l \\ d\ cos(\theta_p) + h \end{pmatrix} \quad (1)$$

$$Q^H = \begin{pmatrix} q_x \\ q_y \end{pmatrix} = \begin{pmatrix} p_x - a_3\ cos(\theta_p + \beta) \\ p_y - a_3\ sin(\theta_p + \beta) \end{pmatrix} \quad (2)$$

Coordinates for point $R$ can be obtained solving the four–bar problem associated with points $S$, $P$, $Q$ and $R$ (or phalanx $\overline{SR}$ proximal, $\overline{RQ}$ middle and $\overline{QP}$ distal). The solution is given by:

$$R^H = \begin{pmatrix} r_x \\ r_y \end{pmatrix} = \begin{pmatrix} cos(\alpha)A - sin(\alpha)B \\ sin(\alpha)A + cos(\alpha)B \end{pmatrix} \quad (3)$$

$$A = \frac{a_1^2 - a_2^2 + \| Q^H \|}{2 \| Q^H \|}; \quad B = \pm\sqrt{a_1^2 - A^2}$$

$$\alpha = tan^{-1}(q_y/q_x); \quad \| Q^H \| = \sqrt{q_x^2 + q_y^2}$$

Two solutions exist, one of them is eliminated by imposing the physiological constrains $\theta_1, \theta_2$ and $\theta_3 > 0$.

An interesting problem occurs when the distal phalanx orientation angle $\beta$ is unknown (a sensor to measure distal phalanx angle is not present). In this case only coordinates for point $P$ can be determined from $\theta_p$ and $d$ as given by equation (1). Without any extra constraint equation, there exists an infinite number of positions for point $Q$ and $R$ that make the finger satisfy the position condition for point $P$. Lee and Rim [15] experimentally determined the coupling relation among finger joint for a normal hand as:

$$\theta_3 = 0.46\theta_2 + 0.083\theta_1^2 \quad (4)$$

Equation (4) was used as a constrain equation to generate (off-line) a lookup table which in turn is used during the simulation to select the orientation angles $\{\theta_3, \theta_2, \theta_1\}$ which satisfy the position for point $P$. This scheme permits the selection of the link angles corresponding to a normal human hand without the use of an extra sensor to measure $\beta$.

Corrections due to adduction-abduction rotation (proximal phalanx) are small since this movement does not generate large *yaw* angles ($\sim$ 20°) in the feedback master structure (Figure 3). To graphically represent finger motion, the adduction-abduction angle for proximal phalanx can be made equal to $\theta_y$ (this approximation is valid for small $l << d$). For RM-II this approximation introduced an error of less than 2%.

Force feedback components, normal force $F_n$ and tangential $F_t$ (Figure 3) can be calculated from the distal phalanx orientation angle $\beta$ and the applied force feedback as follows:

$$F_n = F sin(\beta)$$
$$F_t = F cos(\beta)$$

Calibrated data for each single RM-II sensor was used to generate sensor-dedicated lookup tables. At the beginning of each simulation session, the user is asked to repeatedly open and close his/her hand (equipped with RM-II) in order to determine the user range of motion. During the simulation the lookup tables are used to obtain calibrated data which is interpolated according to the user range of motion.

## SIMULATION SETUP

Two configurations were investigated for the simulation. In the first configuration a Pentium 90 MHz running DOS/MS-Windows was used as a stand-alone host computer. A 12-bits 120kHz A/D board was used to read RM-II sensors while a 12-bit 40kHz D/A board sends force feedback information. A simple wire-frame graphics application was written to test the system. A more elaborate PC-based configuration is still under development. The discussion in this paper will focus on the second configuration where RM-II device is interfaced with a client-server ethernet-distributed architecture using three Unix workstations as shown in Figure 4. This configuration has been fully implemented and tested.

The main process is the *server* program running on the Sun 4/380 (to which RM-II is interfaced). This process calculates object dynamics taking into account collisions, grasping, squeezing, gravity and objects tossing. The *server* reads RM-II data to calculate hand configuration and writes back the corresponding force feedback data. Update information on object deformation, locations and orientations is sent by the server to the HP-775 workstation which runs the graphics process (*graphics client*). This program runs with an average refresh rate of around 28 frames/second when double buffering and Gouraud shading (with one light source) are used. A sound ID is sent by the server to the Sun ELC running a *sound client* program. This ID is then used by the *sound client* to query a database for the appropriate sound to be displayed.

During the distributed simulation the user interacts with two elastic ball objects. Each object has different color and virtual stiffness. A virtual hand is used for the interaction which has

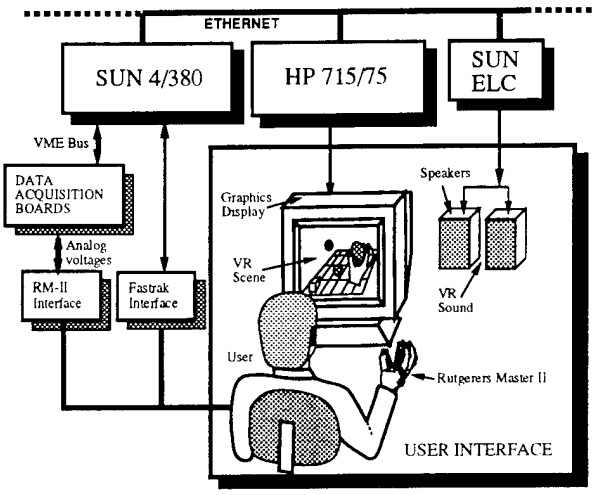



Figure 4: System Architecture

the same kinematics as the human hand, ie. four degrees of freedom per finger. Gravity is simulated allowing the ball to bounce inside the virtual room. When a grabbed object is squeezed individual force feedback signals are applied to each fingertip by the RM-II. In this way the user can "feel" the object allowing him/her to determining its stifness. When collisions between objects occur, an appropriate sound is generated. A sample graphics output of the simulation is shown in Figure 5

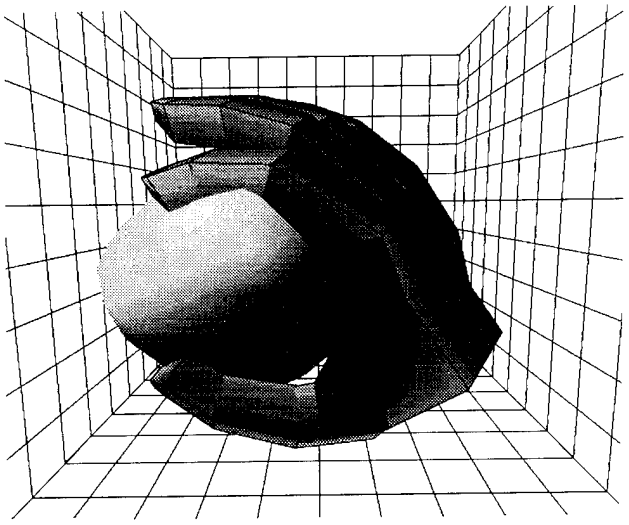


Figure 5: Virtual Enviroment

When an object is grasped, the main loop computes the degree of object deformation by calculating the fingertip position with respect to the object surface. Force generated by the object to each fingertip is calculated using the deformation and the programmed object compliance. Force feedback information is displayed by RM-II while the degree of object deformation is sent to the *graphics client* so virtual objects appear deformed. Object deformation has been previously studied using partial derivatives and finite element methods [11],[21],[23]. The simulation described here, however, uses linearized deformation laws because of the real-time requirement of virtual reality interactions and limited computing power. Hooke's Law, $F_i = k\Delta x_i; i = 1, .., 4$, has been used to relate depth of compression to the generated force. In this way, the equation is kept simple enough for rapid computation while still retaining the ability to model objects of various stiffnesses. The force feedback bandwidth is 14-15 Hz.

The simulation was implemented in C++ language. Figure 6 presents the class hierarchy and simulation interaction. The generic class *RM_II* implements the basic data struc-

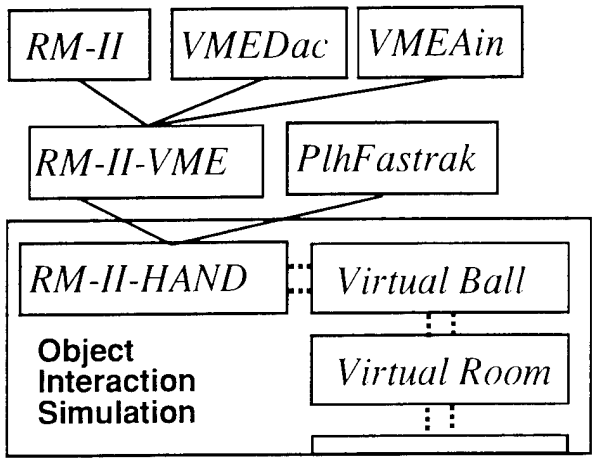



Figure 6: C++ class hierarchy and their interaction

ture and behavior of the RM-II master. This class provides the data structure to hold the readings from 16 sensors corresponding to finger configuration and the 6 readings from hand position-orientation. Calibration offsets and gains for each sensor are also included in the class. The *VMEAin* and *VMEDac* classes provide access to the data acquisition board installed on the VME bus repeater box. The class *RM_II_VME* descends from these three classes and implements the adequate sensor mapping (from I/O analog channels to sensor) and calibration. The class at the bottom of the hierarchy is *RM_II_Hand* which is the class used in the simulation to create the virtual hand. This class descends from classes *RM_II_VME* and *PlhFastrak* which provides access to the Polhemus tracker.

**SUMMARY**

The experimental prototype proved the functionality of the RM-II concept. Further miniaturization of position sensing is possible, however, high precision mechanisms are needed. This is because tolerances among moving parts must be kept small in order to provide precision sensing. The IR displacement sensor and the Hall effect sensors proved to be adequate since they do not produce any contact friction. This represents a clear improvement over the first Rutgers master where pneumatic cylinders were made of steel with rubber gasketed piston which presented high friction (more than 0.5 N). The

tributed simulation system. Human factor test are needed in order to quantify the usefulness of force feedback in the simulation using RM-II.

A stand-alone PC-based architecture is the short term goal of our research. The simulation software will be portted to Sense8 WTK in order to ease usage of RM-II outside our laboratory. It is believed that the RM-II system has wide applicability in all areas that utilize Virtual Reality, such as medicine, entertainment, training, multi-media and telerobotics.

## ACKNOWLEDGMENTS

The research reported here was made possible by the CAIP Center, Rutgers University, with funds provided by the New Jersey Commission on Science and Technology and the CAIP's Industrial Members.

## REFERENCES

1. Brooks F., M. Ouh-Young, J. Batter and P. Jerome. "Project GROPE - Haptic displays for scientific visualization", *Computer Graphics*, Vol. 24(4), pp. 177–185, 1990.

2. Burdea G., J. Zhuang, E. Roskos, D. Silver and N. Langrana. "A portable dextrous master with force feedback", *Presence: Teleoperators and Virtual Environments*, Vol. 1(1), pp. 18-28., 1992.

3. Burdea G. and N. Langrana. "Virtual Force Feedback: lessons, challenges, future applications", *Proceedings of 1992 A.S.M.E. Winter Annual Meeting - Advances in Robotics*, DSC-Vol. 42, Anaheim, CA, pp. 41-47, November. 1992.

4. Burdea G. and D. Gomez. "Actuator System for Providing Feedback to a Portable Master Support", U.S. Patent No. 5,354,162 October 11, 1994.

5. Burdea G. and J. Zhuang. "Actuator system for providing force feedback to a dextrous Master Glove", US Patent 5,143,505, September 1, 1992.

6. Burdea G. and P. Coiffet. *Virtual Reality Technology*, John Wiley and Sons, New York, 400pp., June, 1994.

7. Chapin W. and S. Foster. "Virtual Environment Display for a 3D Audio Room Simulation", *Proceedings of SPIE Stereoscopic Display and Applications*, 12pp. 1992.

8. Exos Inc. "Dextrous Hand Master User Manual", Burlingtion, MA., 1990

9. Exos Inc. New Product Brochure, January, MA., 1994.

10. Gomez, D., G. Burdea and N. Langrana. "The Second-Generation Rutgers Master SGRM", *Proceedings of VR System Conference*, New York City, NY, October, 1993.

11. Gourret J. P. "Simulation of Object and Human Skin Deformations in a Grasping Task", *Computer Graphics*, Volume 23, Number 3, 1989.

12. Iwata H. "A Six-Degree-of-freedom Pen-based Force Display", University of Tsukuba, Japan, 6pp. 1993.

13. Jacobsen S., E. Inverson, C. Davis, D. Potter and T. McLain. "Design of a multiple degree of freedom force reflective hand master/slave with a high mobility wrist", *Third topical meeting on robotics and remote systems*, Charleston, SC, March, 1989.

14. Jau B.M. "Man-equivalent telepresence through four finger human-like hand system", *Proceedings IEEE International Conference on Robotics and Automation*, pp. 834-848. Nice, France, May, 1993.

15. Lee J.W. and K. Rim. "Maximum finger force prediction using a planar simulation of the middle finger". *Proc. Instn. Mech. Engrs.*, Vol 204, pp.160–178. 1990.

16. Marcus, B. "Sensing, Perception and Feedback for VR", *Proceedings of VR System Conference*, New York City, NY, October, 1993.

17. Massie T. H. and J. K. Salisbury "The Phantom haptic interface: a device for probing virtual objects". *Proc. International Mechanical Engineering Congress and Exposition*, DSC-Vol 55-1, pp.295-299. Chicago, IL, November 1994.

18. Minsky M., O. Miling, O. Stele, J. Brooks and M. Behensky. "Feeling and seeing: issues in force display". *ACM Computer Graphics*, 24(2):235–243. 1990.

19. Öhm B.K., W. Hübner and K. Väänäen. "GIVEN: Gesture Driven Interactions in Virtual Environments. A Toolkit Approach to 3D Interactions", *Proceedings Interfaces to Real and Virtual Worlds*, pp. 243-254. Montpellier, France, March, 1992.

20. Richard P., G. Burdea, G. Birebent, D. Gomez, N. Langrana and P. Coiffet. "Human Factors Tests for Virtual Environments: Visual Tracking and Haptic Feedback", *Presence - Teleoperation and Virtual Environments*, November 1994 (submitted).

21. Rijpkema H. and M. Girard. "Computer Animation of Knowledge-Based Human Grasping", *Computer Graphics*, Volume 25, Number 4, July, 1991.

22. Schmult B. "Data analysis with a force-reflective joystick", *Internal Report*, A.T.&T. Bell Laboratories, Holmdel, NJ, 11pp. 1990.

23. Sclaroff S. and A. Pentland. "Generalized Implicit Functions for Computer Graphics", *Computer Graphics*, Volume 25, Number 4, July, 1991.

24. Shimoga K. "Finger Force and Touch Feedback Issues in Dextrous Telemanipulation", *Proceedings of NASA-CIRSSE International Conference on Intelligent Robotic Systems for Space Exploration*, Troy, NY, 20pp., September, 1992.

25. Sundgren H., F. Winquist and I. Lundstrom. "Artificial Olfactory System Based on Field Effect Devices", *Proceedings of Interfaces to Real and Virtual Worlds*, Montpellier, France, pp. 463-472, March, 1992.

26. VPL Research Inc. *"DataGlove Model 2 Operating Manual"*. VPL Research, Inc., Redwood City, CA. 1987.

# Intermediate Representation for Stiff Virtual Objects

*Yoshitaka ADACHI, Takahiro KUMANO and Kouichi OGINO*
TECHNICAL RESEARCH CENTER
SUZUKI MOTOR CORPORATION
2-1 SAKURA-NAMIKI, TSUZUKI-KU, YOKOHAMA, 224 J APAN
PHONE +81-45-943-7111 FAX +81-45-943-7100
E-mail adachi@yrd.suzuki.co.jp

**ABSTRACT**

A method of intermediate space for controlling haptic interfaces is characterized by updating a virtual plane at a low frequency while maintaing a high update rate at force control loop of the interface. By using the virtual plane, the detection of collisions between the tip of finger and virtual objects became independent from the control of the haptic interface. This will enable the haptic interface to display more and more complex surfaces in keeping the same sampling frequency of Impedance control. It was revealed by the experiments with a haptic interface SPICE that an operator could touch and trace smoothly on a curved surface of stiff virtual object, even if the update rate of the virtual plane is relatively low.

**KEYWORDS:** Haptic Interface, Force Display, Virtual Environment, Virtual Reality, Impedance Control

## INTRODUCTION

Haptic interfaces are devices which enable us to interact manually with virtual environments[1-7]. In general, the same interface hardware not only receives kinematic inputs from the human operator's limb, but also exert a reflection froce from the virtual environment on his limb. By using the haptic interface, the operator can feel the sense of touch as if he were directly touching virtual objects.

A haptic interface is generally taken as a device which generates mechanical impedance. Touching a stiff surface is described as the sudden transition from a region of very low impedance to one of very high impedance. Therefore it is desirable that haptic interface has wide dynamic range of impedances. However it is difficult to achieve very high impedance under the stable condition of the system. The limitation in realizing stiff surfaces has been discussed in [8] and [9]. Various kinds of factors decide the limitation, but sampling rate of Impedance control is one of the principal factors for realizing very high impedance.

Information flow of conventional haptic interface generally consists of following three stages (Fig.1).

(1) Tracking the movement of operator's limb with high accuracy.
(2) Detecting collisions with virtual objects.
(3) Generating the mechanical impedance of the haptic interface.

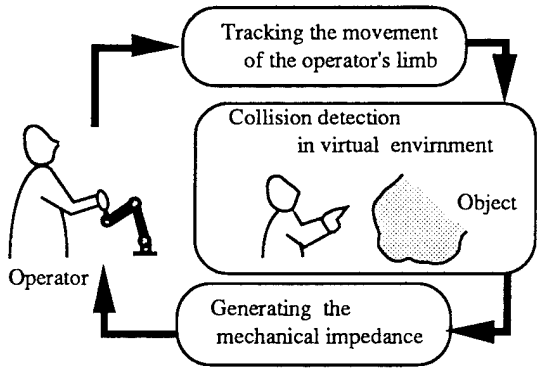



**Figure 1: Conventional method of force reflecting in the virtual environment**

Fast computation in each stage is indispensable to generate stiff surfaces. It is not so difficult to compute in stage(1) and stage(3). However in stage(2) the amount of computation is strongly influenced by complexity of the object shape. For example, though the collision detection with a virtual plane or a virtual ball is relatively easy to compute, the collision detection with free-form surface ( e.g. surface by parametric representation ) requires a large amount of computation and long computation time. Consequently, haptic interface inevitably has a long sampling interval of Impedance control for the computaion, and stiff surfaces can not be represented.

For this problem, intermediate space has been introduced into the haptic interface. Collision detection with virtual objects is computed independently from Impedance control of haptic interface. Geometric transformation in the intermediate space facilitates detection of, and simplifies force vector for, human interaction with simulated objects in virtual environments. Therefore stiff surfaces has been represented even if the virtual objects have complex shapes.

## INTERMEDIATE SPACE

### *Virtual Plane*

The feature of intermediate space is that a virtual environment and Impedance control of haptic interface exchange the information necessary to control by using virtual plane which is frequently recomputed in a



0-8186-7084-3/95 $04.00 © 1995 IEEE

intermediate space( Fig.2 ). In this paper, the tip of an operator's finger is assumed to be a point object in both a virtual environment and the intermediate space. First of all, position of the tip of the operator's finger is calculated from the information of angular sensors attached to a haptic interface. Then, a tangential plane including the nearest point on the complicated surface of the virtual object from the tip of finger is worked out.

The information of this plane is transmitted to the intermediate space which has the same dimension of the virtual environment, and the plane is defined as a virtual plane by following equation(1).

$$ax + by + cz + d = 0 \quad (1)$$

where x,y,z are Cartesian coordinates in the intermediate space. The information of a domain in which the virtual object exists also should be transmitted to the intermediate space. Naturally, the position and the orientation of the virtual plane is frequently recomputed according to the finger movement.

In parallel with this computation, collision detection between the tip of finger and the virtual plane is executed extremely fast. In the case of interference between them, a reaction force is calculated after the collision detection( Fig.3 ). The penetrated depth( $\Delta r$ ) of the tip of finger from the surface of the virtual plane is estimated into the equation(2).

$$\Delta r = |ax_0 + by_0 + cz_0| / \sqrt{a^2 + b^2 + c^2} \quad (2)$$

where $x_0$, $y_0$, $z_0$ are the position of the finger tip. The magnitude of reaction force(Fv) is expressed by the following equation.

$$Fv = K \cdot \Delta r + B\dot{q} \quad (3)$$

where q is the position of the tip of finger. K and B are stiffness and viscosity respectively. Naturally, the reaction force is to be estimated to be zero if the tip of finger is outside of the virtual plane in the intermediate space.

Computation time of getting a tangential plane on a surface of virtual object depends on complexity of the surface. The update rate of virtual plane is to be varied owing to the computation time. However, computation of collision detection between the tip of finger and the virtual plane becomes very simple and it is instantly completed.

*Artificial Friction*

Equation(3) generates only a force in the normal direction of the surface. Such surfaces have no friction in other directions just like a surface of ice. To keep the finger pushing the surface vertically without friction is difficult, because wrong direction of pushing causes a slip on the surface. Artificial friction has been introduced into the virtual plane for this reason( Fig.4 ). Friction force(Fh) is estimated by the following equation.

$$Fh = uV \quad (4)$$

where u is viscous coefficient and V is velocity of tip of finger in the tangential direction to the virtual plane.

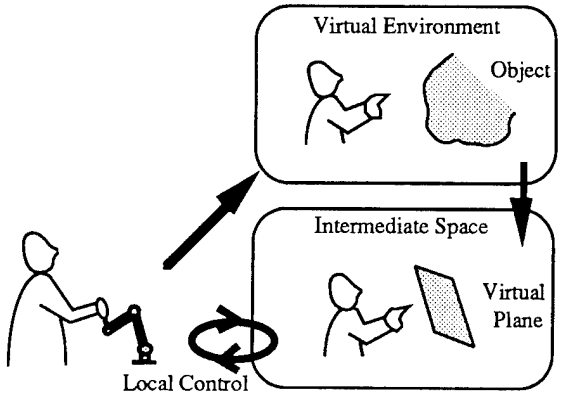



**Figure 2: Intermediate representation of virtual environment by using a virtual plane**

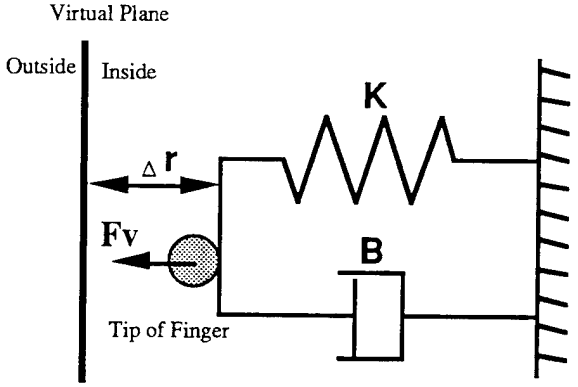



**Figure 3: Force reflecting from the virtual plane**

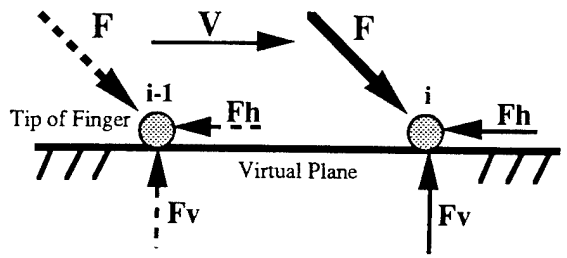



**Figure 4: Artificial friction on the virtual plane**

*Limitation*

If update rate of the virtual plane is moderately fast, the operator can feel curved surfaces. However low update rate makes a bumpy surface like a surface of polyhedron. In addition to this, various kinds of factors also make a bumpy surface. The limitation is given by D = f( V, F, T, M, R ), where V is velocity of the tip of finger, F is contact force, T is update rate of the virtual plane, M is structural inertia of a haptic interface, R is curvature at the surface of virtual object and D is distance between the last position of the tip of finger on the former virtual plane and the first position on the current virtual plane( Fig.5 ). If D is as small as we can not perceive it, the surface will be felt as a smoothly curved surface. D will be given by a simple experiment to check whether operators can find difference in level or not( Fig.6 ). In our experimental system, the limitation of D was estimated as 0.8 mm under the condition of K = 10000 (N/m) and B = 1000 (N/(m/sec)).

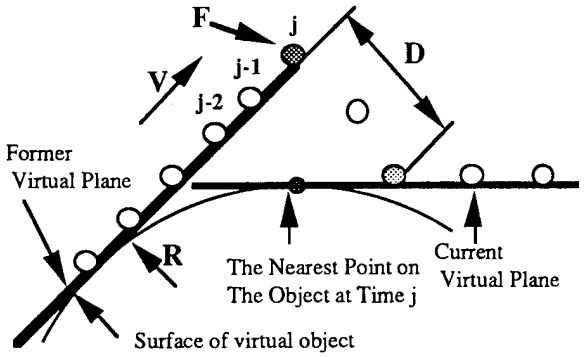



**Figure 5: Limitation of the virtual plane**

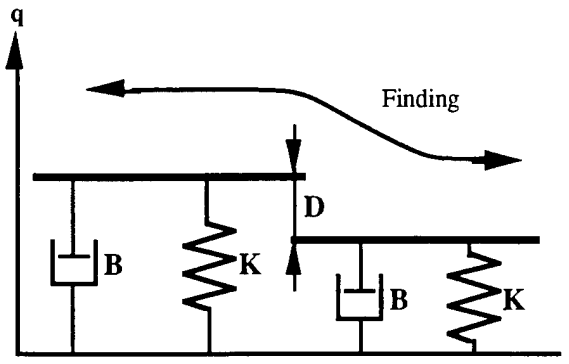



**Figure 6: Experiment of finding a difference in level**

## EXPERIMENTAL SETUP

A haptic interface which is able to generate a suitable reaction force is necessary for giving the realistic sense of touch to the operator. The haptic interface which is called 'SPICE' was utilized for the experiments[10]. SPICE was designed under the consideration as follows, (1) mechanical stiffness, (2) joint structure with lower inertia and less friction, (3) arm structure effective to simplify the control law, (4) necessary but minimum number of sensors and (5) actuators of direct-drive motors with wide range of torque.

SPICE is shown in Fig.7 and Fig.8. An articulated structure with invariant and decoupled arm inertia[11] and six degrees-of-freedom were employed in SPICE. An arm structure and its mass distribution were optimized with the generalized inertia ellipsoid( GIE )[12]. Distribution of GIE is shown in Fig.9.

High resolution optical encoders sense angular position of each joints of SPICE. The grip position and orientation, moreover velocity and acceleration are computed from the information of optical encoders. Nominal spatial resolution is 0.01mm at the center of work space. Direct-drive motors ( brushless DC motors, SinMaywa Industries Ltd. ) were selected for small reluctance cogging and torque ripple. The specification of sensors and actuators are shown in Table 1. Fig.10 shows the result of preliminary experiment in which the relation between force command of the controller and the output of SPICE was examined. The generating force is exactly according to the force command.

High computation performance is required for the controller of SPICE. Two CPUs ( M68030, Motorola Inc. ) with floating point coprocessor are running at 25 MHz clock speed in this system. One simulates the virtual environment and gets a tangential plane on a virtual object, another controls mechanical impedance of SPICE by using a virtual plane. The information of the virtual plane are exchanged through VMEbus( Fig.11). However, these CPUs do not have enough capacity to compute the control algorithms in real time. Thus, three vector processors ( VP, MB92831 by Fujitsu Ltd.) were introduced in this system. Each of them provide a sustained rate of 206 MFLOPS( peak, 32-bit floating point ) at 50 MHz clock speed.

Fig.12 shows general view of the experimental system. IRIS 420/VGX is used for visualization of the virtual environment. An operator seates 100 cm apart in front of a display of IRIS, and holds the grip of SPICE by the right hand.

**Table 1: Specification of sensors and actuators**

| Joint | Brushless DC Motors | | | Encoders |
|---|---|---|---|---|
| | Max Torque [N/m] | Motor Const. [Nm/$\sqrt{W}$] | Model | Resolution [ PPR] |
| 1 | 227.0 | 2.17 | B18-76 | 324,000 |
| 2 | 189.0 | 1.92 | B18-64 | 324,000 |
| 3 | 75.8 | 0.97 | B18-25 | 324,000 |
| 4 | 19.3 | 0.44 | B09-51 | 10,000 |
| 5 | 3.71 | 0.14 | B06-25 | 10,000 |
| 6 | 1.66 | 0.073 | B05-25 | 10,000 |

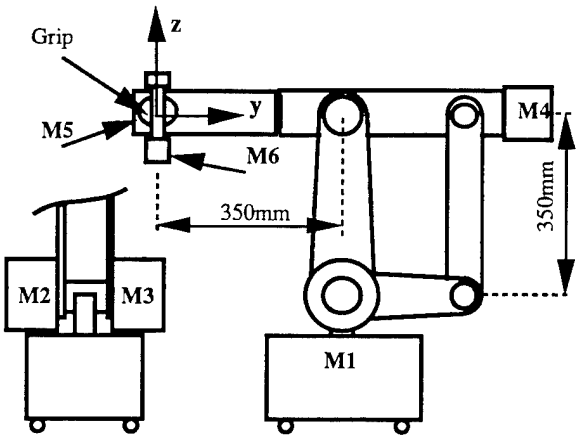



Figure 7: Structure of mechanism of SPICE

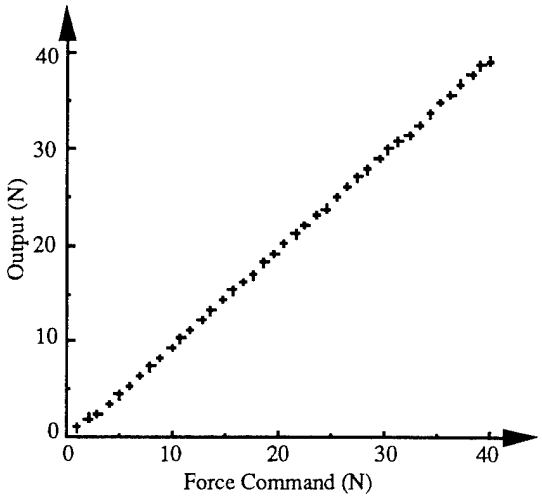



Figure 10: Force command and output of SPICE Y direction at center of work space

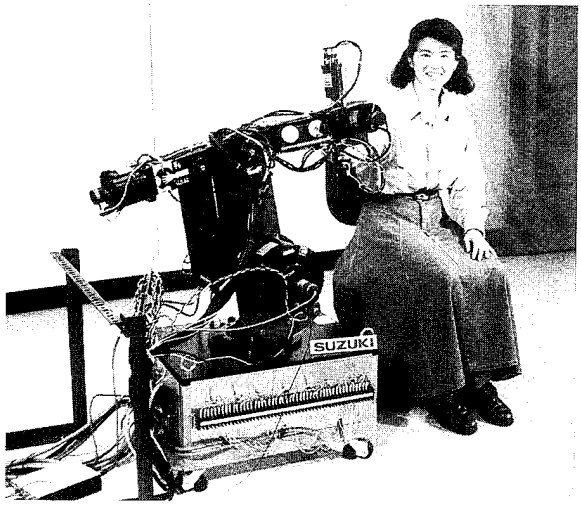



Figure 8: General view of SPICE

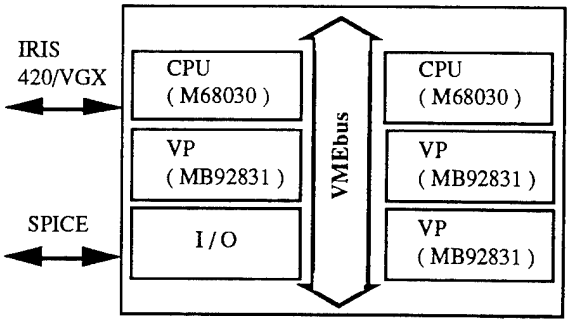



Figure 11: Structure of the controller

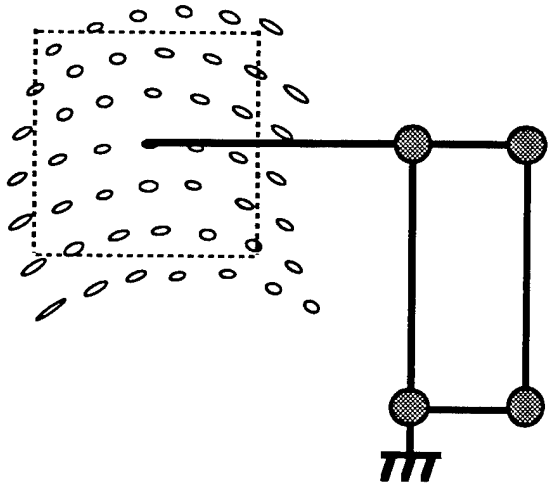


Figure 9: Distribution of GIE

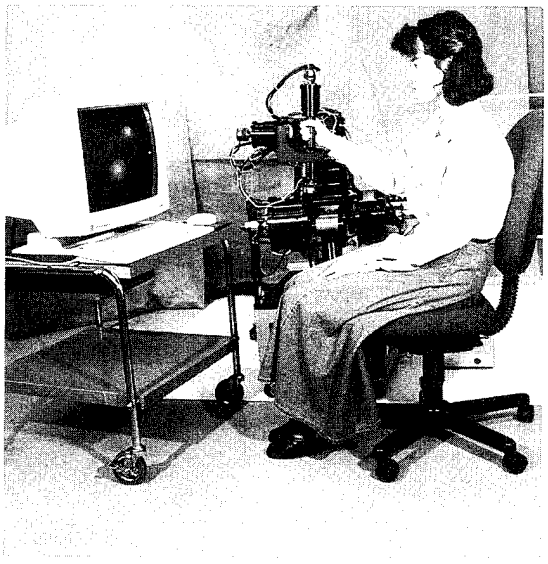


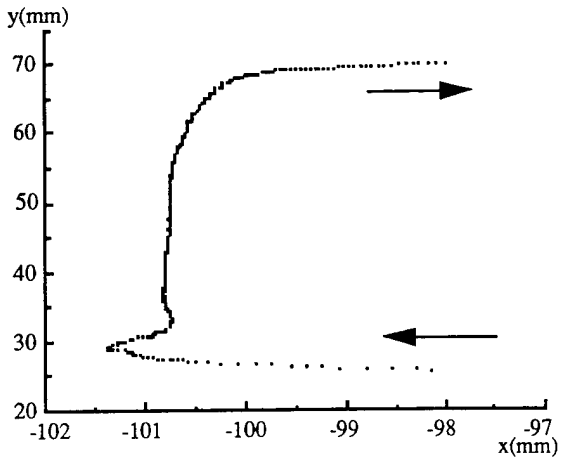
Figure 12: General view of experimental system

## EXPERIMENT

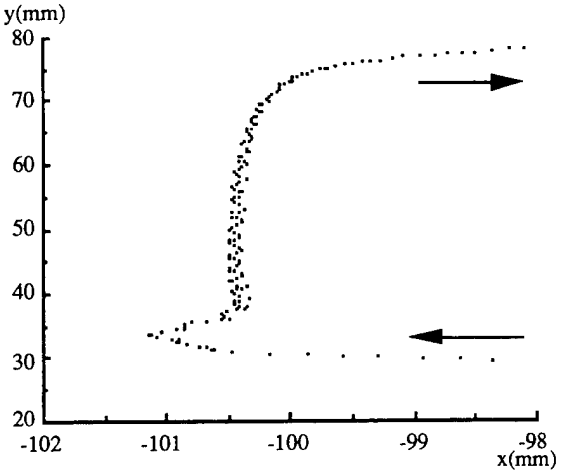
*Experiment(1)*

As a preliminary experiment, a tracing task by conventional approach(Fig.1) was carried out. The operator touched on a stiff wall( K = 10000 N/m, B = 1000 N/(m/sec)) which was placed at X = -100 mm. Three sampling intervals(T), 2mSec (1/T=500Hz), 4mSec (1/T=250Hz) and 6mSec (1/T=167Hz)) were prepared for the experiment.

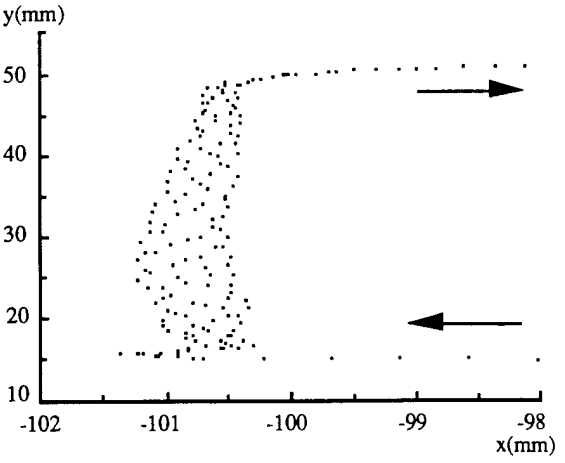
The trajectories projected on x-y plane are shown in Fig.13. The dots show the movement of the tip of the operator's finger at each sampling time. Fig.13(a) indicates that the operator could touch and trace on the surface smoothly. However, when the sampling intervals were 4mSec and 6mSec, unpleasant vibration was occurred (Fig.13 (b),(c)). From the result, it is confirmed that sampling rate of Impedance control should be greater than 500Hz for the implementation of stiff objects.




(a) Sampling Interval : 2 mSec ( 500 Hz )




(b) Sampling Interval : 4 mSec ( 250 Hz )




(c) Sampling Interval : 6 mSec ( 167 Hz )

**Figure 13: Tracing on the stiff wall placed at x = -100 by conventional method**
**Stiffness and viscosity are 10000 N/m and 1000 N/(m/sec) respectively.**

*Experiment(2)*

Tracing tasks with the intermediate space were carried out. The operator traced on a surface of a virtual cylinder with the radius of 75mm(Fig.14 ). Sampling interval of Impedance control by using the virtual plane was 1mSec (1000 Hz). The stiffness and the viscosity of the virtual plane were 10000 N/m and 1000 N/(m/sec) respectively.

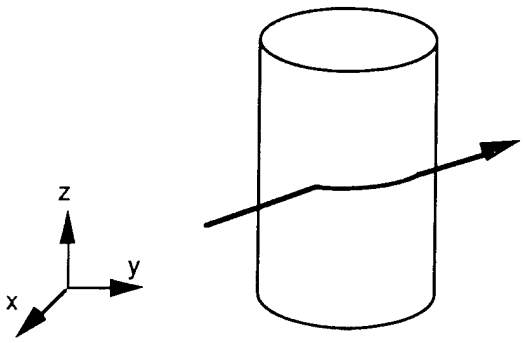



**Figure 14: Touch and trace on the surface of a cylinder by using the virtual plane**
**Stiffness and viscosity are 10000 N/m and 1000 N/(m/sec) respectively.**

Update interval of the virtual plane was 100mSec( 10Hz ). However artificial friction was not introduced in this experiment. The trajectories projected on x-y plane are shown in Fig.15. The dots show the movement of the tip of finger at each 20 mSec. Fig.15 shows the surface of the virtual object is slippery and it is difficult to keep in contact.

Then, artificial friction was introduced on the virtual plane. The viscous coefficient of artificial friction was 600 N/(m/sec). Other condition were the same as before. The trajectories are shown in Fig.16. The dots indicate that the operator could trace easily and smoothly on the stiff surface of the virtual cylinder.

Next, updating interval of the virtual plane was changed to 300mSec(3.3Hz). Other conditions were the same as before. The trajectories shown in Fig.17 indicate that unpleasant vibration was not occured in spite of low update rate. The average velocity of the operator's finger was 20 mm/sec.

To check the effects caused by lower update rate. The updating interval was changed to 400mSec( 2.5 Hz ). Other condition were the same as before. The trajectories shown in Fig.18 indicate that unpleasant vibration due to the bumpy surface was occured. However, under the condition the operator could trace on the surface smoothly as shown in Fig.19 if he moves his finger more slowly. In this case, the average velocity of the operator's finger was 8 mm/sec.

As the results, if update rate of the virtual plane is moderately fast compearing to the movement of the operator's finger, the operator does not feel the bumpy surface. Lower limitation of update rate depends heavily on the traceing velocity of the operator's finger on the surface of virtual objects.

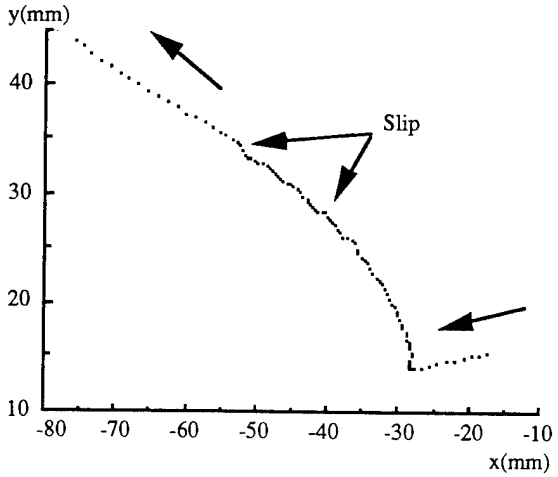



**Figure 15: Tracing on a cylinder without artificial friction. It is difficult to trace on the surface due to slipping.**

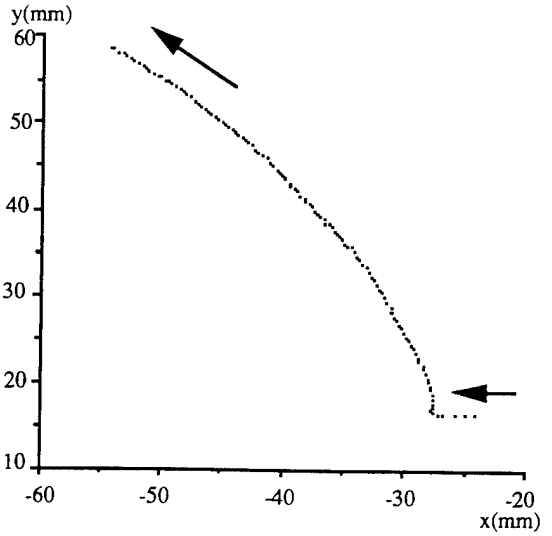



**Figure 16: Tracing on a cylinder with artificial friction It is easy to trace on the surface without slipping.**

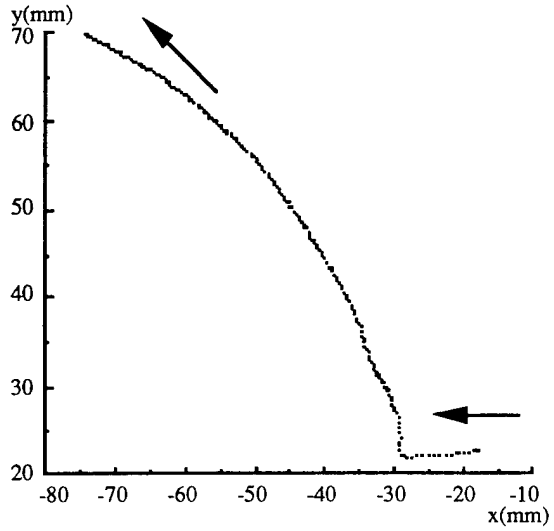



**Figure 17: Tracing on a cylinder with artificial friction Updating the virtual plane was 300mSec (3.3 Hz), and average velocity of finger tip was 20 mm/Sec.**

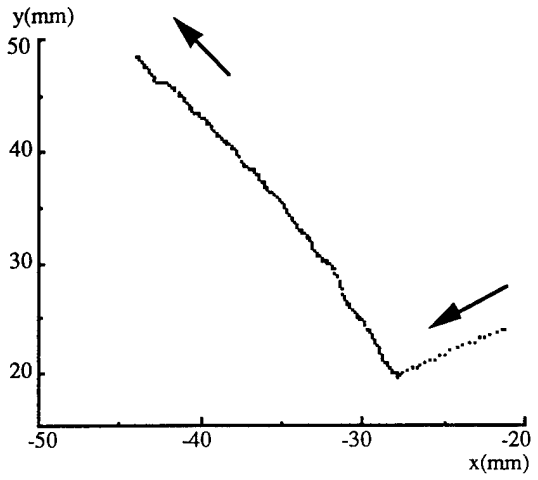



**Figure 19: Tracing on a cylinder with artificial friction Updating the virtual plane was 400 mSec ( 2.5Hz ), and average velocity of finger tip was 8 mm/sec.**

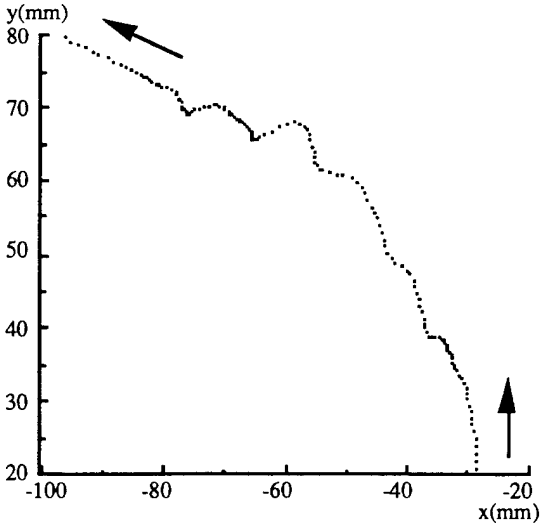



**Figure 18: Tracing on a cylinder with artificial friction Updating the virtual plane was 400 mSec ( 2.5 Hz ), and average velocity of finger tip was 12 mm/sec. Unpleasant vibration was occurred.**

## CONCLUSIONS

The authors have proposed a method of an intermediate geometric transformation that facilitates detection of, and simplifies force vector computation for, human interaction with stiff surfaces in virtual environments. An tangential plane at the nearest point on a virtual surface from the tip of an operator's finger is represented as a virtual plane in an intermediate space. By using the virtual plane, the detection of collisions between the tip of finger and the surfaces of virtual objects becomes independent from the control of a haptic interface. The idea of intermediate virtual plane has two advantages in reducing sampling rate of collision detection and increasing sampling rate of impedance control. Therefore the operator can touch and trace smoothly on stiff and complex virtual surfaces. It was revealed by the experiments with SPICE that an operator could touch and trace smoothly on the curved surface of stiff virtual objects even if the update rate of the virtual plane was slow. The experiments indicated that the intermediate space is helpful and useful for reducing computation and actualizeing complex haptic environments.

## REFERENCES


[1] Millman,P.A., Stanley,M. and Colgate,J.E., Design of a High Performance Haptic Interface to Virtual Environments, Proc. of IEEE Virtual Reality Annual International Symposium '93, pp.216-208, 1993.

[2] Iwata,H., Artificial Reality with Force-feedback: Development of Desktop Virtual Space with Compact Master Manipulator, Computer Graphics, Vol.24, No.4, pp.165-170, 1990.

[3] Adelstein,B.D. and Rosen,M.J., Design and Implementation of a Force Reflecting Manipulandum for Manual Control Research, In Advance in Robotics, ASME Winter Annual Meeting, pp.1-12, 1992.

[4] Howe,R.D., A Force-Reflecting Teleoperated Hand System for the Study of Tactile Sensing in Precision Manipulation, Proc. of IEEE International Conference on Robotics and Automation, pp.1321-1326, 1992.

[5] Brooks,F.P, Ouh-Young,M., Batter,J.J. and Jerome,P., Project GROPE-Haptic Display for Scientific Visualization, Computer Graphics, Vol.24, No.4, pp.177-185, 1990.

[6] Kotoku,T., Komoriya,K. and Tanie,K., A Robot Simulator with Force Generating Function, Proc. of the Second International Symposium on Measurement and Control in Robotics(ISMCR'92),pp.805-810, 1992.

[7] Hirota,K. and Hirose,M., Development of Surface Display, Proc. of IEEE Virtual Reality Annual International Symposium '93, pp.256-262, 1993.

[8] Colgate,J.E., Grafing,P.E., Stanley,M.C. and Schenkel,G., Implementation of Stiff Virtual Walls in Force-Reflecting Interface, Proc. of IEEE Virtual Reality Annual International Symposium'93, pp.202-208, 1993.

[9] Minsky,M., Ouh-Young,M., Steel,O., Brooks,F.B. and Behensky,M., Feeling and sensing: issues in force display, ACM Computer Graphics, 24-2, pp.235-243, 1990.

[10] Adachi,Y., Touch and Trace on the Free-Form Surface of Virtual Object, Proc. of IEEE Virtual Reality Annual International Symposium'93, pp.162-168, 1993.

[11] Asada,H. and Toumi,K.Y., Mechanism and Control of a Lightweight Direct-Drive Robot with Invariant and Decoupled Arm Inertia, Trans. of the Society of Instrument and Control Engineers, Vol.20, No.12, pp. 1161-1168, 1984. (in Japanese)

[12] Asada,H., A Geometrical Analysis of Manipulator Dynamics, Trans. of the Society of Instrument and Control Engineers, pp500-505, 1983. (in Japanese)

# Simulation and Presentation of Curved Surface in Virtual Reality Environment Through Surface Display

Koichi Hirota Michitaka Hirose

Department of Mechano-Informatics
Faculty of Engineering, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113, Japan
E-mail : hirota@ihl.t.u-tokyo.ac.jp, hirota@ihl.t.u-tokyo.ac.jp

## Abstract

*Force feedback can be thought as a concept which was motivated by the phenomenon of contact. Efforts were made to describe cross sections in interfaces and methodologies to realize force feedback from differences in the cross sections. Based on this discussion, the concept of Surface Display was submitted and technical issues for the implementation of the concept were pointed out. A prototype device to demonstrate the concept was created and solutions for technical issues were suggested.*

*The concept of Surface Display is an idea to present the surface of a virtual object itself to the user, rather than the sensation of force or tactile caused by the contact with the virtual object. The prototype device consisted of a mechanism to form arbitrary surfaces and sensors to measure the force affected on the surface. The control and calculation methods for this device are also described.*

## 1 Introduction

The sensation of force has come to be recognized as an important factor for successful manipulation of objects in virtual environments. According to experimental results in the fields of tele-robotics, the sensation of force and contact improves the efficiency and accuracy in manipulation tasks [1]. These results are believed to be applicable as well in virtual environments. If we reflect upon the concept of virtual environments, the similarity of the virtual environment to the real world is considered a fundamental requisite. Hence, the feedback of force is indispensable in providing more similarity of the real world in manipulation tasks in virtual environments. In this paper, the development of a force feedback device to support manipulation tasks in virtual environments is discussed.

## 2 Concept of Surface Display

Force feedback can be thought of as a concept which was motivated by the phenomenon of contact. In the implementation of virtual environments, virtual objects are defined in the computer while the human users exist in the real world. Hence, the interaction between the real user and virtual objects has to be intermediated by a force feedback device. The required

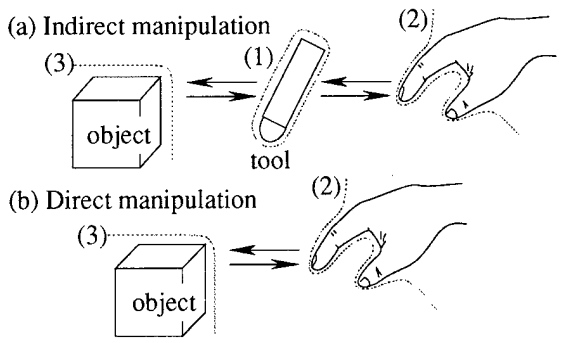



Figure 1: Interface in Contact

structure of the device varies according to the differences in the cross sections of the interface on which we are remarking.

The interfaces for contact in the real world can be divided into two cases: indirect contact using tools and direct contact with hands or other part of our body (see Figure 1). On considering these two situations, we can find three typical cross sections in the interface: (1) the Surface of a Tool, (2) the Surface of the User (ie. skin), and (3) the Surface of an Object as shown in Figure 2 [4].

If we consider the cross section on the surface of a tool, this interface is aimed at the indirect manipulation of a virtual object using a tool. A force feedback device based on this methodology and classification must simulate the relationship between the position and force applied to the tool [2]. In general, a tool has about 6 degrees of freedom or a little more, which is much less than that of the human hand. The contact between the tool and the virtual object occurs in the virtual world, it must be simulated and detected by the computer. A tool in general has a more simple and solid shape than that of the human hand , which makes the calculation and detection of contact between the tool and objects easier.

If we consider the cross section on the surface of a human user, this interface enables the direct manipu-



0-8186-7084-3/95 $04.00 

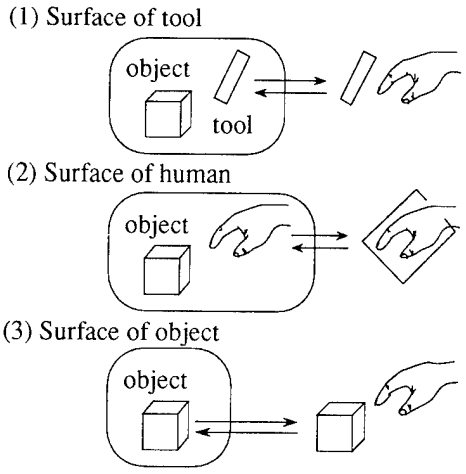



Figure 2: Typical Cross Sections

lation of a virtual object [3]. Similarly, a force feedback device based on this type of methodology and classification must also simulate the relationship between position and force on the surface of the user's skin. Since the majority of the interaction between a human user and a virtual object is realized through the hands, an effective force feedback device of this type should be designed to provide the sensations of force on the surface of user's hand. The human hand has more than 20 degrees of freedom, and has complicated and transforming shapes. This fact makes the design of a force feedback device difficult due to the need to model the human hand in the computer, as well as the need to enable detection of contact between the virtual hand and object. In addition, this type of device has traditionally been implemented as a mechanism which is worn on the user's hand. This makes it difficult to avoid the sensation of "wearing" a device.

If we take a cross section on the surface of a virtual object, the necessary interface attempts to provide the the presentation of the virtual object to the real world and the user. A force feedback device based on this principle must simulate the existence and behavior of the object, including the relationship between the position and force on the surface of the object. In order to implement of this type of device, we can expect some difficulties in trying to present the the shape of the object to the user. On the other hand, there are some advantages with respect to using this type of force feedback device. Namely, the contact occurs in the real world and not in virtual world. Therefore, detection of contact is not necessary with the computer. A computer model of the human hand or other parts of the body are also not needed. Another advantage is the fact that this force feedback device is a "non-wearing" type, which enables users to better distinguish between contact and non-contact state.

Since we believe strongly that the sensation of direct contact with a virtual object was an important issue in virtual environments, we adopted a method to take cross section on the surface of objects for our force feedback environment. We named this way of thinking the concept of surface display.

It is often said that tele-robotic environments and virtual environments are similar to each other. In the tele-robotic environments objects are in the real world around tele-robot. However, in virtual environments, objects are defined in computer. In the implementation of surface display it is a important difference. Tele-environment in the tele-robotic system, is usually unknown in the tactile sense and the contact force between slave arm and the object is a unique information about the object. In contrast, in the virtual environment system, all the information including tactile characteristics, is held in computer and can be referred to at any time. This fact offers facility also in manipulation of object in the virtual world.

## 3 Implementation of Surface Display and Previous Research

If we consider the implementation of Surface Display, the following technical issues are essential:

- The surface of any given shape must be presented to the user.
- All force affected to the surface must be measured.
- The surface must be sensitive to force and the reaction must be determined by simulation.

In addition, as it is not always easy to display the whole shape of the object, only a part of it can be presented to the user. In this case, following technical issue is also important:

- Natural selection of the part to be presented.

Based on the concept of Surface Display, and discussion of technical issues, we developed a prototype device [5][6][7]. In this device, the contact area was limited to the user's fingertip and the contact between the fingertip and the object was assumed to occur only at one point. Therefore, the displayed surface was represented by the tangent plane at the nearest point on the object to the fingertip.

According to the experimental use of this type of prototype, we were able to assess the feasibility of the concept of surface display. However, we also found some problems in the first prototype:

- The presentation of convex or concave corner surface made us feel the sensations of incongruity.
- Interaction with two or more fingers was not possible.
- Tracking of the fingertip by the mechanism cause oppressive sensations to the user.

In the presentation of corner surfaces, the normal vector of the tangent surface turns quickly according to the motion of fingertip, and it disturbs the recognition of shape by the user. Interaction was limited to

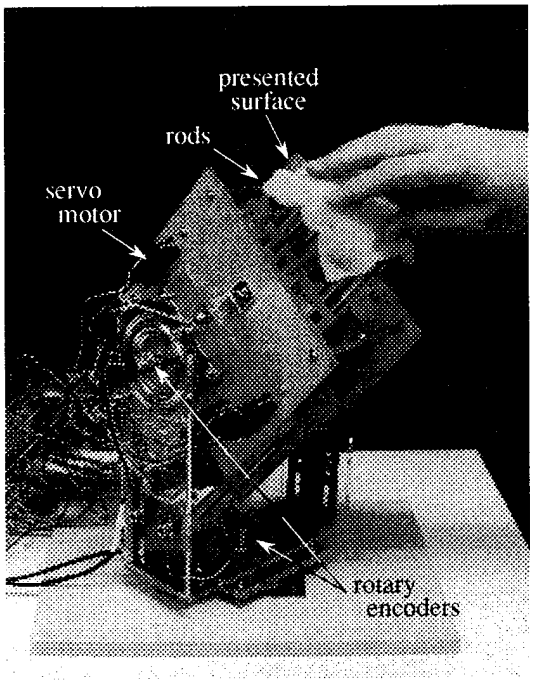



Figure 3: Prototype Device

one fingertip because approximation by the tangent plane method had better accuracy close to the point of contact. Tracking with a mechanical arm was necessary because high accuracy was required to measure fingertip position, from which the contact point on the object and tangent surface at the point were calculated. These problems were proposed to be solved by presenting the shape of the object, not as an approximation by the tangent plane surface, but as the curved surface as it is. In the next section, we will discuss a prototype Surface Display device presenting curved surfaces.

## 4 Prototype Device

In order to demonstrate the concept of Surface Display and to provide some solution for the above mentioned technical issues, a new prototype device was created.

### 4.1 Mechanism

The prototype device is divided in two parts: a Tracking Part and a Display Part. The Tracking Part consists of two degrees of passive freedom to change the position of the Display Part. This mechanism offers the user the ability to select the operation area on the surface of the object (see Figure 3). The rotational motion of each degree of freedom is measured by a rotary encoder, which provides an accuracy of 0.25 degrees. In future prototypes, active tracking with more than six degrees of freedom is being considered.

The Display Part has sixteen parallel rods, whose projecting length is controlled by servo motors attached to each of the rods. The rotational motion

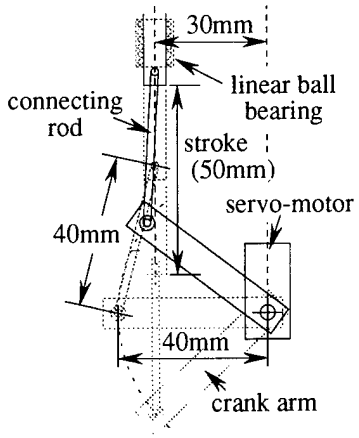



Figure 4: Dimensions of Slider-Crank Mechanism

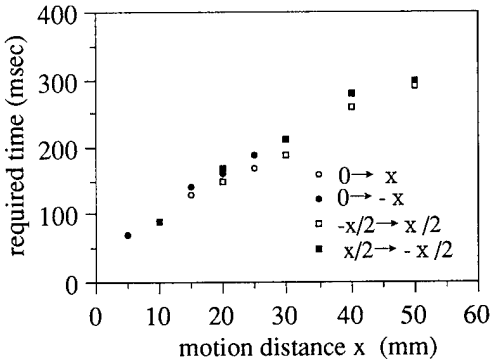



Figure 5: Actuation Speed of the Rods

caused by the servo-motors are transferred into linear sliding motions through a slider-crank mechanism (see Figure 4). According to the design specifications, the rotor of the servo-motor moves 60 degrees in 0.18 seconds with a 1.9 kg-cm torque. When it was mounted on the mechanism, the rods were capable of a the full stroke (50 mm) motion within 300 milliseconds (see Figure 5).

The rods were arranged on the grid points of a $4 \times 4$ square lattice, and the intervals of the grid points are $20 \times 20[mm]$. Each of the rods has a stroke range of $-25[mm]$ to $25[mm]$ relative to its neutral position.

The Display Part presents the given physical shape/surface of an object by changing the projecting length of the rods. For the interpolation of the surface between the rods, a soft foam urethane sheet was mounted on the top of rods.

### 4.2 Control System

The mechanism was connected to a personal computer (PC) (see Figure 6). The status of the Tracking Part Mechanism was measured by rotary encoders and decoded by a mouse interface built into the PC. The servo-motor requires a digital signal input, whose

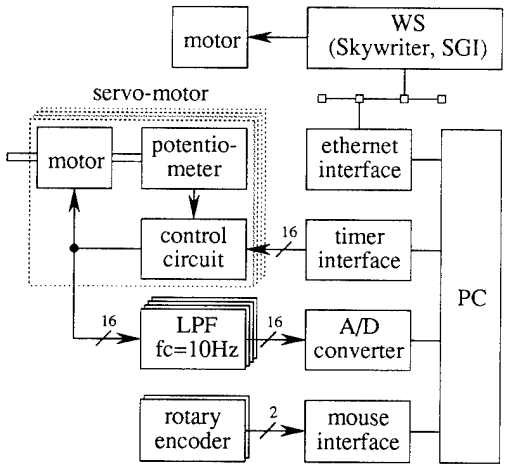



Figure 6: System Block Diagram

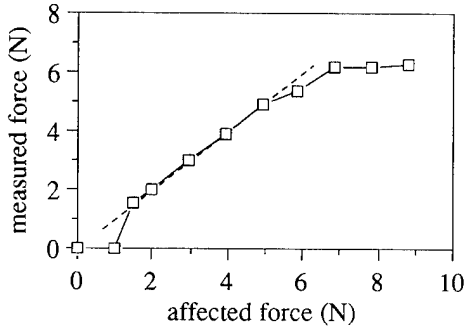



Figure 7: Measurement of Force

high-level pulse width is interpreted into the goal angles of servo control. The timer interface generates pulse signals for each of the motors. The conversion from given projection length to pulse width was done by referencing a table that was calculated in advance with an accuracy of 0.1 mm interval.

Measurement of the force exerted on to the surface of the object by the user was also an essential technical issue in our requirements for Surface Display. In our prototype, the axial force exerted on each rod was measured from the voltage charged on each motor. Torque offered to the motor causes errors in the servo control, and the error reflected on the voltage. As the actuation of the motor by servo circuit was proved to be made by PWM control, the signal was averaged by LPF, whose cut off frequency was designed to be 10 Hz, before the signal was input to an A/D converter interface.

When the motor is close to its goal status, the voltage increases almost proportionally to the force (see Figure 7). In the case where the servo motor was moving towards a new goal status with large error, we could not apply this method and the measurement of the force was suppressed.

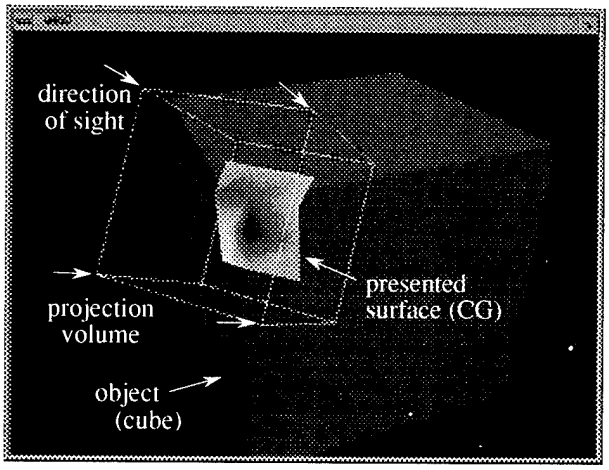



Figure 8: CG Image of Displayed Surface

The connection to the host work station (WS) was made by TCP through Ethernet.

### 4.3 Calculation Process

In the presentation of surfaces, the PC and the WS work cooperatively. The PC receives the projection length data from the WS, and it returns the data of the status from tracking mechanism and the force affected on the rods to the WS.

The tasks of the PC include the following items:

- read data from port (if available)
- set motor control output
- get force and rotary encoder input
- write data to port

All of these procedures are completed at least once within every 100 milliseconds.

The tasks of WS include the following items:

- read port for force and mechanism status input
- update the model of the virtual object (necessary when the object moves or transforms)
- calculate the projection length of rods
- write results to the port

For the calculation of projection length for each rod, points of intersection between the surface of the object and the tips of rods must be calculated. To achieve this, we can imagine a camera attached to the display mechanism (see Figure 8). The camera has an orthographical view, whose direction of sight is always parallel to the axes of the rods. In the images taken by this camera, the axes of rods are represented as points, and the desired projection length of each rod is determined as the distance from the projection plane of the camera to the object surface (see Figure 9). The workstation that we have applied for visual

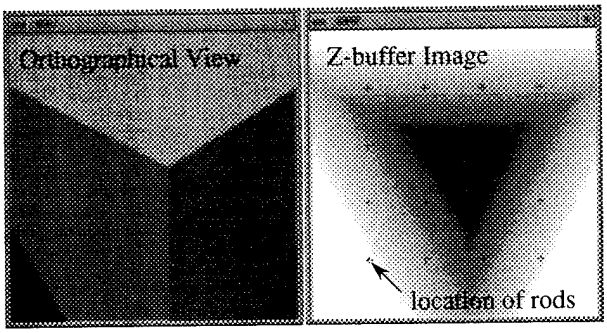



Figure 9: Orthographical View and Z-buffer Image

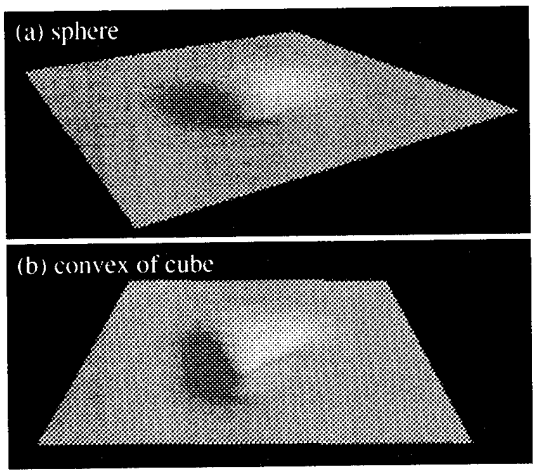



Figure 10: Transcription of Shape

presentation of this virtual environment supports the function of z-buffering, and we tried to measure the the distance by reading out the z value stored in the z-buffer. According to the motion of the object and the presentation mechanism, the orthographical image also changed and the projection length of rods was simultaneously updated.

### 4.4 Interaction Using Force

As an example of the simulation of an object reacting to the force, we tried to make a virtual sheet whose shape transforms according to the amount of force exerted on the sheet. The virtual sheet was defined as a mesh. Height was defined on each of $38 \times 38[grids]$, whose size of lattice is $4 \times 4[mm]$. Displacement of the surface was calculated according to the following expression:

$$dz(d, F) = \frac{F}{F_0}\{\exp(-\frac{d^2}{k}) - \exp(-\frac{d_0^2}{k})\}$$

Where $d[mm]$ is the plane distance from the rods, and $F[N]$ is the intensity of force affected on the rod. Constants appearing in the expression are defined as follows: the maximum distance influenced by the force $d_0 = 44[mm]$, a constant determining the acuteness of transformation $k = 200[mm^2]$, and the constant to determine the sensitivity against force $F_0 = 3.0[N/mm]$. By replacing this function with another function, the transformation characteristics of the virtual sheet can be changed. However, it should be considered that the device represents the surface by rods ordered discretely. According to sampling theorem, the highest frequency that can be presented by this display mechanism is 0.25 /cm. The above mentioned expression and constants were determined based on this discussion and experimentation.

The transformation is calculated as a sum total of the displacement for all the rods in every calculation cycle. The presented surface should be handled in just the same way as the surface of a real object placed in the same place. Interaction with the object is not always made by hand but also made by other objects. We tried to transcribe the shape of a sphere and a convex corner of a cube onto the virtual sheet (see Figure 10).

## 5 Conclusion and Future Work

In our prototype, we developed a mechanism capable of providing surface display of curved surface using parallel rods. This type of mechanism was applicable only for a shape which could be defined as a form of $z = f(x, y)$. However, in many cases, this assumption was not satisfied. Further study is required on shape presentation mechanisms, and the concept of Roboxel and the idea of Robotic Graphics will offer future visions about the presentation of shape [8].

Not only the axial force, but also the thrust force on displayed surfaces must be measured, and the effect of such forces must also be simulated. The degrees of freedom for the measurement of force need not be correspondent directly to that of the actuation and it may be a good solution to distribute sensors on the presented surface.

The tracking part of the force feedback display device must have more than six degrees of freedom with active control. To determine the position and orientation of the display part, the position of the hand or some other part of the users body must be monitored. For the sensor, not so much accuracy is necessary, and spatial sensors such as the Polhemous sensor would be advantageous because there is less sensation of "wearing" it.

In summery, we tried to make a systematic classification of a methodology for force feedback in virtual environments. Based on this discussion, the idea of Surface Display was submitted along with technical issues for implementation. As a prototype, we designed and developed a Surface Display device presenting curved surfaces.

## References

[1] Karun B. Shimoga, "A Study of Perceptual Feedback Issues in Dexterous Telemanipulation", *Proc. of VRAIS'93*, pp. 263 - 279, IEEE, 1993.

[2] Frederick P. Brooks, Jr., Ming Ouh - Young, James J. Batter, P. Jerome, "Project GROPE - Haptic

Displays for Scientific Visualization", *Computer Graphics*, Vol. 24, No. 4, pp. 177 - 185, ACM SIGGRAPH '90, 1990.

[3] Hiroo Iwata, "Artificial Reality with Force Feedback : Development of Desktop Virtual Space with Compact Master Manipulator", *Computer Graphics*, Vol. 24, No. 4, pp. 165 - 170, ACM SIGGRAPH '90, 1990.

[4] Koichi Hirota, Michitaka Hirose, "Surface Display: A Force Feedback System Simulating the Surface of an Object", *Proc. of RO-MAN'94*, pp. 251 - 254, IEEE, 1994.

[5] Michitaka Hirose, Koichi Hirota, Ryugo Kijima, "Human Behavior in Virtual Environment", *Proc. of SPIE/IS&T's Symposium on Electronic Imaging: Science & Technology*, SPIE, 1992.

[6] Michitaka Hirose, Koichi Hirota, "Surface Display and Synthetic Force Sensation", *Advances in Human Factors/Ergonomics*, Vol. 19B, Human-Computer Interaction, pp. 645 - 650, ELSEVIER, 1993.

[7] Koichi Hirota, Michitaka Hirose, "Development of Surface Display" *Proc. of VRAIS'93*, pp. 256 - 262, IEEE, 1993.

[8] William A. McNeely, "Robotic Graphics: A New Approach to Force Feedback for Virtual Reality", *Proc. of VRAIS'93*, pp. 336 - 341, IEEE, 1993.

# Pen-Based Force Display for Precision Manipulation in Virtual Environments

*Pietro Buttolo and Blake Hannaford*
Biorobotics Laboratory, Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195
(206) 543-9378
pietro@ee.washington.edu, blake@ee.washington.edu

## ABSTRACT

In this paper we describe the structure of a force display recently implemented for precision manipulation of scaled or virtual environments. We discuss the advantages of direct-drive parallel manipulators over geared serial manipulators for human-robot interaction application and introduce the serial-parallel structure we chose for our robot which interfaces with the human operator either at the fingertip or at the tip of a freely held pen-like instrument. We derive the statics and the dynamics, and then introduce the optimization criteria that allowed us to choose the dimensional parameters for the force display. Finally we show some of the potential application for this device that will be the subject of following papers.

**KEYWORDS:** force display, haptic interface

## INTRODUCTION

A force display is a manipulator designed to provide and receive kinaesthetic information to/from an human operator. Master manipulators used in telerobotics systems are an example of such devices. Using a master an operator can perform remote manipulation feeling simulated contact with the remote site. A force display can also reproduce feeling coming from a virtual environment, providing in this way a mechanical interface for virtual reality applications.

In the literature it is possible to find various examples of force-display and master devices; parallel mechanism [Ramstein & Hayward 1994], [Hayward & Kurtz 1984], [Millman & Colgate 1991], magnetically-levitated devices, [Salcudean & Yan 1994], [Hollis & al 1990], and others as [Hirota & Hirose 1994], [Sato & al 1994], [Iwata 1994].

The sensation of contact with the real site while operating in a remote station is often referred in the telerobotics literature as telepresence. The same concept can be easily adapted to virtual reality applications. To achieve telepresence the human has to interact with the telerobotics/virtual reality system in a natural way. In an ideal case he should not be able to tell if he/she using the force display or his/her natural tool. For example, a surgeon interacting with a force display should feel as he/she is grasping a scalpel. Therefore, the physical characteristics of the force display must be as close as possible to those of the natural tool. For micro-surgery applications the force-display must have no backlash or lost motion; friction and inertia must be as low as possible.
In this paper we describe the force display we have recently implemented. The design satisfies some specifications required to perform micro-surgery tasks, like high resolution, low inertia and low-friction.

## DESIGN

Let's imagine a surgeon performing a precise incision using a very sharp scalpel. If the surgeon is not trying to use the scalpel as a lever, the force interaction between the tip of the scalpel and the tissue that is being cut is in a 3 dof cartesian space. This is because the contact surface can be approximated by an infinitesimal point. Hence, we designed a 3 dof force display. We decided also that the operator should interact with the manipulator using the tip of a real scalpel or other pointed tool. The main goal was to design a manipulator with very low inertia and friction. In this way the operator does not feel a burden while the scalpel is in free motion, and he/she can feel the high frequency force components generated by the interaction of the scalpel with different kind of tissues.

Geared manipulators do not fit this objective very well. They do not have very high bandwidth, so force information with high frequency components cannot be satisfactory reproduced. In addition backlash and friction phenomena are always present. On the contrary, direct drive manipulators are characterized by very high force generation bandwidth, low friction and no backlash.
The drawback is that direct drive manipulators usually have a higher mass/torque ratio compared to geared

manipulators. For this reason serial direct drive manipulators are characterized by a very high inertia. We decided, therefore, to provide the 2 dof motion in the horizontal cartesian space using a parallel structure. We found a number of 2 dof parallel manipulator in literature. One of the most used structure [Millman & Colgate 1991, Ramstein & Hayward 1994] is the one in Figure 1 where the 2 actuators sometimes are on the same axis, sometimes not:

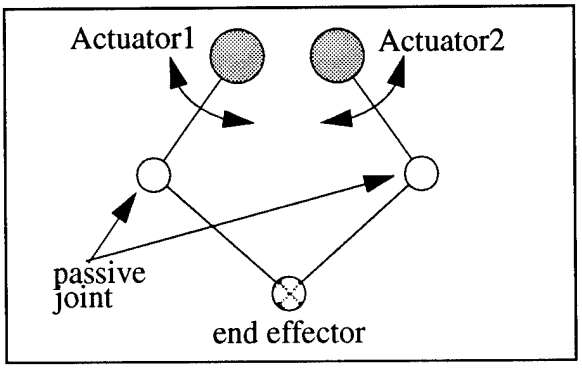



Figure 1

*Schematic representation of a typical 2dof parallel manipulators.*

We decided to add a third actuator to obtain a more symmetrical workspace and to provide more force to the end-effector. The resulting planar structure, see Figure 2, is a redundant actuation system, with 3 actuators and 2 dof.

The third degree of motion is given by a more powerful pair of rotational actuators which rotate the planar mechanism around the axis shown in Figure 2 and Figure 3. Because we are interested in rotation of about $\theta_z = \pm 10°$ across the horizontal plane, it is possible to approximate the rotary motion with a linear motion along the vertical z axes

## STATIC AND DYNAMIC

The evaluation of a parallel manipulator dynamic behavior can be an extremely complex mathematical problem. Because the robot is not an open chain we can not use directly the Newton-Euler approach, and to derive the Hamiltonian is quite complex too. Various approach can be found in the literature [Nakamura 1991, Hayward & al 1994], [Asada & Toumi 1987]. To simplify the problem we decided to consider the 2dof parallel structure as composed of three serial 2 link

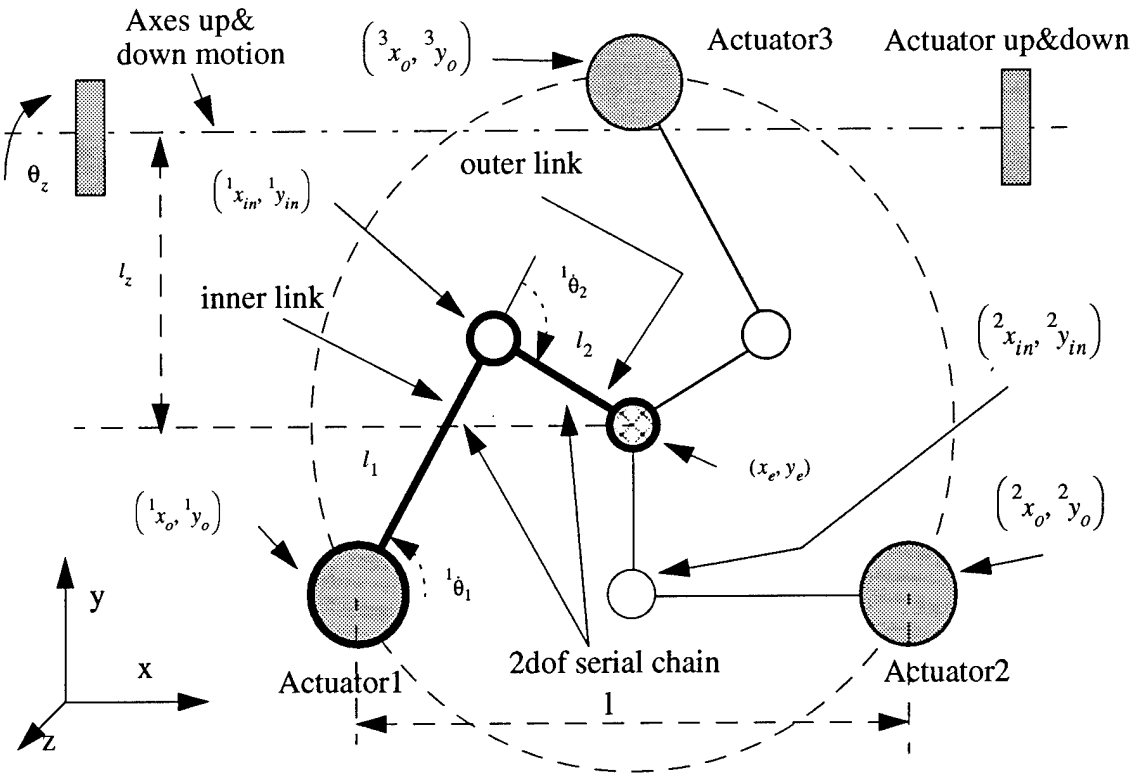



Figure 2

*Schematic representation of the pen-based force display. For the notation refers to the Symbolic Notation paragraph. The shaded circles represent actuated joints. The non-shaded circles represent non-actuated joints. The gridded circle represent the end-effector joint.*

manipulators, connected together at the end-effector. Each serial chain is composed of an inner and an outer link, as shown by the bold line in Figure 2. Between the two links there is no actuator; however it is possible to introduce a fictitious actuator with null mass and null output torque. From now on:

- A superscripts on the upper left corner of any symbol refers to the i-th serial chain. Otherwise the symbol represents the global parallel structure.
- $s_\alpha = \sin(\alpha)$ , $c_\alpha = \cos(\alpha)$
- ${}^i\theta_1, {}^i\theta_2, {}^i\theta_{12} = {}^i\theta_1 + {}^i\theta_2$ are the angles formed by the two links of the i-th serial chain, as shown in Figure 2.
- $\left({}^i x_o, {}^i y_o\right)$ is the position of the origin of the i-th serial chain expressed in the cartesian reference frame (x,y).
- $\left({}^i x_{in}, {}^i y_{in}\right)$ is the position of intermediate joint of the i-th serial chain expressed in the cartesian reference frame (x,y).
- $(x_e, y_e)$ is the position of the end effector expressed in the cartesian (x,y) frame.

Now we derive the static equations. We first consider the planar manipulator and then the vertical motion. Considering only ${}^i\theta_1$, that is the i-th actuator displacement, we get:

$$ {}^i\dot{\theta}_1 = \begin{bmatrix} \frac{c_{i_{\theta_{12}}}}{l_1 s_{i_{\theta_2}}} & \frac{s_{i_{\theta_{12}}}}{l_1 s_{i_{\theta_2}}} \end{bmatrix} \dot{x}_e , \qquad (1) $$

and for the overall planar manipulator

$$ \dot{\theta} = \begin{bmatrix} {}^1\dot{\theta}_1 \\ {}^2\dot{\theta}_1 \\ {}^3\dot{\theta}_1 \end{bmatrix} = J_e^{-1}(\theta)\,\dot{x}_e = \begin{bmatrix} \frac{c_{1_{\theta_{12}}}}{l_1 s_{1_{\theta_2}}} & \frac{s_{1_{\theta_{12}}}}{l_1 s_{1_{\theta_2}}} \\ \frac{c_{2_{\theta_{12}}}}{l_1 s_{2_{\theta_2}}} & \frac{s_{2_{\theta_{12}}}}{l_1 s_{2_{\theta_2}}} \\ \frac{c_{3_{\theta_{12}}}}{l_1 s_{3_{\theta_2}}} & \frac{s_{3_{\theta_{12}}}}{l_1 s_{3_{\theta_2}}} \end{bmatrix} \dot{x}_e \qquad (2) $$

But, we know that

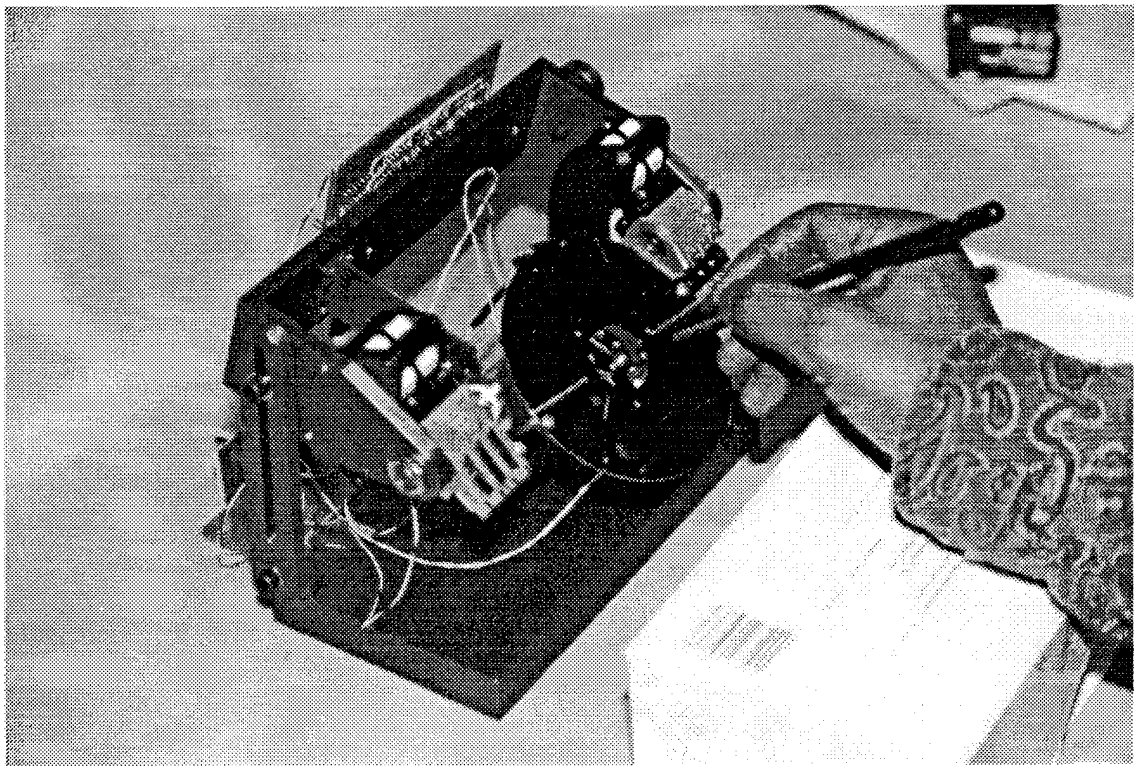


Figure 3

*The Pen Based Force Display*

$$F_e = J_e^{-T}(\bar{\theta}) \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} \frac{c_{1_{\theta_{12}}}}{l_1 s_{1_{\theta_2}}} & \frac{c_{2_{\theta_{12}}}}{l_1 s_{2_{\theta_2}}} & \frac{c_{3_{\theta_{12}}}}{l_1 s_{3_{\theta_2}}} \\ \frac{s_{1_{\theta_{12}}}}{l_1 s_{1_{\theta_2}}} & \frac{s_{2_{\theta_{12}}}}{l_1 s_{2_{\theta_2}}} & \frac{s_{3_{\theta_{12}}}}{l_1 s_{3_{\theta_2}}} \end{bmatrix} \tau \quad (3)$$

Given a desired $F_e$ there are an infinite set of torque vectors $\tau$ that solve the above system of equations. In the next section we will briefly introduce the criterion we choose to solve the redundancy, that will be more completely treated in a successive paper.

The equation for the third dof is approximately given by

$$\tau_z = l_z(mg + F_z) \quad (4)$$

where $\tau_z$ is the torque given by the pair of vertical actuators, $m$ is the mass of the planar 2dof mechanism, and $F_z$ is the external force applied along the vertical axis.

Now we determine the equation for the dynamics. If we break the parallel manipulator at the end-effector, and no force is applied from outside, the equation for the i-th serial chain is

$${}^i\tau = {}^iM\left({}^i\bar{\theta}\right)\ddot{{}^i\bar{\theta}} + {}^iV\left({}^i\bar{\theta}, \dot{{}^i\bar{\theta}}\right) + {}^iG\left({}^i\bar{\theta}\right) \quad (5)$$

and in the cartesian frame[1]:

$${}^iF_e = {}^iM_e\left({}^i\bar{\theta}\right)\ddot{\bar{x}}_e + {}^iV_e\left({}^i\bar{\theta}, \dot{{}^i\bar{\theta}}\right) + {}^iG_e\left({}^i\bar{\theta}\right) \quad (6)$$

Now we consider the interaction forces between the serial chains. We are assuming the gravity force to be zero, because the parallel manipulator works in or close to the horizontal plane, but the same procedure can be applied also in presence of gravity forces. The equations are:

$$\begin{cases} {}^1F_e = {}^1M_e\left({}^1\bar{\theta}\right)\ddot{\bar{x}}_e + {}^1V_e\left({}^1\bar{\theta}, \dot{{}^1\bar{\theta}}\right) + F_{21} + F_{31} \\ {}^2F_e = {}^2M_e\left({}^2\bar{\theta}\right)\ddot{\bar{x}}_e + {}^2V_e\left({}^2\bar{\theta}, \dot{{}^2\bar{\theta}}\right) + F_{12} + F_{32} \\ {}^3F_e = {}^3M_e\left({}^3\bar{\theta}\right)\ddot{\bar{x}}_e + {}^3V_e\left({}^3\bar{\theta}, \dot{{}^3\bar{\theta}}\right) + F_{13} + F_{23} \end{cases} \quad (7)$$

Where $F_{ij}$ is the force exerted by the i-th manipulator on the j-th. Because they are endogenous forces they must have zero sum:

$$F_{21} + F_{31} + F_{12} + F_{32} + F_{13} + F_{23} = 0 \quad (8)$$

Adding together the three dynamic equations and considering an external force $F_{ext}$ we get the dynamics of the 2dof parallel structure in the cartesian frame:

$$\sum_i {}^iF_e + F_{ext} = \sum_i M_e(\bar{\theta})\ddot{\bar{x}}_e + \sum_i V_e(\bar{\theta}, \dot{\bar{\theta}}) \quad (9)$$

The equation for the third dof is approximately given b

$$\tau_z = l_z mg + l_z F_z + I_z \dot{\omega}_z \quad (10)$$

where $\dot{\omega}_z$ is the angular acceleration of the $\theta_z$ joint.

## REDUNDANCY

Because we have 3 actuators for 2 dof there are an infinite number of possible torque vectors $(\tau_1, \tau_2, \tau_3)$ that provide the same force $F_e$. Two possible choices are those that minimize

$$\sqrt{\tau_1^2 + \tau_2^2 + \tau_3^2} \quad (11)$$

or minimize

$$max\left(|\tau_1|, |\tau_2|, |\tau_3|\right) \quad (12)$$

If we choose the Eq (11) we minimize the energy spent by the system, if we choose the Eq (12) we maximize the force that can be applied by the end-effector subject

---

1

$${}^iM_e\left({}^i\bar{\theta}\right) = \begin{bmatrix} 2m_1 + 5m_2 + 2m_1 c_{2^i\theta_{12}} + m_2\left(\left(-c_{2^i\theta_1} - 3c_{2^i\theta_1} + 3\frac{c_{2^i\theta_{12}}}{s^2_{i_{\theta_2}}}\right)\right) & \frac{2m_1 s_{2^i\theta_{12}} + m_2\left(-s_{2^i\theta_1} + 3s_{2^i\theta_{12}}\right)}{s^2_{i_{\theta_2}}} \\ \frac{2m_1 s_{2^i\theta_{12}} + m_2\left(-s_{2^i\theta_1} + 3s_{2^i\theta_{12}}\right)}{s^2_{i_{\theta_2}}} & 2m_1 + 5m_2 + 2m_1 c_{2^i\theta_{12}} + m_2\left(\left(-c_{2^i\theta_1} - 3c_{2^i\theta_1} + 3\frac{c_{2^i\theta_{12}}}{s_{2^i\theta_2}}\right)\right) \end{bmatrix}$$

$${}^iV_e\left({}^i\bar{\theta}, \dot{{}^i\bar{\theta}}\right) = \frac{m_1}{s^2_{i_{\theta_2}}}\left(l_1 c_{i_{\theta_2}} \dot{{}^i\theta}_1^2 + l_2\left(\dot{{}^i\theta}_1 + \dot{{}^i\theta}_2\right)^2\right)\begin{bmatrix} c_{i_{\theta_{12}}} \\ s_{i_{\theta_{12}}} \end{bmatrix}$$

to actuator limits, as it is easy to prove. Because energy is not a concern for our small pen display we decided for Eq (12)

## SYSTEM SPECIFICATIONS

Let's now introduce the requirements for our robot. An operator, when performing a high precision movement grasping a pencil or a scalpel with his/her finger, like a surgeon during a micro-surgery task, can move the tip of the tool in a very small workspace, about $1cm \cdot 1cm \cdot 1cm$ [Tendick 1994], [Anderson 1990]. He/She can apply through the tip of the scalpel a maximum force in the range of $0.5–1\ N$. The pen display must therefore meet the following requirements:

$$\begin{aligned} F^{continuous}_{maxworstcase} &\geq 0.5N \\ F^{peak}_{maxworstcase} &\geq 1N \\ workspace &\geq 1cm \cdot 1cm \cdot 1cm \end{aligned} \tag{13}$$

where for worst case we intend that the manipulator must be able to supply that specified amount of force from wherever inside the workspace, along whatever direction[1].

For peak force we intend the force that can to be applied for a short amount of time without burning the actuator. In a telerobotics application a high force has to be applied on the master side only for a short amount of time, usually when a collision has to be reproduced to the operator. Approximating the $\theta_z$ motion as straight vertical we can consider the motion between the 2dof parallel manipulator and the vertical approximated 1dof link to be uncoupled. So we can decompose our system specification as:

$$\begin{aligned} F^{continuous}_{planar} &\geq 0.5N \\ F^{peak}_{planar} &\geq 1N \\ ws_{planar} &\geq 1cm \cdot 1cm \end{aligned} \tag{14}$$

$$\begin{aligned} F^{continuous}_{vertical} &\geq 0.5N \\ F^{peak}_{vertical} &\geq 1N \\ ws_{vertical} &\geq 1cm \end{aligned} \tag{15}$$

where for simplicity we omitted *maxworstcase*.

## ACTUATION

For mini-robotics applications, the flat coil actuators found inside the hard disk drivers are very interesting and cost-effective actuators. They come in different sizes, depending on the hard disk format, and they have very low inertia and friction. [Marbot 1991], [Marbot & Hannaford 1991], [Buttolo & al 1994].

Because of the system specifications and the flat coil mechanical properties we measured for the various devices, we decided to use the 1.8" actuators for the parallel structure, and the 5.25" size for the up and down motion. The steady state current flowing through the coil that generate a $\Delta T = 120^{o}C$ is 0.65A for the 1.8" and 0.52A for the 5.25". We observed [Buttolo 1994] some damage on the coil starting at an absolute temperature of about $150^{o}C$, at $\Delta T = 120^{o}C$. The torque generated by the actuators is 0.01Nm (1.8") and 0.06Nm (5.25"). For short amount of time we observed we could supply the actuator with up to 2A (1.8" and 5.25"), corresponding to a torque of about 0.03Nm (1.8") and 0.24Nm (5.25"). The range of motion is about $40^{o}$ (1.8" and 5.25").

## DESIGN OPTIMIZATION

In order to choose the manipulator dimensions $l, l_1, l_2, l_z$ we need to define a performance index to be maximized. The robotics literature contains various indices of performance [Yoshikawa 1990, Nakamura 1991], such as the manipulability index introduced by Yoshikawa:

$$w = \sqrt{detJ(q)J^{T}(q)}$$

Because we are mostly interested in high bandwidth force reflection, so that we can achieve a satisfactory telepresence feeling, we define the performance index as

$$PI = \ddot{x}_{maxworstcase} \tag{16}$$

that is the maximum acceleration that can be achieved in the worst case, starting from zero velocity, from any point inside the workspace, along any direction.
We can formalize our design optimization problem as the combination of the boundary equation given by Eq (14) and Eq (15) and

$$\max_{l, l_1, l_2, l_z} (PI) \tag{17}$$

---

1 $F^{continuous}_{planar} = \min_{(x_e, y_e)} \left( \min_{\alpha} \left( \max_{\tau_1, \tau_2, \tau_3} \left( \|F_e\| \,|\, \angle F_e = \alpha \right) \right) \right) \qquad F = J_e^{-T}(\theta)\tau$

$$|\tau_1|, |\tau_2|, |\tau_3| \leq \tau_{max}$$

$l_z$ has to be chosen so that a rotation of $\theta_z = \pm 20^o$ correspond to a linear motion of at least $\Delta z_e = \pm 0.5cm$, so that $ws_{vertical} = 2 \cdot \Delta z_e \geq 1cm$ is satisfied. So it has to be $l_z \geq \frac{\Delta z_e}{\sin(20)} = 1.46cm$. We choose $l_z = 2cm$, so that $ws_{vertical} = 1.37cm$.

The mass of the 2dof structure is 150gr and so, from Eq (4), we derive that a torque of about 0.03Nm is necessary to compensate the gravity force. The pair of 5.25 voice coil can provide a continuous torque of 0.12Nm and a peak torque of about 0.3Nm. Note that the mechanical structure of the robot structure provides no counter-balancing against gravity. Therefore, because 0.03 is the torque necessary to achieve 1g acceleration, our robot can achieve an upward acceleration of 9g. The peak force that can be applied by the end-effector in the up-ward direction is (0.3-0.03)Nm/0.02m = 13.50N. The continuous force is (0.12-0.03)Nm/0.02N = 4.5N.

The force applicable upward and downward is far greater than the minimum requested

$$F_{vertical}^{continuous} \geq 0.5N$$
$$F_{vertical}^{peak} \geq 1N$$

Let's solve now the optimization problem for the planar mechanism. We want to determine $l, l_1, l_2$ so that Eq (14) are satisfied and the performance index $PI$ is maximum. Because of the complexity of the problem we did not look for the global maximum value, but we considered a "good" solution acceptable.

1) We tried to see if, given $(l_1, l_2)$ it is possible to find a relation $l_{f12} = f(l_1, l_2)$ so that

$$PI(l_1, l_2, l_{f12}) \geq PI(l_1, l_2, l) \quad \forall l \neq l_{f12} \qquad (18)$$

We numerically computed $F_{planar}^{continuous}$, $\ddot{x}_{maxworstcase}$, and $ws_{planar}$ for different values of the parameters $l_1, l_2, l$. We found that if

$$l = l_{f12} = 1.03l_1 + 1.59 \cdot l_2 \qquad (19)$$

an equation similar to Eq (18) is verified, but for a small error:

$$PI2(l_1, l_2, l_{f12}) \geq PI2(l_1, l_2, l) \quad \forall l \neq l_{f12} \qquad (20)$$

with

$$PI2 = \max_{l_1, l_2, l}\left( F_{planar}^{continuous} \cdot \ddot{x}_{maxworstcase} \cdot ws_{planar} \right)$$

We will prove at the end that if we choose $l$ from Eq (19), we will find a good solution for the optimization problem.

2) We evaluated $ws_{planar}$ for different values of $(l_1, l_2)$ and $l = 1.03l_1 + 1.59 \cdot l_2$. We found that the planar workspace area depends mostly from $l_1$, while it is almost independent from $l_2$.
So, we choose $l_1 = 2cm$, that gives

$$ws_{planar} \approx 1.5cm^2.$$

3) We evaluated $\ddot{x}_{maxworstcase}$ and $F_{planar}^{continuous}$. for different values of $l_2$, $l_1 = 2cm$ and $l = 1.03l_1 + 1.59 \cdot l_2 = 2.06 + 1.59 \cdot l_2$. The result are shown in Figure 4 and Figure 5.

$\ddot{x}_{maxworstcase}$, that is our performance index $PI$, is maximum for $l_2 = 1.25cm$. From Figure 4 we can see that $F_{planar}^{continuous} = 0.55N$ and so $F_{planar}^{continuous} \geq 0.5N$ is satisfied.
Now, we choose $l$ so that Eq (20) is satisfied instead of Eq (18). This solution

$$\begin{aligned} l_1 &= 2cm \\ l_2 &= 1.25cm \\ l &= 4.05cm \end{aligned} \qquad (21)$$

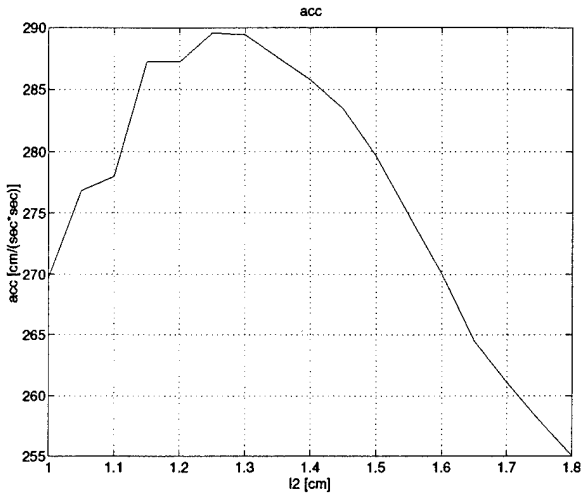



Figure 4

*Plot of* $\max_{l_1, l_2, l}(\ddot{x}_{eminminmax})$ *function of* $l_2$, *with* $l_1 = 2cm$ *and* $(1.03 + 1.59 \cdot l_2/l_1) \cdot l_1$

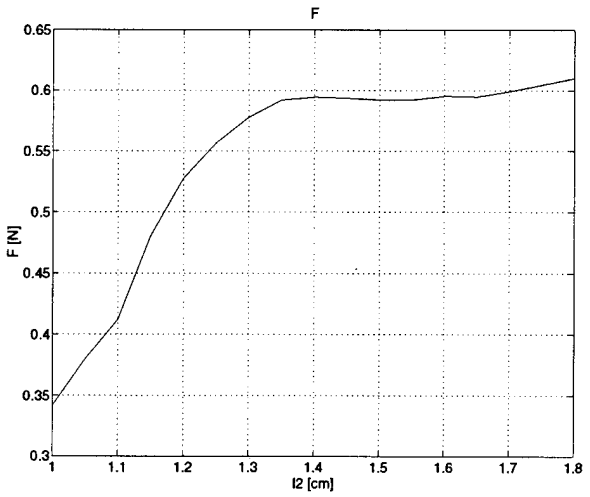



Figure 5

*Plot of* $F^{continuous}_{eminminmax}$ *function of* $l_2$, *with* $l_1 = 2cm$ *and* $(1.03 + 1.59 \cdot l_2/l_1) \cdot l_1$

may not give us the best solution of

$$F^{continuous}_{planar} \geq 0.5N$$
$$ws_{planar} \geq 1cm \cdot 1cm$$
$$\max_{l, l_1, l_2}(PI)$$

but it gives us a good solution of the problem

$$F^{continuous}_{planar} \geq 0.55N$$
$$ws_{planar} \geq 1.5cm^2$$
$$\max_{l, l_1, l_2}(PI)$$

To show it, let's assume that there is another $l^{opt}$, so that

$$F^{continuous}_{planar} \geq 0.55N$$
$$ws_{planar} \geq 1.5cm^2 \qquad (22)$$
$$\max_{l^{opt}, l_1, l_2}(PI) \geq \max_{l, l_1, l_2}(PI)$$

Now, for $l, l_1, l_2$ given by Eq (21) it is $ws_{planar} \approx 1.5cm^2$ and $F^{continuous}_{planar} = 0.55N$, so that

$$PI2(l, l_1, l_2) = 1.5 \cdot 0.55 \cdot PI(l, l_1, l_2)$$

Instead, for $l^{opt}, l_1, l_2$ it must be

$$PI2\left(l^{opt}, l_1, l_2\right) \geq 1.5 \cdot 0.55 \cdot PI\left(l^{opt}, l_1, l_2\right)$$

From Eq (20) it must be

$$PI2\left(l^{opt}, l_1, l_2\right) \leq PI2\left(l^{opt}, l_1, l_2\right)$$

With some substitutions we can see that Eq (22) can not be true.

## CONCLUSIONS

The pen-based force display can be an effective tool for precision manipulation in virtual environment or for scaled telemanipulation. The human operator interacts with the force display in a very familiar way, using a pencil or a scalpel. This configuration can be very effective for micro-surgery.

A drawback of the design of a parallel redundant structure is the high computational requirement needed to solve the dynamic equations and to choose a torque configuration among the infinite possible. On the other hand, this parallel manipulator has a very low inertia, no backlash, almost zero friction, and the actuator redundancy can provide a homogenous force capability. We tried to measure the static friction and it was less than the resolution of our measuring devices, 1grf!

As an additional advantage of our design, multiple closed chains provide an easy way to self-calibrate the mechanical devices [Nahvi 1994]. In our case the presence of a redundant close loop allowed us to self-calibrate all the parameters of the parallel planar device, such as the position of each actuator and the length of the links. This will be the argument of another paper.
We also built a simulation virtual reality test-bed, where an operator can see virtual objects on a video display and touch them using the force display. The feeling coming from touching the virtual object confirmed us that our device is capable of high-frequency force reflection. A future research will be to determinate efficient control algorithm and real-time performance calibration technique [Buttolo & Bratthen 1994].

## REFERENCES

Anderson and Romfh, "Technique in the Use of Surgical Tools,", Appleton-Century-Crofts, 1990 pag1-20

H.Asada, Y.Toumi, "Direct-drive robots," MIT Press, Cambridge, MA

P. Buttolo, P.Bratthen, B. Hannaford, "Sliding Control

of Force Reflecting Teleoperation: Preliminary Studies," PRESENCE Vol3, Num 2, Spring 1994, pag. 158-172

P. Buttolo, D.Y. Hwang, B. Hannaford, "Experimental Characterization of Hard Disk Actuators for Mini Robotics," Proc. SPIE Telemanipulator and Telepresence Technologies Symposium, Boston, October 31, 1994.

V.Hayward, L.Choksi, J.Lanvin, C.Ramstein, "Design and multi objective Optimization of a Linkage for a Haptic Interface," submitted to the 4th International Workshop on Advances in Robot Kinematics 1994

V. Hayward, R. Kurtz, "Modeling of a Parallel Wrist Mechanism with Actuator Redundancy,"McGill Centre for Intelligent Machines CIM-89-4, Montreal, January 1984.

K.Hirota, M.Hirose, "Surface Display: A Force Feedback System Simulating the Surface of an Object", Proceedings 3rd IEEE Int. Work. on Robot and Human Comm. RO-MAN 94, Nagoya JAPAN, July, pag 251-254

R.L. Hollis, S. Salcudean, D.W. Abraham, "Toward a Tele-Nanorobotic Manipulation System with Atomic Scale Force Feedback and Motion Resolution," Proceedings IEEE Micro Electro Mechanical Systems Conference, pp. 115-119, Napa Valley, CA, Feb. 1990.

H.Iwata, "Pen-Based Haptic Virtual Environment," Proc. VRAIS IEEE 93, Seattle WA, pag.287-292

R.L. Kurtz, "Kinematic and Optimization of a Parallel Robotic Wrist Mechanism with Redundancy: Thesis,"Computer Vision and Robotics Laboratory, McGill Research Center for Intelligent Machines, Jan. 1990.

P.H. Marbot, B. Hannaford, "Mini Direct Drive Arm for Biomedical Applications," Proceedings of ICAR 91, pp. 859-864, Pisa Italy, June 1991.

P.H. Marbot, "Mini Direct Drive Robot for Biomedical Applications," MSEE Thesis, University of Washington, Department of Electrical Engineering, August, 1991

P.A. Millman, J.E. Colgate, "Design of a Four Degree of Freedom Force-Reflecting Manipulandum with a Specified Force/Torque Workspace," Proc. IEEE Robotics and Automation, vol. 2, pp. 1488-1493, Sacramento, CA, 1991.

A.Nahvi, J.M.Hollerback, V.Hayward, "Calibration of a Parallel Robot Using Multiple Kinematic Closed Loops,", Proc. IEEE Robotics and Automation, vol. 1, pp. 407-413, San Diego, CA, 1994

Y. Nakamura , "Advanced Robotics - Redundancy and Optimization," Addison Wesley 1991, pag. 205-225

C.Ramstein, V.Hayward, "The Pantograph: A Large Workspace Haptic Device for a Multimodal Human-Computer Interaction," Computer-Human Interaction CHI 94

S.E.Salcudean, J.Yan, "Towards a Force-Reflecting Motion-Scaling System for Microsurgery," Proc. IEEE Robotics and Automation, vol. 3, pp. 2296-2301, San Diego, CA, 1994

T.Sato, J.Ichikawa, M.Mitsuishi, Y.Hatamura, "A New Micro-Teleoperation System Employing a Hand-Held Force-Feedback Pencil, ", Proc. IEEE Robotics and Automation, vol. 2, pp. 1728-1733, San Diego, CA, 1994

K. B. Shimoga, "A Survey of Perceptual Feedback Issues in Dexterous Telemanipulation: Part I. Finger Force Feedback," Proc. VRAIS IEEE 93, Seattle WA, pag.263-270

F.Tendick, R.W.Jennings, G.Tharp, L.Stark, "Sensing and Manipulation Problems in Endoscopic Surgery: Experiment, Analysis, and Observation," PRESENCE Vol2, Num 1, Winter 1993, pag. 66-81

T.Yoshikawa, "Foundations of Robotics," MIT Press 1990, pag 89-90 - pag 190-192

# Panel:

## Whither Force Feedback?

**Chair:**

William McNeely
Boeing Computer Services

# Whither Force Feedback?

Chair
William McNeely, Boeing Computer Services

Panelists
Grigore C. Burdea, Rutgers University
Blake Hannaford, University of Washington
Michitaka Hirose, University of Tokyo
Steve Jacobsen, University of Utah
Kenneth Salisbury, Massachusetts Institute of Technology
Susumu Tachi, University of Tokyo

Force feedback (or haptic simulation) is steadily gaining recognition as an important aspect of virtual reality. It underlies mechanical leverage, body support, sense of balance and sense of contacting real objects. It conveys information that is deemed indispensable in many training, design analysis, and hazardous-environment task simulations. Even in applications where it would nominally be a luxury, force feedback promises to enable superior VR user interfaces that fully exploit hand-eye coordination and full-body interaction.

Alas, despite decades of highly successful usage in VR-like applications such as teleoperation and flight simulation, force feedback is notoriously lagging as a technology for VR. This happens because current approaches draw heavily upon mechanical automation, which lacks the commercial infrastructure enjoyed by other hardware and software components of VR. For example, there is no robot equivalent of the modern, standardized, affordable PC, probably because robots have too long been captive to specialized industrial applications.

Additionally, there is a cultural bias on the part of computer scientists against integrating large, complex mechanical systems into VR. There is an even deeper reluctance to use technology with highly visible safety issues, such as human-robot interactions.

Given this striking mix of opportunities and challenges, what approach to force feedback for VR, if any, is likely to ultimately prevail? This is the central question to be addressed by the panel. The panelists are leading researchers in the area of force feedback for teleoperation and VR. They will describe their current work and address issues that seem critical to answering the central question, such as:

- What are the pros and cons of exoskeletons, manipulators, and robotic graphics? Specifically, how do they compare with regard to real-world concerns such as:
  - user safety
  - force realism
  - infrastructure requirements/assumptions
  - system complexity and portability
  - projected price-performance
- The 2-D desktop paradigm, relying only on hand-eye coordination, proved eminently successful. In light of daunting problems facing full-body force feedback, do we really need it, or is it reasonable to indefinitely restrict force feedback to the hands?
- Is current research on the right track? What about alternatives like:
  - cables?
  - electrorheological fluids (that solidify in electrical fields)?
  - micromechanical body suits?
  - electrodes?
- Which VR applications seem most likely to drive the future development of force feedback technology?
  - training?
  - teleoperation?
  - design?
  - entertainment?
  - communication?

## Grigore Burdea

Widespread use of force feedback in VR simulations requires desk-top inexpensive, easy to maintain and safe masters. Natural interaction requires portability, while dexterous task simulation needs feedback to individual fingers. Portability need not produce fatigue, so masters have to be light. The need for safety, simplicity and lightness is met by micro-actuator pneumatic technology which has an excellent power/weight ratio. One device recently developed in our laboratory is the "Rutgers Master II". By integrating non-contact position sensing within the force feedback system this 80-gram master allows dexterous simulation of hand gestures without a sensing glove. The system is desk-top and self-contained and can work with either a PC or with a Unix workstation.

## Blake Hannaford

Devices for force feedback have very distinctive subjective properties of quality. However, there is no commonly accepted measure of quality through which their



0-8186-7084-3/95 $04.00 

performance can be quantitatively compared. This is a significant barrier to rapid progress in force display technology. This presentation will propose a practical but imperfect measure of performance and show examples from recent devices. A pen based haptic device will be demonstrated.

## Michitaka Hirose

Surface display is a force feedback device which can give us a sensation as if we are actually touching the surface of virtual object. Key technology of the surface display is how to give realistic constraint/force feedback to the user's finger. In my presentation, I will introduce several surface displays developed in our laboratory. And some of shape-forming applications will also be demonstrated.

## Steve Jacobsen

Sarcos Research Corporation (SRC) and the Center for Engineering Design (CED), at the University of Utah, have developed several high performance telerobotic systems for industrial and research use.

These efforts have shown that certain performance characteristics must be simultaneously present in a system if dexterous tele-manipulation is to be achieved by a master commanding a slave manipulator operating in either real or virtual environments. Important characteristics include: (1) high bandwidth force reflection with high stiffness between master and slave, (2) reasonably anthropomorphic kinematics with substantial wrist mobility and sufficient end effector configurability, (3) strength and speed sufficient for natural use by the operator, (4) high resolution force and position display by the master to the operator, and (5) reliable, intuitive, low fatigue operation.

To achieve the above characteristics requires a number of system features. They include: (1) high performance actuators appropriately controlled to operate with minimal time delays between master and slave, (2) actuator systems with virtually no stiction and backlash, (3) real-time system parameter identification for accomplishment of autonomic functions, and for zero balanced gravity compensation, (4) operator control of force gain between master and slave, and (5) sensor systems capable of indicating position, velocity, force and touch.

These fundamental issues, especially as they relate to the design and implementation of masters with force display, will be discussed in the context of ongoing projects within SRC and the CED.

## Kenneth Salisbury

While the use of force reflecting displays to enhance interaction with remote and virtual objects has been of interest for many years, the utilization of such systems is not yet widespread. The quality of force interaction is as important to the successful performance of complex remote manipulation tasks as it is to the performance of more simple tasks such as the operation of switches and controls. Unfortunately performance (and acceptance by users) of a force reflecting display, or haptic interface, depends on multiple, conflicting system attributes. Good bandwidth and resolution clearly enhance the quality of interaction, yet a many-degree-of-freedom system can pose significant challenges to obtaining good dynamic performance. The more degrees of freedom a device has, the more mass and inherent compliance there is to degrade performance. Furthermore, increased complexity and mass tend to yield expensive systems for which intrinsic safety is difficult to maintain.

It is our sense that an important avenue of research in developing better haptic interface capabilities, and use paradigms, is one in which low-cost, intrinsically safe, and easy to use devices are utilized. Because much of the richness in haptic interaction stems from bandwidth and resolution of the force display, these qualities must also be emphasized. In our efforts at MIT we have focused on a relatively simple compromise to the above problems in which we address interactions between a user controlled point and objects in the environment. "Displaying" force sensations which arise from point interactions requires only three active degrees of freedom in the interface and greatly simplifies the mechanics, programming and servo control. The Phantom haptic interface developed in our laboratory permits precisely controlled force vectors to be applied to a user's fingertip or stylus. This device has been used to simulate interaction with a wide range of virtual objects. It has permitted us to explore a variety of haptic rendering and representation issues such as object shape, texture, surface compliance, mobility of objects, and grasping (with two Phantoms). We feel that this device and the styles of virtual object interaction which are evolving from it, form an important class of interaction with computer generated worlds. Because it is a simple, desktop device we feel that it will provide an important vehicle for a broad community of users to join in the development of haptic interaction methods.

## Susumu Tachi

A machine has been developed which generates a virtual haptic space by constructing only the partial space near a contact point based on the real time measurement of human movement and real time control of the display to cover the whole area, which enables the complete representation of the entire environment. The machine consists of a shape approximation device and an active environment display. The former partially approximates the shape of a haptic environment to be presented near a contact point, and the latter follows a human finger motion to complete the approximation to form a virtual haptic space. The latter also provides mechanical impedance of the environment, i.e., inertia, viscosity and stiffness. In my presentation, experimental results using the seven-degree-of-freedom test-hardware developed will be shown by video.

# Panel:

## Networked Virtual Environments

**Chair:**

Michael J. Zyda
Naval Postgraduate School

# Networked Virtual Environments

*Michael J. Zyda, Panel Chair*
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5118 USA
+1-408-656-2305
zyda@trouble.cs.nps.navy.mil

**ABSTRACT**

This panel examines the current and future issues regarding the development of networked virtual environments and teleoperation systems.

**KEYWORDS:** Virtual Reality, Distributed Interactive Simulation, Distributed Interactive Entertainment, Large-scale Virtual Environments.

## INTRODUCTION

The development of multi-user networked virtual worlds has become a major area of interest to the virtual reality community. The realization of high bandwidth wide area communications, the success of World Wide Web applications such as the National Center for Supercomputing Application's Mosaic browser, and government funding of Distributed Interactive Simulation (DIS) has fueled the desire to expand networked virtual worlds beyond local area networks. However, the Internet has proved a challenging environment for real-time applications such as interactive virtual worlds and multimedia. The panel looks at what research groups are doing to meet the challenge and examine the state-of -the-art in networked virtual environment and teleoperation systems.

## MICHAEL ZYDA, SENIOR EDITOR FOR VIRTUAL ENVIRONMENTS FOR PRESENCE

### Panel Chair

Michael Zyda is a Professor in the Department of Computer Science at the Naval Postgraduate School, Monterey, California. Professor Zyda is also the Academic Associate and Associate Chair for Academic Affairs in that department. He has been at NPS since February of 1984. Professor Zyda's main focus in research is in the area of computer graphics, specifically the development of large-scale, networked 3D virtual environments and visual simulation systems. Professor Zyda is a member of the National Academy of Sciences' Committee on Virtual Reality Research and Development. Professor Zyda is also the Senior Editor for Virtual Environments for the MIT Press quarterly PRESENCE, the journal of teleoperation and virtual environments. For that journal, Professor Zyda has co-edited special issues on "Pacific Rim Virtual Reality and Telepresence", on "The Application of Virtual Environments to Architecture, Building and Large Structure Design", and on "Networked Virtual Environments and Teleoperation". Professor Zyda has been active with the Symposium on Interactive 3D Graphics and was the chair of the 1990 conference, held at Snowbird, Utah and is the chair of the 1995 Symposium, to be held in Monterey, California.

*The networking of virtual environments is how we go from one player on a workstation to many cooperating, interactive players at both local and distant sites. Current technology limits us to systems with approximately 300 players using Ethernet and T1 links. We present what is possible today with such technology, what we will be able to achieve near-term and what we need to work on to get us to the large-scale, networked virtual environment of thousands of players. There are hard problems involved in networking virtual environment systems. The key message perhaps is that such design is not done "last" but rather integrated into the software from the start. Another key point is that network protocols for virtual environments must be rapidly reconfigurable while the VE is running.*

## RICH GOSSWEILER, UNIVERSITY OF VIRGINIA

### Fundamentals in Developing Distributed Multi-User Virtual Environments

Rich Gossweiler is the senior Ph.D. student in the User Interface Group at the University of Virginia. He played a major role in designing and developing the underlying distributed virtual environment platform currently employed by the Alice graphics system. For his Ph.D., he is researching application-independent time-critical rendering techniques. Recent work includes an introductory-level tutorial describing how to implement a distributed multi-

participant virtual environment.

*Having observed that it is difficult to begin to explore networked virtual environments, this tutorial is intended for undergraduate students who are competent programmers and who now wish to implement a distributed, multi-participant application. It describes the fundamental concepts of distributed computing for multi-player simulations and includes a C source code template available via the Internet. The template was designed so that students can quickly create their own distributed applications. The template source code uses broadcast communication and a technique called dead-reckoning to improve performance.*

### JOHN MORRISON, MAK TECHNOLOGIES

#### Experiences with DIS-based Virtual Environments

MaK was founded in October 1990 by John Morrison and Warren Katz, two well-known developers of the SIMNET system. MaK has one of the highest concentrations of experienced SIMNET and DIS developers in the industry, measured against companies of any size. MaK personnel have participated in every large DIS simulation contract since the invention of the technology, including SIMNET, Advanced Distributed Simulations Testbed, War Breaker, and the emerging Synthetic Theater of War (STOW).

*Three challenges to building large Networked Virtual Environments are ever-increasing complexity, scalability, and portability. We require a software infrastructure to overcome these challenges. We examine one such example software infrastructure which is currently the basis of dozens of virtual environment efforts. Its use of object-oriented inheritance and its use of an interpreted configuration programming language meets these requirements while achieving the efficiency to support large numbers of entities on current-generation hardware.*

### SANDEEP SINGHAL, DISTRIBUTED SYSTEMS GROUP, STANFORD

#### Strategies for Minimizing Network Traffic for Large-scale Virtual Environments

Sandeep K. Singhal's research is in communication protocols and algorithms for dead reckoning -- the problem of accurately displaying the real-time position, orientation, and structure of objects actually being modeled on remote machines. He is also researching how to enable dynamic multicast channel aggregation by applications. The PARADISE Project at Stanford is a testbed for his work.

*Distributed virtual reality systems require accurate, efficient remote rendering of animated entities in the virtual environment. Position, velocity, and acceleration information about each player is maintained at the player's local machine, but remote hosts must display this information in real-time to support interaction between users across the network. Prior applications have transmitted position information at the local frame rate, or they have relied on dead reckoning protocols using higher derivative information to extrapolate entity position between less frequent updates. These approaches require considerable network bandwidth and at times exhibit poor behavior. We describe a position history-based protocol whose update packets contain only position information. Our evaluation suggests that the position history-based protocol provides a network-scalable solution for generating smooth, accurate rendering of remote entities.*

### MICHAEL MACEDONIA, NAVAL POSTGRADUATE SCHOOL

#### Mega-Scale Virtual Environments

Michael R. Macedonia is a US Army major and a Ph.D. student in computer science at the Naval Postgraduate School. His research is directed toward the development of software architectures supporting large-scale distributed virtual environments.

*We present our ideas in the context of NPSNET-IV, the first 3D virtual environment that incorporates both the DIS application protocol and the IP Multicast network protocol for multi-player simulation over the Internet. The fundamental idea behind the NPSNET approach is to logically partition virtual environments by associating spatial, temporal, and functionally related entity classes with network multicast groups. This is accomplished by exploiting the actual characteristics of the real-world large scale environments that are simulated, and by focusing or restricting an entity's processing and network resources to its area of interest via a local Area of Interest Manager (AOIM).*

### RESOURCES

The special double issue of the journal PRESENCE on Networked Virtual Environments includes articles from all the members of the panel. See *ftp://taurus.cs.nps.navy.mil/pub/PRESENCE_MOSAIC/presence_mosaic.html* for more details. Additional information from each of the members of the panel is available via WWW or email:

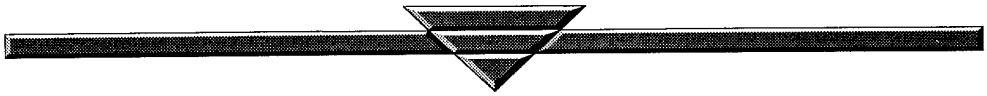
*John Morrison*, jm@mak.com.

*Rich Gossweiler*, ftp://uvacs.cs.virginia.edu/pub/distgame.

*Sandeep Singhal*, http://www-dsg.stanford.edu/SandeepSinghal.html.

*Mike Zyda, Mike Macedonia*, ftp://taurus.cs.nps.navy.mil/pub/NPSNET_MOSAIC/npsnet_mosaic.html.

# Author Index



# Notes

# IEEE Computer Society Press



## Offices of the IEEE Computer Society


*Headquarters Office*
1730 Massachusetts Avenue, N.W.
Washington, DC 20036-1903
Phone: (202) 371-0101 — Fax: (202) 728-9614

*Publications Office*
P.O. Box 3014
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1264
Membership and General Information: (714) 821-8380
Publication Orders: (800) 272-6657 — Fax: (714) 821-4010

*European Office*
13, avenue de l'Aquilon
B-1200 Brussels, BELGIUM
Phone: 32-2-770-21-98 — Fax: 32-2-770-85-05

*Asian Office*
Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, JAPAN
Phone: 81-3-408-3118 — Fax: 81-3-408-3553


# IEEE Computer Society

## IEEE Computer Society Press Publications

**Monographs:** A monograph is an authored book consisting of 100-percent original material.
**Tutorials:** A tutorial is a collection of original materials prepared by the editors and reprints of the best articles published in a subject area. Tutorials must contain at least five percent of original material (although we recommend 15 to 20 percent of original material).
**Reprint collections:** A reprint collection contains reprints (divided into sections) with a preface, table of contents, and section introductions discussing the reprints and why they were selected. Collections contain less than five percent of original material.
**Technology series:** Each technology series is a brief reprint collection — approximately 126-136 pages and containing 12 to 13 papers, each paper focusing on a subset of a specific discipline, such as networks, architecture, software, or robotics.
**Submission of proposals:** For guidelines on preparing CS Press books, write the Managing Editor, IEEE Computer Society Press, P.O. Box 3014, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1264, or telephone (714) 821-8380.

## Purpose

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

## Membership

All members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others seriously interested in the computer field.

## Publications and Activities

***Computer* magazine:** An authoritative, easy-to-read magazine containing tutorials and in-depth articles on topics across the computer field, plus news, conference reports, book reviews, calendars, calls for papers, interviews, and new products.
**Periodicals:** The society publishes six magazines and five research transactions. For more details, refer to our membership application or request information as noted above.
**Conference proceedings, tutorial texts, and standards documents:** The IEEE Computer Society Press publishes more than 100 titles every year.
**Standards working groups:** Over 100 of these groups produce IEEE standards used throughout the industrial world.
**Technical committees:** Over 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.
**Conferences/Education:** The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.
**Chapters:** Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.