Proceedings

of

The First PHANToM User's Group Workshop

September 27-30, 1996
Endicott House, Dedham MA
Massachusetts Institute of Technology, Cambridge, MA

Hosted by

Dr. J. Kenneth Salisbury, AI Lab & Dept of ME, MIT
Dr. Mandayam A. Srinivasan, RLE & Dept of ME, MIT

Sponsored by

Interval Research Corp, Palo Alto, CA
Mitsubishi Electric Research Labs, Cambridge, MA
SensAble Technologies, Inc., Cambridge, MA

The First PHANToM User's Group Workshop
Sept 27-30, 1996
MIT Endicott House, Dedham MA

Final Program

Friday Sept 27

1:00-4:00	Optional Demos (MIT AI Lab & RLE - rides to Endicott House available after demos from AI Lab, by advance arrangement)
1:00-5:00	Arrival at Endicott House
6:00-7:00	Dinner
7:30-8:30	Welcome
8:30-9:30	Refreshments

Saturday Sept 28

8:00-9:00	Breakfast
9:00-10:30	Session 1: Research Effort Overviews
	"Expressive Haptics," Bill Verplank, Interval Research Corporation
	"An Overview of Haptics Research at MIT's AI Lab," Kenneth Salisbury, MIT Dept. of ME and AI Lab.
	"Haptics Research at MIT Touch Lab," Mandayam Srinivasan, MIT Dept. of ME and RLE
10:30-11:00	Break
11:00-12:30	Session 2: Displays and Effects I
	"Protecting the Phantom from the Programmer," Rob Kooper, John Barrus, David Ratajczak, and David Parsons, Mitsubishi Electric Information Technology Center America
	"Taking the mush out of haptics with infinitely stiff walls," Thomas Massie, SensAble Technologies, Inc.
	" The Sense of Torque with a Single Phantom," Dennis Hancock
12:30-1:30	Lunch
1:30-3:00	Free time
3:00-4:30	Session 3: Displays and Effects II
	"Home Haptics," Margaret Minsky
	"Tactile Display of Shape and Vibration," Robert D. Howe, Harvard University
	"Bump mapping for a force display," Yukio Fukui, National Institute of Bioscience and Human Technology
4:30-5:00	Break
5:00-6:30	Session 4: Applications in Teleoperation and Training
	"Using PHANToMs for Tele-surgery: the HMSL system, experiments and experience," Mark P. Ottensmeyer, Jianjuen Hu, James M. Thompson, Jie Ren, and Thomas B. Sheridan, Human-Machine Systems Laboratory, MIT
	"A Haptic Process Architecture using the PHANToM as an I/O Device in a Virtual Electronics Trainer," Scott Davidson, Management Systems and Training Technologies Co.
	" Integrated Haptics Applications: Surgical Anastomosis and Aircraft Maintenance Trainers," Robert Playter, William Blank, Nancy Cornelius, Webb Roberts, Boston Dynamics Inc.
7:00-8:30	Dinner
8:30-9:30	Refreshments

Sunday Sept 29

8:00-9:00	Breakfast
9:00-10:30	Session 5: Tools for Simulated Worlds I
	"Virtual Surface Modelling with "Loosely Coupled" Force Feedback Device," Juli Yamashita, National Institute of Bioscience and Human Technology
	"A Virtual Universe Utilizing Haptic Display," Thomas G. Anderson , Sandia National Labs/U. New Mexico
	"A Graphical Development Environment for Haptics," George C. Williams, The University of Virginia
10:30-10:45	Break
10:45-12:45	Session 6: Tools for Simulated Worlds II
	"Force Feedback in Interactive Dynamic Simulation," Sundar Vedula, Carnegie Mellon University
	"SPI Haptics Library," Wendy Plesniak, Media Laboratory, MIT
	"Graphical and Haptic Manipulation of 3D Objects," Krasimir Kolarov and Diego Ruspini, Interval Research Corporation
	"HL: A Library for the Robust Haptic Display of Complex Graphical Environments," Diego Ruspini and Krasimir Kolarov, Stanford University
12:45-1:45	Lunch
1:30-3:00	Free time
3:00-4:30	Session 7: Data Visualization I
	"Haptic Scientific Visualization," Jason P. Fritz, Kenneth E. Barner, University of Delaware
	"Nanomanipulator Force Feedback Research," Russell Taylor and Jun Chen, Dept. of Computer Science, University of North Carolina
	"Incorporation of Haptics into a 3D Interactive Workbench for Geophysical and Geological Data," Megan E. Clark, P.K. Williams, D. Stevenson and K.Smith, WMC Resources Ltd. and CRC for Advanced Computational Systems
4:30-5:00	Break
5:00-6:30	Session 8: Data Visualization II
	"Haptic Interaction with the Visible Human," Karl Reinig, University of Colorado Center for Human Simulation
	"Interacting with 3-Dimensional Medical Data," Andrew Mor, Mitsubishi Electric Research ITA/Carnegie Mellon University
	"Haptic Interaction Utilizing a Volumetric Representation," Ricardo S. Avila, Lisa M. Sobierajski, General Electric Corporate Research & Development
7:00-8:30	Dinner
8:30-9:30	Refreshments

Monday Sept 30

7:30-8:30	Breakfast
9:00-10:00	Departure (optional bus to airport)

Expressive Haptics

Bill Verplank
Interval Research
verplank@interval.com

A pianist sets fingers on keys, a painter brings brush to canvas, a sculptor presses clay. Why are these skilled and satisfying interactions? What are the qualities of physical manipulation that allow artists to become skilled? How can those qualities be provided in our interactions with computers?

At Interval we are doing research on the design of human-computer interaction with special emphasis on the physical aspects of interaction. For two years now at Interval we have experimented with four PhanTom arms. Our goals are to understand better the role of force-feedback in manipulation both for simulation of virtual objects and for the design new devices.

In addition to our own research, we have supported and learned from a succession of students and faculty. Here are the students we have been working with, along with <e-mail> and (PhD thesis titles). We are pleased to be part of this growing community of researchers.

Margaret Minsky <marg@media.mit.edu>, at MIT and Interval simulated textures. Using forces just in-the-plane (never normal to the plane) she was able to create convincing "bumps" or textures in surfaces. (Computational Haptics, 1995)

Brent Gillespie <b-gillespie@nwu.edu>, at Stanford, built motorized keys to simulate the feel of a piano or harpsichord. He solved many of the dynamic simulation and stability problems inherent in sample-data systems coupled to variable human impedance. (The Virtual Piano Action, 1996)

Karon Maclean <maclean@interval.com>, recently joined Interval from MIT where she emulated the feel of real toggle-switches and sliders with her own 1-dof device. She conducted careful human subject experiments measuring the resolution and fidelity of the emulations. (Emulation of Haptic Feedback For Manual Interfaces, 1996)

Wendy Plesniak <wjp@media.mit.edu> is an Interval Fellow at MIT Media Lab's holography group. She has investigated drawing in 3-D and the value of active force-feedback for supportive constraints and better control. She is currently exploring the simulation of "pets". (PhD thesis proposal: Spatial Haptic Pets)

Diego Ruspini<ruspini@leland.stanford.edu> at Stanford, is extending fast distance and collision calculations developed for robotic path planning to haptic simulation of arbitrarily complex rigid objects, motion of objects and constraints between objects.

Tamara Munzner <munzner@cs.stanford.edu> at Stanford, is hooking up a PhanTom to Pat Hanrahan's Responsive Workbench and will be exploring the spatial correspondence of hand and 3D view.

Sundar Vedula <vedula@cs.cmu.edu> is a student of David Baraff and CMU working with a Phantom to manipulate simulated dynamic systems including contacts and collisions.

With this list of people and project highlights, I propose to describe our enthusiasm for this convergence of virtual haptics with product design, music, computer graphics and robotics.

- Bill Verplank <verplank@interval.com> has a PhD from the Man-Machine Systems Lab at MIT and has taught at MIT and Stanford. Before joining Interval Research, he worked at Xerox on graphical user interfaces and at ID Two, as an "interaction" design consultant.

- Interval Research is 60+ researchers in Palo Alto with a broad charter to "create new industries". All our funding is from Paul Allen.

An Overview of Haptics Research at MIT's AI Lab

Kenneth Salisbury
Dept. of Mechanical Engineering and Artificial Intelligence Lab.
Massachusetts Institute of Technology
jks@ai.mit.edu

1 Introduction

One of the fundamental interests in our research group has been the design of high performance mechanisms and sensors needed to advance the state-of-the-art in robotics. Of central concern has been the development of systems which utilize force information and force control to permit contact-intensive interaction with objects in the environment. It was inspiration from these experiences that motivated our work current work in Phantom-style haptics, and the lessons learned from the robotic activities that have guided our haptic interface design efforts.

We list here some of the devices we have built which pre-dated our Phantom activities, partly to provide a sense of history to our design philosophy and partly to give pointers to the early lessons we learned.

2 A Brief History of Devices

JPL Force Reflecting Hand Controller: a six-degree-of-freedom (DOF) joystick able to monitor user's and motions and exert force and torque vectors on them. Designed pre-1980 this device is used by NASA researchers to develop teleoperator control paradigms. [Bejczy & Salisbury 83]

Salisbury Hand (Stanford/JPL Hand): a three finger, 9-DOF hand featuring force controllable fingers. Completed in 1982, this force controllable articulated hand has been used by many researchers to explore sensing and control issues aimed at increasing robot dexterity. [Salisbury & Craig 82]

Brock fingertip sensor: a six-axis force sensing fingertip permitting contact detection and contact force measurement. This high-bandwidth contact force sensor provided important information on the geometric and temporal nature of forces which occur during contact interaction. Though used as a sensor, it's data hinted at the richness of information available from contact force information; it suggested to us that applying high-bandwidth force stimulation to humans might well provide an important new human-computer interaction modality. [Bicchi et. al, 90]

Whole-Arm Manipulator (WAM): 4-DOF force controllable arm able to contact and interact with objects with all it its surfaces. 1 meter reach. This arm sought to exploit force control capabilities at a large scale, permitting a single, mechanically efficient mechanism, to act as both a sensor and effector. [Townsend 88]

Force controllable actuator: Brushless torque motor with built-in reaction torque sensor for precise torque control. In our never-ending search for higher dynamic range in force application, we

explored the use of a tightly integrated sensor and actuator package, achieving better than 500:1 dynamic range. [Levin 90]

WristHand: 3-DOF wrist coupled with 2 1-DOF curling fingers for mounting on whole-arm manipulator. Controllable grasp forces and curling fingers for rapid stable object grasping. A design study aimed at providing the WAM arm with mechanically adaptive grasping. [Moyer 92] [Anderson 92]

Fast-Eyes Gimbal: a simple yet precise and high-bandwidth 2-DOF camera pointing device. Used by the WAM system for tracking moving objects for catching experiments. [Swarup 93]

Talon-End Effector: A light-weight WAM end effector, exploiting compliant transmission and finger mechanisms in order to grasp both moving and buried objects. [JPL progress report, in press]

Mini-WAM: 4-DOF miniature version of whole arm-manipulator. 1/3 meter reach. Used as force reflecting master for full size WAM and by itself as a manipulator. The warmups for the Phantom design.

Phantom: 6-DOF (3 active, 3 passive) device for exerting precise force vectors on user fingertips. High performance haptic interface. Initially designed as a device to enable users to "touch things on the screen". [Massie 93]

Shah Finger: a new, modular 3-DOF finger aimed at enabling the next generation of fingertip manipulation research. Exploits stiffening transmission techniques to mechanically enhance dynamic range and sensitivity. [ONR Progress report, in press]

Taken together, these projects have helped establish within our group a culture of design which favors simple high-performance force-controllable mechanisms. Yet, none of us have been satisfied with simply building devices. In each case the devices have been interfaced to computers and extensively programmed to demonstrate (and in some cases, disprove) our conjectures of what constitutes useful mechanism.

3 Haptic Rendering Activities

With the advent of the Phantom in 1993 our research took an exciting new turn. Rather than trying to algorithmically interpret sensory information in service of robot perception, we realized that there was an enormous potential for displaying to humans information through correctly modulated force stimulation. The goal of *haptic rendering* borrows from the fields of art and computer graphics, in that is seeks ways to evoke sensations of objects by appropriate sensory stimulation. In the visual domain rendering techniques seek to evoke the sensation of geometry and material properties by providing appropriate optical stimulation. In the haptic domain, rendering techniques similarly seek to provide to human "observers" the stimulation necessary to evoke the sensation of object geometry, behavior and material properties.

Our projects have explored a number of techniques for haptically evoking the properties of objects. These include the potential function based approaches for rendering simple geometry [Massie 96] [Salisbury et. al 95], constraint-based methods for rendering polyhedral objects [Zilles & Salisbury 95], and new tangent-plane based methods for rendering surfaces described by implicit mathematical functions [Salisbury & Tarr 97]. Texture display has also been of interest and we have found that texture sensations can be evoked by both micro-geometric perturbations and friction attributes

[Zilles 95] [Massie 96]. While to rendering compliant objects by finite-element methods has been of interest, traditional finite-element methods are (so far) too computationally expensive for the real-time demands of haptic rendering. A simplification of the finite-element approach developed by [Swarup 95], however, enabled users to interact with thick sheets of compliant material and has been used to guide further efforts in this area.

4 Projects

Our current activities in haptics continue to focus on rendering techniques as well as advancement of haptic interface hardware. We describe briefly some of the projects which support our haptics research.

ARPA Look and Feel: Haptic Interaction with Bio-Materials. This project has two goals. One, in conjunction with Boston Dynamics, Inc., seeks to develop techniques to enable surgical training and tryout. It focuses on compliant material rendering techniques needed to simulate bio-materials. We are currently developing techniques for "palpating-in" material viscoelastic properties needed to populate our models. The other goal of the project is to develop techniques for enhanced teleoperation. In addition to developing surgical robotic devices we are exploring methods for interactively combining virtual haptic constraints with real-time teleoperation to guide and constrain surgical procedures.

VETT Haptics Project. This project's primary goal is to support Naval training activities through the use of haptic interaction capabilities. In addition to supporting basic training and training transfer studies for cognitive and sensorimotor tasks, we are exploring larger workspace haptic interfaces, stimulation in other sub-modalities such as vibration and heat flow, and haptic extensions to the VRML representation for use in modeling training environments.

Reactor Maintenance Robot Simulation. This project explores training and tryout of reactor maintenance tasks via the modeling of reactor and task geometry. A real-time dynamic model of the Schilling Titan robot has been developed and enables users to feel robot dynamics as well as contact interactions. [Anthony & Salisbury 96]

NASA Vision and Touch Guided Grasping. While this project focus on basic NASA needs for performing planetary geology science activities via robot probes, it includes a component which explores methods for providing earth-bound scientists with haptic display of remotely gathered geologic data.

5 Acknowledgments

The work described herein has been supported by a broad range of agencies for many years. Beyond being grateful to them for their generous support, I am even more indebted to the many mentors and hard working students who have inspired and enabled the results described.

6 References

Anderson, Catherine, "The Design of a Compact Actuator System for a Robotic Wrist/Hand," MS Thesis, MIT Dept of Mechanical Engineering, February, 1992.

Anthony, B. and K. Salisbury, "Potential Tunnel Enhanced Telemanipulation" to be presented at SPIE's International Symposium on Intelligent Systems and Advanced Manufacturing, Telemanipulator and Telepresence Technologies III, Boston, Nov 1996.

Bejczy, A.K. and J.K. Salisbury, "Controlling Remote Manipulators Through Kinesthetic Coupling," *ASME Computers in Mechanical Engineering*, Vol. 2, No. 1, July 1983.

Bicchi. A, J.K. Salisbury and D.L. Brock, "Contact Sensing from Force Measurements," *International Journal of Robotics Research,* Vol. 12 No. 3., MIT Press, Cambridge, MA. (also MIT AI Lab Memo No. 1262, October 1990.)

Levin, Michael D., "Design and Control of a Closed Loop Torque Actuator," May 1990. MIT AI Lab Memo, AI-TR 1244.

Massie, Thomas H. "Design of a Three Degree of Freedom Force-Reflecting Haptic Interface", SB thesis, MIT EECS Department, May 1993.

Massie, Thomas H., "Initial Haptic Explorations with the Phantom: Virtual Touch Through Point Interaction," MS Thesis, MIT EECS Dept. February 1996.

Moyer, Thomas, "Design of an Integrated Wrist/Hand Mechanism," MS Thesis, MIT Dept of Mechanical Engineering, February, 1992.

Salisbury, J.K. and C. Tarr, "Haptic Rendering of Implicit 3D Surfaces," submitted to the 1997 Symposium on Interactive 3D Graphics.

Salisbury, J.K. and J.J. Craig, "Articulated Hands: Force Control and Kinematic Issues," *International Journal of Robotics Research,* Vol. 1, No. 1, MIT Press, Cambridge, MA, Spring 1982.

Salisbury, Kenneth, D. Brock, T. Massie, N. Swarup and C. Zilles, "Haptic Rendering: Programming Touch Interaction with Virtual Objects," Proceedings of 1995 ACM Symposium on Interactive 3D Graphics, Monterey, California, April 1995.

Swarup, Nitish, "Design and Control of a Two-Axis Gimbal System for Use in Active Vision," SB Thesis, MIT Dept of Mechanical Engineering, May, 1993.

Swarup, Nitish, "Haptic Interaction with Deformable Objects Using Real-Time Dynamic Simulation," SB Thesis, MIT Dept of Mechanical Engineering, September 1995.

Townsend, William T., "The Effect of Transmission Design on Force-Controlled Manipulator Performance," April 1988. MIT AI Lab Memo, AI-TR 1054.

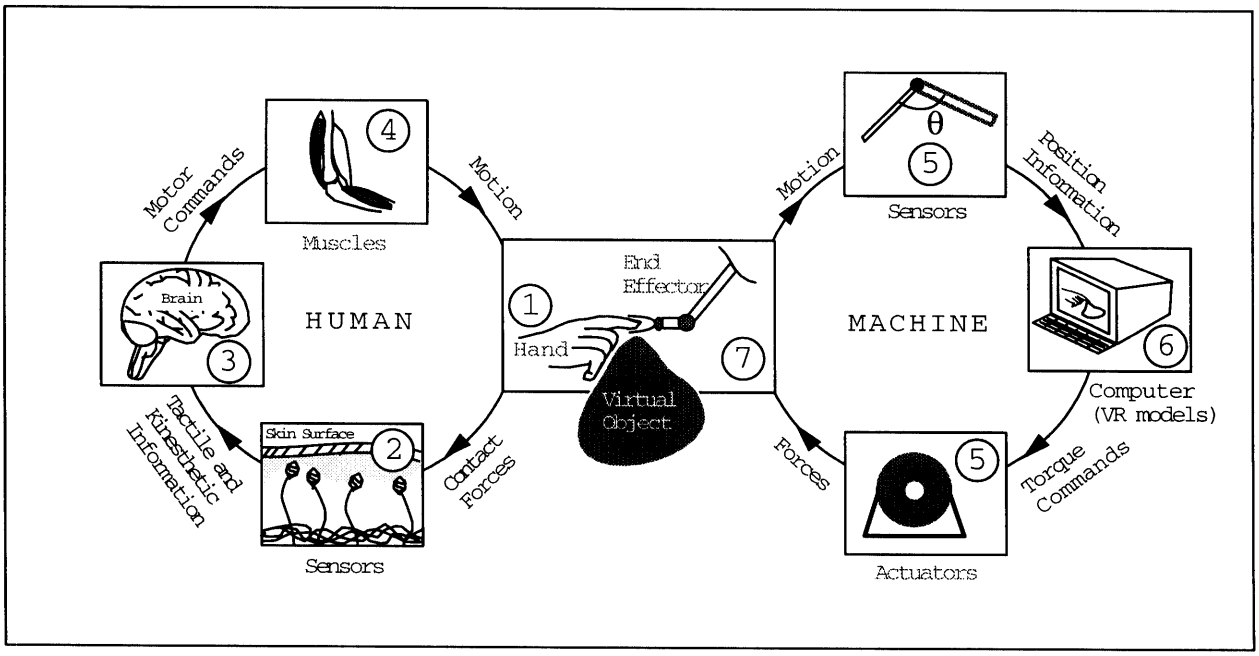
Zilles, Craig and K. Salisbury, "A Constraint-Based God Object Method for Haptic Display," proceedings of IROS-95, Pittsburgh, Aug 6-9, 1995.

HAPTICS RESEARCH AT THE MIT TOUCH LAB

Mandayam A. Srinivasan
Laboratory for Human and Machine Haptics
36-796, Massachusetts Institute of Technology
E-mail: srini@mit.edu

The goals of research conducted at the "MIT Touch Lab" are to understand human haptics, develop machine haptics, and enhance human-machine interactions in virtual reality and teleoperator systems. To gain a deeper understanding of human haptics, multidisciplinary investigations involving skin biomechanics, neurophysiology, psychophysics, motor control, and computational models are employed. Typical projects involve the measurement of human capabilities in specified manual tasks employing computer-controlled electromechanical devices, and the determination of the biomechanical, neural and perceptual mechanisms that underlie performance in these tasks. To develop haptic machines that enable the user to touch and feel virtual environments, electromechanical devices are designed, simulation and rendering software are developed, and studies on the human perception of virtual objects under purely haptic and multisensory conditions are conducted. The results of this research are also beneficial to hand therapy, intelligent prosthesis design, and the development of autonomous robots that need to perform human-like functions in unstructured environments.

Figure below illustrates the subsystems and information flow underlying interactions between human users and haptic interfaces to virtual environments. A review of human haptics, machine haptics, and the relationship between the two is given in Srinivasan (1995). The following list, ordered according to the numbers in the figure, gives brief summaries of research topics that are being investigated at the Touch Lab.



1. Biomechanics of Touch: Mechanical behavior of human fingerpads is measured using high precision robots and analyzed using finite element models to better understand how forces on the fingers due to contact with objects are converted into tactile information (Gulati and Srinivasan, 1995; Dandekar and Srinivasan, 1995).

2. Tactile Neurophysiology: The neural signals sent from the skin of the fingerpad to the brain during tactile sensing of object properties are recorded and analyzed (slip and microtexture: Srinivasan et al 1990;LaMotte and

Srinivasan, 1991);(shape: Srinivasan and LaMotte, 1991;LaMotte et al, 1996);(softness: Srinivasan and LaMotte, 1995).

3. Human Perception: The human ability to perceive object properties such as shape, texture and softness is measured using computer-controlled apparatus and psychophysical methods(Tan, et al, 1994,1995; Beauregard, et al, 1995).

4. Motor Action: The human ability to control contact forces during manual exploration and manipulation is measured and its relationship to sensory limitations is investigated (Srinivasan and Chen, 1993; Karason and Srinivasan, 1995; Beauregard and Srinivasan, 1996).

5. Haptic Device Development: Computer controlled electromechanical devices that can be programmed to convey the "feel" of virtual objects to the human user have been developed (Beauregard, et al, 1995; Srinivasan, et al, 1996).

6. Software Tools for Simulated Worlds: Software is being developed to create interactive virtual worlds and to display their visual, auditory, and haptic attributes to the human user (Morgenbesser and Srinivasan, 1996; Hou and Srinivasan, 1996).

7. Human-Machine Interactions: Experiments are performed to investigate how controlled alterations in visual, auditory, and haptic displays affect human perception and performance (Srinivasan, et al, 1996; Hou and Srinivasan, 1996). The results are useful in overcoming some of the technological limitations and are applicable to the design of optimal human-machine interaction paradigms.

8. Computational Theory of Haptics: A computational theory of haptics is being developed to provide a theoretical framework for information processing and control strategies common to both humans and robots performing haptic tasks.

In the past few years we have developed device hardware, interaction software, and psychophysical experiments pertaining to haptic interactions with virtual environments. The Linear Grasper is a device capable of simulating mechanical properties of objects such as compliance, viscosity, and mass along a single dimension. The Planar Grasper has a two-dimensional workspace, and software for simulating rigid walls, corners and springs have been developed. With the advent of the PHANToM, a variety of haptic rendering algorithms for displaying the shape, texture and compliance of objects have been created. All the three devices have been used to perform psychophysical experiments aimed at characterizing the sensorimotor abilities of the human user and the effectiveness of computationally efficient rendering algorithms in conveying the desired object properties to the human user. In the following sections we highlight some of the results of experiments on human abilities and the development of software for haptic interactions.

Purely Haptic and Multisensory Perception and Performance

Although technological limitations impose major constraints on haptic interface design, human abilities and limitations are also important in determining the design specifications for the hardware and software that enable haptic interactions. With this viewpoint, we have designed and conducted simple experiments to rapidly determine the human factors for the design of haptic interfaces (Tan, et al, 1994). More precise psychophysical experiments have been carried out with computer-controlled apparatus to measure human haptic resolution in discriminating fundamental physical properties such as stiffness, viscosity, mass, velocity and acceleration (Tan, et al, 1995; Beauregard, et al, 1995). During these discrimination tasks, haptic motor performance was also recorded and later analyzed. The results have led to the postulation of "Temporal force control - spatial force discrimination (TFC-SFD) hypothesis" which states that subjects apply a stereotypical force vs. time profile (almost linear ramps) and discriminate on the basis of force vs. position profiles (Beauregard and Srinivasan, 1996). Implications to the design of virtual environments include specifications on how accurately the object dynamics need to be simulated and what parameter values will ensure discriminable objects.

Psychophysical experiments have also been conducted to investigate how controlled alterations in visual, auditory, and haptic displays affect human perception. For example, in experiments on discrimination of stiffness of virtual springs, intentional skewing of the visual display of spring deformation relative to the kinesthetic sense of hand position resulted in a drastic misperception of stiffness (Srinivasan, et al, 1996). This is an example of how altered mappings of different modalities in multisensory virtual environments can enhance the range of properties perceived by the user. Experiments are underway to assess the influence of contact sounds on the perception of object stiffness.

Interaction Software Development

To display object shape, a haptic rendering algorithm called "Force Shading" has been developed (Morgenbesser and Srinivasan, 1996). It employs controlled variation in the direction of the reflected force vector to cause a flat or polyhedral surface to be perceived as a smooth convex or concave shape. Perceptual experiments that validate the algorithm have also been conducted. It has been demonstrated that a modification of this technique can successfully display surface texture. An implication is that polygon-based geometric model of an object can be common to both graphics and haptics; appropriate "shading" in each modality causes the object to be perceived as smooth or textured.

To facilitate rapid building of specific virtual environments, a tool kit called "MAGIC" has been developed (Hou and Srinivasan, 1996). It provides the user with virtual building blocks that can be displayed visually and haptically. The user can select primitive shapes such as cylinders, spheres, cubes and cones; move them and change their size, stiffness, and color; combine several primitive shapes to form a new, more complex object; save the scene for future use. Using this toolkit to design mazes, perceptual experiments on visual-haptic mappings and interaction paradigms have been performed. More recently, a new haptic rendering software called "HaptiC-Binder" has been developed to enable the user to interact with general polyhedral objects (Basdogan and Srinivasan, 1996).

Acknowledgments

The funding for most of the work described here was provided by ONR and NAWC/TSD. The author wishes to thank Dr. Cagatay Basdogan for preparing the figure and Dr. Lee Beauregard for help with preparing this document.

References

Basdogan C and Srinivasan MA, HaptiC-Binder: A software algorithm for touching and feeling arbitrary three dimensional objects in virtual environments, Unpublished document, 1996.

Beauregard GL, Srinivasan MA, and Durlach NI, Manual resolution of viscosity and mass, Proceedings of the ASME Dynamic Systems andControl Division, DSC-Vol. 57-2, pp. 657-662, ASME, 1995.

Beauregard GL and Srinivasan MA, Sensorimotor interactions in the haptic perception of virtual objects, The Engineering Foundation Conference on Biomechanics and Neural Control of Movement, 1996.

Dandekar K and Srinivasan MA, A 3-dimensional finite element model of the monkey fingertip for predicting responses of slowly adapting mechanoreceptors, Proceedings of the 1995 Bioengineering Conference, Eds: R. M. Hochmuth, N. A. Langrana, and M. S. Hefzy, BED-Vol. 29,pp.257-258, 1995.

Gulati RJ and Srinivasan MA, Human fingerpad under indentation I: static and dynamic force response, Proceedings of the 1995 Bioengineering Conference, Eds: R. M. Hochmuth, N. A. Langrana, and M. S. Hefzy, BED-Vol. 29, pp.261-262, 1995.

Hou IA and Srinivasan MA, Multimodal virtual environments: MAGIC toolkit and visual-haptic interaction paradigms, Unpublished document, 1996.

Karason SP and Srinivasan MA, Passive human grasp control of an active instrumented object, Proceedings of the ASME Dynamic Systems andControl Division, DSC-Vol. 57-2, pp. 641-647, ASME, 1995.

LaMotte RH and Srinivasan MA, Surface microgeometry: Neural encoding and perception, In Information Processing in the Somatosensory System, Eds: O. Franzen and J. Westman, Wenner-Gren Intl. Symposium Series, Macmillan Press, 1991.

LaMotte RH, Lu C, and Srinivasan MA, Tactile neural codes for shapes and orientations of objects, In: Information processing in the somatosensory system, Eds: O. Franzen, R. Johansson and L. Terenius. Birkhauser Verlag AB, Basel, 1996 (In press).

Morgenbesser HB and Srinivasan MA, Force shading for haptic shape perception (accepted for presentation at the ASME winter annual meeting), 1996.

Srinivasan MA, Whitehouse JM and LaMotte RH, Tactile detection of slip: Surface microgeometry and peripheral neural codes, J. Neurophysiology, Vol. 63, No. 6, pp.1323-1332, 1990.

Srinivasan MA and LaMotte RH, Encoding of shape in the responses of cutaneous mechanoreceptors, In Information Processing in the Somatosensory System, Eds: O. Franzen and J. Westman, Wenner-Gren Intl. Symposium Series, Macmillan Press, 1991.

Srinivasan MA and Chen JS, Human performance in controlling normal forces of contact with rigid objects, In Advances in Robotics, Mechatronics, and Haptic Interfaces, DSC-Vol. 49, ASME 1993.

Srinivasan MA, Haptic Interfaces, In Virtual Reality: Scientific and Technical Challenges, Eds: N. I. Durlach and A. S. Mavor, Report of the Committee on Virtual Reality Research and Development, National Research Council, National Academy Press, 1995.

Srinivasan MA and LaMotte RH, Tactual discrimination of softness, J. Neurophysiology, Vol. 73, No. 1, pp. 88-101, 1995.

Srinivasan MA, Beauregard GL, and Brock, DL, The impact of visual information on haptic perception of stiffness in virtual environments,(accepted for presentation at the ASME winter annual meeting), 1996.

Tan HZ, Srinivasan MA, Eberman, B and Cheng, B, Human factors for the design of force-reflecting haptic interfaces, In: Dynamic Systems and Control, Vol. 1, Ed: C. J. Radcliffe, DSC-Vol.55-1,pp. 353-359, ASME, 1994.

Tan HZ, Durlach NI, Beauregard GL, and Srinivasan MA, Manual discrimination of compliance using active pinch grasp: the roles of force and work cues, Perception and Psychophysics, Vol. 57, No. 4,pp. 495-510, 1995.

Protecting the PHANToM from the Programmer

Rob Kooper, John Barrus, David Ratajczak, David Parsons

MERL - A Mitsubishi Electric Research Laboratory
201 Broadway
Cambridge, MA 02139

kooper@cc.gatech.edu, barrus@merl.com, dratajcz@mit.edu, parsons@svl.meitca.com

(617) 621-7500

ABSTRACT

Anytime you connect motors and amplifiers to a computer, you make it possible for the computer to break something or hurt someone. However, careful design of hardware and software can permit full use of a device within limits imposed by safety concerns. It is trivial to cause one such device, the Phantom [Massie and Salisbury '94] from SensAble Technologies, to damage itself and in fact it is quite difficult not to do so. In this paper, we describe a robust solution to this problem.

INTRODUCTION

Programming SensAble Technologies Phantom is difficult for many reasons, one of the foremost being that it is quite easy to write a program that will destroy the machine, as is shown in Example 1. This program will cause the Phantom arm to accelerate until it reaches its mechanical stops. High electrical currents will continue to flow to the motors and the insulation on the motor windings will melt and short out the motors.

```
#include <phantom.h>
void main(void) {
    /* Torque in Newtons */
    setPhantomTorques(10, 10, 10);
    sleep(1000);
}
```

Example 1. Application that will break the Phantom.

At MERL, we have been working on a system which makes it impossible for a programmer to hurt the Phantom in any way. This system is a combination of hardware and software which monitors both the Phantom and the actions of the software driving it, thus eliminating dangerous or fatal software commands.

The system protects the Phantom in four ways:

1. Thermal protection: If the motors are driven with too high of a current for too long, the amplifiers are disabled while the motors cool down.
2. Speed limits: If the end of the arm accelerates beyond a certain maximum speed, no additional torque is applied to the arm.
3. Torque limits: Torque requests sent to the Phantom motors are kept within safe limits.
4. Workspace limits: Resistive forces are applied in order to decelerate the Phantom when the user moves it too close to its mechanical workspace constraints.

Maximum speed, torque, and workspace limits are built into our safety system, but the programmer has the option of specifying more cautious limits while debugging Phantom programs.

DESIGN PHILOSOPHY

Providing a safe environment for the Phantom requires careful consideration of the entire system. For instance, if some of the safety features are implemented in software, that software must not be influenced by factors which are out of the control of the architect of the system, such as bugs in other people's programs. This consideration moved us to develop an embedded controller which removes the possibility of someone else's software disabling our safeguards. An additional advantage of using an embedded controller is the opportunity to take over some of the processing normally done by the application computer, leaving more time for the application to run.

The following are some of the things we considered when designing the Phantom safety system:

- Never assume motors are at ambient temperature when application starts.
- Provide for absolute torque (current) limits at all times.

- Do as many calculations as possible on dedicated processor(s) to reduce the workload on the application computer.
- Reduce or eliminate complexity for the programmer.
- Allow safe addition and removal of objects while Phantom is operating.

CONTROLLER ATTRIBUTES

Based on the above design philosophy, we created a controller for the Phantom which has the following attributes:

- Thermal simulation runs on embedded controller.
- Thermal simulation cannot be stopped by removing power or by programmer error and thermal protection cannot be shut off.
- Automatic conversion between forces and torques and between angles and XYZ positions done.
- Temperature, position, velocity, and error information provided to application programmer on request or at specified intervals.
- Reduces computational load on application computer.
- Eliminates dangerous forces caused by new objects appearing in same location as users hand.

In order to accomplish the above tasks, we provide additional processors between the Phantom hardware and the application computer. One of those processors is essentially an embedded controller which controls the Phantom directly using information received from the application computer over an Ethernet connection (see Figure 1).

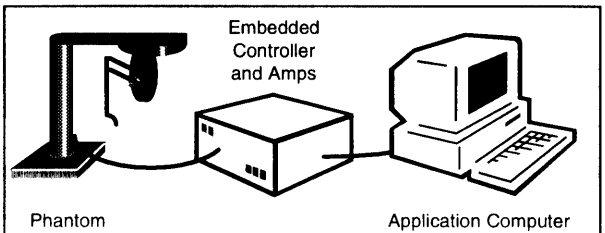


Figure 1 Phantom setup

Using this controller, the programmer no longer needs to set and read bits directly on the Phantom plug-in board. The user can create small packets of information, like torque requests, and convey those directly to the controller over the Ethernet. Information is conveyed back to the application using the same communication channel. Since this is network based, many computers can receive the same information at the same time, although only one can have direct control of the Phantom.

CONTROLLER IMPLEMENTATION

A library of abstract C++ classes is provided to the application programmer. These classes provide network connectivity, temperature checks and predefined objects and are used to communicate with the controller. This enables the programmer to quickly build an application that can connect with the Phantom controller and issue commands and display Phantom specific information.

The current implementation of the controller is based on a 90Mhz Pentium running a real time UNIX variant (LynxOS). The communication between an application computer and the controller is handled by TCP/IP over Ethernet.

Thermal Protection

Since thermal protection is so vital, a unique solution was developed to prevent the motors from burning out. In addition to the thermal simulation of the motors running on the controller, we designed and built an electrical system which mimics the thermal performance of the motors. The voltage at one point in the electronic circuit precisely tracks the heating and cooling of the motors' internal windings and housing. Powering down the hardware has no affect on the cooling of the motors nor does it adversely affect the analog electronic simulation. The tracking voltage is always measured on start up to determine the current temperature of the motors for the software simulation. Additional electronics monitor the tracking voltage and automatically shut off the amplifiers if the voltages, and therefore the internal motor temperatures, reach dangerous levels. Even if the controller suffers an unexpected software failure, the motors will still be protected.

Torque and Velocity Limits

The Phantom controller converts the forces supplied by the user, or calculated internally, to torques. After this calculation, and before supplying these values to the Phantom, the controller checks to see if these values are within the range specified by the user. If the not, the Phantom controller scales the force vector to keep it within range.

When converting the angles of the Phantom to a position, the Phantom controller calculates the velocity of the Phantom. If the velocity is outside of the range sup-

plied by the user, the Phantom controller will not apply a force until the velocity comes within the set limits.

Workspace Limits

One of the predefined objects in the controller is a model of the reachable workspace of the Phantom. This model is used to decelerate the Phantom whenever the probe approaches either the limits of the Phantom's reach or fixed obstacles in the workspace, such as a desktop. This provides an important operating safety feature, and also supplies useful haptic feedback to the user. The geometry and haptic rendering algorithms of the workspace model can be easily changed.

SENSABLE'S APPROACH

SensAble Technologies has a Phantom library (called GHOST) which does all of the thermal calculations and works with a watchdog timer on the amplifier box to reduce the chance of burning out motors. However, it has no provisions for the situation in which a person using the Phantom might shut off the software when the motors are hot and then restart the program immediately. On restart, the GHOST library must assume that the motors are at ambient temperature because the GHOST library has no way of measuring the real temperature. If the motors are hot on startup then it is possible to burn out the motors using SensAble's software. In order to guarantee that the motors are at ambient temperature, more than 15 minutes would have to pass before restarting the program because of the amount of time it takes for the motor housing to cool.

FUTURE WORK

Unfortunately, due to limitations in capacitor technology, the electronics that mimic the motor temperature are not very reliable or stable and require frequent recalibration. Other methods of thermal tracking are being pursued, in addition to the software based thermal simulation.

The speed of the Ethernet communication is too slow. Maximum round-trip communication using the current embedded controller is about 500 Hz which prohibits the representation of stiff objects. Although rewriting the Ethernet drivers for our system might speed it up a bit, we are also considering other communication technologies like FireWire, Univeral Serial Bus and High Performance Parallel Interface when they become more widely available. Also, the use of a simple stiff wall model in the controller like the one used in the UNC Phantom library might be an effective alternative [Mark et al. '96].

Another way to alleviate problems with communication speed is to remove the need for high speed communication completely. If a haptic model could be loaded into the controller and the controller could then do all of the necessary calculations for delivering the correct force to the user, the applications computer would not have to worry about haptic calculations at all and the Ethernet communications speed would no longer be a bottleneck. In the near future we will be considering a model format or file format for storing and communicating haptic information to the controller. This format will provide for transmission of not only haptic object geometry but also surface properties such as friction. The controller could implement a friction model such as the one described in [Zilles and Salisbury '95].

ACKNOWLEDGEMENTS

The authors would like to thank the people at SensAble Technologies for the quick response and support every time we broke our Phantom.

REFERENCES

[Mark et al. '96] William R. Mark, Scott C. Randolph, Mark Finch, James M. Van Verth, and Russell M. Taylor II, "*Adding Force Feedback to Graphics Systems: Issues and Solutions*", Computer Graphics (SIGGRAPH'96 Proceedings), 30(2), pp. 447-452.

[Massie and Salisbury '94] T. Massie and K. Salisbury, "*The PHANToM Haptic Interface: A Device for Probing Virtual Objects*", Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems, Chicago, IL, November 1994.

[Zilles and Salisbury '95] Craig B. Zilles and J. Kenneth Salisbury, "*A Constraint-based God-object Method For Haptic Display*", Proceedings of the International Conference on Intelligent Robots and Systems (IROS '95).

Taking the Mush Out of Haptics with Infinitely Stiff Walls

Thomas Massie

thomas@sensable.com
SensAble Technologies, Inc.
26 Landsdowne Street
Cambridge, MA 02139
617-621-0150

Abstract

Convincingly hard walls and free space that feels free are perhaps the two most important criteria for a useful haptic interface system. For this reason, several studies have been done on optimal software algorithms for creating very hard walls. Approaches to creating hard walls have including increasing spring constants to the ragged edge of instability, introducing hardware and software damping, and even inertia cancelling impulses upon user contact. I will present a novel method that I developed which (neglecting hardware deflections) effectively creates a wall with infinite stiffness.

The algorithm is easy to understand and is best described as the analogy to a classical P.I.D. (Proportional Integral Derivative) controller. Most haptic interface algorithms to date implement the Proportional term in the form of a spring constant which pushes back on a user when the user penetrates a virtual surface. Some algorithms go one step further an implement the Derivative term as a damper proportional to the velocity that a user is penetrating a virtual wall. The P and D terms have physical analogies, are easily conceptualized, and don't have adverse side effects due to discontinuities when the user makes a transition from free space to a wall or vice versa. Implementing the Integral term in a virtual environment is not as straightforward, but has profound implications for wall stiffness. I will present the problems I encountered in adapting this classical control algorithm to virtual haptic environments, the remedies I tried, and the final working solution which includes a temporal-spatial filter. Source code and live demonstrations will be presented at the Phantom User's Group.

Description of the Approach

For simplicity, this algorithm is described in one degree of freedom, but is extensible to a full three degree of freedom controller. The user's position in space is defined as X_{user}. If the surface that we wish to represent as a hard wall occurs at $X_{user}=0$, then for values of X_{user} greater than 0, the user is considered above the surface and for values of X_{user} less than 0, the user is considered to be penetrating the surface. The desired position is at the surface when the user is penetrating the surface, and the desired position is the user's position when the user is above the surface (in free space).

Specifically, $X_{des}=0$ when $X_{user}<0$ and $X_{des}=X_{user}$ when $X_{user}>0$. Now, E_p is defined as the difference between X_{user} and X_{des}.

For a wall with a "proportional controller," the rule for determining the reaction force is simply $Force=K_pE_p$. This is best described physically as a virtual spring. For a wall with a "proportional derivative controller," $Force=K_pE_p + K_dE_d$, where $E_d=d(E_p)/dt$. The derivative term is best described as a virtual shock absorber. Finally, for a wall with a full "proportional integral derivative controller", $Force=K_pE_p + K_iE_i + K_dE_d$, where $E_i= \int E_p\, dt$. The closest physical analogy that comes to mind for the integral term is a virtual pneumatic cylinder that starts being filled with air when the user enters the surface.

What function do each of these three terms contribute to a hard wall? The **further** a user pushes into a surface, the *harder* the proportional term will push back. The **faster** a user pushes on a surface, the *harder* the derivative term will push back. And finally, the **longer** a user pushes into a surface, the *harder* the integral term will push back.

A direct application of this classical PID controller would work very well for simulating hard, stiff virtual walls, except there is an issue of what to do with the integral term, K_iE_i, when the user moves from contacting or penetrating the surface to being in the free space above the surface. As they are defined above, $E_p=0$ and $E_d=0$ when the user moves to free space. But because E_i is the integral of E_p and E_p is 0 in free space, E_i will remain at some finite, non zero value when the user moves to free space. Imagine a surface that pushes on a user long after the user is no longer touching the surface!

Improvements

So can we make a special case and simply set E_i to zero when the user enters a region of free space? No, this does not work. Ironicly, the integral term always pushes the user to the exact place where the special case is applied - right at the transition from surface to free space. If E_i is set to zero as soon as the user leaves the surface and enters free space, then a casual user exerting a constant force into the surface would be "jacked up" by the "pneumatic cylinder" only to be dropped back below the surface repeatedly. How do I know this? This was the first solution that I tried!

Observing the defect in the first approach of setting $E_i=0$ as soon as the user left a surface led to the second approach. Specifically, instead of setting E_i to zero as soon as the user left the surface, a region of epsilon thickness was defined above the surface. If the user was above the surface, but within epsilon of the surface, the E_i term was allowed to stay constant. When the user moved at least epsilon from the surface, the E_i term was set to zero. As one might expect, this approach was only marginally better than simply setting E_i to zero in the region of free space, but it was worth a try. Furthermore, pursuing this approach further led to the realization that any discontinuity in E_i would probably lead to problems. E_i had to be the integral of E_p while penetrating the surface, and E_i had to be zero in free space, but discontinuities in E_i with respect to time or space had to be eliminated.

A temporal decay function was first used to attenuate the E_i term when the user moved to free space. At each iteration of the algorithm, $E_i=E_i / (1 + \text{beta})$, when the user was outside of the wall. This was a definite improvement over the earlier discrete one-step elimination of E_i. It was during the process of searching for the optimal value of beta that the realization was made that beta should be position dependent. The algorithm was then changed to $E_i=E_i / (1 + (\text{beta}*x))$, where x was the distance the user had traveled above the surface. If the user was either very near the surface, or if not much time had passed, the E_i term would not be attenuated much, but if many time steps have passed, or if the user was very far from the surface, then E_i would be greatly attenuated.

The results of this algorithm were very promising. Surfaces could be made to feel infinitely stiff and no odd discontinuities in force were perceivable. However, the surfaces had an "active" quality that was very distracting. That is, the wall often seemed to impart more energy to the user than the user had put into the wall.

The final tweak to the algorithm, which might be considered a "hack" or "insight" depending on the reader's point of view, was to let the derivative (velocity damping) term remain active, even after the user had left the wall, but only so long as $K_dE_d<K_iE_i$. (E_d was redefined as $d(X_{user})/dt$ rather than just $d(E_p)/dt$ for this calculation.) And in the case where $K_dE_d>K_iE_i$ and the user was in free space, E_i was set to zero instantaneously. If the user was leaving the wall quickly, this part of algorithm would avoid imparting more energy than needed to the user. Adding this branch in the algorithm made a tremendous difference in the quality of the feel of the wall.

Conclusion:

The purpose of this endeavor was to improve the basic software algorithms for creating virtual haptic surfaces by implementing an integral term. This paper describes a reliable, working solution, and how I came to realize this solution. Perhaps most significant was the realization that the transition of the integral term between the wall and free space would require a temporal-spatial filter, with velocity considerations. Although this paper may not contain the optimal implementation of the integral term, I have shown that it is both possible and worthwhile to include the integral term in haptic algorithms. It is my hope that others might look for and find a better implementation.

The Sense of Torque
With
A Single Phantom Haptics Device

Dennis Hancock
Hewlett-Packard Laboratories
dennish@best.com

The Phantom is an example of a haptics I/O device that provides 6 degrees of freedom (DOF) input control (with the optional gimbal attachment) but is limited by design to only 3 degrees of freedom (linear) force output. In real life when we move three dimensional objects around an environment with our hands, we feel stopping forces and twisting torques during object collision. Humans use this 6 DOF feedback to intuitively accomplish path planning (e.g. removing the milk pitcher from the back of the refrigerator shelf.) It might be supposed the lack of output torque from the Phantom would limit its use to only linear placement of generalized 3D objects in a synthetic scene. However this is not entirely true. Recent experiences with a single Phantom reveal there is a sense of torque the user experiences which aids in the placement of 3D objects. To be sure, there is no real torque in the classic mechanics. The implication for this observation is that many path planning and CAD synthetic assembly tasks can be accomplished with a single Phantom, saving the cost of a second Phantom and computer system.

We have written software that computes collision detection/ force-torque generation for a 3D polygonalized object moving in a similarly characterized static world. Objects in this world can be convex or concave. Currently, no surface friction models or material compliance models have been implemented. Because of the lack of output torque capability of the Phantom device, we cannot negotiate object placement/orientation in our synthetic world to mimic the experience in the real physical world. Two Phantoms used together can display torques along the two directions perpendicular to the moment arm described by the line connecting the endpoints of the two arms. But this architecture effectively doubles the cost of the hardware, both for the haptics display and the computers. Our experience with a single Phantom suggests there are some classes of placement tasks that can be effectively performed even though no output torque can be applied.

Consider the simple scene shown in Figure 1a. The stick object encounters the solid block at a point contact, and a torque $\underline{t}$ is generated by the vector force $\underline{F}$ provided by the the users hand operating on the moment arm $\underline{R}$ formed from the point of contact to the location of force application. In the Figure, we see that the moving object initially pivots around the point of contact due to this torque and a user of a haptics device would expect to feel a twist in their hand. The evolution of the forces and torques in this example depends on many factors and several scenarios could develop. But the important point is that someone experiencing these forces can involuntarily negotiate a realistic path around the obstructing object.

The condition shown in Figure 1b shows the situation of no torque generation because there is no moment arm to act on. The user would only feel a force of opposition to motion and no torque. This comment pertains not only to a real world physical situation but also a synthetic situation where one could have a fully 6 DOF haptics display. But we all know that an applied torque can make this object rotate about this point of contact. In the synthetic

world experience where a person is using a haptics device to move objects, he can voluntarily provide this torque with his hand and rotate the object. The force/torque diagram for this case in shown in Figure 1c and should be compared with the diagram of Figure 1a. The only difference in these figures is that the force vector in 1c is at the point of contact. This is the force/torque diagram that arises with a haptics device with only a linear force output and no torque output. No matter where the stick collides with the object there will be no moment arm.

It has been our observation that there is a sense of torque a person experiences in these situations that aids in placing parts in a 3D world. To be sure there is no real torque--there can't be. But there is the kinesthethic sense that one can roll an object around another in a believable way. We make the following observations about this phenomenon:

1) when the user of a haptics device with only 3 DOF output linear force voluntarily supplies the torque of collision with his hand, there is a sense of torque that aids in the negotiated placement of objects in a 3D world.

2) This condition is greatly aided when the colliding objects can be visualized and the person's experience with the real world can be applied.

3) This sense of torque can only be experienced for the orientations consistent with the force vector applied by the users hand. Thus, no apparent twisting forces can be felt and current Phantoms cannot be expected to mimic screwdrivers and wrenches. (However, devices equipped with 3DOF torque instead of 3DOF linear force output could mimic the displacement of rotated objects with the voluntarily application of linear forces. These two display options are complementary.)

Our current hardware system consists of 2 Hewlett-Packard 735 UNIX workstations running asynchronously of each other. The machines run in the real time priority mode and both exchange information with each other using Berekely sockets over an Ethernet connection. One machine is dedicated solely to stereoscopic rendering of our graphics scene. This 125MHz computer runs code utilizing the Starbase graphics library and renders a 3300 flat shaded polygonal scene of the Ford Mustang at 20Hz. Stereoscopic viewing of the CRT is accomplished with non headtracked CrystalEyes eyewear. The second workstation is a 100 MHz, 50 Mflop machine that runs our OS non-specific collision detection/force-torque generation code. Both codes are written in C to avoid any execution time penalties currently associated with C++. This machine currently is executing at the 1 hapton* level due to an I/O bottleneck to the Phantom. We estimate that this hardware-software system is potentially capable of 10 haptons.

Acknowledgements. I gratefully acknowledge the financial and materiel support of Fred Kitson of the Hewlett-Packard Laboratories for this project. And hats off to Mike Goss for his assistance and PC wizardry. I would also like to thank the Hewlett-Packard Company for past support on other unreported haptics projects. Finally, I would like to thank NASA Goddard for initially funding my collision detection algorithm work in support of the Hi Gain Antenna Study for the Hubble Telescope.

* We define the hapton as the number of triangle pair collisions that can be computed 1000 times per second. Spatial accuracy of collision computation must be done to a precision of 0.1% or greater and the triangles must be intersecting on edge. The hapton is a measure of haptics computer system performance in analogy to the polygon/sec as a unit of graphics system rendering performance.

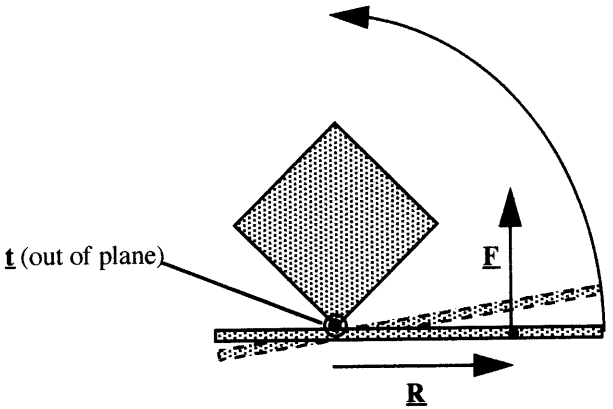


Figure 1a. Rod pivots due to torque **t** created by applied force **F** acting on moment arm **R**

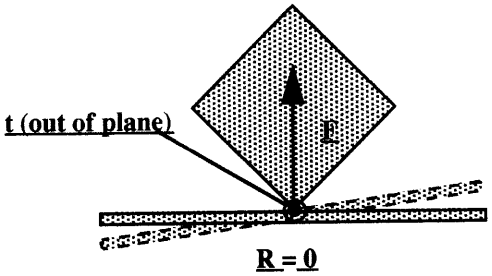


Figure 1b. Rod pivots due to voluntarily applied torque t. Force F acts on no moment arm, creating no torque.

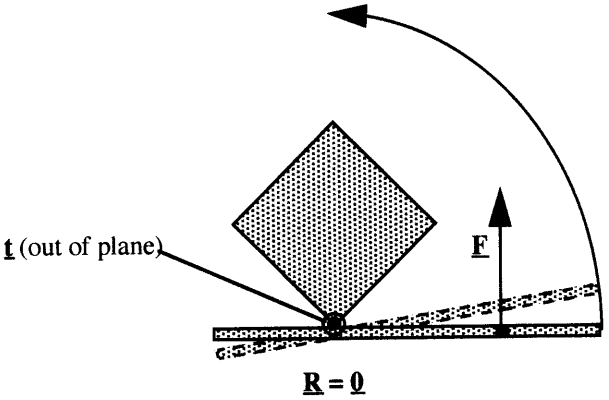


Figure 1c. Rod pivots due to torque **t** voluntarily applied by user. Force **F** acts on no moment arm, creating no torque.

Home Haptics Laboratory

Margaret Minsky
marg@media.mit.edu

The time has come to revisit the pin-block, usually found in toy stores for making 3D pictures of your face. It is 4" x 6" array of metal pins, available in novelty stores for about $10. It's a great toy; most people use it to make 3D impressions of their faces, hands, etc. In this essay, I will show you some recent observations using the pin-block as an "object to think with", to highlight certain issues in human haptics and display design.

1 Haptic interaction with 3D Images

Many researchers in the community have realized that a large scale pin-array, with fast actuators, is an interesting haptic display. But perhaps not everyone has realized that without actuation, it already is a perfectly good, though highly limited teleoperator mechanism.

If you place the pin-block on top of a radio console with buttons and knobs, they all show up nicely. If you touch the 3D images, the surfaces (as sampled by the pinheads) are rigid. If you push on the button surfaces, they operate the buttons. You can choose a radio station. However, if you try to turn the volume up, you can't. This device transmits the shape and size of all the controls, as well as the button feel, but of course it can only transmit normal forces, along the pins, and no torques or shears.

In that simple experiment, the block has been unexpectedly transformed from an imaging device to a control device with haptic feedback.

2 Textures and Materials

In my recent work, I showed that certain properties of textures can be perceived when shear forces only are available to the finger. The pin-block provides the dual filter, providing normal forces only from the environment, to the fingertips. What happens if you use this rigid teleoperator to feel more subtle textures and materials?

To test this, I placed the pin-block on each page of "Feely Bugs", a tactile book for kids. Each bug is covered with a textile. Lacy bug can neither be seen nor felt through the block. Velcro bug can be seen as a raised shape, but the block does not have enough resolution for the individual hook shapes to be transmitted. Puffy bug is a big success in haptic transmission. It can hardly be seen, but the boundaries of the spongy fabric can easily be felt by scanning the fingertips across the block.

Compliance is such an important property of textured materials that it can provide a sense

of presence, despite the fact that the surface texture available to the fingertips (the smooth scaly texture of the pin heads) is radically different from the soft fuzzy texture of the fabric.

3 Living Things

What do you need to transmit the presence of another person's hand? I was surprised at how animate the image of a hand in the pin-block is. One part of the surprise is finding that you can interact haptically with what appeared to be a visual representation. But there is more to the surprise. It is okay that the skin texture is different than real skin. The transmission of compliance is very important to the sense of living tissue. Scanning across the pin-hand with a single fingertip one can see and feel that the fingerpads, even the whole hand taken together, act as bags of incompressible fluid. When you push on one the right side of the palm, the pins on the left side push up. And of course, you can feel deliberate motions from the other hand as the person flexes and moves.

4 Summary

In summary, the unactuated pin-block toy is a subtle laboratory for contemplating issues about haptic display such as the role of normal and shear forces, the relative importance of compliance and position, and the components of animacy and presence.

Tactile Display of Shape And Vibration

Robert D. Howe
Harvard University
Division of Engineering and Applied Sciences
Pierce Hall, 29 Oxford St., Cambridge, MA 02138
howe@deas.harvard.edu

1 Introduction

Like most haptic interfaces, the Phantom provides force feedback, where a single force vector is applied to the end of the finger or arm. In general, haptic perception also relies on cutaneous information, produced by distributed sensations across the skin. We are developing new devices to generate this type of feedback, including both small-scale shape and high-frequency vibration. Shape is useful for both object recognition and for manipulation control. Knowledge of object shape permits determination of the local surface normal direction, which is important for predicting frictional behavior. The shape of the object at the contact with the finger tip also determines the kinematics of the contact, and thus whether the object can pivot or roll against the finger. Vibrations are produced by transient events such as contact and slip, and humans make extensive use of this type of information in manipulation. Vibrations also allow us to distinguish textures and detect contamination by stroking a finger over a surface. Beyond the system development aspects, our work is directed at understanding how tactile information relates to task properties and perception. We are also creating algorithms for synthesizing tactile feedback in virtual environments.

2 Shape display

To recreate small-scale object shapes, we have developed a number of tactile shape displays. These devices consists of a regular array of pin elements which rest against the user's finger tip (Figure 1). Shape memory alloy (SMA) wires raise and lower individual elements to approximate the desired surface shape on the skin. Each element can produce over 1 N of force and up to 3 mm of height variation. Current versions include 6x4 and 10x1 element arrays (Kontarinis et al. 1995). We have implemented a feedforward controller and fluid cooling, which raises the bandwidth of the otherwise slow SMA actuators (Howe et al. 1995a). In contrast with other tactile shape displays, these devices combine the small size and weight that permits mounting on a haptic interface like the Phantom, with the high force levels essential for portraying object shape under force reflection.

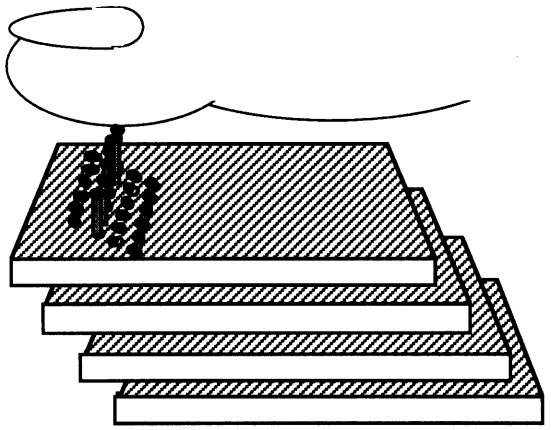


Figure 1. Shape display and human finger tip. Individual pins are raised and lowered by shape memory alloy actuators to approximate the shape of objects.

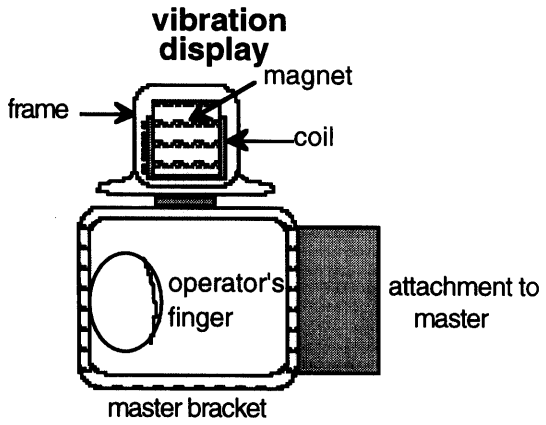


Figure 2. Vibrotactile display system. Vibrations generated by the vibration display are transmitted through the square sheet metal frame to the user's finger (shown end-on).

These devices can also be used to relay tactile information in teleoperated manipulation. Tactile array sensors in the gripping surface of the remote robot's hand measure the pressure distribution at the contact with grasped objects. This sensor's output is sampled by a computer, which applies signal processing algorithms and drives the shape display at the operator's finger tip. One application for these systems is in minimally invasive surgery, where shape information is particularly important for tasks such as finding hidden anatomical features and assessing tissue properties (Howe et al. 1995b). In our laboratory experiments, we have demonstrated that this system can be used to locate simulated tumors hidden in compliant "tissue."

3 Vibration display

In addition to the low-frequency distributed information provided by shape display, physiological and psychophysical studies suggest that it is helpful to provide high-frequency vibratory information. The frequencies of interest here are from a few dozen Hz to over 1 kHz. Since this is in the audio frequency range, miniature loudspeakers are easily modified to create prototype vibration display devices. The paper cones and metal frames are removed, and the remaining structure containing the coil and central diaphragm are mounted "upside down," so the base containing the permanent magnets is free to move in space (Figure 2). Passing current through the coil generates a force against the magnet, which accelerates the relatively massive base and produces an inertial reaction force against the manipulator. In our tests, inexpensive speakers produced a 3 mm range of motion, and up to 0.25 N peak force at 250 Hz.

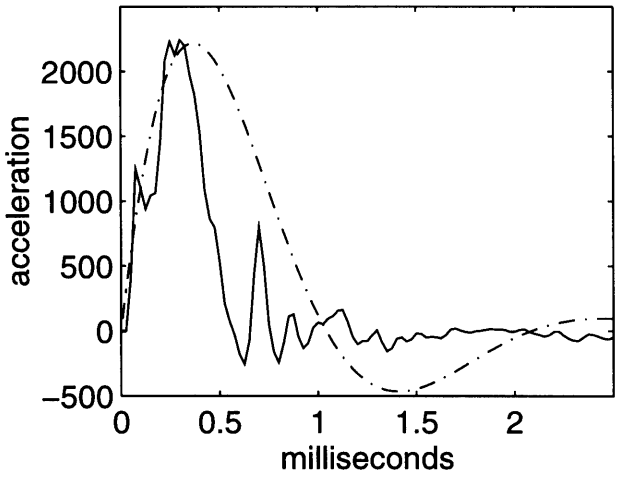


Figure 3. Recorded and synthesized vibration waveforms for tapping an aluminum block with a steel tool tip.

To investigate the role of vibrotactile feedback in various tasks, we have used a telemanipulation system equipped with a vibration sensor on the remote robot's gripper and a vibrotactile display on the user's finger tip (Kontarinis and Howe 1995). Manipulation performance could be compared with and without vibrotactile feedback, and correlated with task properties. In inspection and exploration tasks like assessing surface roughness, perception of vibrations is a key part of the task, so vibrotactile feedback is essential for success. In other manipulation tasks, vibrations reveal the state of the system, and thus aid in execution. One example is needle biopsy, where vibrations denote the puncture event, and vibration feedback can reduce reaction times or permit minimization of forces. Some manipulation tasks, such as assembly of close-fitting parts, are limited by the ability to coordinate forces, so vibrotactile feedback is of marginal value. Current work is directed at better understanding the relationship between tactile feedback and task performance, and optimizing sensor and display device characteristics.

3.1 Virtual vibrations

Vibration feedback is also useful in virtual environments, which raises the question of how to generate the appropriate waveforms. As an initial study, we have examined the problem of hardness perception based on tapping a surface with a tool (Wellman and Howe 1995). We began by measuring the waveforms that are produced in actual tapping with an instrumented stylus. A simple exponentially damped sinusoid model was fit to data (Figure 3), parameterized by desired hardness and approach velocity. This model was then used to produce waveforms in an experiment that assessed the validity of this characterization. Using these simple waveforms

for vibrotactile display in a virtual environment allowed subjects to accurately distinguish surface hardness. In ongoing work, we are examining algorithms for representing other transient events and surface textures, and issues in synchronizing vibrotactile and force feedback.

Acknowledgments

This paper describes the work of a number of people in the Harvard Robotics Laboratory, including Dimitri Kontarinis, William Peine, and Parris Wellman. Financial support was provided by the Office of Naval Research under grants N00014-92-J-1814 and N00014-92-J-1887.

References

R. D. Howe, D. A. Kontarinis, and W.J. Peine, "Shape memory alloy actuator controller design for tactile displays," Proc. 34th IEEE Conference on Decision and Control, New Orleans, December 13-15, 1995a.

R. D. Howe, W. J. Peine, D. A. Kontarinis, and J. S. Son, "Remote palpation technology for surgical applications," *IEEE Engineering in Medicine and Biology*, 14(3):318-323, May/June 1995b.

D. A. Kontarinis and R. D. Howe, "Tactile display of vibratory information in teleoperation and virtual environments," *Presence*, 4(4):387-402, 1995.

D.A. Kontarinis, J.S. Son, W.J. Peine, and R. D. Howe, "A Tactile sensing and display system for teleoperated manipulation," Proc. 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, May 1995, p. 641-646.

P. Wellman and R.D. Howe, "Towards realistic vibrotactile display in virtual environments," T.E. Alberts, ed., Proc. ASME Dynamics Systems and Control Division, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, San Francisco, November 12-17, 1995, DSC-Vol. 57-2, p. 713-718.

[Note: These and related references are available on our WWW site, URL http://hrl.harvard.edu/people/faculty/howe/howe.html.]

Bump mapping for a force display

Yukio FUKUI
National Institute of Bioscience and Human-Technology
1-1, Higashi, Tsukuba, Ibaraki 305, Japan
fukui@nibh.go.jp

ABSTRACT: *Haptic surface features such as undulation and texture are represented both by shape profile itself and by deviating normal direction just like bump mapping in the computer graphics technology. These two factors which form apparent shape through human haptic exploration are investigated. Through psychophysical experiments they are found to contribute independently to human haptic perception of the shape profile, and the shape profile itself is more effective compared to the pseudo profile derived from normal direction deviation.*

1. INTRODUCTION

When we touch an object of little friction such as melting ice, we receive a reaction force to the normal direction of the point where we touch it with our fingers. Because of the integration of surface tangent becomes the surface profile, we can guess the entire shape profile by obtaining the sequence of surface tangent or the surface normal as a substitution of it through haptic exploration. However we can find usually the shape profile through tracing along the outline of it. Here these two facts imply that we can perceive a shape profile by obtaining the sequence of either the surface geometry itself, or its differential information. Here arises a question that which component is more dominant for haptic perception of the whole shape profile when we touch and explore a curved object of little friction. Fortunately, for a representation of virtual objects with a force display, we can control independently these shape profile parameters (geometry and its differential information). To modify the surface normal direction, we introduce a "bump mapping" method which is used in computer graphics to making a realistic shaded image. As a related work, M.Minsky developed a virtual 3D haptic display by controlling the 2D driving force[1].

2. SHAPE RECOGNITION EXPERIMENTS

Through consideration of the previous chapter, we picked up two shape profile parameters that may control the human perception of a shape profile. Hereafter we call these parameters a "position parameter" and a "normal parameter" respectively. Next question is whether these two parameters works linearly onto the human perception or not. Here the linearity means whether the effects of these two parameters can be mapped onto a single psychophysical scale or not. Therefore we performed psychophysical experiments to investigate the problem.

2.1 Method

We used a virtual cylinder as a target object for recognition which could be modified by two shape profile parameters. The cross sections of the modified object are shown in Fig.1. Equations of the cylinder using parameter t is

$$\begin{aligned} x(t)&=a(t)*\cos(t) \\ y(t)&=a(t)*\sin(t) \\ a(t)&=r*(u*\cos^2(2t)+1), \end{aligned} \qquad (1)$$

where r is radius of the cylinder and u is a position parameter. The directions of reaction force are obtained by differentiating Eq.1 with t, then rotating 90 degrees and u is substituted by v as

$$\begin{aligned} nx(t)&=(2+5v-8v*\cos(2t)+5v*\cos(4t))*r*\cos(t) \\ ny(t)&=(2+5v+8v*\cos(2t)+5v*\cos(4t))*r*\sin(t). \end{aligned} \qquad (2)$$

Here v is a normal parameter. Here the parameters u and v can be changed independently, becoming two shape parameters which determine apparent haptic shape profile.

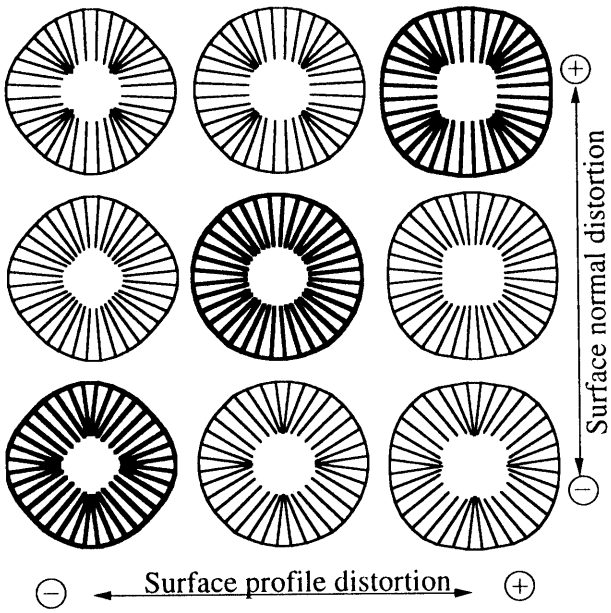


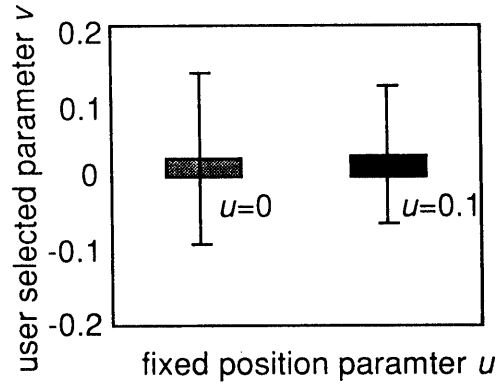
Fig.1 Two types haptic distortion

Figures show cross sections of cylinder-like 3D virtual object. Subjects can explore inner side of them by PHANToM. The bars from the surface toward the center show the directions of reaction force when cylinder sides are touched. From left to right, peripheral shape profiles are distorted. From top to bottom, directions of reaction force distorted. The bold line figures are consistent with surface profile and direction of reaction force as are real objects.

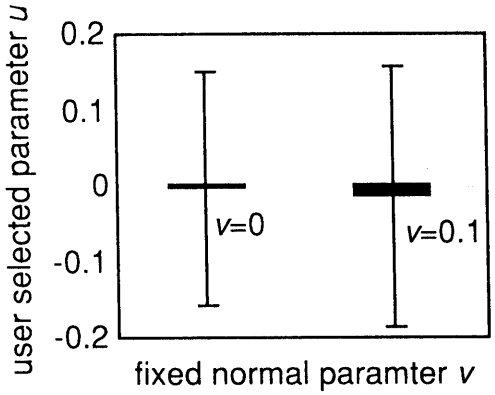
One of parameters (position or normal) is set fixed and the other is required to be changed by the subject. Eight subjects are participated in this experiment. A Subject can touch and trace along the inner surface of the cylinder using PHANToM, and is asked to make the cylinder feel most "smooth" by changing the parameter with specified keys on the board. The experiments were performed at first the position parameter fixed, then the normal parameter fixed. The actual radius r of the cylinder was 2cm, then 5cm.

2.2 Result and Discussion

The figure 2 shows the result of the experiment described in the previous section. There found no significant differences between the cases of different fixed parameter values. This means that one parameter value with which subjects feel "smooth" does not depend on the other. If the two shape profile parameters could be mapped onto a single psychophysical scale, then one parameter value of the most smooth case should have depended on the other.



(a) fixed position parameter case



(b) fixed normal parameter case

Fig.2 Results of experiments

3. QUANTITATIVE EVALUATION

We performed experiments to measure the equivalent psychophysical amount of deformation made by the two parameters. We used undulation on a flat surface. Figure 3 shows the shape profile and direction of reaction force.

3.1 Method

Two surfaces are presented haptically to the subject. One is a reference, and the other is a working surface of which the amount of undulation can be controlled by the subject. The shape profile is represented as

$$y=u*\cos(2\pi x/p)/2 \quad (3)$$

where u is a position parameter and p is a constant period of undulation, while x and y represent the coordinates. The direction vector of reaction force is represented as

$$(\pi v*\sin(2\pi x/p), p) \quad (4)$$

where v is a norm parameter. Note that parameters u and v have a dimension of length in this case.

Three experiments were performed. The first one presented normal shape shown in Fig.3(a). The subject is asked to make the perceived amplitude just same as the reference. In the second experiment, the reference plane is a shape in Fig.3(b), while the work plane is Fig.3(c). The third experiment exchanged them.

(a) normal shape (u=v>0)

(b) reaction force deviating shape (u=0, v>0)

(c) profile only deviating (u>0, v=0)

Fig.3 Shapes used in the experiment

3.2 Result and Discussion

The results of three experiments are shown in Fig.4. From the results we found that the shape representing parameter is more effective than the direction representing parameter when they are presented individually.

4. CONCLUSION

Human haptic perception, as well as visual perception, is subject to characteristic distortions. We investigated this phenomenon to make the most of this human illusion for simplifying haptic displaying algorithms. We found that both the shape profile itself and the surface normal direction depict the apparent shape independently while haptic exploration. We can use both factors superimposed just like bump mapping in computer graphics techniques.

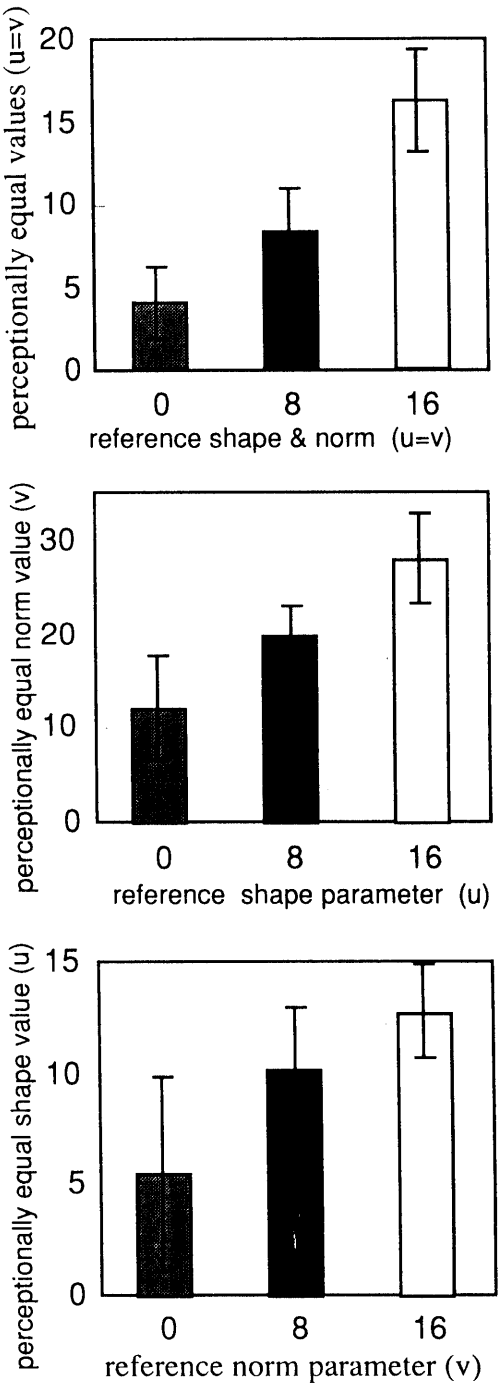


Fig.4 Equal perceptinal deformation

5. Reference

[1] M.Minsky, M.Ouh-Young, O.Steele, F.Brooks and M.Behensky, "Feeling and Seeing: Issues in Force Display", Computer Graphics, Vol.24, No.2, pp.235-243, 1990

Using PHANToMs for Telesurgery: the HMSL system, experiments and experience

Mark P. Ottensmeyer, Jianjuen Hu, James M. Thompson, Jie Ren, Thomas B. Sheridan
Human-Machine Systems Laboratory, MIT, Cambridge, MA

Abstract:

Many groups have used the PHANToM Haptic Interface arms to interact with virtual environments, but they can also be used as both master and slave in a teleoperator system. In the Human-Machine Systems Laboratory at MIT we are involved in a project to examine a number of issues related to the development of a telesurgery system. We have used a pair of "tool handle" PHANToMs to provide force feedback to the telesurgeon and as a platform for a dedicated telesurgical tool. The telesurgery system also provides for audio communications between the surgeon and an assistant at the remote site, and transmission of a color video image of the surgical task. The issues that we investigated were control of the teleoperator when subject to system time delays and the interaction between the telesurgeon and assistant under various time delay conditions. To support these efforts, the system includes hardware and software to generate delays, and surgical simulators to test the performance of the telesurgeon/assistant team.

The system and the experiments will be described, and the results and our experience will be discussed.

Introduction:

The first use of a pair of PHANToM haptic interface arms in a teleoperator mode seems to have been a small demonstration made of two small units set up in a simple position (P) control loop by Rhonda Massie in 1993 [Massie, 1996]. The development of the "tool handle" version opened up the possibility of a larger workspace teleoperator with excellent force feedback potential. Two of these units were acquired by the HMSL by early 1995, and with the addition of a teleoperated surgical tool, have become the heart of the HMSL telesurgery system.

Telesurgery is an exciting new development in the field of telemedicine, which seeks to extend the reach of a surgeon to areas that are either difficult or dangerous for a doctor to reach (e.g. Antarctic research station or battlefield). A number of other groups are active in this area, both in development of systems and in early clinical use. SRI Inc. in California [Green, et al., 1995], have developed a two armed, five degree of freedom, bilateral force reflecting telesurgery system which provides the surgeon with accurate force feedback and stereoscopic views of the surgical site. It has been demonstrated favorably in performing surgical tasks on simulators with animal tissue. At Johns Hopkins Medical Institute, in Baltimore, a simpler system called tele-mentoring is in clinical use [Kavoussi, et al., 1994]. An expert surgeon uses a telerobotically controlled laparoscopic camera to instruct a surgeon learning a new procedure. The expert can be almost an arbitrary distance away from the operating room, yet still be able to impart his knowledge to the learning surgeon.

The HMSL telesurgery system has been used to look at two aspects of telesurgery not specifically addressed by either of these groups, namely what the effects of introducing time delays in the system are, and how to coordinate the interaction between the telesurgeon and the assistant. Time delays enter the system from primarily two sources: large distances over which communications may occur, and limited communication channel bandwidth. To transmit image data efficiently with limited bandwidth, image compression hardware is used, but this introduces at least a 1/4 second delay between each site. With increased time delay and a system with force feedback, there is the potential for instability to develop in the control loop which is clearly unacceptable in a surgical milieu. SRI and Johns Hopkins use short, direct links between master and slave to avoid non-trivial delays in their systems. The issue of surgeon/assistant interaction springs from the recognition that a less skilled assistant on the scene is unencumbered by any shortcomings of the teleoperator and can serve as extra pairs of hands, eyes and ears under the direction of the surgeon. We wanted to know how best to assign surgical subtasks within the team to yield best performance. The system we used and the experiments we performed will be discussed next.

HMSL telesurgery system: (figure 1)

The heart of the system is the combination of the PHANToM arms with a teleoperated surgical tool which gives the surgeon physical access to the remote environment. Between the three d.o.f. available from the PHANToM and one further spatial degree (roll axis) controlled by the tool, the system is suitable for performing laparoscopic surgical tasks. Laparoscopy is a form of minimally invasive surgery, in which long surgical tools are inserted through sealed tubes inserted in the abdominal wall, permitting the operation to be performed without ever exposing the internal organs to the outside air.

When used in non-delayed teleoperator mode, the arms are controlled with a standard PD (position +

derivative) control scheme [Sheridan, et al., 1995]. The control loop runs at approximately 500Hz, which provides reasonable force feedback. The tool, which controls the roll axis and the position of any of three interchangeable tool tips (gripper, shears, laparoscope) uses P control, but does not provide force feedback. In laparoscopy, the surgeon's primary feedback is visual, so complete force feedback was not considered essential.

When time delays are simulated by the system, a novel control scheme, called Fuzzy Sliding Mode control is used to prevent the instability mentioned earlier. Developed by Hu [Hu, 1996], this scheme modifies the sliding mode control algorithm by varying the thickness of the boundary layer using a fuzzy logic algorithm which takes into account position and velocity differences between the command and the current state of the master and slave. It effectively removes small perturbations locally before they can grow into larger instability.

In addition to force feedback, the video image from the laparoscope is displayed to both the surgeon and assistant, and audio communication is provided between the team members. In addition to the teleoperated tool, the surgeon controls a mouse pointer to help give concise instructions to the assistant. The mouse pointer is displayed on both monitors. A commercial audio-video delay generator is used to synchronize the audio and video signals with the delays in the teleoperator system. Thus, the assistant deals with the "patient" in real time, while the surgeon must contend with the delays.

Experiments and sample results:

To simulate surgery, we chose a subset of surgical training tasks [Steele, et al., 1994] designed specifically for laparoscopy. These tasks included grasping and transferring objects between grippers, using the scissors, laparoscopic clip appliers and the laparoscope (camera). For each task, the tools used were rotated between surgeon and assistant to test all combinations of the interaction, and four levels of (fixed) time delays were tested. These levels included a "standard" with the surgeon acting directly, the surgeon acting through an undelayed teleoperator, and surgeon acting through teleoperator delays of 0.6 and 1.2 seconds, round-trip. A representative example of the results, for the use of scissors task, is shown in figure 2. In this task, lengths of suture material had to be grasped and cut at specific locations, simulating removal of excess suture after knot-tying. Notable results include the following: (1) completion time when the surgeon was using the undelayed teleoperator was the same as when the surgeon acted directly, (2) completion times increased linearly with time delay when the surgeon controlled either active tool (gripper or scissors), and (3) time was roughly constant when the surgeon controlled only the laparoscopic camera, delegating the use of the active tools to the assistant.

Discussion, conclusions and recommendations:

From the results, which were similar across all of the test tasks, there are a number of encouraging findings. First, the similarity in performance between the "bare-handed" and undelayed tele-surgeon indicates that the PHANToM based surgical teleoperator performs very well for these tasks. All of our results could have been called into question if the undelayed teleoperator had caused significant performance degradation. Second, by delegating active tasks to the assistant, the team can complete tasks as quickly under time delayed conditions as without delay. This supports the possibility of using tele-mentoring-style systems over longer distances, where the expert controls only the camera. The drawbacks to telesurgery are clearly demonstrated when the surgeon takes an active role, in this case increasing completion time by a factor of four for a delay of 1.2s. This would suggest that surgeons may not be able to perform active tasks over long distances.

In observing the participants performing the tasks, a number of observations are relevant to these conclusions. When subject to time delays, people using teleoperators often spontaneously begin to use a "move and wait" strategy [Ferrell, 1965], in which they make small incremental motions, then wait for the feedback to arrive before initiating further motions. We saw this very clearly in these experiments, especially at the 1.2s delay level. This move and wait cycle adds significantly to completion times for active tasks, and prevents real continuous control of the tools by the surgeon. When controlling the camera, the surgeon was still subject to the delay, but since the field of view was large enough, imprecise positioning of the camera was not a factor that contributed to longer completion times. The other major observation was that with the use of the Fuzzy Sliding Mode control, in eliminating the small perturbations, the fine force feedback useful to the surgeon in the final stages of positioning the tools was diminished by the algorithm. This is something of a two-edged sword, because though the loss of some feedback makes the task more difficult, the surgeon likely would not be able to compensate for the asynchrony between command and the corresponding feedback.

As a result of these experiments, we have some of the first quantitative measurements demonstrating the differences between different surgeon/assistant team interaction modes for varying time delays. Future work should include extensions to this work, including the development of a more versatile telesurgery system (e.g. open surgery), more realistic surgical simulations (e.g. bleeding vessels, motion due to breathing), and work to develop further methods to reduce the effects of the time delays.

Acknowledgment:

This work is supported by the U.S. Army Medical Research, Development, Acquisition and Logistics Command (Prov.) under Contract No. DAMD17-94-C-4125. The views, opinions and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy or decision unless so designated by other documentation.

References:

[Ferrell, 1965] Ferrell, W.R. (1965) "Remote Manipulation with Transmission Delay," IEEE Transactions on Human Factors in Electronics, vol. 6, pp24-32.

[Green, et al., 1995] Green, P.S., Hill, J.W., Jensen, J.F., Shah, A.S. (1995) "Telepresence Surgery," IEEE Engineering in Medicine and Biology Magazine, vol. 14, no. 3, pp324-329.

[Hu, 1996] Hu, J. (1996) "Stable Force Reflecting Control of a Telerobotic Surgery System with Time Delay," Master's thesis, department of Mechanical Engineering, MIT, Cambridge, MA.

[Kavoussi, et al., 1994] Kavoussi, L.R., Bender, J.S., Moore, R.G., Zenilman, M.E., Partin, A.W., Satava, R.M. (1994) "Telerobotic Assisted Laparoscopic Surgery: Initial Laboratory and Clinical Experience," Urology, vol. 44, no. 1, pp15-19.

[Massie, 1996] Thomas Massie (1996), SensAble Technologies, personal communication.

[Steele, et al., 1994] Steele, R.J.C., Hosking, S.W., Chung, S.C.S. (1994) "Graded Exercises for Basic Training in Laparoscopic Surgery," Journal of the Royal College of Surgery, Edinburgh, vol. 39, pp112-117.

[Sheridan, et al., 1995] Sheridan, T.B., Thompson, J., Hu, J., Ottensmeyer, M., Ren, J. (1995) "Human Factors in Tele-Inspection and Tele-Surgery: Cooperative Manipulation and Time Delay," Annual Report for USAMRDALC, contract no. DAMD17-94-C-4125.

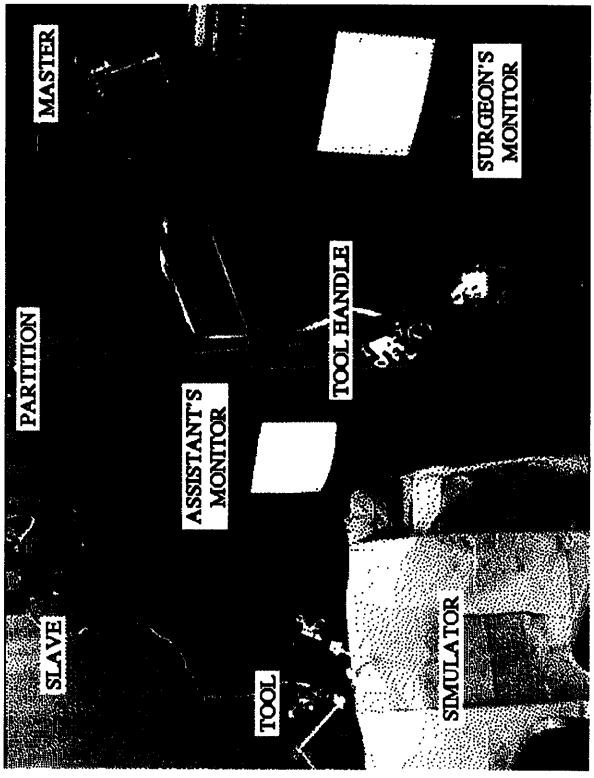


Figure 1: HMSL Telesurgery System. Master and slave could be separated by large distances

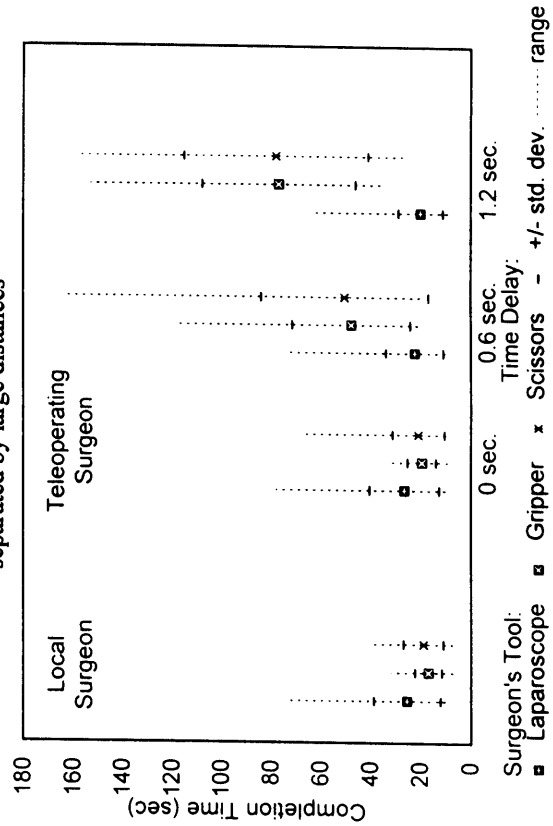


Figure 2: Use of Scissors experiment results.

A Haptic Process Architecture using the PHANToM™ as an I/O Device in a Virtual Electronics Trainer

Scott W. Davidson
Management Systems and Training Technology, Co.
2300 Ninth Street South, Suite 502
Arlington, Virginia 22204
scott@rdvax.ntsc.navy.mil

This paper describes a software architecture and use of the PHANToM™ as an I/O device in a Virtual Environment (VE) application. The VE application models the sight, sound, touch, and simulation of an electronics test console used to teach navy students basic electricity and electronics. The Model 130 Test Console, shown in Figure 1, is a portable electronics trainer that provides a prewired AC/DC power supply system, an input-output system, and an in-circuit faulting system to teach students the concepts of basic electronic theory and trouble-shooting. The Model 130 Test Console supports approximately 270 circuit boards and allows the insertion of faults using switches located at the bottom of the console (Nida, 1987).

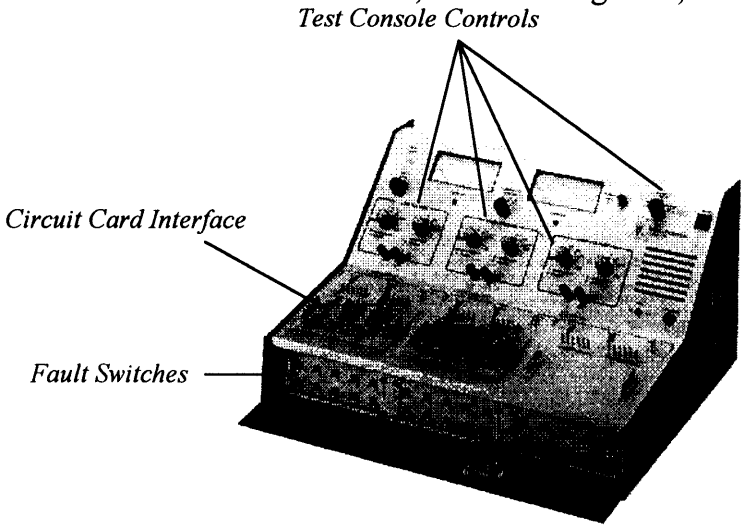


Figure 1. The NIDA Model 130 Test Console.

VIRTUAL MODEL 130 INTERACTION

The user interacts with the simulated Model 130 in the context of a virtual workbench as shown in Figure 2 (Weigand, 1994). The PHANToM™ tracks the user's hand in the environment and is shown as a small 3-D cursor in the scene; this shape is also used for the multimeter positive and negative probes. Interaction within the VE involves touching toggle buttons, multimeter probes,

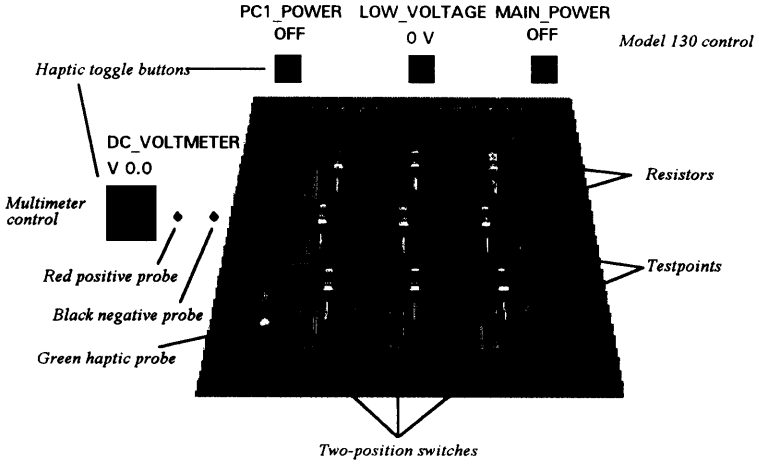


Figure 2. The Virtual Model 130 Test Console.

and testpoints as well as switching on and off the two-position switches on the circuit board.

SOFTWARE DESIGN

The software architecture is shown in figure 3. The main software infrastructure is provided by the commercial product dVs/dVise from Division Inc. The user-interaction is almost totally defined in the context of dVise. dVise is an interactive authoring tool for building and experiencing virtual environments (Division, 1995). For the virtual electronics application, it was most important to capture the aperiodic *touch events* generated by the trainee and to initiate action updates as a result of those events. The dVise actions cause aperiodic updates to the state of the simulation based on either the touch of a toggle button or the selection of a testpoint with the multimeter. *Switch events* are initiated by the haptics process when the user moves a two-position switch on the circuit board.

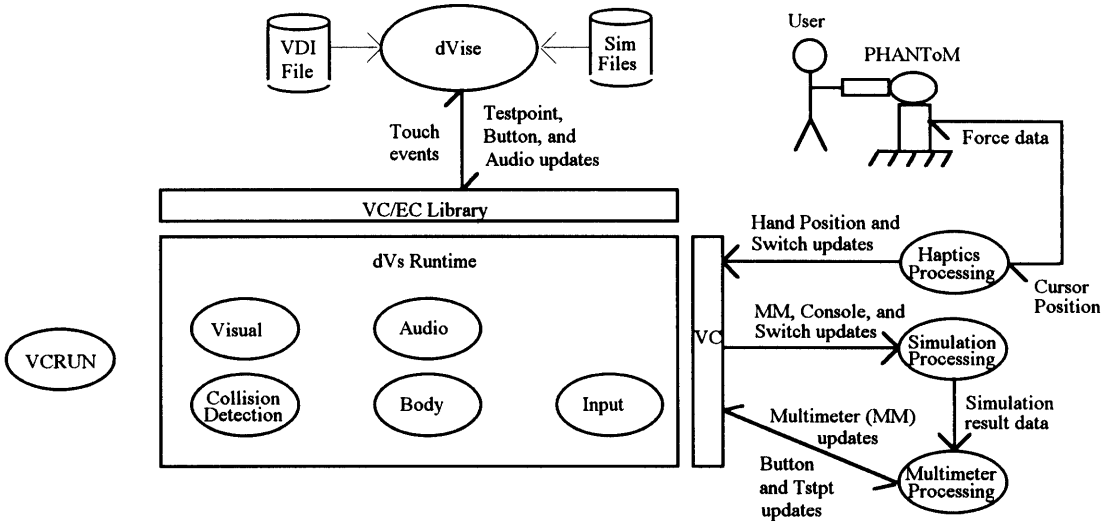


Figure 3. Virtual Model 130 Software Architecture.

Touch Events. The haptics process tracks the PHANToM™ position and provides it as hand "sensor" data to the body actor (an "actor" denotes a standard Division runtime process). The body actor provides an interface between the user and the VE and monitors movement of the body. Likewise, the collision actor monitors the movement of the hand and determines when the hand has touched an object. The collision actor, however, only does coarse-grain collision detection to determine when two objects have collided. The touch events are registered with dVise which are then used to issue testpoint or toggle button updates to the dVs runtime database; dVise also issues an action to the audio actor to play a sound as a result of any touch event. The multimeter process is notified of touch events related to multimeter control and updates the appropriate control variables in the simulation. Additionally, the multimeter process is responsible for updating the numeric display of the multimeter at 30 Hz. The simulation process is notified of console switch settings and updates control variables in the simulation.

Switch Events. In addition to monitoring the hand, the haptics process also updates the dVs runtime database with changes to the two-position switches. The switch state updates are passed to the Model 130 simulation process to reflect the switch manipulations.

Simulation Model

The first task in developing the virtual Model 130 was creating a simulation of the console electronics and providing a reconfigurable model of various circuit cards that can be used with the console. DYMOLA (DYnamic MOdeling LAnguage) was used to perform the complete modeling of the console, circuit cards, and multimeter. DYMOLA is an object-oriented language and a program for modeling large continuous-time systems with discrete events (Cellier, & Elmqvist, 1993). Models are hierarchically decomposed into submodels, and model reuse is supported through libraries containing model types or classes. Connections between submodels are described by defining an interface which represents physical coupling. The physical coupling for this application is represented by the electrical interface between the circuit card and console.

Haptic Model

The haptics process uses an explicit representation for modeling the circuit card components, haptic toggle buttons, and positive/negative probes of the multimeter shown in Figure 2 (Zilles, 1995). Most of the interaction involves touching static testpoints and toggle buttons in the scene. However, there are five instances of a two-position switch with articulation as shown in Figure 4.

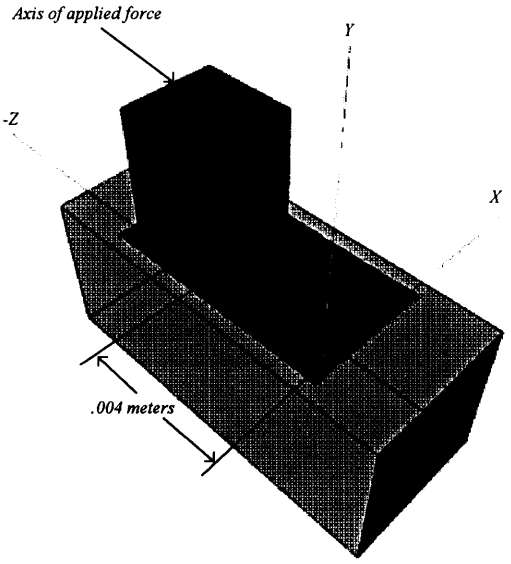


Figure 4. Two-position Switch with Mechanical Stops

The switch has one-degree of freedom in translation along the z axis with mechanical stops. In Figure 4, the switch is shown 4 millimeters extended from its home position. Simulation switch files were needed to describe the switch motion to the haptic system and are loaded via dVise. The switch simulation files describe the number of states, initial position, minimum value, maximum value, motion increment, threshold force, and simulation parameter control. Switch dynamics, however, are not computed. If the threshold force is exceeded in the direction of motion, the geometry is transformed by the motion increment amount. Additionally, the switch state data is updated in the dVs runtime database.

Visual Model

The visual model was developed using ModelGen from MultiGen Inc. The model includes external files representing common components used on the virtual circuit board and includes five two position switches, nine resistors, thirteen testpoints, and one circuit

board. Texture mapping was used on all components to add realism. Toggle buttons for the virtual Model 130 and multimeter were also created to control the simulation using the haptic interface.

Audio Model

Audio sounds representing interactions with the Test Console were recorded using Silicon Graphics audio tools and saved as .aif files. The sounds were played back for every "touch" event defined in the VDI database.

REFERENCES

Cellier, F. E., & Elmqvist, H. (1993). Automated Formula Manipulation Supports Object-Oriented Continuous-System Modeling. IEEE Control Systems, Vol 13, No. 2.

Division Inc. (1995). dVise Developer Guide. Division Inc.

Nida Corp. (1987). Nida Model 130 Test Console Technical Manual. Nida Corp.

Weigand, T. E. v. (1994). The Virtual Workbench & the Electronics Training Task. Internal Memo, VETT Project at MIT/RLE.

Zilles C. (1995). Constraint-based God Object Methods for Haptic Interfaces. MIT Masters Thesis.

Integrated Haptics Applications: Surgical Anastomosis and Aircraft Maintenance

Robert Playter, Bill Blank, Nancy Cornelius, Webb Roberts, Bob O'Toole
Boston Dynamics Inc., Cambridge, Massachusetts USA

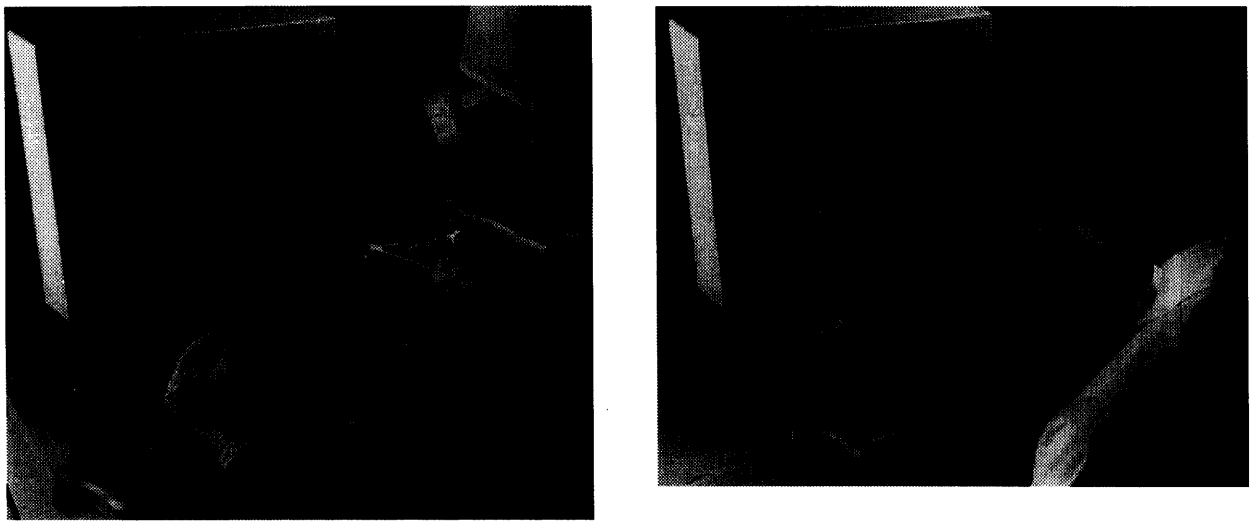


Figure 1: Virtual Trainers for Surgical Anastomosis and Aircraft Maintenance

Introduction

People use their hands to touch, manipulate, and learn about the world around them. Interaction through touch and manipulation is known as haptic interaction. BDI is developing simulation-based training systems in which haptics plays a prominent role. One application is a trainer for surgical procedures, such as end-to-end anastomosis and palpation of calcified arteries. Another application is a trainer for maintenance of military aircraft.

Our basic approach is to create dynamic computer simulations of the systems to be trained, and to connect the simulations to 3D computer graphics and 3D haptics. The graphics provide the sight and sound of the system, while the haptics allow the user to touch, feel, grasp, and manipulate the simulated system. The dynamic simulation calculates motions and forces that allow the simulated system to behave as a real system would behave when touched and manipulated. Haptic algorithms, including contact detection and contact force calculation, mediate interactions between the dynamic simulations and force feedback devices.

The haptics system we developed for these applications, *TangibleReality*^{TM}, uses SensAble Phantoms for the high fidelity force feedback, fast dedicated PCs to simulate virtual objects and compute haptics algorithms, and Silicon Graphics workstations for high quality 3D graphics and sound. In this paper we describe two integrated training applications, present some of the technical challenges we faced in making the integrated systems, and discuss Tangible Reality, the common software foundation the applications are built upon.

Virtual Aircraft Maintenance Training

Cadets learning to maintain US military aircraft receive about half of their hands-on experience using full-scale mock-ups of the various aircraft systems. (The other half of the hands-on training is done on actual aircraft.) These mock-ups take up a lot of space, require substantial logistics for operations and maintenance, are specialized for each aircraft type, and are very expensive. We developed the Virtual Aircraft Maintenance Trainer to explore the feasibility of replacing these expensive mock-ups with less expensive

computer hardware and software. If successful, virtual maintenance training could have substantial benefits, including reduced cost, training in a smaller space, use of standard commercial hardware, and training during the aircraft design phase. In the ideal case, a single VR training system could replace the mock-ups for all aircraft systems for all types of aircraft.

The virtual maintenance trainer we built allows a user to perform the essential elements of two real-world maintenance training tasks performed by the US Marines on the AV8B aircraft, a vertical take-off and landing attack fighter. (You may have seen Arnold Schwartzenegger fly an AV8B in "True Lies".) One task is diagnosis of a failed Radar Warning Receiver System Built-In-Test (RWR BIT). The second maintenance task is adjustment of the Vernier/Non-Linear Nosewheel Steering Linkage. Next we summarize these tasks as implemented on the virtual trainer.

Avionics Task: RWR BIT

The RWR BIT is a self testing function of the avionics system. It is run from the cockpit of the aircraft. Setting up and running the RWR BIT requires the technician to push buttons, throw toggle switches, and turn dials in the cockpit, as well as observe indicator lights and listen for audible tones. During the RWR BIT, an abnormal Threat Light Display on the front panel of the cockpit indicates a failure that needs to be isolated. Diagnosing the RWR BIT failure requires additional cockpit procedures as well as electrical system diagnosis in the rear avionics bay.

To support the RWR BIT, the virtual maintenance trainer includes functional models of the AV8B that allow the user to move about the aircraft, operate cockpit controls, observe avionics system behavior and measure electrical signals with a virtual voltmeter probe. 3D computer graphic models of the virtual trainer include graphic texture maps and accurate geometry to simulate the appearance of the aircraft. Haptic simulations of the cockpit panels, buttons, dials, switches, and electrical pins allow the technician to see, feel, and hear the avionics system operations. Functional models of the aircraft electrical system allow the user to detect and diagnose the RWR BIT failure.

Vernier/Non-Linear Steering Linkage Adjustment

The Vernier/Non-Linear Steering Linkage is a linkage mechanism that is responsible for shifting between fine and coarse steering of the nosewheel as a function of rudder pedal position. The physical device is a four-bar linkage that is located in the nosewheel hatch. The linkage adjustment procedure involves detaching the linkage from its neighbors, manipulating the linkage through its range of motion, measuring the forces required to move the linkage between two hard stops, and adjusting those forces by tightening or loosening a friction nut on the linkage.

To support the adjustment of the nosewheel steering linkage the virtual maintenance trainer includes simulation models of the AV8B that allow the user to move from the exterior of the plane to the nosewheel hatch, observe the nosewheel steering linkage as it is driven by rudder pedal inputs, disconnect the linkage from the steering column, manipulate the linkage, adjust the tightness of the friction nut, and reconnect the linkage. The dynamic simulation of the linkage obeys the kinematic and dynamic constraints of the multi-link system and allows interactive attachment or detachment from the rudder pedal actuator, and joint friction adjustment. The user can feel the shapes of the links in the mechanism, as well as the forces needed to move it.

The technical challenges associated with the virtual maintenance project were related to the graphic and haptic complexity of the aircraft. We used Wavefront Advanced Visualizer to design the aircraft cockpit in the form of general polyhedra. We then created an object loader and generalized contact detection algorithms that allowed us to see and touch these objects with a tool represented by a polyhedron. Even though we did not model all the features of the cockpit, we modeled enough of them that the computational burden of contact detection was significant. We used bounding box algorithms to limit the number of candidate objects that needed to be processed at any one time. The user interacted with virtual objects in this application through a single finger or tool tip. To make the interaction with cockpit features convincing we made haptic algorithms that supported friction, sharp features such as corners, and hard surfaces. We designed our dials to have flat sides in order to make them easier to turn with a single finger. This same technique could have been used to turn the linkage nuts in the nosewheel bay. Instead we programmed the instantaneous removal

or installation of a nut or used a slider on the graphical user interface to determine nut angle. State machine logic tracked the sequence of operations performed by the user in the cockpit to enable certain electronic functions and to trigger visual and audible cues such as flashing lights or tones.

The Virtual Surgical Trainer

The Virtual Reality Anastomosis Trainer is an integrated system that allows users to practice end-to-end anastomosis, a common surgical procedure. The user holds real surgical tools, which are connected to force feedback devices. The user holds an instrumented needle holder in one hand and a forceps in the other, allowing them to manipulate two virtual tube organs and suture them together. The surgical tools are attached to Phantom force feedback devices. The forces of interaction between the tools and the simulated tubes are displayed to the user through the force feedback devices, while the visual images of the interaction are displayed through 3D computer graphics.

The simulated elements in the anastomosis demonstration are two deformable tubes, forceps, needle holder, needle and suture. In a typical suture procedure, forceps are used to grasp and stabilize a tube with one hand while puncturing the tube with a needle and needle holder held in the other hand. The user then releases the needle and regrasps it from the inside of the tube. The needle and suture are then drawn through the tube from outside-to-inside. This process is repeated inside-to-outside on the second tube. The suture is then pulled tight.

The technical challenges of this project were driven by the need for efficient simulation of the deformable tube. We used a spline based representation of the tube to simplify the simulation, graphics, and haptics. We simulated movement of the tubes by modeling the translation and rotation at "control points" of the spline. Local "puckering" of the tubes that resulted from poking or plucking were only represented in the graphics, so the user could not touch these local deformations with the other tool. We used a haptic model for each surgical tool that was comprised of line segments. Rather than model all the interactions needed for tying a knot in the suture, knots are automatically formed when the tubes are drawn together the correct amount.

Tangible Reality

The Virtual Aircraft Maintenance Trainer and Virtual Surgical Trainer we developed are both built upon a common software system called *Tangible Reality*. Tangible Reality includes many of the software elements needed for haptics-based VR systems. It includes support for:

- Dynamic object simulation
- Deformable tube simulation
- 3D object loading
- Fast contact detection
- Contact force calculation
- Dual force feedback device drivers
- Kinematics for force feedback devices
- Low-level force-feedback device drivers for Windows95 and Windows-NT
- 3D graphics and sound on SGI computers

This list can be broken into three broad software groups that apply to all interactive haptic environments: dynamic simulation, haptics algorithms, and 3D graphics and sound.

We use a *simulation generator* to create efficient physics-based dynamic simulations of objects and linkages from high level descriptions of an object's geometry, topology, and material characteristics. Once created, dynamic simulations predict the behavior of virtual objects and insure that they move according to the laws of physics. Physical behavior can help them to appear realistic. Physics-based models are based on the equations of motion for the object and sets of parameters that can be changed to tune behavior. Mass, moment of inertia, compliance, gravitational attraction, shape, friction, and viscosity are among the many physical parameters that we adjust to influence behavior.

Haptic algorithms mediate the exchange of forces between simulated objects and the user. The two most important haptic algorithms are those for detecting when objects are in contact and for computing the forces exchanged when contact occurs. We use several contact detection algorithms that are optimized for real-time performance and that are tuned to the object model type. For example, we use a linear-time Lin and Canny type algorithm for contact between convex rigid polyhedra, and a grid-depth type algorithm for point contact on polyhedra with mild concavities.

To calculate contact forces between objects, we use penalty method techniques. Characteristics such as surface compliance, friction, textures, puncturability, and surface feature resolution are controlled by the contact force model.

The 3D computer graphics we use are relatively conventional, including standard SGI texture-mapped graphics, with stereo presentation and simple triggered sounds.

Conclusion

In this paper we have summarized two examples of integrated haptic training applications, and said a few words about the underlying software system upon which they are built. Our goal at BDI is to build ever more capable training applications that include sight, sound, and touch, while further expanding the foundation of software that supports the development.

Acknowledgments

The Virtual Aircraft Maintenance trainer project was supported by the Joint Strike Fighter (JSF) program, and done in conjunction with Dr. Dave Fowlkes at the Naval Air Warfare Center, Training Systems Division and with Dr. Ken Salisbury at the MIT Artificial Intelligence Laboratory. The surgical anastomosis simulator is being developed as part of DARPA's Advanced Biomedical Technologies Program, and has also received support from Dr. Tom Krummel at the Penn State, Hershey Medical Center, Department of Surgery.

Virtual Surface Modelling with "Loosely Coupled" Force Feedback Device

Juli Yamashita and Yukio Fukui

National Institute of Bioscience and Human-Technology, A.I.S.T., M.I.T.I.
1-1, Higashi, Tsukuba, Ibaraki 305, Japan
E-mail: juli@nibh.go.jp, fukui@nibh.go.jp

1. Introduction

CAD (Computer Aided Design) is one of the most important application fields that require 3D haptic/force feedback. Industrial designers have long been forced to manipulate 3D forms with 2D input/output devices such as mice and CRT displays without haptic feedback, that has given designers uncertain, indirect, and unnatural feelings of shapes they are creating.

The authors have been working on "ViSurf" system, a 3D virtual surface modeler with force feedback, which allows users to deform a B-Spline [Yama93, 94] and triangular polygonal surface directly, without touching control points or vertices. ViSurf has two kinds of force feedback devices; one is a 6 D.O.F. Cartesian type manipulator controlled by velocity and position which has been developed by Dr. Fukui, one of the authors [Yokoi94, 95], and the other is a PHANToM. To control the two different devices in the same way and to bridge the gap of control latency among three processes, which are haptic rendering, object geometry handling, and graphic rendering, this paper proposes a "loose coupling" architecture and a Feature-Based Haptic Rendering Protocol. The architecture separates haptic rendering process from others and they communicate asynchronously using the protocol. As a result, 1700 Hz PHANToM controlling latency has been achieved constantly independent of graphics refreshing rate and communication frequency.

2. ViSurf System

Fig.1 shows the ViSurf system configuration and some specifications of haptic feedback devices. Currently, only one device is available at one time.

2.1 Tool-Based Direct Form Deformation Interface

In ViSurf system, a user deforms a surface just by pushing it with a "virtual tool"; the surface will be deformed as being pushed and you will feel force feedback through the handle of the force feedback device. Several kinds of tools of different deformation effects, such as curving, twisting, and cutting, have been implemented as shown in Fig. 2 [Yama94]. This tool-based interface is very intuitive and easy to understand, since it (1) is independent of form representations. Though the modeler has two different underlying form representations, which are B-Spline surface (of degree 3, a sheet of patch can be handled at one time) and triangular polygonal surface, "push and deform" interface is the same and the same kind of tools give the same (or at least very similar) deformation. In addition, different from existing CAD systems, a user can deform surfaces without touching special "handles," such as control points of B-Spline surface and vertices of polygonal surface. (2) has force feedback. For example, a "cutter" tool will become very difficult to use without force feedback, because the user should keep the tip of the tool on the surface to be cut in 3D space just by visual feedback.

2.2 "Loosely Coupled" Haptic Device and Feature-Based Haptic Rendering

Another unique point of ViSurf system is "loose coupling" of haptic feedback device and object geometry data. Usually, geometry of the virtual world is maintained where it is most

frequently referred; in this case, on the PC controlling PHANToM or the manipulator, since they require 100~kHz order control frequency while visual rendering needs 30~60Hz. Let us call this "tight coupling" of world geometry and haptic device (Fig.3-a). Tightly coupled system has two problems. One is that geometry handling can not always keep high control frequency for haptic rendering; for example, topological change, such as cutting open a surface, requires many memory allocations which take time, and collision detection process becomes slower when the number of geometry objects increases. The other problem is to keep the coherency of the world geometry database at the graphic renderer. Any change of geometry should be sent to the workstation over the EtherNet; when a large number of vertices are changed, it evokes a large network traffic that also slows down the control cycle of haptic device.

To solve these problems, we propose a "loose coupling" architecture of geometry database and haptic device. ViSurf has a loose coupling architecture as shown in Fig. 3(b): The Graphic workstation maintains all geometry data. It receives sequences of 6 DOF data (or cursor trajectory) from haptic device, detects collision between surface objects and the 3D cursor (or the device), deforms surface if needed, returns local surface feature data (such as a collision point and a normal vector there) to the device, and renders surface objects graphically. A haptic feedback device sends it's 6 DOF position and rotation data over the EtherNet to the workstation, either periodically or eventually. It renders surface haptically based on the features. Each feature needs its haptic rendering algorithms which must be "light" enough to keep high control latency.

We named this communication protocol for loosely coupled haptic device that exchange surface features a "Feature-Based Haptic Rendering Protocol." The loose coupling architecture with the protocol has given good results. Although the communication frequency between the PC and the workstation is at most 30 Hz, which is equal to the graphic rendering latency in this case, more than 1700 Hz PHANToM controlling latency has been achieved constantly, which gives very fine feelings of surface shape. Even during a time consuming form manipulation such as cutting, it feedbacks force based on features which are already received and keeps the tip of a "tool" out of surface.

3. Discussion and Future Work

Haptic rendering based on discrete feature information is an approximation of the true form, but its quality can be better than that of tightly coupled system for prepared features, since the device control latency is guaranteed. In addition, the protocol has such merits that robustness, device independence (different force feedback devices can be used in the same way), and independence of form representations (e.g., CSG, solid, surface, etc.).

Currently, no feature, a plane, an wedge, and a corner, have been implemented as surface features, and much more are definitely needed. Especially, features for free-formed surface are indispensable. Interpolating algorithms of features should also be developed. It is a challenging problem to determine a set of features sufficient to represent a certain class of surface features. The protocol should also be extended to support object characteristics other than shape, such as stiffness and texture. We are also aiming at cooperative virtual CAD system over the network.

References

[Yama93] Yamashita, J. and Y. Fukui: "A Direct Deformation Method", Proc. of IEEE VRAIS'93, 1993.

[Yama94] Yamashita, J., H. Yokoi, Y. Fukui, and M. Shimojo : "A Virtual Surface Modeler for Direct and Regional Free Form Manipulation", Proc. of ICAT '94 (The Fourth International Conference on Artificial Reality and Tele-Existence), 1994.

[Yokoi94] Yokoi, H., J. Yamashita, Y. Fukui, and M. Shimojo : "Development of the Virtual Shape Manipulating System", Proc. of ICAT '94 (The Fourth International Conference on Artificial Reality and Tele-Existence), 1994.

[Yokoi95] Yokoi, H., J. Yamashita, Y. Fukui, and M. Shimojo: "Development of 3D-Input Device using Adaptive Control", IEEE International Conference on Neural Networks, Vol.V, pp.2709-2714, 1995.

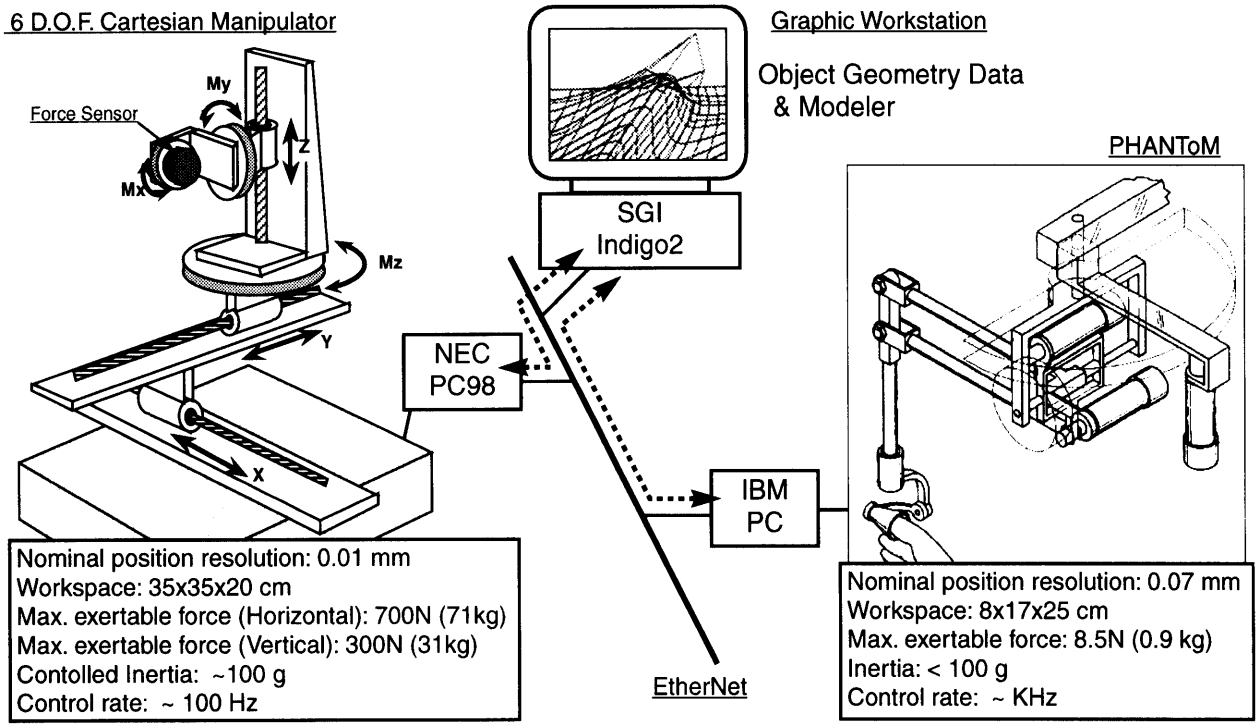


Fig. 1 ViSurf System Configuration The system has 2 force feedback devices which asynchronously communicate with a graphic wokstation over the EtherNet (dotted lines with arrows). Haptic rendering on each PC is done based on shape features sent by the modeler on the workstation.

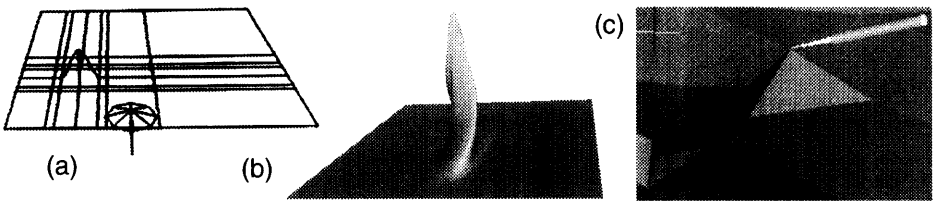


Fig. 2 Deformation of Virtual Surface
(a) A "push" of a small curving tool made a projection. (b) Result of a twisting tool. (c) A cutter tool.

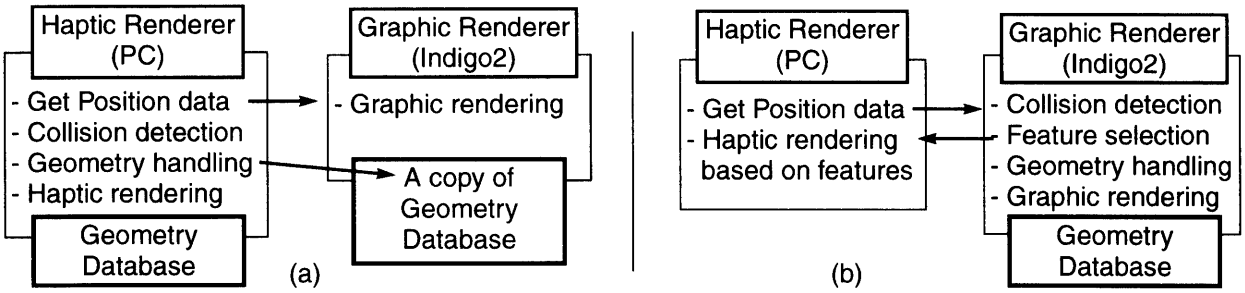


Fig. 3 Load Balancing (a) Haptic rendering and geometry maintaining processes are tightly coupled. (b) Loosely coupled system.

A Virtual Universe Utilizing Haptic Display

Tom Anderson
Sandia National Labs
Albuquerque, NM 87185-0318
Dept. of Electrical and Computer Engineering
University of New Mexico
Albuquerque, NM 87131
email: tganders@unm.edu

Abstract

This paper summarizes a virtual reality universe application in which a user can travel between four virtual worlds through the use of haptic buttons. Each of the worlds demonstrates different aspects of haptic rendering which together create a wide base for force feedback effects. Specifics of the rendering algorithms will be discussed along with possible uses and modifications for other real-life applications.

1. Introduction

Haptics is a growing field in the virtual reality (VR) community, and will continue to grow as its significance becomes more clear. Increases in computing power and more sophisticated haptic interfaces have already shown the value of the sense of touch in VR, and while it is still a relatively new field, its potential is enormous. Haptics can add a new dimension to data visualization, as well as increase one's sense of immersion, and create a more effective human computer interface.

At Sandia National Laboratories, haptic research has focused on creating a new level of interaction with VR environments. The PHANToM haptic interface, a serial, three degree of freedom, force feedback device was incorporated into the EIGEN virtual environment on a Silicon Graphics Indigo 2 Impact workstation for researching haptic rendering. Although the PHANToM has worked well in EIGEN with this research, the principles in this paper apply for other haptic devices and VR environments. A virtual universe application was created to demonstrate the force rendering algorithms, in which four separate worlds could be accessed by pushing haptic buttons. The four worlds (Illusion, Weight, Field, and Poly) and the aspects of haptic rendering that they represent are described below.

2. General Interaction

The virtual universe application begins in a control panel, a virtual wall with four buttons each containing a picture of the world to which it leads. A small sphere, the cursor, is controlled by the PHANToM. The wall is haptically rendered as a normal force that is proportional to penetration depth [2]. The buttons, which are flush, have a different spring constant than the wall and can therefore be moved into the surface. At a threshold value the button force discontinues, and the user appears in the new world. Several first time users have commented that they like the effect of the button's force suddenly stopping, leaving them in the new world.

This general structure works well for demonstrations, yet it could be effective for other applications as it is a convenient way to move between related sets of data. The different worlds could be used to display parts of a larger data set and be easily traversed, or they could contain general information that one would often access from a main world.

3. Friction, Textures, and Intersensory Discrepancy

The first world, Illusion, received its name from an exploration of intersensory discrepancy between our visual and proprioceptive senses, of which it has been shown that vision is dominant [3]. In Illusion, the user can see and touch a sphere inside a box. By changing a slide bar, the graphical representation of the sphere changes to an ellipsoid (lengthened along the x axis) while the haptic

representation remains the same. When the user moves the cursor across the ellipsoid in the x-y plane, the cursor visually moves farther than when the ellipsoid is circled on the y-z plane. Yet physically the user is still touching a perfect sphere. This effect leads many users (though not all) to believe that the sphere actually 'feels' longer as well. The illusion is more effective as the user spends more time feeling the sphere while looking at it. It also increases effectiveness with the introduction of friction and texture. This dominance of sight over touch does not negate the value of haptics, yet by realizing this concept, one might take advantage of people's ability to smoothly incorporate discrepancies among their senses. In some situations, an object can feel different than it looks and still seem real.

After the initial inclusion of friction and texture, a larger variety of different materials was desired for simulation. The friction in these materials is currently accomplished with damping effects that are proportional to penetration depth. In simulating friction, a key concern is stability. Initially, the cursor's current and previous positions were projected to the sphere's surface (or any object's surface in a general case) and a vector was created from subtracting the projection of the previous point from the projection of the current point, which gave the direction of the friction. Stability was then achieved by averaging this vector over a history.

The textures were created by modulating the surface of the sphere with a combination of several sine waves and several square waves of differing amplitudes and frequencies. Then the materials were described with combinations of friction and texture that were intuitively created and fined tuned until things 'felt right'. The materials created include wood, both polished and rough; sandpaper; cobblestone; a magnetic material; water and a thicker viscosity fluid; rubber; plastic; and a sticky material.

There are several concepts of interest in creating these materials. In materials that were not supposed to be compliant, the visual image of the cursor was kept from entering the sphere (because the force is proportional to penetration depth, the cursor partially moves into an object it is touching) which made it seem harder. This is an example of a use for the intersensory discrepancy described above. With the magnetic material, the magnetic attraction needed to approach zero as the cursor moved towards the sphere for stability reasons. The sticky sphere was created by adjusting the friction to have an inward radial component. The water sphere consisted of only a damping term (the cursor could feely move through the sphere). It was enhanced by using alpha blending, so it was transparent which added to its realism. Finally, in a method similar to that of the sticky sphere, a damping term applied in the outward radial direction added to the spheres' stiffness when desired.

4. Dynamics

The second world, Weight, is an application in which the cursor is attached to a weight by a spring, visually modeled as a sphere on a rubber-band. The directions of the forces on each are computed by subtracting their center coordinates and normalizing the resultant vector, and the magnitude is computed from their distance apart. A gravity term is added to the force on the weight. The motion of the object is modeled from physics, and thus, given the forces, the acceleration is found from $\mathbf{a} = \mathbf{f} / m$, where $\mathbf{a}$ is a vector representing acceleration, $\mathbf{f}$ is a vector representing force, and m is the mass of the object. Velocity and position are then found by integrating acceleration over time, using a chosen time value in the equations rather than finding it from the servo rate. A virtual floor was then added on which the ball could bounce. The sound of this collision was presented in which the volume was determined from the ball's velocity.

Taking advantage of the freedom in choosing the time variable lead to an interesting result. By decreasing the value of only the time variable, the object not only moved slower but also felt more massive and felt as if it were moving through a viscous fluid. This effect, for example, might be used in combination with damping to create the effect of a clay material in an object modeled by point masses (i.e. vertices on a polygonal object). One might also take advantage of the fact that the force on the cursor does not necessarily need to be the same force applied to the mass. This could be used to change how things feel, while maintaining a physics model.

Of the four worlds, Weight requires the least amount of programming code, yet it is often people's favorite application. Users can feel the momentum and inertia of the sphere, both of which occur naturally from the motion equations described above, which is a powerful effect.

5. Vector Field Representations

The third world is called Field and contains a mapping from a vector field to the forces felt by a user. The vector field, electric potential, is created from virtual point charges which a user can place or delete to make any arbitrary formation. Then the electric potential, which is referred at infinity, is mapped onto forces that the user can feel. This allows a user to search the field and find places where the individual charges cancel, thus leaving 'weak' spots where the cursor can explore. One should note that a user is not feeling actual forces created within the physics, but instead is feeling a mapping of data into forces. Because the user is feeling an abstraction, this can be a powerful tool in understanding various vector fields or even other types of data.

Another feature of this world is the interface of the PHANToM to the EIGEN craft's movements. The cursor is in a box, similar to that in Illusion, except that when the cursor touches one of the walls of the box, the ship moves appropriately. This is an intuitive form of navigation and also works well with general exploration because the user can easily change between a craft movement and haptic exploration. A user would move the craft until an object was inside the box, and then would have haptic access to it. This makes large areas available for interaction in fine detail.

6. A Polygonal Force Representation

A main focus in the haptic work at Sandia National Laboratories was the incorporation of force feedback into already existing technology. Because applications were previously mainly visual, this incorporation of haptics was greatly encouraged by methods that would work well in combination with already defined graphical methods. Therefore, an algorithm was developed in which the user could touch arbitrary polygonal data sets [5], i.e. surface defined objects which are common in graphics.

This algorithm creates a force that is normal to the currently touched polygon, and interpolates over edges as the cursor moves to different polygons. The magnitude of this force is defined by penetration depth which presents a visualization issue. The cursor can be lost within a compliant object. To resolve this issue, a 'Bendable Polygon' algorithm was used in which the current polygon was divided into six polygons, and neighboring polygons were also split to make the object remain seamless as shown in Figure 1. When Gouraud shaded, this technique works well and can lead to lower levels of detail, as objects can then be described by larger polygons.

The vertices of the object are then attached by springs to a framework consisting of base points from which the graphical vertices can be pulled away. With the inclusion of additional springs among the vertices and to the cursor, the object is deformed as the cursor pushes into it. The friction and textures can be added for material properties. As polygonal data sets increase in size, windowing of the local area can decrease computational time and make them renderable in real time.

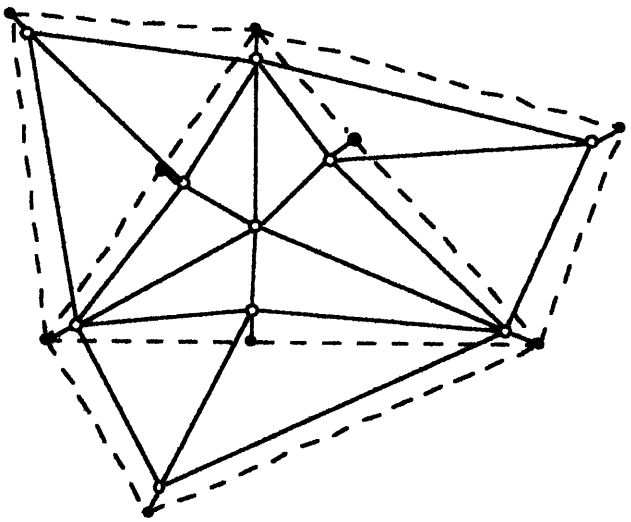


Figure 1: Graphical deformation in the polygonal representation of forces. The dashed lines represent the object while it is not deformed, and the solid lines represent the polygons that the user sees after it is deformed. Filled in circles are base points and open circles are graphical vertices, which are connected by springs.

7. Conclusions

The four worlds represent a base for the haptic rendering that will soon be applied to other real-life applications. In addition to finding applications for the concepts described and finding the most effective methods of haptic exploration, there are continued efforts to increase the base of haptic

effects. Work is currently being done on developing a more effective friction models and texture creation. Combinations of visual, haptic, and audio effects combined will continue to be explored. More advanced user interfaces, such as a haptic control panel that moves with the ship, similar to that in an airplane, is being researched. Such an interface would include switches, buttons, etc., and could possibly replace the mouse completely. The polygonal method of force representation is being expanded to work with larger data sets, and other surface modeling techniques are being explored. Overall, the modeling that has been done has shown the potential of haptics in the virtual environment, and these successes are expected to continue.

8. References

[1] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics, Principles and Practice*, Addison-Wesley Publishing Co., Reading, MA, 1990.

[2] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles, "Haptic Rendering: Programming Touch Interaction with Virtual Objects", ACM Symposium on Interactive 3D Graphics, Monterey, CA, 1995.

[3] R. Welch and D. Warren, "Immediate Perceptual Response to Intersensory Discrepancy", Psychological Bulletin, Vol.88, No.3, pp.638-667, 1980.

[4] T. Anderson, "A Polygonal Method for Haptic Force Generation", submitted to IEEE / Virtual Reality Annual International Symposium, 1997.

[5] SensAble Devices Inc., "The PHANToM" literature from SensAble Devices Inc. 225 Court St., Vanceburg, KY 41179.

[6] C. Maples and C. Peterson, "MUSE, A functionality-Based, Human-Computer Interface", The International Journal of Virtual Reality, Vol. 1, No. 1, pp. 2-9, Winter 1995.

[7] P. Buttolo, B. Hannaford, B. McNeely, "Introduction to Haptic Simulation", Tutorial 2A, IEEE/VRAIS tutorial notes, March, 1996.

[8] R. Klatzky and S. Lederman, "Toward a Computational Model of Constraint-Driven Exploration and Haptic Object Identification", Perception, Vol. 22, pp. 597-621, 1993.

[9] M. Minsky, M. Ouh-Young, O. Steele, F. Brooks, and M. Behensky, "Feeling and Seeing: Issues in Force Display", Computer Graphics, Vol. 24, No. 2, pp. 235-243, 1990.

[10] F. Brooks, M. Ouh-Young, J. Batter, P. Kilpatrick, "Project GROPE - Haptic Displays for Scientific Visualization", Proceedings of SIGGRAPH '90, Dallas, Texas, August 6-10, 1990.

9. Acknowledgments

I would like to thank George Davidson and Arthurine Breckenridge for their support on the research leading to this paper.

Synthetic Force Feedback Projects at the U. of Virginia.

George Williams and Dennis Cosgrove

This paper reviews the projects at The User Interface Group at the University of Virginia surrounding the Phantom force feedback device. The first portion of this paper relates our implementation of an integrated graphics and force feedback development environment. The remainder focuses on projects underway using this environment.

An Interactive Development for Haptics

When we first received a Phantom device, we sought to integrate it into Alice, an interactive, 3D graphical development system for the desktop PC developed here at the University of Virginia. Alice is a system for quickly prototyping 3D graphical scenes and scripting simple animations for 3d objects(1). We have benefitted from an Alice front-end for creating haptically-able applications mainly because the system allows us to interactively develop and debug such applications with rapid turnaround time.

A Word About Alice

Alice is a 3D interactive graphics programming environment. The goal of Alice has been to make it easy for novice programmers to develop interesting 3D environments and to explore the new medium of interactive 3D graphics. Alice is primarily a scripting and prototyping environment for 3D object behavior, not a 3D modeler. The primary way of interacting with the objects in the scene is through writing short scripts in an interpreted language. As scripts are executed, users can manipulate objects with a mouse, move the camera around, and make objects react to the mouse and keyboard.

The scripting language for Alice is called Python. Python is an easy-to-learn interpreted and object-oriented language(2). The Alice API supplements the core language of Python with a set of methods and classes that control the graphics hardware. The user creates and manipulates graphical objects in Alice by typing simple commands in a Python command-line shell. The following line-by-line script might represent a typical Alice session, (comments follow the '#':

```
>> cube=AnObject('cube')  #make an object of class 'AnObject' and make
                          #it look like a cube
>> cube.Move(Left,1)    #move the cube to the left by 1 unit
>> cube.Turn(Right,90,1)  #turn it to the right by 90 degrees over 1 second
>> foobar=ASequence( cube.Move(Left,1,1), cube.Turn(Right,90,1) )
         #move it to left over a second and then to the right over a second
```

The cube is rendered and the animations are played out in a separate graphics window as the user is typing in the interpreter.

Alice and Haptics

Alice is a powerful system for creating interactive 3D graphical environments.

One of its more powerful features is its extensibility. External hardware I/O devices, such as tracking systems and, here, the Phantom, have been easily integrated into the system. Programmers can write fast low-level C or assembly code and wrap the functions in Python and make them visible through the interpreter. A programmer can hide the low level details of the implementation while providing a easy-to-use abstraction to the user.

The architecture for our particular haptics-integrated Alice system is simple and reflects partly the design of the Armlib service developed at UNC(3). A Phantom servo loop process attends to the Phantom and runs on one machine (a 166MHz PC), and the Alice rendering/animation process runs on another PC. Both keep a separate database of the objects in the simulation, but maintain synchrony when necessary by communicating via an ethernet connection and TCP/IP. Again, programmers develop applications through the Alice API. As before, normal graphical objects are instantiated from the 'AnObject' class as in the example. Haptic objects are created from the 'AFeelable' class, a new class (derived from 'AnObject') provided to support haptic objects. Haptic objects inherit all the methods of graphical objects as well having haptic properties. In the above example, substitution of 'AnObject' by 'AFeelable' yields the same graphical display, with the addition of the cube presented haptically by the Phantom device.

'AFeelable' extends 'AnObject' with methods that can modify what we call the properties of *inertia* and *magnetism.* The amount of effort needed to move certain objects with the Phantom can be changed interactively by changing the *inertia* property of a haptic object. Setting the *inertia* to maximum renders the object immovable. Objects can also 'attract' the Phantom probe point as well. This is achieved by setting an object's *magnetic* property temporarily or until the probe point has collided with the *magnetic* object. We don't currently support any notion of friction or gravity, although these are useful properties for haptic objects.

Works in Progress in Force Feedback

Enhancing the Worlds-In-Miniature Metaphor with Force Feedback

The WIM, or Worlds-In-Miniature interaction technique, was a technique developed here at The University of Virginia exploring a simple interface for interaction, object selection, object manipulation, and navigation in graphical, immersive virtual environments(4). Using the WIM technique, an immersed user (head-tracked with head-mounted display) interacted with the virtual environment through a small hand-held miniature, a scaled-down version of the life-size virtual environment. A user could move objects within this life-size environment by selecting and manipulating the object's WIM counterpart. In the real world, participants are, of course, not holding real scaled-down miniatures, but are using hand-held, passive props.

We are looking at using the Phantom device to enhance the hand-held props by imparting haptic properties to the virtual objects in the miniature. This should add an entirely new dimension to the interaction technique in many ways. With haptics, miniatures can possess a weight or inertia, reflecting in some manner how hard the life-size object would be to manipulate. Also, haptically-able miniatures could support some notion of how objects associate,

potentially improving on user perception of object associations(5) through visual feedback alone. For example, consider a furniture-moving task using the WIM metaphor. We might wish to associate a chair with the floor, so that the chair can move anywhere in the room, but must slide along the floor to do so. With haptic feedback through a WIM, a user realizes immediately that the chair can only slide along the floor as it is pushed. In addition, when the miniature chair hits a wall in the miniature room, the chair, as well as the hand pushing it along, stop. Visual techniques alone supporting these associations and constraints can seem awkward.

A truly immersive and bi-manual technique for exploring these ideas is not yet possible due to the physical constraints of a force-feedback device like the Phantom. Nevertheless, we are working on a non-immersive condition using the aforementioned Alice development system.

BI-MANUAL TASKS AND HAPTIC FEEDBACK

Many tasks that one might want to mimic using synthetic haptic feedback are bi-manual in nature. Bi-manual tasks require both hands to work in concert but asymmetrically. Often, in these tasks, the less dominant controls the coarse component of the task, providing the reference frame for the dominant hand, which performs the detailed part of the task(6). Writing words on paper is a classic example; as one hand forms the intricate letters, the other hand constantly shifts and adjusts the paper underneath. The WIM metaphor is a tactile, bi-manual interaction technique using passive props.

For some tasks, it may be enough to have passive props. For other tasks, true-to-form haptic feedback may be required for both hands (for example, unscrewing the lid on a jar.) Still, other tasks may warrant haptic feedback, but only in one hand. In such tasks, the dominant requires the true-to-form haptic response, while the other hand does not, but retains control over the task's reference frame. In this way, only one haptic device, held by the dominant hand, may be required.

We are exploring this idea of haptic asymmetry in bi-manual tasks by mimicking the task of an artist hand-painting an object. In our version of the task, a virtual object is loaded in the integrated graphic-haptic environment and is logically attached to a tracking device. The tracker is held in the less dominant hand. As the hand rotates or translates the tracker, the tracked object in the scene moves faithfully. The tracker in no way resembles the object rendered, but merely acts as an imprecise prop for it(7). The prop is then moved within the vicinity of the Phantom work space. The right hand, using the force feedback device, then probes the object. The right hand "feels" the object through the Phantom simulation of it; the left hand of course does not, feeling only the prop.

We wish to eventually implement a simple painting program where users can actually paint in the texture space of the surfaces of the haptic objects. We can them compare this interface with commercially available applications for hand-texturing objects through a 2D graphics window using a mouse. We have done enough work in this area to offer the following observations:

1) It's not as strange as one might think.
2) Users quickly get tired holding the tracker.

3) Some users report 'sensing' haptic response in the hand not receiving simulated haptic force when the other hand receives the simulated haptic feedback.

To alleviate the problem in #2, we tried a technique where the haptic object's center could be rigidly attached in space relative to the Phantom, yet the rotations of the hand-held tracker register with the object. Thus, users could hold the tracker anywhere in tracking range while also controlling the rotational reference frame of the task.

This research in haptic asymmetry is exploratory and is a work in progress. We hope to eventually understand more deeply the limitations and strengths of this haptic interaction technique.

References

(1) "Alice: A rapid prototyping system for interactive 3D graphics," UIST '96.
(2) Python language reference @ http://www.python.org.
(3) ArmLib by anonymous ftp@ ftp://ftp.cs.unc.edu/pub/packages/GRIP/armlib/ , The University of North Carolina.
(4) "Virtual Reality on a WIM," Stoakley, Richard, SIGCHI '95.
(5) "Object Associations," A Simple and Practical Approach to Virtual 3D Manipulation, Bukowski and Sequin.
(6) Guiard, Y., Asymmetric division of labor in human-skilled bimanual action, Journal of Motor Behavior, 19, p.486-515.
(7) Hinckley, et.al., "Passive Real-World Interface Props for Neurosurgical Visualization", ACM CHI'94, pp.452-458.

Special Thanks:
I'd also like to thank Sensable Devices for their help in getting us set up with our Phantom!!

Force Feedback in Interactive Dynamic Simulation

Sundar Vedula and David Baraff
The Robotics Institute
Carnegie Mellon University
{vedula,baraff}@cs.cmu.edu

Abstract: *It is well understood that force feedback coupled with visual display can be an important two-way communication channel in human-computer interaction. In this work, we present ways of integrating force feedback with dynamic virtual worlds that evolve in time based on physical laws. The Phantom™ haptic interface is used for force feedback, and is integrated with existing dynamic simulation algorithms running on a separate workstation, so that objects can be manipulated in real time and the corresponding forces felt back by the user. Various issues involved in communication between the processors, interpolation, and friction are discussed.*

1 Introduction

In recent years, there has been increased interest in virtual environments that evolve *physically*, or based on forces that act on objects that make up the environment. In this work, we explore techniques and issues involving integrating a force feedback device based on point interaction, such as the Phantom™ with a physically based dynamic simulation system. The idea is to be able to build a framework that enables users to make virtual environments by giving objects the desired physical properties (mass, material, shape, etc.) and then being able to interact with them - pushing them around and actually feeling the forces that one would feel if real objects were pushed around with a tool.

Recent work on dynamic simulation of non-penetrating rigid bodies by Baraff[3], Mirtich and Canny[6], has shown that it is now possible to simulate the physics of interacting rigid bodies in real time. We use a simulation system based on [3] to build a virtual world with interacting bodies, that handles collisions, contacts, and second order dynamics between rigid, polyhedral objects. Given the set of external forces on the system, the dynamics equations are incrementally integrated to yield the new state, backtracking when collisions occur. Constraints are modeled as constraint forces, and friction is implemented as Coulomb friction.

Srinivasan and Salisbury [7] provide a good description of current issues and challenges in haptic feedback. Mark et. al.[4] provide a general overview of issues and solutions to problems in adding force feedback to graphics systems, although their discussion is limited to static models. Adachi [1] addresses the problem of haptic display of curved surfaces using an intermediate representation, which we use as the base for our local kinematic update model (although not for curved surfaces). Minsky et. al. [5] address various techniques for surface texture display, and provide a good overview of control methodologies suitable for haptics.

2 Client Server model

2.1 Motivation

While update rates of 25 to 30 Hz are sufficient for visual display, it is well known that stable force display requires force updates at least 1000 times a second. Given the complex computation that dynamic simulation involves, it is clear that we need to decouple the dynamic simulation of the virtual world from the low-level force control loop. Presenting a model of the world neighboring the location of the virtual probe evolving in time to the force control program would be the ideal solution. The idea is similar to that of clipping in computer graphics, where the subset of the model outside the viewing volume need not be rendered. There is also the inherent advantage that the feel of the model at the point of interest is independent of its complexity at places far away from there.

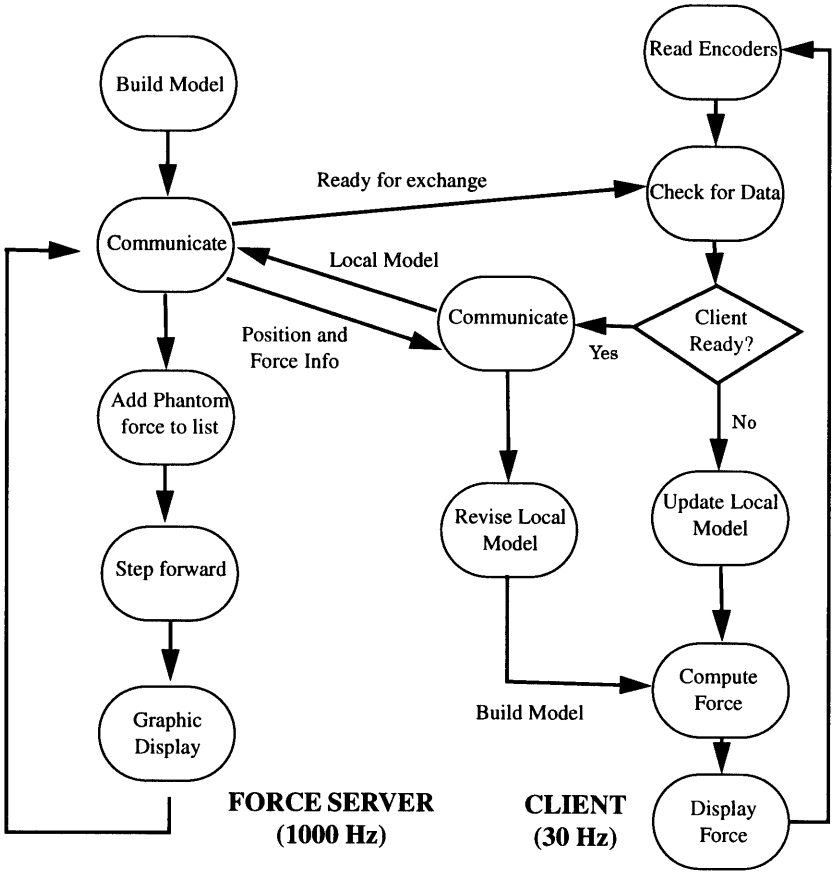


Figure 1: Our Client-Server model

2.2 Local Model

Our model handles the above problem by having a force *server*, that queries the haptic device for position and commands forces to it, and an application *client*, that models the evolution of the virtual world in time based on forces that act on the individual bodies and constraints on their motion (in short, runs the dynamic simulation). The client, as soon as it is ready, queries the force server for the position of the probe, computes an appropriate local model, and sends it over to the force server. It also receives information on the forces that the user tried to exert on the object(s) in the model, so that these may be integrated during the next time step. An inherent advantage of this setup is that it is trivial to have multiple devices and simulations interacting with each other in various combinations. The sequence of computation is as shown in Figure 1.

The amount of information communicated to the force server is a big factor in the performance of the system. One extreme is giving it knowledge of the entire geometry of the world, which causes a huge degradation in the haptic servo rate, and is obviously unnecessary. The other is to compute by simulation, forces that a body such as the probe would feel if it were interacting with other bodies and just send those forces to the controller. Our approach tries to combine the advantages of a limited geometry model while being able to passively control the device (that is not have it driven to a "desired" position all the time).

The actual force displayed is computed using a spring model, where it is a linear function of the penetration into the nearest plane that is a part of the local model. The user now needs to exert the negative of this displayed force to the probe to keep it from flying away (which he/she automatically will if the probe is being held firmly). To give the user the feel of "pushing" virtual objects, we apply this same force to the object being pushed, and then have it respond appropriately one time step later. It is important that this force be scaled suitably because of the different time scales on both processors. The delay is only visual and goes unnoticed. So rather than trying to measure the force that the

user is pushing with, we actually force him to do so, and then add an equal but opposite force into the list of external forces that the simulator has to integrate over the next time step.

2.3 Local Model Schemes

We have investigated three schemes for a local model that is sent to the force controller. The first is that of a simple plane that is closest to the probe, as investigated in [1]. While this works for static objects, objects that are moving cause a force discontinuity and therefore a nasty clicking sensation at the frequency at which their position is updated, which is about 30 Hz for our system. In the second scheme, velocities of the bounding points of the plane are sent across to the force server each time (two end-points for an edge in 2-D simulation), and the position interpolated appropriately. This works fine for objects moving linearly or those that have large moments of inertia and are hard to spin (the worst case is when the object rotates about one of the two edge points).

In the third, we use a complete kinematic model for the intermediate representation on the force server between the dynamic updates. In 2-D, the motion of any plane can be fully described by its angular velocity about the center of rotation, and the linear velocity of the center of rotation itself. For a freely moving dynamic object, the center of rotation corresponds to the center of mass, therefore the position of the plane closest to the probe in local co-ordinates, along with the world-space position, velocity, and orientation of the center of mass of the object would be a complete kinematic model of the motion of the plane. With this local model for force display, the problems mentioned in the previous section are eliminated, and we have a stable force-position relationship.

Although more sophisticated approaches such as a true dynamic model with local force integration are possible, we feel that the overhead increase is not justified in our case given the well-behavedness of the kinematic update model. This would be a problem in systems where sudden, high forces are exerted, or if the dynamic simulation is too slow, causing large changes in velocity each time step.

3 Surface Friction

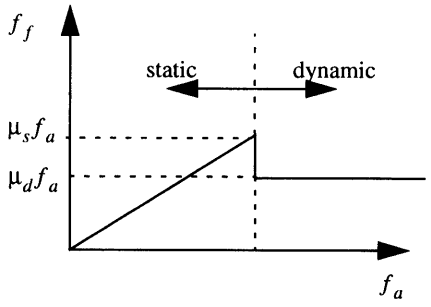


Figure 2: Coulomb variation of frictional force f_f with applied force f_a

Manipulating dynamic objects without friction feels like pushing ice cubes on a floor with a pen. We have implemented a Coulomb friction model, treating the tip of the probe as a small finite plane that rubs on other surfaces. Coulomb friction states that frictional forces always arise to oppose impending (static friction) or actual motion (dynamic friction), with the variation as shown in Figure 2. The main problem with modeling friction on a position sampled device such as the Phantom is that without knowing the applied force (no force sensors), there is no direct way to guess the direction of impending motion, and therefore to compute the frictional force. However, if we know that the probe is being moved, we know that a force has to be applied opposing the direction of motion. This makes dynamic friction easy. Static friction is implemented as a stiff spring about the point of penetration into the surface (or a point when the velocity goes below a threshold), but this still is relatively noisy because of the fact that any velocity estimate breaks down at when movement is very slow. Another problem is the fact that there is often frequent switching between static and dynamic friction modes. In spite of this, the addition of friction adds a great deal of manipulation capabilities to the otherwise slippery probe.

4 Implementation Issues and Results

We use the 1.5X Phantom^{TM} interface, controlled by a 200 MHz Pentium Pro PC running Linux. Dynamic simulation runs on a 150MHz SGI R4400. On a discretely sample device like the Phantom that has no tachometers or accelerometers, velocity can only be computed by differentiating the position signal, which is highly noisy. We use a simple exponential filter on the velocity that uses the last 100 samples to compute an estimate (equivalent to a lag of about 10 milliseconds) Although this, like most other approaches produces slightly noisy data for velocities close to zero, it works reasonably well otherwise. Client-server communication is achieved using non-blocking TCP/IP sockets over an Ethernet network. A useful observation was that it is important to transfer all information with one socket read/write call, to avoid a nasty degrade of performance on multiple socket calls while waiting for a round-trip packet. The degradation would be a lot lesser with UDP, at the expense of reliability.

We have achieved a haptic servo rate of over 10kHz on the force servo, which is much faster than real-time requirements for the Phantom. Dynamic simulation is in 2-D, while haptic and graphic display are both in 3-D. TCP/IP communication consists of about 120 bytes of information sent back and forth between the two workstations once every simulation loop, which happens about 30 times a second. This is also the update rate of the graphics display, giving a smooth animation. Thus the graphics loop is real-time, while the force feedback loop runs at well over real-time. The feel is similar to that of pushing objects around on a smooth plane with a pen (not quite one's finger because of all the tactile force information missing).

5 Conclusions

We have established a framework and shown that it is possible to integrate force feedback with physically based dynamic simulation and graphics using a client-server based system. This framework makes it trivial to connect multiple devices to one hardware system, and thus have distributed users manipulating the same virtual environment. Future work involves design of better interfaces for 3-D interaction, and different control strategies (including active ones that actually drive the device). We also see many interesting issues in the use of such a system to perform fine manipulation tasks, possibly using more than one haptic interface in tandem.

6 Acknowledgments

This work was funded in part by an Office of Naval Research YIP award and by Interval Research Corporation. Most of the work on friction was done while one author (Vedula) was at Interval, and he would like to thank the haptics group there for many useful discussions.

7 References

[1] Adachi, Y., Kumano, T., Ogino, K., "Intermediate Representation for Stiff Virtual Objects". Proc. IEEE Virtual Reality Annual International Symposium, pp. 203-210, 1995.

[2] Bailey, M., Johnson, D., Kramer, J., Massie, T., "So Real I can almost Touch It: The Use of Touch as an I/O Device for Graphics and Visualization". Siggraph Course Notes #37 (New Orleans, Louisiana), 1996.

[3] Baraff, D., "Interactive Simulation of Solid Rigid Bodies". IEEE Computer Graphics and Applications, 15:63-75, 1995

[4] Mark, W.R., Randolph, S.C., Finch, M., Van Verth, J.M., Taylor II, R.M., "Adding Force Feedback to Graphics Systems: Issues and Solutions". Computer Graphics(Proc. SIGGRAPH) pp. 447-452, 1996.

[5] Minsky, M., Ouh-Young, M., Steele, M., Brooks, F.P. Jr., Behensky, M., "Feeling and Seeing: Issues in Force Display". Computer Graphics (Proc. 1990 Symposium on Interactive 3D Graphics) pp. 235-243, 1990.

[6] Mirtich, B., and Canny, J., "Impulse-based Simulation of Rigid Bodies". Proceedings of 1995 Symposium on Interactive 3D Graphics, April 1995.

[7] Srinivasan, M.A., and Salisbury, J.K., "Chapter 4: Haptic Interfaces", in Virtual Reality: Scientific and Technological Challenges, Eds: N.I. Durlach and A.S. Mavor. National Academy Press, Washington D.C., 1994.

SPI Haptics Library

Wendy Plesniak, John Underkoffler
MIT Media Laboratory
wjp@media.mit.edu jh@media.mit.edu

Abstract

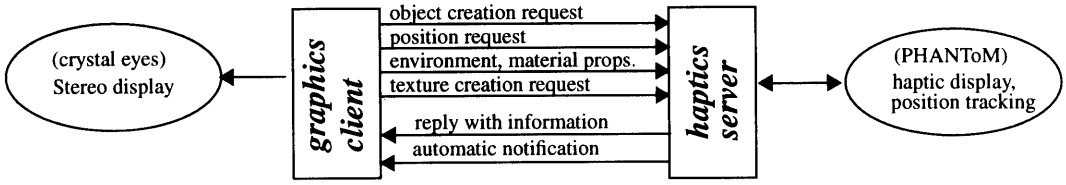
For haptics applications in which the PHANToM haptic interface is controlled by a PC, we have built a network-based client-server system so that high-quality computer graphic rendering can accompany haptic simulation. The system currently consists of a 586 (Pentium) PC to drive the PHANToM and an SGI Indigo Elan to render stereo computer graphics. The machines communicate over Ethernet using TCP/IP.

Position query and object creation commands are sent from the graphics host to the PC running the haptics simulation. Informational replies and automatic notifications are sent from the PC back to the graphics host. Thus the PC-based haptics library is made available to applications running on higher-end graphics workstations.

The haptics library supports the creation and deletion of a wide variety of primitive objects, textural and bulk material properties so that rendering a haptic scene is accomplished by making simple calls from a client program. Currently the system simulates static models, and dynamic simulations are being developed.

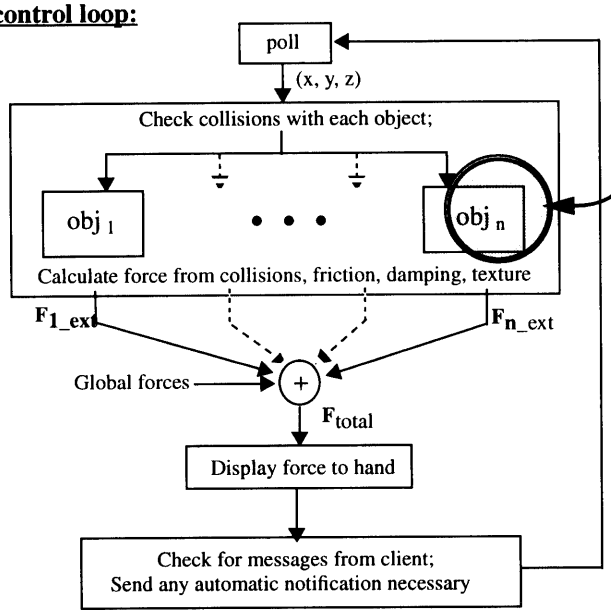
Haptics client-server

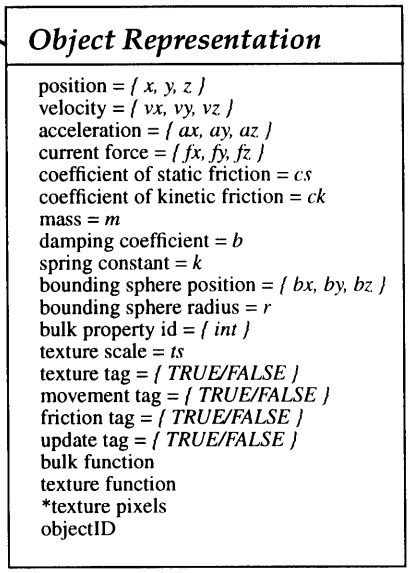
Our network-based client-server system has been developed for applications demanding both high quality haptic and visual rendering. The haptics server recognizes commands from a client graphics machine for object creation and deletion, activation and deactivation, setting global drift forces and global viscosity, specifying surface damping and friction (static and kinetic), creating textures on objects, and for position-query. In turn, the client receives confirmations of successfully-completed commands, requested position information, and automatic notifications when buttons tagged as "active" have been touched. This system, which models a world of static objects, has proven to be robust; the only noticeable visual latency occurs when network traffic is heavy. The basic system is shown below.



The haptic control loop, which accommodates a servo-loop of about 4000 pts/sec, is constructed as shown below:

Haptic control loop:





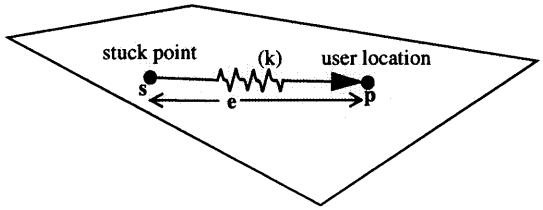
Within each cycle, the location of the stylus is reported, and its collision with all objects is determined. If the stylus is in contact with any object, the force in the direction of the object surface normal is computed as well as forces from specified surface properties (friction, damping, texture). Force contributions from all objects are added to global drift or damping forces, and the result is displayed to the user. Object state, as shown above, is updated as necessary. Any requests from the client are then addressed before position polling repeats.

There are obviously many ways to model the surface and bulk properties of materials, and the computation of forces generated when we interact with them. Because the only directly measured state provided by the PHANToM is position, indirect methods must be used for computing physical quantities involving velocity and acceleration. Below, we provide a brief description of some simple modeling which has yielded satisfying simulation of some haptic phenomena.

Friction and damping modeling

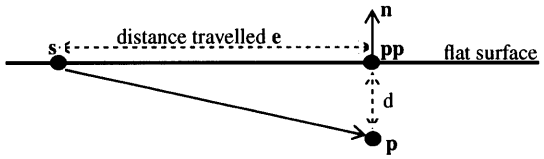
Frictional force, $\mathbf{F}_f$, can be approximated as follows; a "stuck" point, **s**, at which a user first contacts a surface, or the point at which velocity along the surface becomes zero, is recorded. We model a second surface (with stiffness k) at this point, oriented so that its normal opposes the user's direction of motion. Static friction is modeled as the resisting force required to "break through" this second surface, before being able to continue motion across the real surface. After sufficient force has been applied to overcome static friction, movement is permitted and **s** is dragged along. Then, kinetic friction is modeled as a smaller opposing force whose magnitude depends on the amount of normal force, $\mathbf{F}_n$, exerted by the user.

Friction modeling on a surface



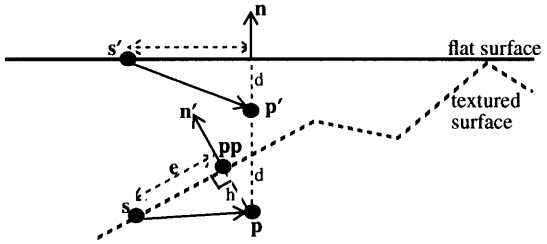
Thus, if $k\mathbf{e} < (c_s |\mathbf{F}_n|)$, then $\mathbf{F}_f = k\mathbf{e}\mathbf{f}_t$; else $\mathbf{F}_f = (c_k |\mathbf{F}_n|)\,\mathbf{f}_t$, where c_s is the coefficient of static friction, c_k is the coefficient of kinetic friction, and $\mathbf{f}_t$ is the unit vector in the direction of the tangential component of force. The stiffness coefficient, k, of the "resisting" surface is currently not user-specified. Computation of the distance travelled, **e**, along a flat surface, textured surface, or sphere is shown in the diagram below.

Computing the distance travelled on flat surfaces



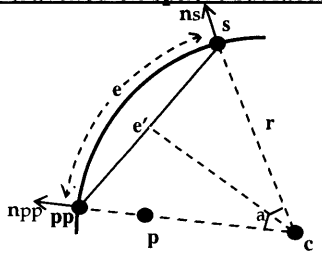
$\mathbf{d} = (\mathbf{p} - \mathbf{s}) \cdot \mathbf{n}$
$\mathbf{pp} = \mathbf{p} - d\mathbf{n}$
$|\mathbf{e}|$ = distance travelled in surface = $|\mathbf{pp} - \mathbf{s}|$

Computing the distance travelled on textured surfaces



$\mathbf{h} = d(\mathbf{n} \cdot \mathbf{n}')$
$\mathbf{pp} = \mathbf{p} - h\mathbf{n}'$
$|\mathbf{e}| = |\mathbf{pp} - \mathbf{s}|$

Computing the distance travelled on sphere surfaces



$\mathbf{e} = |\mathbf{r}|a$
$a = 2\text{asin}\{\ |\mathbf{e} - \mathbf{s}| / 2|\mathbf{r}|\ \}$
$|\mathbf{e}| = 2|\mathbf{r}|\ \text{asin}\{\ |\mathbf{e} - \mathbf{s}| / 2|\mathbf{r}|\ \}$

or, **e** can be approximated with **e'**:
$|\mathbf{e}'| = |\mathbf{pp} - \mathbf{s}|$

Damping within surfaces can also be computed according to the user-specified damping coefficient, b. We simply record five most recent points along the stylus trajectory within a surface and use them to compute an approximate velocity, $\mathbf{v}_{avg}$. From this information, the damping force, $\mathbf{F}_d = b\mathbf{v}_{avg}$, is computed and added to $\mathbf{F}_n$ and $\mathbf{F}_f$, to form the total force that a user is exerting on an object.

Texture simulation

We currently specify object textures in two ways, either algorithmically, or by using a greyscale image (a 2D array of pixels). The texture simulation procedes in the following manner: if, upon polling, the stylus is found to be in contact with a texture-mapped surface, then either a function is evaluated or a texture map is indexed to indicate the surface relief at the current sample position. A displacement value ranging from {0 - texture scale} is added to the current position which permits the stylus to continue to sink into the surface unperturbed for that offset distance. Both texturing methods are outlined below.

Algorithmically-defined textures:

If sample point collides with textured object:

map currently-polled sample (x1, y1, z1) to (u, v, z2);
evaluate texture function at (u, v) and four surrounding points;
vector cross product to approximate new normal (ntx, nty, ntz);
inverse-map (u, v, z2+offset) back to (x1', y1', z1');
replace original normal with (ntx, nty, ntz);
compute forces with new position and normal..

Image-defined textures:

If sample point collides with textured object:

map currently-polled sample (x1, y1, z1) to (u, v, z2);
index texture map at points around sample point;
interpolation yields a height, offset, at (u, v);
approximate new normal (ntx, nty, ntz);
inverse-map (u, v, z2+offset) and (ntx, nty, ntz);
compute forces with new position and normal.

Recent and future work

The server design reported above does not permit reliable network-based simulation of inter-object dynamics and thus does not permit the user to rearrange elements in the scene by physically interacting with them. We are currently redesigning the system to provide a more complete simulation of object dynamics (to be displayed both visually and haptically.) The revised design permits objects to be created and to interact with each other and with the user. Objects like spheres, planes, cubes and other obstacles can be tagged as fixed or moveable, and this status can be changed as the application runs. During the inter-object collision detection stage, the bounding sphere of each object tagged as moveable is checked for intersection with those of other objects. This operation requires $(N^2 - N)/2$ simple compares per loop, where N is the number of movement-tagged objects. Objects which may be exerting force on other objects in the scene are tagged for update, and the current force acting on each tagged object is computed and recorded.

Finally, the objects that are tagged for update have their states recomputed, and the graphics host is notified of the new information at a rate of approximately 30 updates/sec (not every control loop cycle). This permits us to keep the haptics loop as tight as possible so that the simulation runs smoothly, and allows us to update the graphics loop at the approximate limit of graphics performance.

Haptics Library: summary

The SPI haptics library uses these models and others to provide a flexible platform for quickly mocking up haptic simulations using the PHANToM haptic interface, that accompany computer graphic stereo display of the same objects. The complexity of computer graphic rendering, handled by the client, does not interfere with server control loop speed; thus visual simulations rendered by the client can be arbitrarily elaborate. The calls currently available to client applications are listed below:

Available Client Calls

PhantomPollPos(*&x, &y, &z, button*)
PhantomGetForce(*&fx, &fy, &fz*)
PhantomSetNumReadAttempts(*n*)
PhantomOpen()
PhantomClose()
PhantomSetVolumeLimits(*xMin, xMax, yMin yMax, zMin, zMax*)
PhantomGetVolumeLimits(*&xMin, &xMax, &yMin &yMax, &zMin, &zMax*)

(cont'd)

PhantomDefinePlane(*xPnt, yPnt, zPnt, xNorm, yNorm, zNorm, stiffness*)
PhantomDefineImgTx(*xRes, yRes, xDist, yDist, *imagefile*)
PhantomDefineSphere(*xCenter, yCenter, zCenter, xRad, yRad, zRad, stiffness*)
PhantomDefineButton(*xMin, xMax, yMin, yMax, zMin, zMax, stiffness, activeside*)
PhantomDeletePlane(*planeID*)
PhantomDeleteSphere(*sphereID*)
PhantomDeleteButton(*buttID*)
PhantomActivatePlane(*planeID*)
PhantomDeactivatePlane(*planeID*)
PhantomActivateSphere(*sphereID*)
PhantomDeactivateSphere(*sphereID*)
PhantomActivateButton(*buttID*)
PhantomDeactivateButton(*buttID*)
PhantomTranslateObject(*objID, xTrans, yTrans, zTrans*)
PhantomGetButtonHit()
PhantomNumButtonHits()
PhantomSetPlaneFriction(*planeID, cs, ck*)
PhantomSetPlaneDamping(*planeID, b*)
PhantomSetSphereFriction(*sphereID, cs, ck*)
PhantomSetSphereDamping(*sphereID, b*)
PhantomSetPlaneTexture(*planeID, amp, freq, texid*)
PhantomSetSphereTexture(*sphereID, amp, freq, texid*)
PhantomSetViscosity(*b*)
PhantomSetDriftForce(*fx, fy, fz*)
PhantomReInitialize()
PhantomForcesOn()
PhantomForcesOff()

Acknowledgments

This work was primarily sponsored by the Honda R&D Corporation. Much of the work was performed at the MIT Media Lab and at Interval Research Corporation, where one of the authors, W. Plesniak, was sponsored as an Interval Fellow. Equipment support was provided by the Intel Corporation.

Graphical and Haptic Manipulation of 3D Objects

Krasimir Kolarov[1], Diego Ruspini[2]

Interval Research Corporation[1], Stanfrord University[2]
kolarov@interval.com

1. Research Interests

We are interested in identifying new ways of human - machine interaction via haptic and graphical devices. In particular we have been working toward developing a cooperative graphic and haptic interface that allows us to manipulate and sculpture 3D objects more effectively. That includes compact and efficient representation of the object as well the ability to "feel" the 3D object - different textures, friction, etc. - and sculpt and model it through the Phantom. We are assuming that the combined graphical and haptic interface to 3D objects will allow us much richer and powerful interaction as well as the ability to perform tasks that are not possible with the existing technology.

2. Joint Haptics project with Stanford University.

During the past year we started by building an interface based on the UNC haptics library [1] that includes on one side the Phantom and a PC server to it and on the other side an SGI for graphical visualization and manipulation of virtual objects. The communication between the PC and the SGI is done over the Internet. That setup allowed for effective use and experiments with simultaneous haptic and graphical interface with virtual 3D objects. It also allowed us to study the problems of latency over the Internet, transfer of complicated 3D objects, as well as graphic visualization.

As part of this effort we built a joint research project between Interval Research Corp. and the research group of Prof. Oussama Khatib from the Robotics lab at Stanford University. This project explores the possibility of interactions between remote sites with the Phantom arms that include:

human (using a Phantom) -to- human (using a Phantom);

human (using a Phantom) -to- dynamic, realistic, 3D simulation;

human (using a Phantom) -to- Puma robot (master-slave teleoperation).

3. Student projects in Advance Robotics at Stanford University

We initiated and organized joint projects with the students in the Experimental Robotics class at Stanford in 1995 and 1996 taught by Prof. Oussama Khatib. That included equipping a Pentium PC and a Phantom with the right software and hardware for the project, organizing, directing and supervising the student's efforts. The project was successful in that in a few weeks, the students produced interesting and ambitious demos illustrating the advantages of combined use of haptic feedback and graphics. It also provided a good basis for extending the existing system to more complicated 3D objects and meshes. I will describe the projects that the students did, some of which were successfully demonstrated at the Experimental Robotics conference at Stanford in July 1995.

3.1 The Virtual Xylophone.

The first project built a virtual xylophone. The graphics were done on SGI Inventor and included the color xylophone and a mallet that was controlled by the Phantom (see Figure 1). The sound changed depending on the acceleration with which one plays the xylophone. For sound experiments there was a provision for changing from xylophone to an organ sound as well. A number of interesting issues relating to the interaction of haptic, visual and sound feedback were raised.

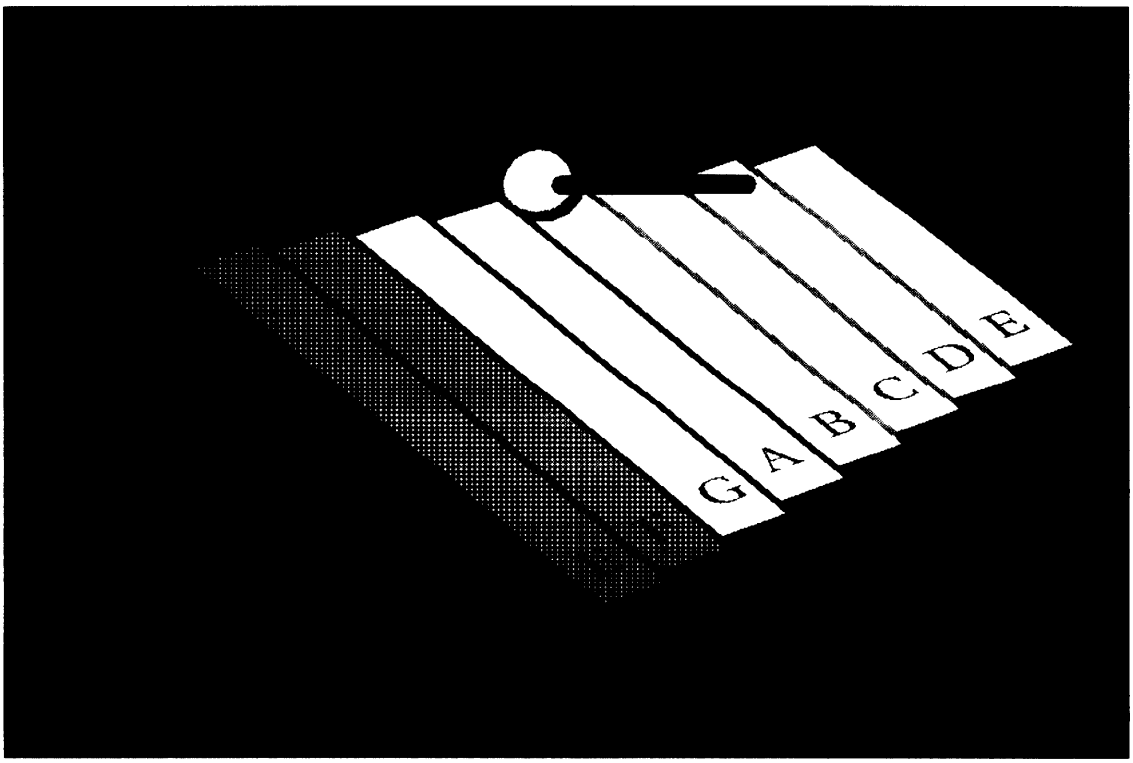


Figure 1. Graphical view of the Virtual Xylophone.

3.2 Haptic Roaches.

The second project described involves a "Roaches" game. The graphical environment (done on the SGI with Inventor) includes a large playpen with cylindrical obstacles, a number of roaches and a small sphere representing the users' finger (controlled by the Phantom) - see Figure 2. The goal is for the user to kill the roaches by pinning them down from above. The roaches have certain amount of intelligence and try to hide underneath the obstacles which the user needs to put aside in order to reach the roaches. All graphical objects are inherently 3 dimensional and the game is enhanced via sound and visual effects. The criteria for the user's performance in the game is time, which adds another dimension to the project.

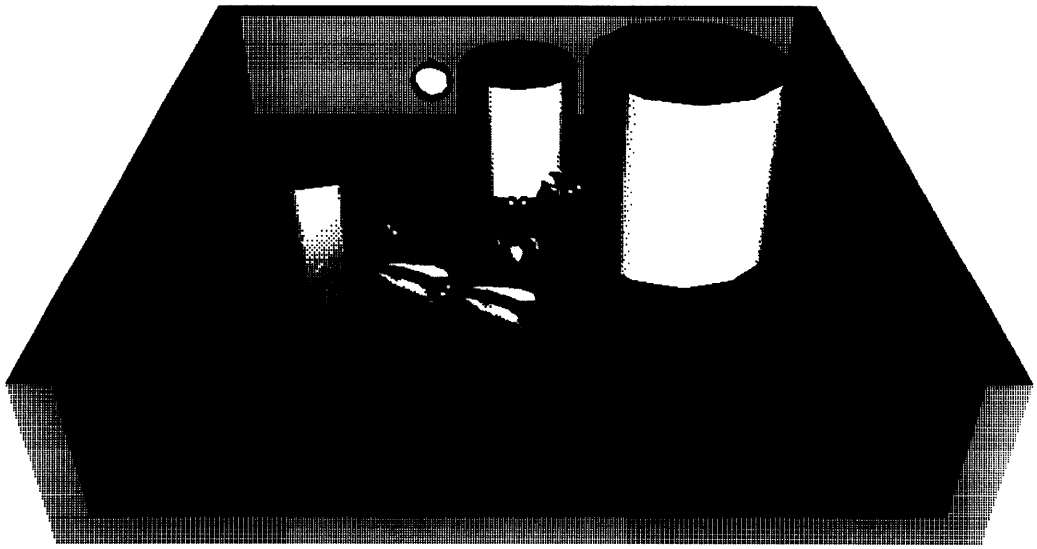


Figure 2. The game of Haptic Roaches.

3.3 Haptic Exploration of rigid 3D objects.

The third project describes our first joint efforts with Diego Ruspini from Stanford on graphic and haptic exploration of rigid 3D objects with surfaces described as polygonal meshes. Examples include a virtual 3D staircase, 3D bunny and space shuttle. The newest results and the theoretical foundation of this work are further explained in a separate paper for this Workshop [2]. Our

overall goal is to extend the capabilities of our haptics library to support a more powerful and general set of modeling primitives (for manipulating arbitrary complex rigid objects), to allow the haptic server to operate with a greater amount of autonomy from the host computer, and simulate a wide range of virtual environments. On the next stage the support libraries provides a means of allowing the developer to specify constraints between the objects in the environment and for controlling the motion of objects in the virtual world. The last part of the effort will involve modeling the contact forces caused by contact and collisions between the objects in the virtual environment. This work will build on research being done in the areas of fast distance/collision calculation and dynamic simulation at Stanford [3].

REFERENCES

[1] W. Mark, S. Randolph, M. Finch, J. Van Verth, R. Taylor II, "Adding force feedback to graphics systems: issues and solutions". *SIGGRAPH' 96 Proceedings*, pp. 447-452, August 1996.

[2] D. Ruspini, K. Kolarov, O. Khatib "Robust Haptic Display of Graphical Environments", *Proceedings of the First Phantom User Group Workshop*, Boston, September 1996.

[3] S. Quinlan, "Efficient distance computation between non-convex objects", *International Conference on Robotics and Automation*, April 1994.

Robust Haptic Display of Graphical Environments

Diego C. Ruspini[1], Krasimir Kolarov[2] and Oussama Khatib[1]

Robotics Laboratory[1]
Stanford University, Stanford, CA 94305

Interval Research Corporation[2]
1801 Page Mill Road, Palo Alto, CA 94304

Introduction

We are developing a new haptic interface library "HL." With a Application Programmers Interface (API) almost identical to that of "GL", a graphics hardware interface library used on all Silicon Graphics workstations, haptic environments can quickly and efficiently be incorporated into graphics applications. The library uses a multi-level control system to effectively simulate contact with virtual environments. Work has also begun to help guarantee that the behavior of the haptic device is stable even if the simulated model is too complex to be simulated accurately.

In our implementation contact, friction and other forces are not applied directly to the haptic device, but to a virtual "proxy" or representative object in the virtual world. This object, similar to the god-object proposed by Zilles and Salisbury [Zilles94], has a known mass, velocity and acceleration and can therefore be accurately simulated as it is effected by forces in the virtual environment. The haptic device and a virtual spring/damper are then used to keep the error between the virtual and real objects configuration to a minimum.

To allow complex models to be simulated, fast collision detection between the proxy and the virtual environment is essential. In our implementation a hierarchy of bounding spheres is automatically constructed when objects are introduced [Quinlan94]. This hierarchy can be quickly traversed to prune objects that are too far from the proxy to effect its movement. Only nearby obstacles need to be checked fully to see if they lie in the path of the proxy. The use of hierarchical bounding volumes reduces the computation costs enough to allow complex environments to be simulated.

Many graphic interfaces, including "GL," allow modelers to specify surface normals on the vertices of polygonal surfaces. This information is used to alter the lighting model on the surface to give it the appearance of being smooth. The normal information is used by our system to create a similar haptic effect. The constrained motion of the proxy is computed using the local surface orientation inferred by the supplied normals. Once the constrained motion is found, it is projected back onto the actual object surface. The proxy position is then advanced along this projected motion direction. This gives the user the appearance that they are on a smooth surface while actually moving along the underlying polygonal model. This technique produces results similiar to that proposed by [Morgenbesser96].

By using a virtual proxy the haptic servo controller task is reduced to minimizing the error between the configuration of the proxy and position of the haptic device. Reducing position error of a mechanical system is problem that has been dealt with extensively in the robotics literature [Craig89]. In our current implementation a strait forward operational space proportional derivative (PD) controller is used to guarantee stability even in the presence of a large number of objects. The low level control loop can be separated from the contact/proxy update loop. By running the control loop at a high fixed clock rate stability is more easily ensured and the fidelity of the haptic display can be made to degrade gracefully, as the complexity of the environment is increased.

2. The Client Application

Our current haptic interface runs on two computers connected via ethernet and follows a common client/server metaphor. The haptic client is where a users application is run. In our implementation very little work is performed by our interface on the client CPU. This was intentional so that the haptic display would not interfere with the performance of the users application. The bulk of the work required to render a haptic scene is performed by the haptic server. Its job is to receive all modeling information from the client, track the position of the haptic device, update the position of the virtual proxy, and send control forces back to the haptic device.

The "HL" Library allows users to define objects as a collection of primitive objects -- points, line segments, or polygons. Transformations are provided to allow objects and primitives to be translated or rotated freely. Surface normal and texture coordinates can be associated with polygons vertices to allow smooth or textured surfaces to be defined. Any number of objects can be created, and objects can call other objects allowing the creation of complex object hierarchies. All primitives specified between the time an object is opened and closed belong to the new object. The new object does not become visible to the haptic display until it is closed. This condition ensures that a given object is not made touchable until it is completely specified. A special object (Object 0) is always displayed. This object, possibly empty, forms the root of the object hierarchy. This slight difference from the "GL" specification allows the object hierarchy to be continuously defined, unlike in "GL" where time is divided into discrete frames. This continuity in the definition of the virtual environment is important since the haptic server may be running at a servo rate many times that of the application program.

On the client side almost all functions are small wrapper procedures that pack the functions arguments into a packet to be transmitted to the haptic server. The communication protocols supported by the library fall into roughly three categories, commands which are issued by the client and do not require a response, queries where the client waits for a response from the server, and events which are issued by the server when a specified condition has occurred (e.g. the user has touched an object.) Commands that do not require immediate action are queued up into a large packet and only transmitted if the buffer becomes full, or the buffer is explicitly flushed. Queries always flush the queue before waiting for a response from the server. Query

functions do not return until the desired response is received or an error occurs. When events are received they are placed in an event queue until they can be serviced by the client application.

3. Model Construction

Once the modeling commands are received from the client, they must be stored in a form suitable for haptic rendering. Vertices are transformed into the local object frames and meshes and sequences of line segments are broken into a set of independent convex bodies. Intersections tests between the convex primitives and the path of the virtual proxy are performed by the Gilbert distance algorithm described in [Gilbert88]. This algorithm computes the distance between two convex polyhedrons in $O(n+m)$ time, where n and m are the number of vertices in each polyhedra. Each polyhedra is specified as the convex hull of a set of vertices. No additional information about the edges or the facets of the convex hull, or the order of the vertices is required. This algorithm returns the nearest point to each polyhedra as the affine combination of the vertices of the polyhedra. The affine weights can be used to interpolate surface normals and texture coordinates from the vertices of the underlying polygon. While this algorithm is more general then what is required by our current system, in the future we hope to extend our system to allow the proxy to represent more complex objects.

Because each object is normally constructed from a large number of primitives, a naive test of checking each primitive with the path of the proxy would be prohibitively expensive. In general the proxy will be in contact with at most a small fraction of the underlying primitives. To make use of the spatial coherence inherent in the object, a hierarchical bounding representation for the object is constructed. The bounding representation is based on spheres similar to the one described in [Quinlan94]. This hierarchy of bounding spheres is constructed by first covering each polygon with small spheres in a manner similar to scan conversion in computer graphics. These spheres are the leaves of an approximately balanced binary tree. Each node of this tree represents a single sphere that completely contains all the leaves of its descendants.

After covering the object a divide and conquer strategy is used to build the interior nodes of the tree. This algorithm works in a manner similar to quick-sort. First an axis aligned bounding box that contains all the leaf spheres is found. The leaf spheres are then divided along the plane through the mid-point of the longest axes of the bounding box. Each of the resulting two subsets should be compact and contain approximately an equal number of leaf spheres. The bounding tree is constructed by recursively invoking the algorithm on each subset and then creating a new node with the two sub-trees as children.

Two heuristics are used to compute the bounding sphere of a given node. The first heuristic finds the smallest bounding sphere that contains the spheres of its two children. The second method directly examines the leaf spheres. The center is taken as the mid-point of bounding box already computed earlier. The radius is taken to be just large enough to contain all the descendant leaf nodes. Note that a given node is not required to fully contain all of its descendant nodes only their leaf nodes. The method that generates the sphere with the smallest radius is used for the given node. The first heuristic tends to work better near the leaves of the tree, while the second method produces better results closer to the root. This algorithm has an expected $O(nlogn)$ execution time, where n is the number of leaf spheres.

4. Updating Proxy Position

At each time step the virtual proxy moves in the direction that reduces the error between it and the actual finger position, subject to the constraints imposed by the objects in the environment. In our current implementation the proxy is modeled as a sphere with a radius specified by the user. Because of small numerical errors polygons that are intended to share a common edge may in fact contain gaps. The proxy should therefore be at least large enough to avoid slipping through these holes in the underlying model, otherwise no restriction is placed on the size of the proxy. Most often a user would like to make the proxy large enough to be easily visible on a graphics display.

To determine if a collision occurs between the proxy and an object in the environment, the volume swept by the virtual proxy, as it moves during a given time period, is checked to see if it penetrates any primitive in the environment. In our system the time step is always small enough so that the path of the proxy can be approximated by a straight line segment. Because the proxy is modeled as a sphere the comparison is reduced to determining if a line segment comes within one proxy radius of a given polygon. A sphere containing this line segment is checked with the bounding sphere hierarchy of each object. If the two spheres do not touch, then the underlying primitives cannot interfere with the path of the proxy, and no further testing is required. If the spheres penetrate, a collision is still possible and the child nodes of the object are checked recursively to see if they also touch the bounding sphere of the path. If a leaf node is reached the proxy path and primitive are checked using the Gilbert distance algorithm. Because a primitive may have more then one leaf sphere, and the path bounding sphere may intersect more then one of these leaves, a comparison cache is kept to quickly determine if a distance test has already been performed at this time step, thus avoiding extraneous calls to the distance algorithm.

When a collision between a primitive and the proxy path is found, a plane normal to the obstacle in the configuration space of the proxy is calculated at the point where the proxy will first make contact with the obstacle. Once all contacts have been found the proxy position can be moved until it makes contact with the closest contact planes. Planes that fall below this new point cannot at least locally affect the motion of the proxy and can therefore be pruned. If the proxy reaches the users position no further movement is required. Otherwise a new constrained motion direction is needed. In [Zilles94] Zilles and Salisbury propose using Lagrange multipliers to find the constrained motion of the "god-point." The paper describes a solution where the allowable region of motion is restricted to the intersection of all the contact planes, when in fact at most three of the constraint planes can be active at one time.

Determining the contact plane can be solved with standard quadratic programming techniques like those described in [Gill86], however this problem has many simplifications that make a less complex and faster solution possible. In our implementation this problem is solved by taking its dual. The solution is independent of translation or scaling, thus the proxy can be shifted to

the origin and the distance from the proxy point to the desired goal position can be scaled to one. All contact planes, since they are in contact with the proxy, now go through the origin. In this dual each contact plane is represented by a point, defined by the negative components of the contact plane normal. The negated normals represent the outward normals of the local-free space region. The active constraint planes are found by finding the point nearest to the goal position on the convex hull of the contact plane points and the origin. An example of this duality is illustrated in Figure 1 below.

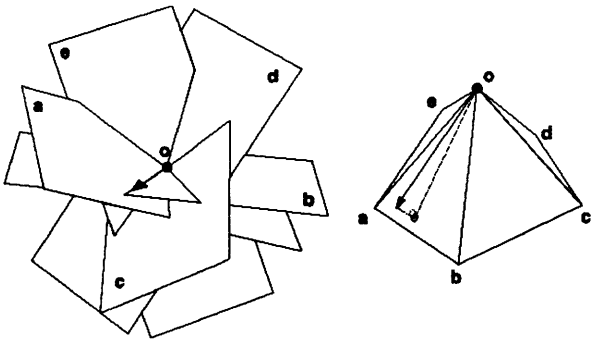


Figure 1: Duality of Constrained Motion Solution

This problem is now in a form that can be solved using the same distance algorithm as was used for the primitive contact test [Gilbert88]. This algorithm returns the nearest point as the affine combination of points on the convex hull. Each point with a positive contribution to the solution indicates that the corresponding contact plane constrains the motion of the proxy. By using the dual the active contact planes can be found in $O(n)$ time, where n is the number of contact planes and without the addition of any large software package. Once the (at most) three active constraint planes are found the desired motion of the proxy can be quickly found and the iteration can continue.

5. Force Shading

Most graphic interfaces allow modelers to specify surface normals on the vertices of polygonal surfaces. This information is used to alter the lighting model on the surface to give it the appearance of being smooth. To create a similar haptic effect, when contact between the proxy and a polygon (containing user specified normals) is detected, a new normal at the surface is calculated. This is done by interpolating the specified vertex normals using the weights (returned by the distance algorithm) that specify the contact point. This interpolation is very similar to the interpolation done for Phong shading in computer graphics. The computed normal is used to specify a new contact plane going through the contact point. This plane is used instead of the true contact plane to compute the motion of the proxy as described above. Once the constrained motion is found, this motion is projected back onto the actual contact surface. This process is illustrated in Figure 2.

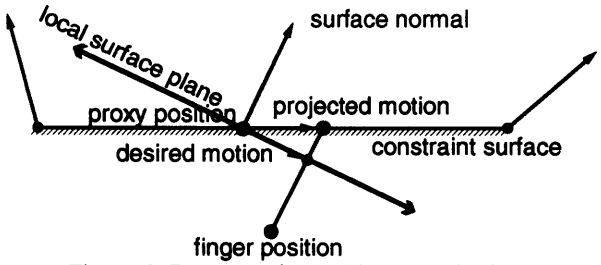


Figure 2: Proxy Update with Force Shading

The difference between the simulated smooth surface and a normal surface is illustrated in figures 3 and 4. In the example the difference between the actual user position and the position of the virtual proxy are shown as the users finger follows a circular path around a ten sided polygonal approximation of a circular object. For compactness only the first quadrant of the circle is shown. Because the constraint surface is curved, the differences between the position of the proxy and the finger are best illustrated in polar coordinates. The angular displacement corresponds roughly to the amount to which the users finger is pulled tangentially as the user moves around the circle. The radial displacement indicates roughly the amount of force the user feels pulling the finger toward the surface of the object.

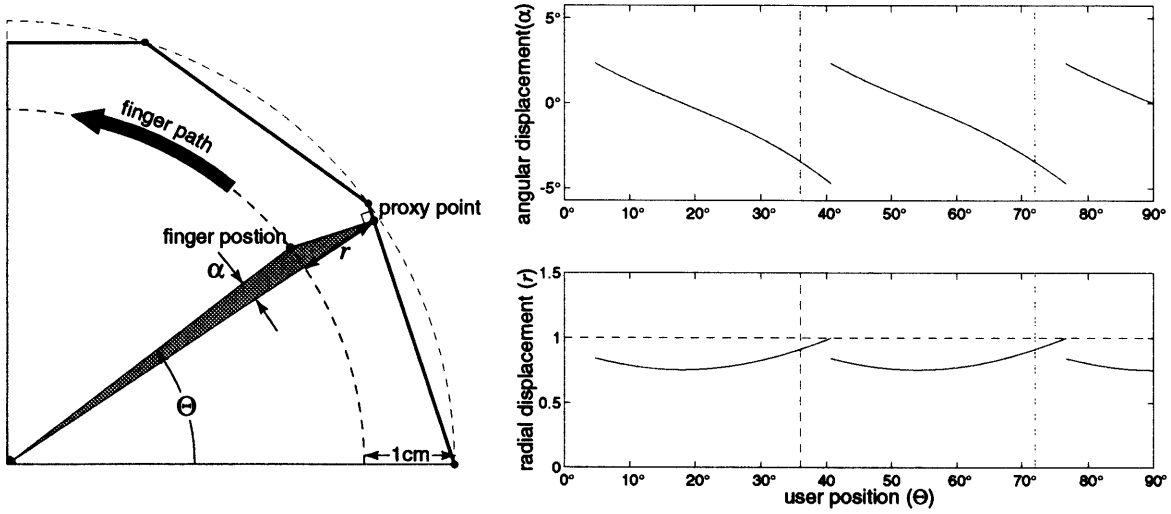


Figure 3: Virtual Proxy Path

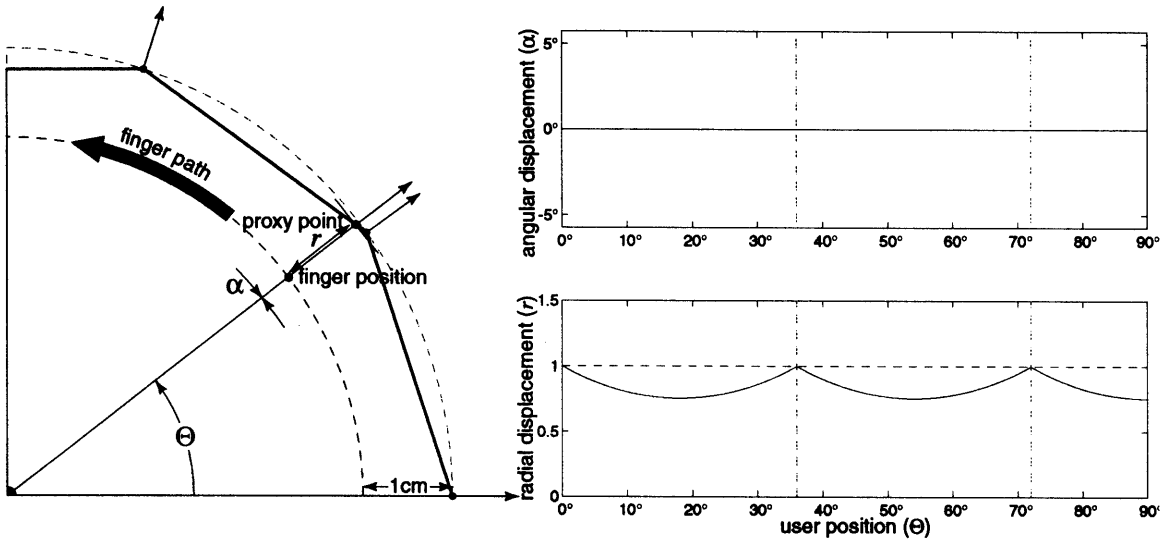


Figure 4: Proxy Path with Force Shading

As is seen in Figure 3 a strong discontinuity occurs when the proxy finally reaches the edge of the object. In Figure 4 surface normals have been specified on the vertices as shown. The result movement of the proxy shows that no tangential force is felt by the user as the finger is moved around the object. This is what would be expected if the user where moving around a perfectly circular object. In the radial plot it can be seen that the proxy still follows the underlying surface of the object. However the motion of the proxy has no discontinuities and therefore the surface appears smooth to the user.

It should be noted that this technique may increase the distance between the user position and the proxy. This makes ensuring stability much more difficult. It is possible to specify normals and polygons that make the resultant motion of the proxy unstable. However for normals typically seen in practice proxy movement appears to be well behaved.

6. Haptic Control

By using a virtual proxy the haptic servo controller task is reduced to minimizing the error between the configuration of the proxy and position of the haptic device. Reducing position error of a mechanical system is problem that has been dealt with extensively in the robotics literature. In our current implementation a simple operational space proportional derivative (PD) controller is used to guarantee stability even in the presence of a large number of objects. The low level control loop can be separated from the contact/proxy update loop. By running the control loop at a high fixed clock rate stability is more easily ensured and the fidelity of the haptic display can be made to degrade gracefully, as the complexity of the environment is increased.

7. Results

Our haptic library has been successfully used on models containing more the 2000 polygonal primitives. In our tests the client computer was an SGI Indigo2 High Impact running IRIX 6.2. The haptic server was a 90Mhz Pentium running Linux 2.0.2, connected to a PHANToM haptic device. Communication was through a standard ethernet TCP-IP connection. The haptic server gave stable results for that many polygons even with position gains over 1600 Newton/meter, and no artificial damping. The cycle time of the proxy update loop shows only an approximately O(logn) growth with the number of polygons. It is hoped that with a faster CPU, even more complex models can be simulated.

Future Work

At present our current library does not allow objects to be moved or manipulated. This is a large drawback since one of the main goals of our project is to allow people to better interact with the virtual environment. Work to allow the proxy to interact with the objects in the environment has already begun.

Bibliography

[Gilbert88] E. G. Gilbert, D. W. Johnson and S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," IEEE J. of Robotics and Automation, Vol.4, No. 2, April 1988.

[Gill86] P. Gill, S. Hammarling, W. Murray, M. Saunders and M. Wright, "User's Guide to LLSOL," Stanford University Technical Report SOL 86-1, (January 1986).

[Massie94] Massie, T.M., Salisbury, J.K., "The PHANToM Haptic Interface: A Device for Probing Virtual Objects." ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1994, In Dynamic Systems and Control 1994 (Chicago, Illinois, Nov. 6-11), vol. 1, pp.295-301.

[Morgenbesser96] Morgenbesser, H.B., Srinivasan, M.A., "Force Shading for Haptic Shape Perception." ASME Winter Annual Meeting 1996.

[Quinlan94] Quinlan, S., "Efficient Distance Computation between Non-Convex Objects," Int. Conference on Robotics and Automation, (April 1994).

[Zilles94] Zilles, C. B., Salisbury, J. K., "A Constraint-based God-object Method for Haptic Display." ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1994, In Dynamic Systems and Control 1994 (Chicago, Illinois, Nov. 6-11), vol. 1, pp.146-150.

Haptic Scientific Visualization

Jason P. Fritz, Kenneth E. Barner
Applied Science and Engineering Laboratories
University of Delaware/A.I. duPont Institute
Newark, DE 19716
Email: *barner@asel.udel.edu*

Abstract

Haptic scientific visualization is a technique used to represent scientific or mathematical data using a haptic interface. The haptic visualization project implements a number of algorithms for the haptic rendering of data using the PHANToM. These algorithms depend on the nature of the data, and the desired representation such as a wire frame mesh or a solid surface. Simulation enhancements are also added to improve the human interpretation of the data. The resulting system can be used investigate non-visual information display and the human comprehension of those representations.

1 Introduction

Scientific visualization is a technique used to explore scientific data by representing it in a form more suitable for human comprehension. Traditionally, this form has been only visual, but humans explore new objects and environments using all of our senses, especially the active sense of touch. Haptic interface technology, therefore, can provide an additional medium for data analysis, and is especially beneficial to blind or visually impaired people. Previously, blind people had no access to interactive 3D representations of scientific/mathematical data. Haptic interfaces, like the PHANToM, provides a higher bandwidth medium (in terms of information) for non-visual exploration of virtual environments than any other non-visual sense.

The haptic visualization project at the Applied Science and Engineering Laboratories [2] investigates techniques for the effective haptic display of scientific data, and to create a software framework for this implementation. The hardware for this project consists of the PHANToM connected to a 120 MHz Pentium running Windows NT, and a Silicon Graphics Crimson for graphical display (see Fig. 1). Through an object-oriented approach, the software framework implements techniques for rendering data as points, lines, surfaces, or vector fields. The available rendering methods depend on the dimensionality of the data to be visualized, which can be up to three dimensions for the 3 DOF interface used in the current system.

Haptic enhancements to the simulations are used to extract more information from the data. In graphical information display systems, enhancements such as numbered axes and colors are typically added to ease information extraction. These enhancements do not have direct haptic equivalents, but can be represented non-visually through means such as speech output, haptic grid planes and texture variations. Texture and friction can also be used in a haptic display in an analogous manner to false coloring in a visual display.

2 Haptic Visualization System

The visualization system consists of both a haptic and graphics subsystems. Although the main goal is the development of the haptic components, graphics are used to aid sighted and low vision users, and aid in development and debugging. The haptics and graphics systems have different operating conditions, and are thus kept separate, Fig. 1. The haptic environment is generated on the PC, with a control loop frequency of 1kHz. The graphics are rendered on an SGI using OpenGL. All environment information resides on the PC,

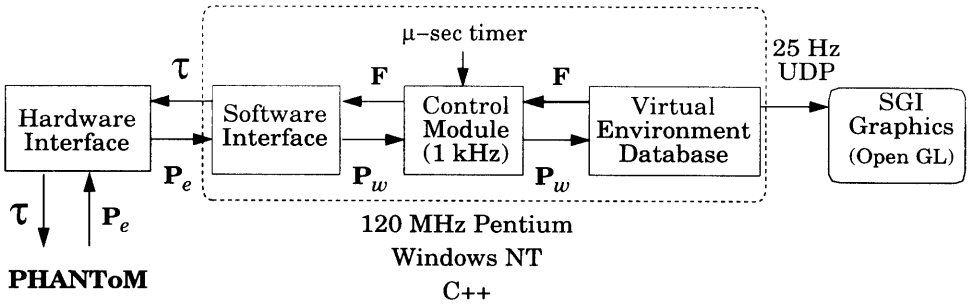


Figure 1: System setup. $\mathbf{P}_e \doteq$ encoder position, $\mathbf{P}_w \doteq$ world position, $\mathbf{F} \doteq$ force, $\tau \doteq$ torque.

with changes sent over Ethernet to the SGI using UDP at a rate of 25Hz. Timing is accomplished through a custom built μ-second timer.

The visualization system software is being developed in an object-oriented fashion using C++. The software structure is such that the haptic interface is independent of the haptic environment, Fig. 1, allowing for the use of different haptic interfaces. Under this structure, force and interface position are the only variables transferred between the haptic interface and the software environment. The haptic interface, to the PHANToM in this case, is implemented as a class. Thus, the Phantom class is used as the software interface to the PHANToM for the transformation of position and force/torque vectors from joint space to world space and vice-versa. This class also handles additional PHANToM I/O and a software controller for stability.

The haptic environment is implemented as a database class. This database class defines and controls the interaction of all of the objects, their haptic methods, force profiles, and textures. Haptic methods compute a displacement or penetration vector, which is then used by the force profile to compute the constraint or normal force in addition to any friction forces. This structure allows, for instance, constraint forces to range from linear, such as a spring-damper model, or non-linear, such as a button profile. There is also a class for stochastic processes and filter implementation for properties such as texture and overall system stability. This structure is designed for flexibility and simplicity to make it relatively easy to add new algorithms and create complex haptic simulations.

3 Objects and Rendering Methods

Data comes in many forms, and there is no general solution to render different types with one method. Ideally, these algorithms should be simple to implement, and fast enough to run in real-time, in addition to presenting the data in a form that is easy to comprehend. As previously mentioned, data is organized by its dimensionality as 1D vectors, 2D matrices, or 3D matrices. For each dimension type, there are a number of algorithms available for rendering. Each of the data types currently used in our visualization system and their rendering methods are briefly discussed next.

One-dimensional data is rendered in a plane as abscissa and domain values the same way that graphic plots render vectors. These values are left as discrete points or connected with lines. To feel discrete points is space, a sphere is created around the haptic interface point (IP). Now perform collision detection between the sphere and the points. The resultant displacement vector is the normalized sum of the displacement vectors of each point intersecting the sphere, expressed as

$$\mathbf{d} = \frac{1}{M} \sum_{i=1}^{M} (r - \|\mathbf{p}_i\|) \hat{\mathbf{p}}_i$$

where M is the number of intersected points, r is the sphere radius, and $\|\mathbf{p}_i\|$ is the distance from the ith point to the center (IP) in the direction of $\hat{\mathbf{p}}_i$. The resultant $\mathbf{d}$ is then modified according the object force profile. This sphere concept can also be used to feel lines, where $\mathbf{p}$ is now the shortest vector from the line

to the sphere center (the IP). Conversely, lines can also be rendered by creating a force field around the line which generates forces to attract the IP when it is in close proximity to the line. This is similar to a gravitational force except that there are no forces generated when the IP is on the line.

Two-dimensional data can be rendered as a height map on a 2D lattice where the heights of the data points correspond to the values of the matrix. These points can be left as discrete points, connected with lines to create a mesh, or connected to create a solid surface. For points and lines (wire frame mesh) the sphere technique implemented for 1D data is used. When rendered as a surface, triangular polygons and an algorithm called the Shadow Point are used. The Shadow Point method was derived from the God-object method [6]. The God-object computes the location of the point on the surface where the IP would be if the surface could be infinitely stiff. The solution is found by solving a constrained minimization problem using Lagrange multipliers, where the constraints are the planar facets. There are three possible solutions depending on the number of constraints. For one constraint it is the point on the constraint plane closest to the IP. Two constraints gives the closest point on the line defining the intersection of two planes, and three constraints results in the point defining the intersection of three or more planes. This is the approach that the Shadow Point takes to find the same solution using fewer operations. Since the display of 2D data surfaces renders the polygons on a rectangular grid, only those facets located in a rectangular window around the IP (independent of the IP height) are checked for collisions. This window takes advantage of the non-global aspect of point interaction, and allows a surface to be rendered with a large number of polygons.

Three-dimensional data can be rendered as discrete points, a 3D mesh, a 3D polyhedron, or a vector field. The points and mesh methods are rendered as mentioned above. A polyhedron is the general case of the height map surface. The Shadow Point is also used in this case, but collisions are checked with all facets before the initial contact. Therefore, a faster algorithm increases the maximum number of polygons that can be rendered. Rendering the data using a 3D vector field typically only maked sense when the data itself represents a 3D vector field. The vectors are arranged on a 3D rectangular lattice, and result in a scaled version of the force vector at that point in space. Interpolation is used when the IP is not exactly on the lattice intersections, namely zeroth order (voxel representation), linear, quadratic, or cubic depending on the smoothness desired.

4 Environment and Haptic Enhancements

While adequate for feeling relative shapes, the previously discussed rendering methods do not provide information about the specific values of the data. Therefore, additional features must be added. One feature is speech. Using a text-to-speech (TTS) converter, the coordinates of the data or the location of the IP in the data coordinate system can be spoken at any time. Another feature is the equivalent of grid lines called grid planes. Grid planes generate a small, but perceivable force when the IP passes through them, and are useful for navigation and scale information.

Friction and texture are added to enhance the overall simulation, and to extract more information from the data. We have implemented simple models for dry friction ($\mathbf{F}_{static} \leq \mu_s \mathbf{F}_n$ and $\mathbf{F}_{kinetic} = \mu_k \mathbf{F}_n$), viscous friction ($\mathbf{F}_v = -B\mathbf{v}$), and drag friction ($\mathbf{F}_d = \frac{1}{2}\rho A \mathbf{v}^2$, where ρ and A are the viscosity of the medium and the cross-sectional area of the object respectively). To prevent the dry friction model from generating forces when not in motion, the friction force is zero below a small threshold velocity. "Anti-friction", which is in the same direction as velocity, can be generated by using small negative coefficients. These coefficients, though, can cause system instability if they over compensate for the friction and damping of the hardware coupled with the human user. Since the PHANToM has no force sensors to measure the applied force, estimates are made from position samples [5]. The range of friction coefficients is limited due to stability concerns, however, they can be discretized into distinguishable levels to correspond to discretized colors.

Texture is a much more complicated property to model. As with friction, the goal is to render distinguishable textures, not to exactly model real textures. We have developed a framework to implement a number of different algorithms. In general, there are two ways to represent texture, which have equivalent counterparts in graphics: texture mapping and vector perturbation [1, 3]. Texture mapping maps texture patches on a surface, and includes Minsky's lateral gradient algorithm for height maps [4]. Vector perturbation generates a

texture force vector that is added to the normal constraint force, perturbing its magnitude and/or direction. This method includes 1D sampling (temporal or spatial) of a continuous surface with or without additive noise, and a lattice representation of a texture. One dimensional sampling obtains or generates the texture vector according to the given model. For example, consider $\mathbf{F}_t = \sin(x) + \sin(y) + z_o + \mathbf{w}$ where x and y are respective IP components, z_o is the height of a plane, and $\mathbf{w}$ is a stochastic process. The lattice representation is the same as that used for rendering a 3D vector field, but the lattice itself can be one, two, or three dimensional. In general, the stochastic texture vectors are generated by filtering white noise, expressed as $\mathbf{F}_t(x, y, z) = \mathbf{h}(x, y, z) * \mathbf{w}(x, y, z)$, where the filter impulse response $\mathbf{h}$ is convolved with a 3D white noise process, $\mathbf{w}$. The stable filter can range from an all pass (limiting the magnitude for hardware stability), to a complex model such as an autoregressive moving average (ARMA) or Markov Random Field. The lattice structure allows for the incorporation of spatial dependencies of the texture vectors. More control of the spatial frequencies is another benefit from the lattice structure. The major constraint on texture rendering is the bandwidth of the haptic interface.

5 Conclusions and Future Work

Using this framework, we have developed a number of simulations. These include 2D line graphs and 3D surface and mesh plots, and 3D vector fields. Simple texture has also been added to surfaces. Other simulations include a dynamic peg insertion program, and a representation of spherical sound waves that can be frozen in time. Feedback from users has been positive.

There is a lot of potential future work for this project. The software framework is currently only a skeleton and needs further development. Texture modeling will also need further development and experimentation. Only basic applications have been created, without extensive human factors research into the comprehension value that they have. In addition to the haptic visualization project, there are two near future projects that will use the PHANToM. One will investigate the possibility of using the PHANToM for hand tremor filtering, in addition to a software filter. The other project would involve mounting a PHANToM to a wheelchair to be used as a teleoperation master.

References

[1] D. S. Ebert (editor), F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: a Procedural Approach.* AP Professional, Boston, MA, 1994.

[2] J. P. Fritz. Haptic techniques for scientific visualization. Master's thesis, University of Delaware, 1996. Work in progress.

[3] S. Haruyama and B. A. Barsky. Using stochastic modeling for texture generation. *IEEE Computer Graphics and Applications*, pages 7–19, Mar. 1984.

[4] M. M. Minsky. *Computational Haptics: The Sandpaper System for Synthesizing Texutre with a Force-Feedback Haptic Display.* Doctor of philosophy, Massachusetts Institute of Technology, 1995.

[5] K. Salisbury, T. Massie, D. Brock, N. Swarup, and C. Zilles. Haptic rendering: Programming touch interaction with virtual objects. In *ACM Symposium on Interactive 3D Graphics*, 1995.

[6] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *IROS '95*, 1995.

6 Acknowledgements

This project is funded by the National Science Foundation, Grant # HRD-9450019, with additional support from the Nemours Foundation Research Program.

Nanomanipulator Force Feedback Research

Jun Chen†
Russell M. Taylor II

Department of Computer Science
University of North Carolina, Chapel Hill

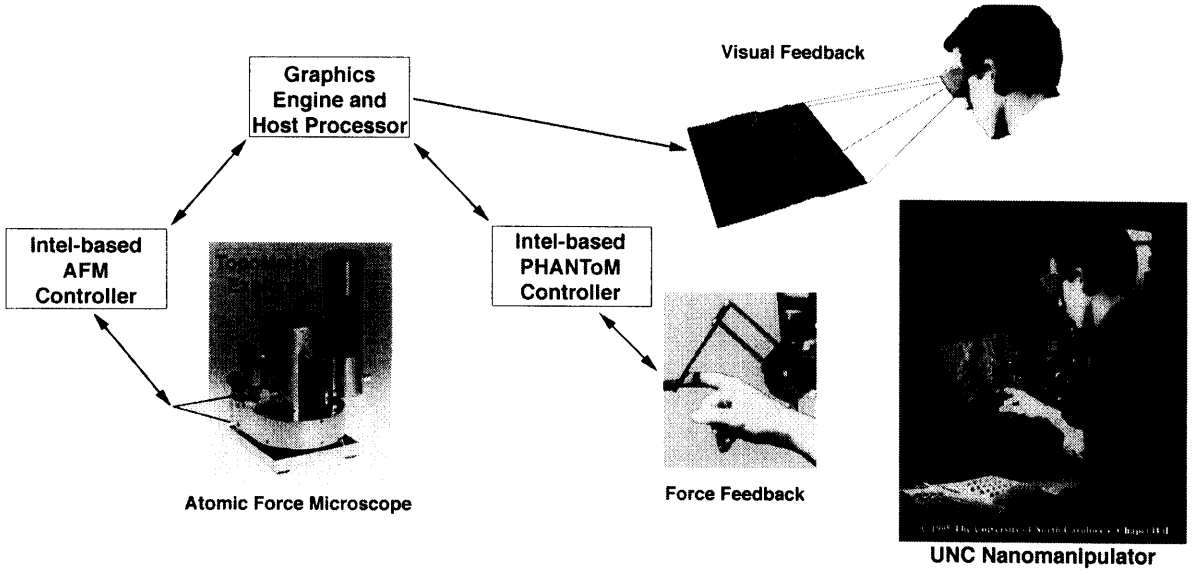


Figure 1: The Nanomanipulator is an example of a computer graphics system to which force feedback has been added. The system comprises a graphics engine, force display and microscope controller. These computers are connected through an Ethernet switch, which provides a dedicated 10Mb/s path to each. The force, graphics and microscope programs each run asynchronously at their own maximum rate. The complete application from the user's point of view is shown in the lower right.

Abstract

The Force-Feedback research at the University of North Carolina at Chapel Hill investigates and develops methods for providing high quality force feedback to the users of real applications. Armlib is a distributed force-feedback software library with support for high quality hard surfaces. We present Armlib as an interface to the PHANToM and the Nanomanipulator as an application using Armlib.

Category: Systems.

CR Categories: C.3 (real-time systems), H.1.2 (User/Machine Systems), I.3.7 (Virtual reality), I.6.8 (distributed simulation)

Keywords: haptic, force, scientific visualization, interactive graphics, virtual environment, telepresence, teleoperation, virtual world.

† Contact Author: CB #3175, UNC, Chapel Hill NC 27599-3175. (919) 962-1906 chenju@cs.unc.edu

1. Introduction

Force feedback provides direct perception of three-dimensional(3D) objects and directly couples input and output between the computer and user. The Nanomanipulator project uses force-feedback techniques in conjunction with stereo graphics to allow human interaction with objects on the nanometer scale. The addition of force display to computer graphics system provided users with a stronger sense of understanding of the surface structure.

2. The Nanomanipulator

The Nanomanipulator project is a collaboration between the departments of Computer Science and Physics at the University of North Carolina at Chapel Hill. The project is developing an improved, natural

interface to scanning probe microscopes, including Scanning Tunneling Microscopes (STM) and Atomic Force Microscopes (AFM).

The project has studied the control of SPMs, using force feedback to allow the user to feel the surface as the microscope probe touches and/or modifies it. The Nanomanipulator displays a high-quality rendering of the surface being scanned as the data arrives in real time.

Users not only can feel the topography of the surface, but also can feel during the modification of the surface. We are exploring how to modulate friction, stiffness and adhesion of the simulated surface based on measured parameters (friction, adhesion) of the scanned surface.

Scanning Probe Microscopes work by rastering a tip across a surface, sampling its height at locations on a regular grid. They can also modify the surface using either voltage pulses or by physically pressing the tip into the surface. This modification is normally done under program control, with experiments consisting of scanning the surface before and after a change. Normally, there is no feedback during the modification event. The Nanomanipulator uses a force-feedback probe to allow the user to directly control tip motion during modification and feel the changes as they are occuring. This provides much finer control over modification and has enabled new and fruitful types of experiments.

Experience has shown that the nM greatly increases productivity by acting as a translator between the scientist and the instrument being controlled. The scientist can concentrate on interacting with the surface rather than with the interface[Taylor-93].

3. Armlib

To support the multiple research projects involved in force feedback at UNC-CH, and to provide a common interface to the different devices in the laboratory, we have developed a force-feedback library we call Armlib.

3.1 Features

The application programmer does not need to know much about force-feedback technology to use the software library. Some features of the library and their motivation for inclusion are:

Device Independence

It supports several varieties of PHANToM force-feedback device and allows additional devices to be supported by writing a fairly simple "device-driver" for them. The software library automatically scales positions and forces so that an application can use any supported device. To specify which force-feedback device to use, the user simply sets a UNIX environment variable before running the application.

Operate Multiple Devices Simultaneously

The Armlib allows an application to use multiple force-feedback devices simultaneously. This ability is useful for multi-user or telepresence applications, or to provide a user with a force-feedback device for each hand.

Distributed Operation

The library has a client-server structure, with the client half of the library running on the same computer as the application program and the server half running on the computer that has the hardware interface to the force-feedback device. The distributed nature of the library provides an important degree of flexibility in a research environment because it has the ability to run the main application on almost any machine connected to the Ethernet. Additionally, the separation of the server and client tasks allows the server computer to devote 100% of its processing power to the force feedback task to maintain an high update rate (in excess of 500 Hz).

High Performance

The library includes a number of features to provide high-quality forces. These features include asynchronous message passing between client and server to reduce wait time and the ability to download hard surfaces and surface textures to the server.

3.2 Intermediate Representations

[Adachi-95] shows the importance of an *intermediate representation* (their term, which we adopt) for a force model that is updated infrequently by the application code but allows high update rates on the force server. Armlib implements two general intermediate representations, *plane and probe* and *point-to-point spring*, and extensions to these types (multiple probes, friction, discontinuity prevention).

Plane and Probe

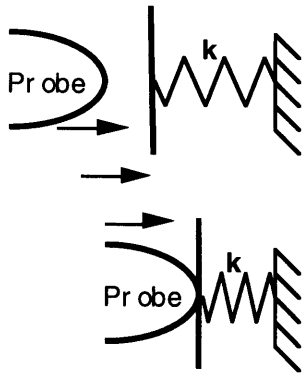


Figure 2: A hard surface is approximated by a plane connected to a spring. When the probe encounters the plane, a spring force with spring constant k is applied. Very high k produces a surface that feels hard. Force is applied normal to the surface.

In the *plane and probe* model, the force server keeps models of a plane in the working volume which the probe can contact. [Adachi-95] When the probe penetrates the plane, a restorative spring force that depends on the depth of the penetration is applied. This provides a surface with controllable sponginess against which the user can push (see figure 2).

Using this model, the application computes a local planar approximation to the surface at the user's hand location each time through its main loop. This allows the user's hand to slide around on a firm plane (with force updated at around 1 kHz), with the plane's position being updated by the application based on local information (at around 20 Hz).

Figure 3 shows how this works in the Nanomanipulator. The Nanomanipulator takes advantage of local surface approximations and surface textures to allow force update rates in the 1000Hz range, while continuing to refresh the graphics display and the representation of the local surface at a more moderate 20 Hz.

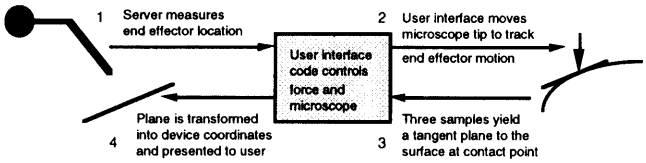


Figure 3: Nanomanipulator application determines a local plane approximation to send to the force server. As the probe moves, the tip tracks it on the surface and sends a new plane equation describing the local surface.

Point-to-Point Springs

The complexity of the model maintained by the force server is limited, since each additional computation performed by it reduces the force update rate. Some applications cannot produce a rapidly-computable local approximation to the force. Examples are rigid-body simulations with many colliding bodies and molecular dynamics simulations of all atoms in a protein. For these applications, there is no simple approximation to the whole calculation, so any force applied directly from the model will have a slow update rate.

In order to keep the forces on the probe stable, there must be some sort of smooth change in force between simulation updates as the probe is moved. To provide this, Armlib employs a method that uses a simulated spring to connect the probe endpoint to the point of contact in the simulation, as shown in figure 4.

The method is the same used in [Surles-92] for mouse-based interaction. It was implemented on the graphics host by Yunshan Zhu using a device-specific predecessor to Armlib. We have re-implemented it in the Armlib force server.

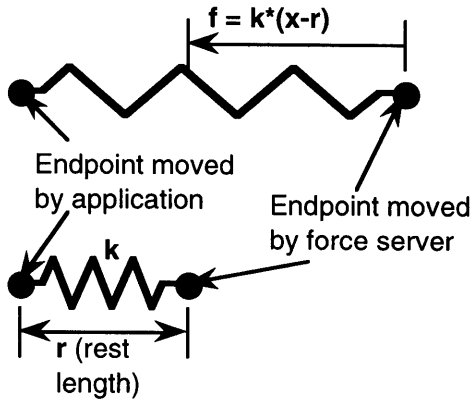


Figure 4: Point-to-point spring, where the one point is moved by the application (at maybe 1 Hz) and the other is moved by the force server (at around 1 kHz). The two are coupled by a spring with constant **k**.

In this method, the application controls the motion of one endpoint of the spring at its slower update rate, while the other end of the spring follows the probe motion at the force update rate. The spring adds force both to the probe (pulling the user's hand towards the point of contact) and to the application (adding forces into the simulation). Adjustment of the spring constant controls the tightness of the coupling between application and probe; a looser spring produces small forces in the application while a tighter spring causes more discontinuity in the force when the application endpoint moves.

In order to prevent the user from moving the probe too rapidly, it is possible to add viscosity into the force-server loop. This makes it feel as if the probe is moving through oil, and tends to keep the probe from moving large distances (and thus adding large forces) between simulation time steps.

3.3 Surface Friction

The surface model described above presented forces to the probe only in the direction normal to the surface. This feels to the user as if all surfaces are made of oiled glass, with the probe tending to slip off convex surfaces and into concave regions.

[Adachi-95] demonstrates the importance of surface friction in allowing the user to explore a surface without slipping. [Minsky-90] implemented a 2-D static friction texture using surface slopes. We have implemented a friction model that includes both static and kinetic components and is rapidly computed. Adjustment of the parameters produces surfaces that feel like concrete, sand, rubber, skin, or cloth. Figure 5 shows the parameters of our model graphically; an explanation follows.

The friction model is that of a surface populated by snags being probed by a flexible tip. When the tip is not stuck in a snag, it moves across the surface

opposed by a friction force (with coefficient of kinetic friction *kK*) that is linearly dependent on its velocity. When the probe encounters a snag, it sticks there until the probe moves more than *dSnap* units away from the sticking point. While snagged, a force pulling the tip to the center of the snag (with spring constant *kStick*) that is linearly dependent on distance takes effect.

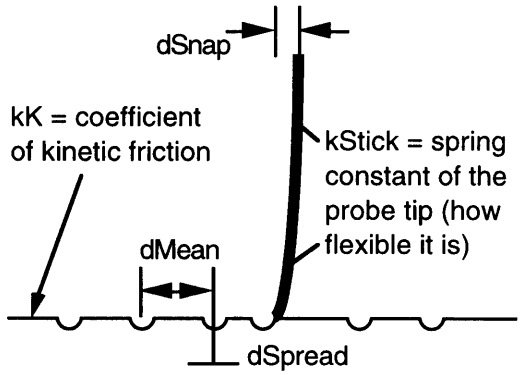


Figure 5: Surface friction model. The tip slides across the surface against viscous friction *kK* until it hits a snag. Snags populate the surface with mean distance *dMean* between them, uniformly distributed within *dSpread*. The tip sticks in the snag, bending with spring constant *kStick* until moved more than *dSnap*, then jumps free.

The snags tend to hold the probe in place on the surface even when the user is not moving it. This provides a natural "station keeping" on surfaces with high snag density (such as sandpaper).

The snags are placed around the surface with a mean distance between snags of *dMean*, uniformly distributed within *dSpread*. In fact, we populate the surface with snags dynamically, rather than laying them out statically (which would be very difficult given the changing parameters and surfaces). After leaving a snag, the tip encounters another placed with uniform probability between *dMean-dSpread/2* and *dMean+dSpread/2* units away, regardless of the direction traveled.

An important characteristic of this friction model is that it is rapid to compute the force each iteration of the server loop, even though the behavior can be quite complex. This is important to avoid reducing the feedback loop update rate, so that we can still present hard surfaces to the user.

4. Publicly Available Resources

The source code and documentation for the software library are available from the address: http://www.cs.unc.edu/Research/force

References

[Adachi-95] Adachi, Yoshitaka, Takahiro Kumano and Kouichi Ogino, "Intermediate Representation for Stiff Virtual Objects," *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS'95)*, March 11-15, Research Triangle Park, NC. pp. 203-210.

[Colgate-93] Colgate, J. Edward, Paul E. Grafing, Michael C. Stanley and Gerd Schenkel, "Implementation of Stiff Virtual Walls in Force-Reflecting Interfaces," *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS'93)*, pp. 202-208.

[Mark96] Mark, William, Scott Randolph, Mark Finch, James Van Verth and Russell M. Taylor II, "Adding Force Feedback to Graphics Systems: Issues and Solutions," to be published in *Computer Graphics: Proceedings of SIGGRAPH '96*, August 1996.

[Minsky-90] Minsky M., Ming Ouh-young, O. Steele, Frederick P. Brooks Jr., M. Behensky, (1990), "Feeling and Seeing: Issues in Force Display", Proceedings ACM Siggraph Symposium on Interactive 3D Graphics", 1990, 24(2), pp 235-243.

[Surles-92] Surles, Mark C., "An Algorithm With Linear Complexity For Interactive, Physically-based Modeling of Large Proteins." *Computer Graphics: Proceedings of SIGGRAPH ;'92*, Volume 26, Number 2, July 1992. pp. 221-230.

[Taylor-93] Taylor, Russell M., Warren Robinett, Vernon L. Chi, Frederic P. Brooks, Jr., William V. Wright, R. Stanley Williams and Erik J. Snyder, "The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope," *Computer Graphics: Proceedings of SIGGRAPH '93*, August 1993. pp. 127-134.

Acknowledgments

Support for this work was provided by grant number RR02170 from the National Institutes of Health National Center for Research Resources, Frederick P. Brooks, Jr., PI. The project is also supported by NSF HPCC ASC-9527192, NSF ARI DMR-9512431, and NSF CISE CDA-9504293 grants for nM. The SARCOS arm was provided by ARPA. The Argonne Remote Manipulator is on loan from Argonne National Laboratories.

We thank Frederick P. Brooks, Jr. and William V. Wright, investigators on the NIH grant, for support and ideas throughout this work. We thank other students, in particular Kimberly Passarella-Jones, Bill Mark, Scott Randolph, Mark Finch, James Van Verth, Brian Grant for work on Armlib and the ideas presented here.

Incorporation of Haptics into a 3D Interactive Workbench for Geophysical and Geological Data

M.E. Clark[1], P.K. Williams[1], D. Stevenson[2] , K.Smith[2] and N.J Archibald[3]
[1]WMC Resources Ltd, [2]CRC for Advanced Computational Systems, [3]Fractal Graphics
emails: mclark@aigpe.com, p.williams@wmc.com.au, duncan@cbr.dit.csiro.au, kevin@cbr.dit.csiro.au, nja@ned.dem.csiro.au

INTRODUCTION

WMC Resources Ltd's geophysical and geological staff at the Kambalda Mine in Western Australia have commenced a project in conjuction with the Australian Cooperative Research Centre for Advanced Computational Systems and Fractal Graphics to use virtual reality platforms to enhance 3D interactions and 3D visualisations of complex geophysical data sets. The Phantom will be incorporated into a virtual reality platform to faciliate editing and modeling of voxel data and to provide additional feedback on aspects of the data..

The Mine Environment

Exploration and planning in a mine environment involves the combination of geological data and remotely-sensed geophysical data on rock properties gathered from the surface, drill holes and mine openings. Both data sets are used to construct 3D interpretations of the geology, 3D models of the geophysical properties such as magnetic susceptibility or electrical resistivity (Willliams, 1996) and 3D models of specific rock types.

The complexity of the data sets and the high value associated with improving exploration success and decreasing mine development require rapid, interactive interpretation and modeling of the data.

The Current State of Technology

3D visualisation of geological data, drill hole and mine development information is standard in the mining and exploration industry. Platforms are generally high-end workstations. PC-based systems are used, but as yet are not the industry standard due to the high-end graphics requirements, data volumes and the need for data security.

Input for this data is usually 2D CAD, GIS, and vector data and 1D drill-hole information. Editing of the data is done at the 2D and 1D input level.
3D visualisation is generally achieved through surface-rendering techniques.

The Future

3D visualisation of geophysical data and models will become standard as the progress is made with data processing, inversion modelling and understanding the complex relationship of rock properties to mappable rock units.

The following forces will drive the industry towards the use of interactive 4D virtual reality as a standard for mine and exploration planning:

i) the value gained from efficient use of geological, geophysical and infrastructural data for mine planning and exploration,
ii) the increasing use of remotely sensed geophysical rock property data,
iii) the need for a real-time "picture" of the mine to support mine automation (Vassie *et al.*,1996).

THE PROJECT

Aims and Objectives

The project team aims to apply 3D interactive visualisation and haptics to the interpretation of complex geophysical data sets. The project will be based on developing an existing interactive virtual reality platform into a prototype workbench for use in day-to-day mining and exploration operations. One of the main objectives of the project is to develop a fast, interactive environment using the platform for manipulation of both vector and voxel data with emphasis on geological and geophysical systems. The incorporation of haptics will assist portraying and interacting with the data in a perceptively fluent manner.

Method

The input data will be line, surface and volume data from drill-hole information, surface interpretations and geophysical data. Voxel data will be generated which satisfy boundary conditions and constraints imposed by geological observation andd physical laws. Haptic rendering of the voxel data will be followed by the assignment of attributes to the voxel data to allow haptic manipulation. A high-performance SGI Onyx workstation will be used with a Virtual Workbench (Poston & Serra, 1996) from the Institute of System Sciences, at the National University of Singapore, as the platform.

REFERENCES

Williams,P.K. 1996: Using Geophysics in Underground Hard Rock Nickel Mining- A Question of Vision and Value. pp.55-75 *in* Howarth,D.,Gurgenci, H., Sutherland, and Firth, B., (Eds.) Proceedings of the 1996 Mining Technology Conference, Fremantle, WA, 10-11 September. CMTE.

Poston, T. and Serra, L. 1996: Dextrous Virtual Work. Communications of the ACM, May 1996, 29:5, pp.37-45.

Vassie, R., Stokes, A, Gibbs,D., and Durrant-Whyte, H. 1996: Automation of Open-Pit Mining. pp 9-15 *in* Howarth,D.,Gurgenci, H., Sutherland, and Firth, B., (Eds.) Proceedings of the 1996 Mining Technology Conference, Fremantle, WA, 10-11 September. CMTE.

Haptic Interaction with the Visible Human

Karl D. Reinig

University of Colorado Center for Human Simulation

Introduction

The Visible Human Dataset TM represents a complete submillimeter photographic and radiological description of both a male and female cadaver. At the University of Colorado Center for Human Simulation we are working on methods to both graphically and haptically interact with the anatomy represented by the data. The effort has lead us to develop generic algorithms to haptically interact with either voxel or polygonal data.

Before the Visible Human data can be used for haptic interaction it must be both segmented and classified, that is, each voxel in a volume of interest must be labeled with its proper tissue type. We are nearly finished with a one-year effort to completely segment and classify the Visible Human Male. We have also developed techniques to generate both polygons and their texture maps directly from the segmented and classified data.

Haptic algorithms can be developed to interact with volume data, polygonal data, or a hybrid of the two. We originally created (Summer of 95) algorithms to palpate complex polygonally defined surfaces. Polygonal models efficiently define volumetric boundaries. Relatively simple algorithms can be developed to rapidly find the intersections of tools with polygons. And simple Phong techniques can be used to produce smoothly varying surface normals. Since we can make polygonal-based models directly from any of our segmented anatomical structures, we can see them and feel them as well.

We are working through two potential disadvantages of polygonal models for haptic interaction. The first problem is the effort required to produce multiple polygonal objects from high resolution segmented data. The second problem has to do with being sure which tissue types the tool is interacting with based solely on boundary crossings.

For some haptic simulations it has been easier for us to develop the force feedback directly from the segmented voxel data. Our voxel-based algorithms use digital differential analyzer techniques [1] to efficiently determine the tool interaction with multiple tissues. One disadvantage of the voxel-based algorithms is the large memory required to store high resolution volumes of interest.

Algorithms

The anatomical simulators we are developing each involve a tool coming in contact with and possibly penetrating different materials. There is generally some part of the tool which does the shearing and some part that is just drag. For example, the tip of a needle shears tissues during insertion while the length of the needle provides friction on the way in and on the way out. The shearing portion of a scalpel is spread out over the sharp (generally curved) leading edge. The rest of the blade creates drag and can be used to deform materials, but it does not cut. The general problem then is to determine which materials the tool is shearing and which surfaces the tool is in contact with but has not sheared. If these questions can be answered and the appropriate forces generated at 1500 Hz or better for arbitrarily complex models, then you can create effective haptically correct anatomical simulators.

Between our polygonal-based and voxel-based algorithms we can calculate appropriate haptic forces for the general problems described above. Our voxel-based models allow penetration of any number of objects. However, we have yet to implement a good surface friction model directly from the voxel data. Our polygonal-based models currently handle the generic surface and internal shear problems for one object bounded by another. We are in the process of adding the multiple object capabilities of the voxel-based models to our polygonal-based models and the surface friction capabilities of our polygonal-based models to our voxel-based models.

Simulators

The following briefly describes three visually and haptically accurate anatomical simulators that we are in the process of developing.

Needle Insertion Simulator

Our Needle Insertion Simulator (NIS) puts the PHANToM behind a shell machined from the back of the Visible Human Male. The user inserts the needle anywhere they want in the back and attaches the tip to the PHANToM. The shell provides orientation for the user as well as a pivot point for the needle. The pivot point allows us to produce torque in addition to the usual three degrees of force. The user may push the needle in any desired direction. As the needle tip is pushed further into the back, the torque preventing redirection increases. The user feels each anatomical structure and interface that the needle tip passes through, as well as the summation of the drag along the needle length. The user feels the needle bump up against bone or wedge into the elastic tar of the inter-vertebral disk. If they should happen to push the tip into a major artery they feel the subtle loss of resistance as the tip shears only blood while the length of the needle still provides friction. They also feel the pulsation of the blood. The user may request updates of Anterior-Posterior and Lateral X-rays that show the position of the needle with respect to the anatomy (the same as they see in the clinic). They can also request updates of the needle position in the original Visible Human volume. The NIS, which runs on a Pentium processor PC, was demonstrated at the 1995 annual meeting of the American Society of Anesthesiologists (Atlanta) where it won "Best of Show" for scientific exhibits. The University of Colorado Health Sciences Center currently uses the NIS to help train anesthesiologists to do celiac plexus blocks.

Surgical Cutting Simulator

The haptic portion of our surgical cutting simulator uses algorithms similar to those of the needle insertion simulator. The major difference between the surgical simulator and the NIS is the ability to show the 3-D cut in real-time. Real-time 3-D graphics are maintained using a method we call Solid Shells. Briefly, the method of Solid Shells starts with a texture mapped polygonal model at the full resolution of the original visible portion of the volume anatomy. As the topology of the scene changes, surfaces are cut or torn, the texture mapped polygonal model is modified to represent the new surface. Because changes to the model are incremental the method is able to update topology at virtual reality rates. To the user, the model appears to be solid since anything they do to it results in the proper surfaces being rendered. The method requires the graphics power of an SGI maximum impact or better. We will be demonstrating a form of the surgical cutter which allows a surgeon to palpate and cut into an eye during the 1996 Centennial Annual Meeting of the American Academy of Ophthalmology (Chicago).

Dental Simulator

Our dental simulator allows a person to probe a tooth in search of anomalies. In its current form, the user sees and feels the tool tip make contact with a tooth. When they "scratch" a healthy portion of enamel, the surface feels slick and they are unable to penetrate the tooth. When the probe slides into a carries (region of decay), they feel additional surface friction. If they push a little harder in that area they see and feel the probe sink into the hard tar of the decay. When they try to pull out, they must pull back through that same tar giving the classic carries tug back. The tooth that we currently use came (courtesy of Bill Lorensen) in the form of 161 CT slices from a GE industrial scanner. We have modified the data to simulate carries and produced both polygonal surface models, using marching cubes [2], and reconstructed X-rays from the altered data. We use texture maps both for realistic graphics and to designate surface friction and shear strength. This gives us very high resolution control of the haptic parameters.

Conclusion

Haptic feedback has added a tremendous amount of realism to our simulators. Before the PHANToM, users of our surgical cutting simulator felt as though they were moving a wand through air. It was easy to carve up a leg, it was very difficult to create the cuts you wanted. Now they feel the tool-tissue contact. They feel the friction of the cut including variation due to the different materials they are cutting. Without haptics, the Needle Insertion Simulator and the Dental Simulator would hardly be worth using.

With the PHANToM comes a simple challenge. It will give you the position, your algorithms must rapidly determine the appropriate force. With each new application, we are increasing the scope of our algorithms.

1) Foley, van Dam, Feiner, and Hughes, "Computer Graphics - Principles and Practices", Addison Wesly, 1990, pp 74-78.

2) W. E. Lorensen and H.E. Cline. "Marching cubes: A high resolution 3d surface reconstruction algorithm". Computer Graphics, 21 (4), July 1987.

Interacting with 3-Dimensional Medical Data Haptic Feedback for Surgical Simulation

Andrew B. Mor[1,2], Sarah Gibson[2], Joseph T. Samosky[3]

abm@ri.cmu.edu gibson@merl.com jsamosky@mit.edu

ABSTRACT

This paper describes different methods tested to haptically explore voxel-based data acquired through medical scanners. The goal of this project is a multi-modal virtual reality surgical simulator. This simulator will allow surgeons and surgical residents to practice and rehearse surgical procedures using patient-specific data. Several methods for calculating the displayed force were investigated and are presented. Additional medical dataset operations, graphical interfaces, and implementation issues are also presented.

I. INTRODUCTION

Through a collaboration between MERL, Brigham and Women's Hospital, MIT, and CMU, we are developing a volume-based surgical simulation system to accomplish three tasks: assist in the training of surgical residents, help established surgeons prepare for procedures by rehearsing the surgery using actual patient data, and provide an aid to navigation which can be used in the operating theater during a procedure. To achieve this goal, this collaboration is focusing on 3 different areas: deformable object simulation, real-time volume rendering, and voxel- based haptic simulation. The Phantom, a haptic interface developed by Sensable Technologies, Inc., is currently being used for force display.

Three-dimensional, voxel-based data is the natural medium for performing surgical simulation because it is the native format of scanned medical data, including MRI and CT. Also, these methods will allow us to model the complex interior structures present in human anatomy, which is critically important for deformable tissue simulation. In contrast, surface based models only approximate surfaces in the medical data and cannot accurately incorporate internal structure. By interacting directly with the medical data, instead of a polygonal model fit to surfaces in the data, the Phantom can more accurately represent the underlying structure of the patient.

[1]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA

[2]MERL - A Mitsubishi Electric Research Lab, Cambridge, MA

[3]Massachusetts Institute of Technology, Cambridge, MA

While many researchers have implemented systems utilizing the Phantom, only a small number have attempted to interact with three dimensional, voxel-based data. Avila and Sobierajski at General Electric Corporate Research and Development were the first to utilize voxelized data in a haptic simulation. This voxel-based data presents a novel problem for determining object boundaries and forces to display to the user due to the relative sparseness of the data with respect to the resolution of the Phantom. Additionally, when dealing with segmented voxel data, which consists of a binary classification map, additional filtering methods are required to simulate a smooth surface.

In our current work, three methods for calculating the displayed force were investigated: interpolating between stored normal vectors at each voxel location; trilinear interpolation of smoothed intensities at the vertices with calculation of the gradient to get direction; and factoring the raw binary data with a "Gaussian sphere" around the Phantom position, again using the gradient to calculate the normal direction. In this work, three types of data were utilized: raw binary segmentation data; filtered versions of the above raw data; and subsampled voxel data built from geometric primitives.

II. DATASET ACQUISITION, PREPROCESSING, AND VISUALIZATION

The initial dataset acquired for this project was a T-1 weighted proton density MRI image of a normal male knee. The dataset size was 256x256x124 with a voxel size of 0.625 x 0.625 x 0.9mm. This dataset contains eight million voxels, while a higher resolution dataset currently being worked on contains 24 million voxels. The slices were then hand segmented into the major anatomical structures: bone (femur, tibia, fibula, patella), cartilage (femoral, tibial, and patellar), lateral and medial menisci, and anterior and posterior cruciate ligaments. Hand segmentation was performed due to the difficulties presented to automatic segmentation techniques by MRI data. Unlike CT images, where image intensity is directly related to density, MRI intensity is based on physical parameters which can vary greatly within one structure but sometimes minutely between neighboring bodies.

When viewing the bony structures acquired from the segmented dataset, the surface of the bones do not appear smoothly segmented, as shown in Figure 1. In

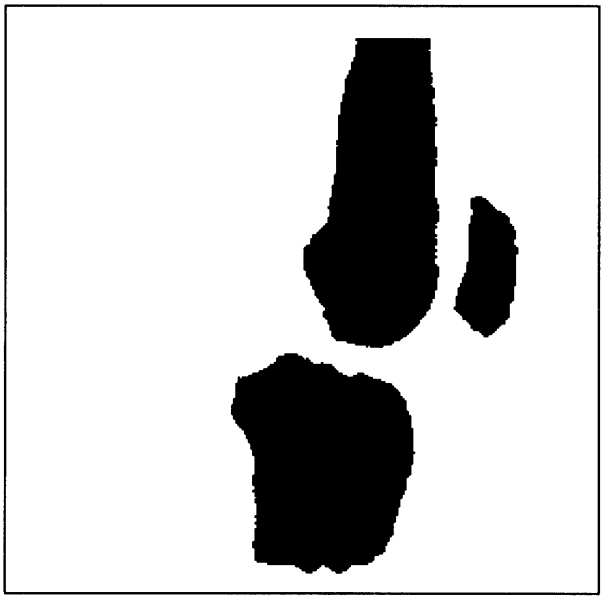


Figure 1 Raw segmentation data

fact, when the dataset is viewed in a direction orthogonal to the original slice plane, it is clear that large inter-slice errors, due to segmentation being performed on a per-slice basis, are present. These types of errors will be present in almost any segmented dataset, due to the difficulties inherent in segmentation, whether performed by hand or by automatic and semi-automatic techniques. Therefore, it was necessary to filter the dataset to achieve a surface that can be displayed haptically to the user. Without smoothing, a small change in the Phantom position could cause erratic behavior in the direction of the displayed force. Batch filtering was performed using a Gaussian filter in the frequency domain. A filtered version of the previous image is shown in Figure 2. The bumpiness present in the original image is still present, but is now surrounded by a smooth gradient of values. Note that this filtering is not required for datasets that are built from geometric primitives and are inherently smooth.[1]

Two methods for visually displaying the dataset were utilized. The first method employs fast volume rendering on a multiprocessor SGI Challenge.[2] This method utilized the 3-D texture memory and implemented the shear-warp algorithm to display depth.[3] The second method employs a flexible 3D sectional visualization system (SectionView) to display the slice of data and where in that slice the user is currently pointing with the Phantom.[4] The two methods are complementary in that the volume rendering presents a global view of the user's position

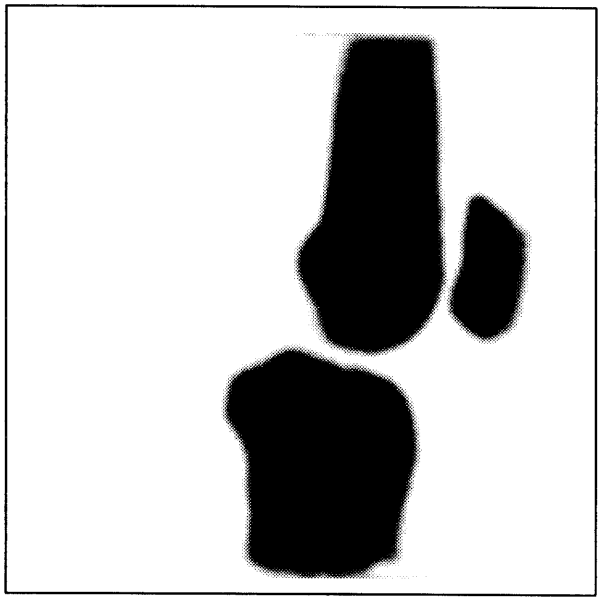


Figure 2 Filtered segmentation data

in the dataset, while the section method presents the local view of exactly what the user is interacting with. If the original MRI dataset is loaded and displayed using the sectional method, the system allows the user to see the original available data around what she is feeling, providing a natural interface to explore an area of interest. To maximize the refresh rate on the haptic interface, the graphical interfaces were run on a separate machine with position data broadcast over an Ethernet network.

III. Methods for Force Display

As mentioned above, three methods of calculating the displayed force vector were investigated. The different methods trade-off speed of execution and storage size of the voxel-based models. The dataset contains eight million voxels, where each voxel can contain multiple datapoints relating to parameters of the voxel.

The first method that was implemented interpolated between stored normal vectors at each voxel center. Utilizing models generated from geometric primitives, each voxel contained both a magnitude and a normal direction. To minimize storage size, the normal direction was stored as three bytes, a byte for the component in each direction. An inverse distance metric was used to interpolate the displayed force from the intensities and normal directions at each vertex location. While this method is fast, it required at least three times the amount of storage compared to using the

intensity alone. Additionally, this method tended to flatten out curves that closely paralleled the axis directions, due mainly to the discretization of the normal vector when stored as a triplet of bytes.

The second method involved placing a "Gaussian sphere" around where the user is pointing and performing local smoothing, followed by application of a gradient operator to calculate the displayed force. This "Gaussian sphere" is just a simple Gaussian filter extended to three dimensions, so that the scaling factor decreases with radial distance from its center. By performing the smoothing at run-time instead of by pre-processing, the number of occupied voxels is minimized, an important consideration for a later part of our research involving deformable, voxel-based objects. This method operates by precomputing the Gaussian coefficients for a range of squared distances. For each update cycle, a 5x5x5 area around the Phantom tip is factored and summed to calculate the current intensity, as in the following equation. In this equation:

$$I_{xyz} = \sum_{i=-2}^{2}\sum_{j=-2}^{2}\sum_{k=-2}^{2} \frac{1}{\left(2\pi\sigma^2\right)^{3/2}} e^{-\frac{d^2}{2\sigma^2}} I_{[x]+i,[y]+j,[z]+k}$$

I_{xyz} is the intensity being calculated; the brackets around the indices i, j, and k imply rounding to the nearest integer, and therefore the nearest voxel location; and d^2 is the distance from the $([x]+i,[y]+j,[z]+k)$ voxel position to the location of the Phantom tip. The smoothing is then repeated at points around the current position to calculate the normal direction. While this method holds great promise for future work and research, it currently does not operate smoothly and displays a very choppy force to the user.

The last method currently being investigated and researched utilizes trilinear interpolation to calculate intensities between voxel centers and then central differences to determine the local surface normal. Trilinear interpolation is an extension of simple interpolation in 1D to three dimensions and is guaranteed to be continuous. As in Figure 3, to calculate the intensity at a point within a cube comprised of the eight surrounding vertices, interpolate first between the vertices along common edges in the x-direction. Then, using the four values returned from that interpolation, interpolate in the y-direction. The two values calculated from these last interpolations are then used to interpolate to the value at the point of interest. Trilinear interpolation is also used to calculate the intensity at six surrounding points to determine the normal direction using central differences. This method does require preprocessing of the data to create a smooth region

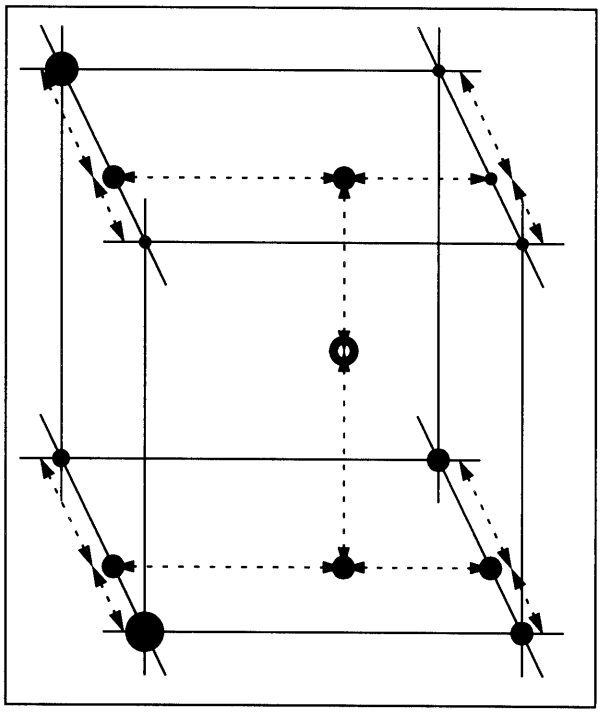


Figure 3 Trilinear interpolation

containing a ramp of values between free space and the object. If the smoothed region is not sufficiently wide, the effective gain in the local region can be great enough to cause unstable behavior. This method convincingly displays the forces associated with the voxel-based data, and provides the cornerstone for a surgical simulator.

IV. Implementation Issues

As with many digital hardware systems, difficulties arise due to problems with the discretization of time. For instance, in a continuous time system, velocity is just the instantaneous derivative of position. But, in the Phantom system, the change in position from one update cycle to the next does not look anything like the actual velocity due to two factors, the discretization of time and the discretization of space due to the encoders. Therefore, velocity of the Phantom tip was calculated as a moving average over the previous 0.01 seconds, using a ring buffer to store the change in position and the elapsed time for each update cycle. Viscosity was implemented in the usual manner, as a force acting opposite to the velocity vector. Viscosity was applied whenever the user penetrated the surface of an object. This could occur because the direction of the displayed force was generated from the local gradient. When the user penetrates past the smoothly varying intensities on the surface of the object, the gradient goes to zero because all the local voxels possess the maximum intensity. A pseudo-friction model was used, where a

viscosity force was applied when moving along the surface of the object. The magnitude of the viscosity was less than or equal to a preset maximum value, to simulate Coulomb sliding friction.

Object rotations were implemented utilizing a quaternion to rotation matrix transform. The quaternion was calculated approximately ten times a second, and the object was "rotated" by rotating the user position in the opposite direction. In this way, the user position is transformed, and then just indexed into the voxel space in the usual manner. These rotation matrices are also sent over the network to the graphical interface to provide feedback as to how the object is rotating.

As was mentioned above, the binary segmentation data was filtered to provide smooth gradient values to the trilinear interpolation. Unfortunately, this filtering did cause some "buzzing" in a couple of locations in the dataset. When two bones were sufficiently close together in the segmentation map, the filtered intensity values between them would be artificially high. For instance, when a byte is used to store the intensity, values as high as 40 were seen between the fibula and the tibia and also between the femur and the tibia. These high intensities in the free space between the bones were large enough to cause unstable behavior of the Phantom. One view of this behavior is to think of the Phantom tip as bouncing between the two sides of a valley, where the sides are formed by the bones in the model, with the bouncing caused by overshoot of the tip. Because of this "buzzing," new filtering techniques, such as morphological filters which will not increase the size of the object, are being investigated.

These methods were all implemented on a Silicon Graphics Indigo2 Extreme equipped with a R4400 processor running at 250MHz. The optimized version of the trilinear interpolation method updated at approximately 4800Hz, well above the accepted minimum of 1000Hz.[5] This method was also ported to a 200MHz Pentium Pro PC running Windows95. On this machine, using the Microsoft Visual C++ compiler, optimized code ran at 6500Hz, a 35% improvement.

V. Conclusion

This research provides a solid groundwork toward voxel-based modeling and interaction for surgical simulation. A fast haptic display method, trilinear interpolation with a local gradient operator, for interacting with static voxel data was implemented and tested. This knee model was shown to a local orthopedic surgeon, who provided helpful feedback. More user input will be garnered during the next stage of development of the overall surgical simulator. Three dimensional voxel-based simulators like this one, because they utilize raw segmentation data instead of surfaces that approximate the data, minimize the approximations needed to accurately display medical data and are more appropriate for surgical simulation and training.

VI. Acknowledgments

The authors would like to thank the MELCO Wellness Business Group for providing the financial support for this collaboration and the Mitsubishi Electric Research Laboratory for providing equipment and resources necessary for this research.

We would also like to thank Dr. Scott Martin of Brigham and Women's Hospital for his input on procedures and his feedback on the feel of the simulator, Gordon Douglass for his support with the computing facilities, and Dr. Shin Nakajima of Brigham and Women's Hospital and Akira Sawada of MELCO for their work in hand-segmenting the knee dataset.

References

[1] Avila, Ricardo and Sobierajski, Lisa, personal communication, Summer 1996.

[2] Fraser, Robert, "Interactive Volume Rendering using Advanced Graphics Architectures," Whitepaper, http://www.sgi.com/Technology/volume/VolumeRendering.html

[3] Lacroute, Philippe G., "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," Stanford Technical Report CSL-TR-95-678, September 1995.

[4] Samosky, Joseph T., "SectionView--A System for Interactively Specifying and Visualizing Sections Through Three-Dimensional Medical Image Data," M.S. Thesis, Massachusetts Institute of Technology, 1993.

[5] Srinivasan, Mandayam A., "Haptic Interfaces," in Virtual Reality: Scientific and Technical Challenges, eds. Durlach, N.I. and Mavor, A.S., National Research Council, National Academy Press, 1994.

[6] Gibson, S., Samosky, J., Mor, A., Fyock, C., Grimson, E., Kanade, T., Kikinis, R., Lauer, H., McKenzie, N., Nakajima, S., Ohkami, H., Osborne, R., Sawada, A., "Simulating Arthroscopic Knee Surgery using Volumetric Object Representations, Real-Time Volume Rendering and Haptic Feedback," MERL Technical Report #TR96-19, 1996.

Haptic Interaction Utilizing a Volumetric Representation

Ricardo S. Avila and Lisa M. Sobierajski

GE Corporate Research & Development
Schenectady, NY 12345

Introduction

One major challenge when developing applications with the PHANToM is the high refresh rate required for haptic interaction [Massie-94]. This refresh rate, which is typically at least 1 KHz, is much greater than the 30 Hz rate that is considered ideal for visual interaction. Achieving this haptic refresh rate in complex, static environments can be difficult, and achieving this rate while interactively modifying and visualizing the environment is an even bigger challenge.

The majority of the haptics work done to date has focused on geometric representations of objects, such as planes and spheres [Salisbury-95]. However, there are several compelling reasons to consider a volumetric representation when performing haptic interaction [Iwata-93][Avila-96]. Some of the strengths and weaknesses of a volumetric approach to haptic interaction are covered in this paper. In addition, we outline a basic approach to haptic interaction utilizing a volumetric representation, including methods for visualizing, feeling, and modifying a volume. Finally, some applications of this approach are discussed.

A volume is considered to be a 3D regular rectilinear grid containing a set of properties at each vertex, or voxel. In its simplest form, a voxel contains a single scalar value, usually indicating the density of material in the local area. An interpolation method is used to obtain a continuous scalar field. We have found trilinear interpolation to be adequate for most data sets.

There are two main reasons why we are investigating a volumetric representation for haptic interaction. First, many scientific and medical scanning devices, such as CT and MR medical scanners and confocal microscopes, produce volumetric data in the form of stacks of images. It is therefore desirable to investigate the benefit of haptic interaction with this representation directly. Second, volumetric representations are well suited for fast local access and are generally independent of scene complexity. Computing forces, modifying object properties, and rendering local changes to a complex structure within a volume can be done very efficiently. For this reason it may be beneficial to convert a geometric scene to a volumetric representation for haptic interaction [Kaufman-93].

One of the main difficulties with a voxel-based representation is the large memory overhead. In our current implementation of these techniques, we allocate 1 byte per voxel for the density value. This is sufficient if we only wish to feel, view, and modify a volumetric isosurface. An additional 3 bytes are allocated if we want to interactively paint a volume with high fidelity. If we wish to visually render the volume as a shaded translucent object we store an additional 2 bytes for an encoded gradient direction, and one byte for the magnitude of this gradient. We use this encoded normal only for visual rendering since force calculations require higher accuracy. Additional volume properties can be assigned through a 1 byte index value. When working with a 256^3 voxel data set the memory required for this implementation ranges from 16 to 128 MB.

Haptic Rendering

The detection of a collision between the haptic sensing location and an object is a fundamental operation of haptic interaction. Fortunately, this operation can be performed on a complex volumetric isosurface in a quick and efficient manner. This can be accomplished by evaluating the density function at the haptic sensing location D_h and performing a simple comparison against the isosurface density value D_i. If $D_h > D_i$ then a collision has occurred. Figure 1 illustrates this with a two-dimensional example. Evaluating a single trilinear interpolation function and performing a

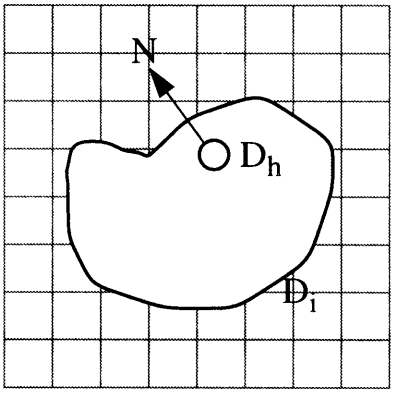


Figure 1: A comparison of the density function at the haptic sensing location D_h against the isosurface density D_i reveals a collision has occurred. The local gradient N is then computed using central differences.

comparison is an inexpensive computation and can easily be performed at the rates required for haptic interaction. Extending collision detection beyond a point contact model is an area we intend to study further.

In order compute a simple linear force response (Hooke's Law) when making contact with an isosurface we need to compute the penetration distance of the haptic sensing location into the isosurface. This is a difficult operation, but it turns out that the scalar density field can be used as an indicator of penetration distance. Using this assumption, we create a set of transfer functions that assign not only stiffness force magnitudes based on scalar density, but also viscous force magnitudes. While this indicator has proven effective for many data sets, there exist several for which this is not a valid assumption. We are investigating additional methods which are more resilient to varying gradients immediately within an isosurface.

Volume Rendering

In our haptic interaction method, we visualize the volumetric scene using an accelerated ray casting technique [Sobierajski-95]. Initial near and far bounds on each ray used to compute the image are obtained from an enclosing polygonal approximation of the scene. During haptic interaction, the view point remains fixed, and the near and far bounds are updated during the modification filtering process. Ray casting is a flexible rendering technique that allows for the display of hard, solid surfaces as well as amorphous objects such as clouds or smoke [Sobierajski 94]. In addition, ray casting can be used to efficiently update the image since only those pixels that potentially show a modified portion of an object must be computed. The cost of updating the image can be distributed across the haptic interaction loop by recasting only a small number of rays during each iteration.

Volume Modification

We perform volume modification by applying a filter to the properties stored in the local voxel neighborhood. Virtual tools such as a paintbrush, an airbrush, a carving knife or a toothpaste tube can be simulated by applying different filtering operations to the properties stored in a voxel. As long as the extent of the filter remains small, this filtering operation can be done efficiently within the haptic interaction loop.

Applications

We have applied these techniques toward both volume visualization and modeling tasks. When visualizing complex objects the ability to interactively feel three-dimensional structure has been found to be a useful tool. If an unimportant structure is obstructing the user's view, it is often desirable to interactively remove the structure to reveal an internal feature. These techniques have also been found to be helpful when modeling a free-form surface. The ability to simultaneously see, feel, and modify provides an intuitive method for creating an object.

References

[Avila-96] R.S. Avila and L.M. Sobierajski, "A Haptic Interaction Method for Volume Visualization," *Visualization '96 Proceedings*, (October 1996).

[Iwata-93] H. Iwata and H. Noma, "Volume Haptization," *IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pp. 16-23 (October 1993).

[Kaufman-93] A. Kaufman, R. Yagel, and D. Cohen, "Volume Graphics," *IEEE Computer* **26**(7), pp. 51-64 (July 1993).

[Massie-94] T.H. Massie and J.K. Salisbury, "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," *Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems,* Chicago, pp. 295-302 (November 1994).

[Salisbury-95] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles, "Haptic Rendering: Programming Touch Interaction with Virtual Objects," *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pp. 123-130 (April 1995).

[Sobierajski-95] L.M. Sobierajski and R.S. Avila, "A Hardware Acceleration Method for Volumetric Ray Tracing," *Visualization '95 Proceedings*, pp. 27-34 (October 1995).

[Sobierajski 94] L.M. Sobierajski and A.E. Kaufman, "Volumetric Ray Tracing," *1994 Symposium on Volume Visualization*, pp. 11-18 (October 1994).