Haptics: From Basic Principles to Advanced Applications

Course Notes for SIGGRAPH '99
August, 1999

Organizer

Ricardo S. Avila	GE Corporate Research & Development

Speakers

Ricardo S. Avila	GE Corporate Research & Development
Cagatay Basdogan	Massachusetts Institute of Technology
Thomas Massie	SensAble Technologies
Diego Ruspini	Stanford University
Kenneth Salisbury	Massachusetts Institute of Technology
Dan Staples	SensAble Technologies
Russell Taylor	University of North Carolina at Chapel Hill

Abstract

This course provides a thorough introduction to haptics covering its history, techniques, and recent advances with a particular emphasis on applications. The first half of the course will serve as a basic introduction to haptic devices, human psychophysics, haptic rendering techniques, and implementation issues. The second half of the course will cover several advanced application areas including assembly and path planning, modeling deformable objects, telemanipulation, scientific applications, and modeling and rendering volumetric objects. The course will conclude both morning and afternoon sessions with hands-on demonstrations.

Course Schedule

9:00 - 9:45	**Salisbury**	**Introduction** History, basic psychophysics, haptic devices
9:45 - 10:30	**Ruspini**	**Haptic Modeling and Rendering** Haptic modeling and rendering techniques
10:30 - 10:45		**Break**
10:45 - 12:00	**Staples**	**Implementation** Issues when building haptic applications Hands-on demonstrations
12:00 - 1:30		**Lunch Break**
1:30 - 2:30	**Avila**	**Volume Haptics** Volume-based techniques Assembly and path planning application
2:30 - 3:00	**Basdogan**	**Deformable Objects** Geometric and physically-based models
3:00 - 3:30	**Salisbury**	**Telemanipulation** Telemanipulation and surgery
3:30 - 3:45		**Break**
3:45 - 4:15	**Taylor**	**Scientific Applications** Molecular docking and nanomanipulation
4:15 - 5:00	**Massie**	**Advanced Applications Demonstration** Live demos of advanced applications

Speaker Biographies

Ricardo S. Avila
General Electric Corporate Research & Development
1 Research Circle, KWC220A
Niskayuna, NY 12309
Phone: (518) 387-6632
Email: avila@crd.ge.com

Ricardo S. Avila is a Graphics Scientist in the Electronic Systems Laboratory at General Electric's Corporate Research and Development Center in Niskayuna, New York. His current research interests include computer graphics, scientific and medical visualization, volume rendering, haptic rendering, and object-oriented software development. He has presented material in the volume visualization courses at Siggraph and Visualization conferences since 1994. He formerly worked at the Howard Hughes Medical Institute applying visualization research toward neurobiology. He received a BS and MS in Computer Science from the State University of New York at Stony Brook.

Cagatay Basdogan
Massachusetts Institute of Technology
77 Massachusetts Avenue, Room 36-758
Cambridge, MA 02139
Phone: (617) 253-5306
Email: basdogan@mit.edu

Cagatay Basdogan, Ph.D., is a research scientist at the MIT Research Laboratory of Electronics (RLE) and a member of MIT Touch Lab since 1996. His research interests include medical simulation and robotics, computer graphics, human-computer interactions, virtual reality technology, haptic interfaces, and computer simulation and biomechanical modeling of human movement. He is the (co) author of over 20 conference and journal publications in these areas. Currently, he is involved in technical development of a project that investigates haptic interfaces, human perception, and cognitive performance in multimodal virtual environments. In collaboration with a medical team from Harvard Medical school, he also leads the efforts at MIT
in developing a surgical simulator.

Thomas Massie
SensAble Technologies
215 First Street
Cambridge, MA 02142
Phone: (617) 621-0150
Email: thomas@sensable.com

Thomas Massie founded SensAble Technologies to commercialize 3D touch interfaces for computers. He received a BS in Electrical Engineering and an MS in Mechanical Engineering while at the M.I.T. Artificial Intelligence Lab. Both his bachelors and masters theses were concerned with development of the PHANToM haptic interface. He continues to conduct research on both the hardware and software aspects of haptic interfaces and provides technical assistance to a number of universities and corporations throughout the world which are incorporating haptic interfaces into computer graphics applications.

Diego Ruspini
Stanford University
Stanford University
Robotics Laboratory, Room 122
Gates Computer Science Building 1A
Stanford, CA 94305-9010
Phone: (650) 725-8810
Email: ruspini@cs.stanford.edu

Diego Ruspini is currently pursuing his doctoral degree in computer science at Stanford University. His research has focused on the robust haptic display of virtual environments from graphical models. This work has lead to the development of a haptic rendering library with the capabilities of taking arbitrary surface geometries as well as shading, friction, and texturing information and displaying them haptically to a user. His research has also extended into the development of physically based models for the real-time simulation of dynamic environments. Mr. Ruspini received his Masters degree from Stanford University in computer science in 1991 based on his work on the dynamic simulation of robotic workcells. Mr. Ruspini received his Bachelors degree in 1989 from the University of California at Berkeley.

Kenneth Salisbury
MIT
Dept of Mechanical Engineering & Artificial Intelligence Lab
NE43-837
545 Technology Square
Cambridge, MA 02139 USA
Phone: (617) 253-2773
Email: jks@ai.mit.edu
Web: http://www.ai.mit.edu/people/jks/jks.html

Kenneth Salisbury received his BSEE in 1975, MSME in 1977 and Ph.D. in Mechanical Engineering in 1982, all from Stanford University. As Principal Research Scientist at MIT in the Department of Mechanical Engineering and as a member of the Artificial Intelligence Laboratory, Dr. Salisbury's research interests focus on mechanism design and control applied to robotic and haptic devices. In addition to his research at MIT, he has been involved numerous consulting and technology transfer activities, holds many patents in haptic and robotic technology, co-authored "Robot Hand and the Mechanics of Manipulation" (MIT Press, 1985), and has authored many papers in the fields of telemanipulation, haptic technology, robot force control, and design and control of robot hands and arms.

Dan Staples
SensAble Technologies
215 First Street
Cambridge, MA 02142
Phone: (617) 679-4500 x4551
Email: dstaples@sensable.com

Dan Staples manages software development at SensAble Technologies, including development of the GHOST SDK which enables software professionals to write haptics applications easily. Prior to joining SensAble, Dan managed development of 3D solid modeling and graphics products at Intergraph and Unigraphics Solutions. Dan holds a B.S. in Mechanical Engineering from Rice University and an MBA from Vanderbilt University.

Russell Taylor
University of North Carolina at Chapel Hill
Department of Computer Science
Sitterson Hall CB 3175
Chapel Hill, NC 27599-3175
Phone: (919) 962-1701
Email: taylorr@cs.unc.edu

Russell Taylor is a Research Assistant Professor in the Computer Science Department at the University of North Carolina at Chapel Hill. He is currently leading the Nanomanipulator project, an effort which provides a virtual-reality interface to Scanning Probe Microscopes including a force-feedback hand controller as part of the interface. Russell has also worked on the force-feedback molecular docking program and on VR projects using force feedback.

Contents

An Introduction to Haptics

Kenneth Salisbury

Dept. of Mechanical Engineering and
Artificial Intelligence Lab.
MIT

Outline

- Introduction
- The Basics of Haptic Psychophysics
- Haptic Devices Past and Present

According to Webster….

Main Entry: *hap·tic*
Pronunciation: ˈhap-tik
Function: adjective
Etymology: International Scientific Vocabulary, from Greek *haptesthai* to touch
Date: circa 1890
1 : relating to or based on the sense of touch
2 : characterized by a predilection for the sense of touch <a haptic person>

Merriam-Webster, Incorporated, http://www.m-w.com/cgi-bin/dictionary

What is haptics?

- Physical interaction via touch
- Uniquely bi-lateral sensory modality
- Touching and interacting with real, virtual and remote environments

Why is it interesting and important?

- Primal
- Intuitive
- Pervasive
- Expressive
- Unexplored….

Nomenclature:

haptic: an adjective, as in "a haptic interface"

haptic interaction: the act of touching objects

haptics: use as a noun, the study/practice haptic interaction

haptically: making use of touch interaction

haptic interface: device permitting human to have touch interaction with real or virtual environments

haptisize - bad English :) but, like sensorize, found

haptical - yikes, no, no.

Nomenclature:

human haptics: human touch perception and manipulation

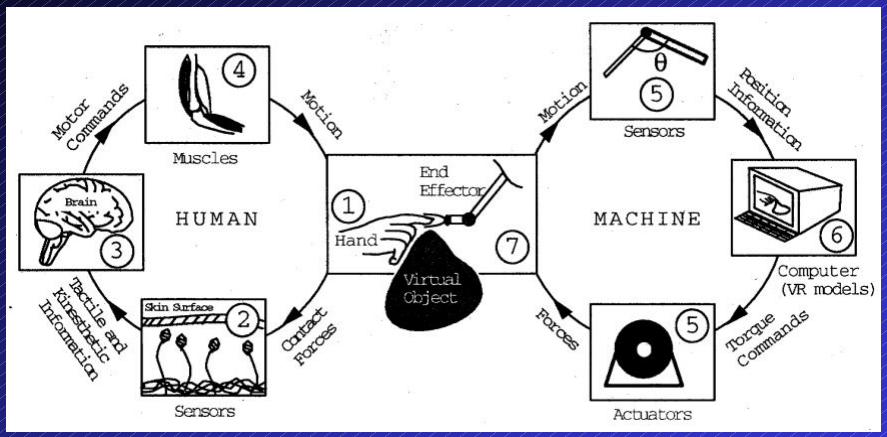
machine haptics: concerned with robot arms and hands

computer haptics: concerned with computer mediated haptics

Haptic interaction occurs in many contexts

- Human haptics
  - every-day manipulation
  - tools, controls
  - music, art, etc.
- Machine haptics
  - autonomous robots
  - remote manipulator systems
  - surgical robots, etc.
- Computer haptics
  - training
  - design
  - entertainment, etc.

Haptic interaction with virtual objects: Information and power flows



Courtesy Mandayam Srinivasan, MIT

Basics Of Haptic Psychophysics

Outline

- What do humans do with haptics?
- Terminology
- Human Sensory System
- Human Sensory Performance
- Multi-Modal Issues
- Human Mechanical Abilities

What do humans do with haptics?

- Exploration and Manipulation
  Motor actions and sensing occur simultaneously
  Manipulation is motor dominant
  Exploration is sensory dominant
- Perception
- Communication
- Expression

Haptic Terminology:

• ***Tactile Information***: "referring to the sense of contact with the object, mediated by the responses of low-threshold mechanoreceptors innervating the skin.. within and around the contact region".

Kinesthetic Information: "Referring to the sense of position and motion of limbs along with the associated forces conveyed by the sensory receptors in the skin around the joints, joint capsules, tendons, and muscles together with neural signals derived from motor commands. (Sometimes referred to as proprioceptive)" [from Srinivasan in Durlach and Mavor, 95]

• Characterized by simultaneous use of multiple information channels.

• In practice, we distinguish between tactile array displays and net force displays.

Human Sensory System - 1

• Determining limb position and motion

- Sensory receptors
  - Joint capsules
    - free nerve endings
    - Ruffini, Paciniform corpuscles (stretch, vibration)
  - Tendons (tension via Golgi organs)
  - Muscles (stretch, rate of stretch via spindles)
  - Skin around joints
    - rapidly and slowly adapting afferents (stretch)
    - Pacinian corpuscles (vibration)
- Muscle commands

Human Sensory System - 2

- Determining contact conditions and object properties: Information sources
  - Cutaneous mechanoreceptors
    - near surface, high spatial resolution
      slowly adapting (SAI, Merkel)
      rapidly adapting (RAI, Meissner)
    - more deeply, low spatial resolution
      slowly adapting (SAII, Ruffini)
      rapidly adapting (RAII, Pacinian)
  - Muscle commands

Human Sensory System - 3

- Determining contact conditions and object properties: Quantities derived from cutaneous mechanoreceptors
  - temporal and/or spatial information
  - normal indentation
  - lateral skin stretch
  - relative tangential motion
  - vibration
  - micro texture
  - shape (at mm size)
  - compliance
- Net sensations are determined by integrating kinesthetic and contact information - no local synapses so all data goes to brain

Human Sensor Performance: resolution and sensitivity

- Absolute detection thresholds
  - surface texture .1 micro-meters
  - static skin displacement 20 micro-meters
  - transient temperature variations .05 C
  - 2 point resolution 1 mm at fingertips
  - localization resolution .15mm
  - position reproduction 2 mm at fingertips
  - pressure .03 Newton/cm^2

Human Sensory Performance: human hand's JNDs

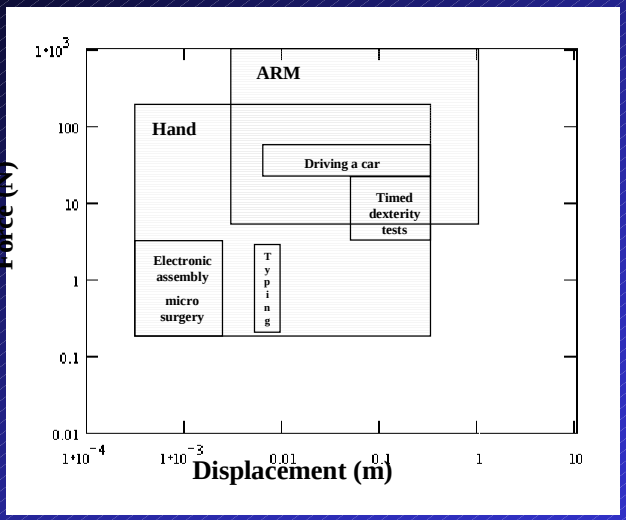
- Just noticeable differences (JND) in active touch
  - length 10%
  - velocity 10%
  - acceleration 20%
  - force 7%
  - stiffness 3% (softer surfaces) - 8% (hard surfaces)
  - viscosity 14%
  - mass 21%
  - rigidity perceived at 25N/mm

Multi-Modal Issues

- visual information *strongly* influences haptic perception
- auditory information *weakly* influences haptic perception
- haptic sub-modalities of vibration, tactile array and temperature stimulation *enhance sense of presence*
- spatial and temporal *registration* of vision, haptics and audition are important

Understanding multi-modal performance demands is critical for guiding technical development.

Human Mechanical Abilities



Courtesy of Lynette Jones, MIT

Haptic Devices Past And Present

Outline

- Haptic stimulation modalities
- Basic device characteristics
- Example devices: Passive
- Example devices: Active
- Other stimulation modalities
- What makes a good haptic interface

Haptic stimulation modalities

- force and position
- tactile
- vibration
- thermal
- electrical

Basic device characteristics

- degrees of freedom (number of joints)
- active and/or passive (force reflecting or not)
- grounding location (grounded versus exo-skeletal)
- sensing quality (resolution, maximum and dynamic range)
- actuator quality (resolution, maximum and dynamic range)
- bandwidth

Example Devices: Passive

- Ground-based
  - keyboards, knobs
  - trackballs, mice, pens
  - joysticks
  - MicroScribe-3D (Immersion)
- Exo-skeletal
  - Dexterous Hand Master (U. Utah/EXOS)
  - Gloves (VPL, Virtual Technologies)
- Hand-held
  - Optical (Optotrack)
  - Electromagnetic devices (Polyhemus, Ascension)
  - Accelerometer devices (InterSense)

Example devices: Active, Exo-skeletal

- 1-6 degrees of freedom
  - UTAH/Sarcos Research Arm
  - CyberForce (Virtual Tech.)
  - Rutgers Master (Burdea, Rutgers Univ.)
  - PERCRO Human Interface (Scuola Superiore S.Anna)

Example devices: Active, Ground based - 1

- 1 Degree of freedom
  - Steering Wheels
    - Hard Driving (Atari)
    - Ultimate Per4mer (SC&T[2])
- 2 Degree-of-freedom
  - Pens and Mice
    - Pen-Based Force Display (Hannaford, U. Wash)
    - MouseCAT/PenCAT (Hayward, Haptic Tech., Canada)
    - Feel-It Mouse (Immersion)
  - Joysticks
    - Force FX (CH Products)
    - Sidewinder Force Feedback Pro (Microsoft)

Example devices: Active, Ground based - 2

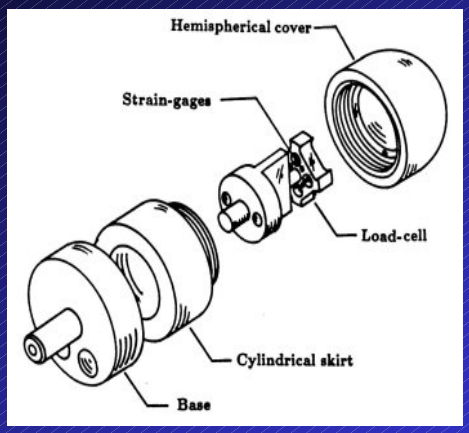
- 3 Degree-of-freedom
  - PHANToM (SensAble Technologies)
  - Impulse engine (Immersion)
- 6+ Degree-of-freedom
  - Teleoperator masters (MA-23, Argonne, CRL)
  - Freedom 6/7 (Hayward, MPB Technologies)
  - 6DOF (Cybernet)
  - PHANTOM Premium 6 DOF

Other stimulation modalities

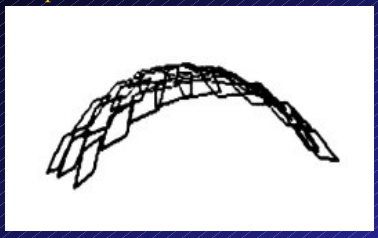
- Vibration and tactile arrays (Howe, Harvard)
- Thermal stimulation (Ottensmeyer, MIT)
- Tactile and Thermal Glove (Scuola Superiore S.Anna. Italy)
- Electrical (Bach-y-Rita)

A Story ...

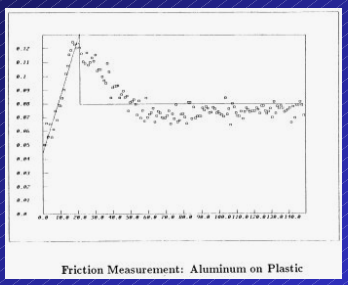
A Force Sensing Fingertip



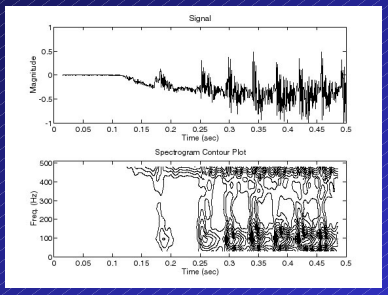
Shape



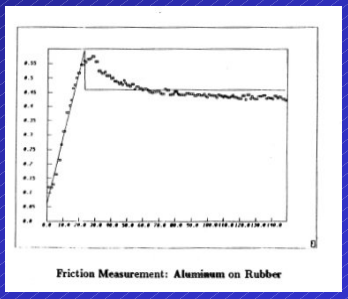
Friction/Texture



Dynamics

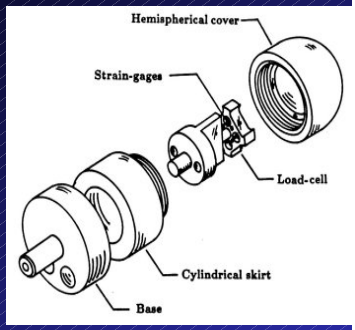


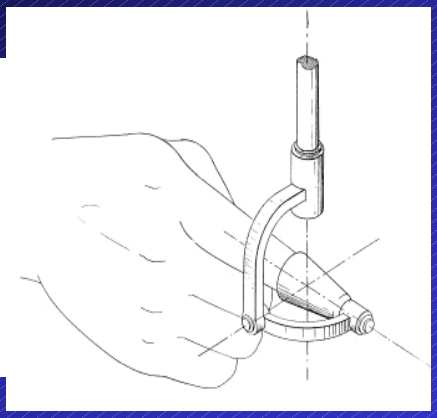
time →



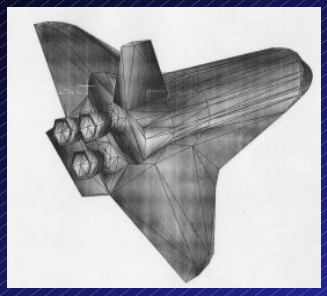
time →

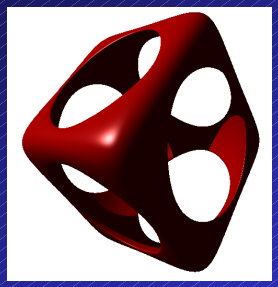
Touching and being touched...

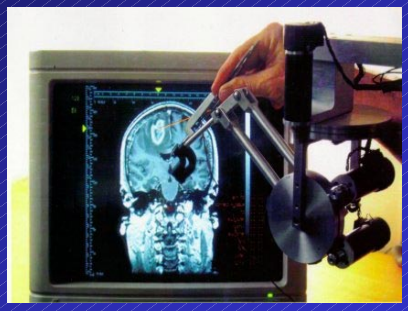





PHANTOM







What makes a good haptic interface? - Performance 1

- Transparency and fidelity are the goals
  - how do we get them?
- Good intrinsic mechanical behavior
  - low mass
  - balanced
  - high structural stiffness
  - high structural resonance frequency
- Easy to backdrive
  - low mass
  - low friction

What makes a good haptic interface? - Performance 2

- Efficient transmission
  - low friction
  - impedance matched
- Good sensing of interface state
  - resolution
  - dynamic range
  - low hysteresis
- Good actuation
  - bandwidth
  - resolution
  - dynamic range

What makes a good haptic interface? - Performance 3

- Some considerations and tradeoffs
  - Contrast and bandwidth are important
  - Consider point versus whole-hand interactions
  - Consider tool versus finger interactions
  - Complexity goes up by N!
  - More degrees of freedom result in *lower* performance for given cost/volume
  - High stiffness can lead to high friction
  - Bandwidth limited by lowest structural resonance so keep K/M large
  - Larger M requires more power flows for a given bandwidth

What makes a good haptic interface? - Market viability

- Intrinsic safety
- Low cost
- Convenience
- Reliability
- Ready-to-use
- Extensible
- Proper weighting of performance specifications

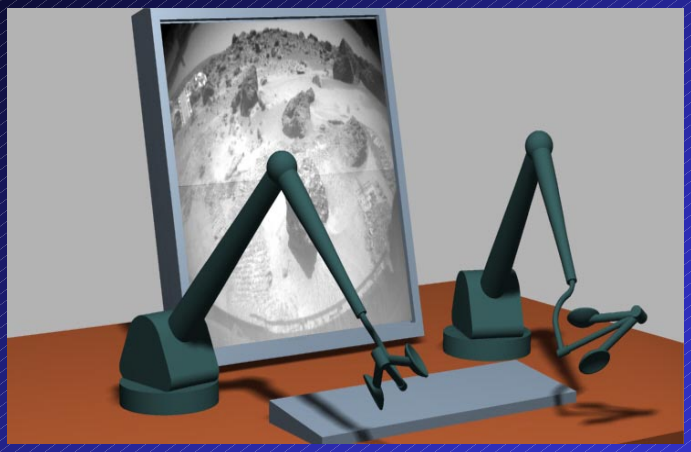
The Future

- **Challenges**
  - We need to:
    build the right devices
    provide the right software
    make it extensible
    find the right markets
  - Computer haptics should be:
    pervasive
    smaller, cheaper, faster...
    convenient
    for multiple fingers, hands, persons
    shared

The Future

- **Opportunities**
  - Building applications
    concrete
    abstract
  - Providing means for
    communication
    expression
    exploration
  - Understanding humans

MarsScape - Virtual and Remote Geology



Experiments with Enhanced Telesurgery

Photo courtesy Hank Morgan

Suggested Reading - 1

Brooks, Jr.., F. P., M. Ouh-Young, J. J. Batter, and P. J. Kilpatrick. "Project GROPE: Haptic Displays for Scientific Visualization," Computer Graphics: Proc. SIGGRAPH '90, August 1990, 177-185, Dallas, TX.

Burdea, G. "Force and Touch Feedback for Virtual Reality" John Wiley & Sons, Inc., 1996.

Durlach, N. I. and Mavor, A. S. (Eds.) "Haptic Interfaces," in *Virtual Reality: Scientific and Technical Challenges*, Report of the Committee on Virtual Reality Research and Development, National Research Council, National Academy Press, 1994.

Hayward, V. Astley, O.R. 1996. Performance Measures for Haptic Interfaces. In Robotics Research: The 7th International Symposium. Giralt, G., Hirzinger, G., (Eds.), Springer Verlag. pp. 195-207.

Cutkosky, M. R., & Howe, R. D. (1993). Dynamic Tactile Sensing, IEEE Trans-Actions on Robotics and Automation, 9(2), pp. 140-151.

Suggested Reading - 2

Salisbury, J.K. and M.A. Srinivasan, "Phantom-Based Haptic Interaction with Virtual Objects," IEEE Computer Graphics and Applications, September-October 1997, Vol. 17, No. 5. IEEE Computer Society.

Sheridan, Thomas B, "Telerobotics, Automation and Human Supervisory Control," MIT Press 1992.

Tan H., et al, "Human Factors for the Design of Force-Reflecting Haptic Interfaces" Dynamic Systems and Control, ASME DSC-Vol. 55-1.

Shimoga, K. B. (1993). A Survey of Perceptual Feedback Issues in Dexterous Telemanipulation: ... (Parts I and II). Proceedings of VRAIS 1993, pp. 263-279, Seattle, WA.

Vertut, J. and P.Coiffet "Teleoperation and Robotics: Evolution and Development," Kogan Page, London/Prentice-Hall N.J., 1995 (Vols A and B)

Haptics Web Pages of Interest:

Haptics Community Web Page (wonderful collection of information and pointers): http://haptic.mech.nwu.edu

Minsky's Haptics bibliography: (classic collection of literature citations): http://marg.www.media.mit.edu/people/marg/haptics-bibliography.html

Bill Buxton's Directory of Sources for Input Technologies (excellent set of pointers to many types of input devices): http://www.dgp.toronto.edu/people/BillBuxton/InputSources.html

Workshop on Human and Machine Haptics (watch for forthcoming book): http://cdr.stanford.edu/touch/workshop

Haptics-E: The Electronic Journal of Haptics Research http://www.haptics-e.org

Telemanipulation

- In the beginning there were master-slave manipulators
  - terminology, origins, characteristics, etc.
- Commercial Applications
- Surgical Robots and Telemanipulators
- Clinical Application: Minimally Invasive Surgery
- Toward Enhanced Telesurgical Medicine

Terminology

- Master-Slave System
- Teleoperator, Telemanipulation, Telerobotics
- Bilateral versus bimanual
- Force-reflection, Backdrive ability
- Masters: Kinematic replica versus generalized
- Control: Joint-by-joint versus Cartesian frame

Telemanipulation Pioneers

- Raymond Goertz - Argon Nationals Laboratories
- Jean Vertut - CEA, Saclay France
- Ralph. Mosher - General Electric
- Carl Flatau - TRI Corp.
- Tom Sheridan - MIT
- Antal Bejczy - NASA/JPL
- and many more …

Telemanipulators - desirable characteristics

- Back-driveability
  - inertia, friction
- Low Inertia
- High Stiffness
- High natural frequency
- Isotropy
  - inertia, compliance, friction
- Low hysteresis

Telemanipulators are not new

- Began in mid-1940's
- Used in mission-critical operations to enable and extend human dexterity
- Early devices share many component technologies with new surgical robots
- Experience has proven their value and reliability

Where are Telemanipulators Used?

- Nuclear/Hazardous Operations
  - weapons fabrication
  - material handling
  - reprocessing
- Undersea Operations
  - well head maintenance
  - geological/biological research
  - salvage
- Space Operations - Shuttle Remote Manipulator
  - inspection
  - payload deployment/retrieval
  - human support

Commercial Telemanipulation

- Central Research Laboratories
  - Hazardous materials handling
- ALSTOM - Schilling Robotics
  - Undersea operations
- Spar Aerospace
  - RMS - shuttle manipulator
- Jet Program
  - Fusion reactor maintenance

Central Research Labs

http://www.centres.com/index.htm

Introduced first mechanical master slave system in 1949 for nuclear industry. Began electromechanical master slave system development in 1953 (w/ Argon Labs.) Have manufactured and delivered over 8000 master-slave manipulators with installations in over 25 different countries. This represents about 70% of the free world market.

Schilling Robotics (ALSTOM Automation)

http://www.schilling.com

Schilling Development, Inc. was founded in 1985 and delivered its first telerobotic manipulator system just one year later. Designed to be deployed undersea, this remotely controlled "robot arm" could manipulate objects and perform work that was once the exclusive domain of skilled divers. Schilling is now the leading supplier of telerobotic manipulator systems for Remotely Operated Vehicles (ROVS) and manned submersibles used in commercial, scientific, and military applications.

Spar Aerospace

http://www.spar.ca/space/telrbtcs.htm

Spar provides telerobotics and robotics derived technology and systems to support national space agencies' around the world and commercial enterprises' experiments and servicing activities in space, and to assist in such terrestrial activities as environmental clean-up operations, in which human contact could be hazardous. Spar's most famous invention is Canadarm, the space shuttle's robot arm. The next generation of space robotics, the Mobile Servicing System, will be used to build and maintain the International Space Station, the largest international science project ever undertaken.

The JET Project

(*from: http://www.jet.uk*)

JET (Joint European Torus), which is jointly funded and staffed by Euratom and 15 European countries, represents the culmination of many years of fusion research. JET is the world's largest magnetic confinement fusion experiment which aims at confirming the scientific theory of fusion and the scientific feasibility of nuclear fusion for power generation.

Remote Handling Equipment

Robots or Servo-Manipulators?

It has been essential to develop a general purpose remote handling system which can adapt to the changing configurations and conditions of the JET machine. This has been achieved by developing a system which makes use of special Manipulators to extend the operators own arms into the radioactive environment. These Manipulators provide the operator with a sense of 'touch' and together with the associated Closed Circuit TV system, create a sense of being inside the Torus. The net effect is to enable the human operator the do the tasks even though it is being done within a hostile environment. It is not considered appropriate to use a fully programmed or robotic type of approach for JET remote maintenance.

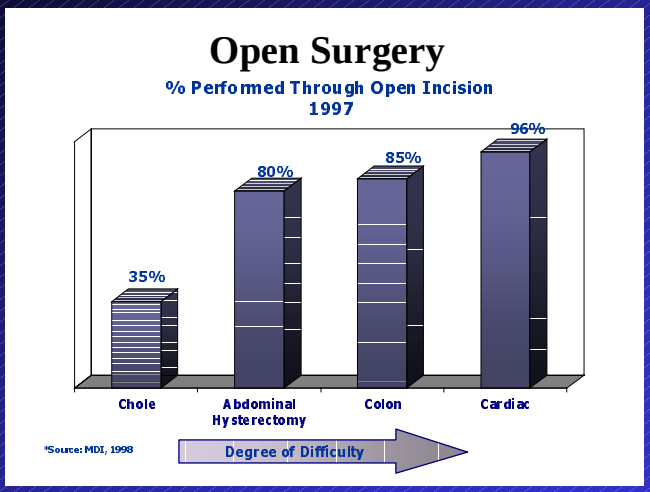
Telemanipulation for Medicine?

- Why do it at all?
  - Alignment with anatomical features
  - Precision fixturing and shaping
  - Access in confined spaces, minimally
  - Mobility, Dexterity
  - Enhanced performance through
    - scaling
    - filtering
    - "melding"
    - information overlay
    - shared motion control

Medical Robotics - the first wave

- Computer-aided fixturing and cutting
- Minerva
  - Neurosurgery
  - http://dmtwww.epfl.ch/imt/robchir/Minerva.html
- Integrated Surgical Systems
  - RoboDoc - Hip Replacement
  - http://robodoc.com

Minimally Invasive Telemanipulator?



Why Use Telemanipulators for Minimally Invasive Surgery (MIS)?

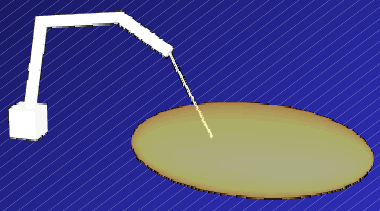
- Increased dexterity and precision
- Improved intra-cavity range of motion
- Visual immersion
- Intuitive motion
- Tremor reduction
- Scaled motions
- Regular, integrated interface
- Collaborative potential...

Medical Telerobotics - the second wave

- Computer Motion, Corp.
  - AESOP camera positioner
  - Zeus Surgical Robot
  - http:\\www.computermotion.com
- Intuitive Surgical, Inc.
  - Intuitive Surgical System
  - http:\\www.intusurg.com

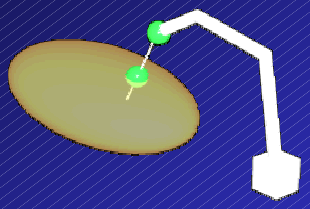
Kinematics of MIS Systems

- JPL RAMS
  - active constraint at entry point
  - 2 freedoms allocated to stationary entry constraint
  - http://robotics.jpl.nasa.gov/tasks/rams/homepage.html



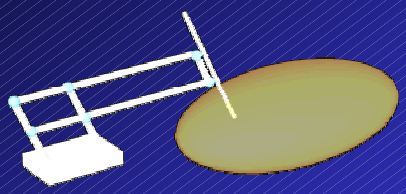
Kinematics of MIS Systems

- Computer Motion
  - natural center at entry point
  - passive freedoms allow stationary entry point



Kinematics of MIS Systems

- Intuitive Surgical
  - remote center at entry point created by linkage



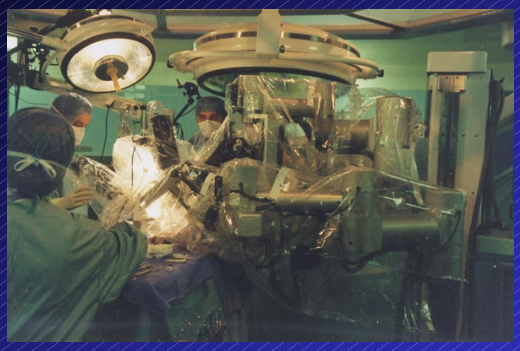
Component Technologies

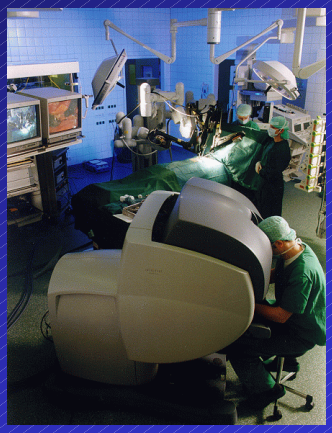
- 6 degree-of-freedom mobility + grip
- Mechanical cable transmissions
- Indexed (clutched) motion
- Scaled motion and force
- High positioning resolution
- Moderate force reflection
- Computer mediated closed loop servo control
- Real-time coordinate transforms between master and slave
- Stereo viewing of remote scene

MIS Telesurgical Tools

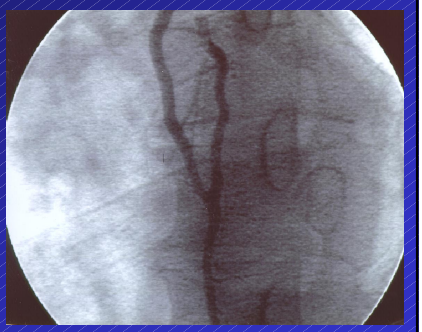
Pictures courtesy Intuitive Surgical, Inc.

Surgical Telerobotics in the Operating Room





Closed-Chest Cardiac Bypass



Toward Enhanced Telesurgical Medicine

- Motion Constraints
- Information overlay
- Multiple Surgeons
- Remote Surgeons
- Training
- Preoperative Planning

General Telerobotics References

- Sheridan, Thomas B, *Telerobotics, Automation and Human Supervisory Control*, MIT Press 1992.
- Vertut, J. and P.Coiffet *Teleoperation and Robotics: Evolution and Development*, Kogan Page, London/Prentice-Hall N.J., 1995 (Vols A and B)
- Burdea, Grigore and P. Coiffet, *Virtual Reality Technology*, John Wiley & Sons, ISBN 0-471-08632-0 (Cloth) June 17 1994.
- Presence: Teleoperators and Virtual Environments, MIT Press. (in print and electronic)
- The NASA Space Telerobotics Program
  http://ranier.hq.nasa.gov/telerobotics_page/telerobotics.shtm

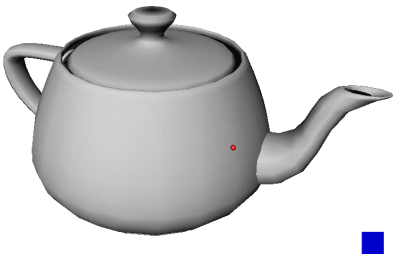
Medical Telerobotics References

- International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), http://neuromedia.ukc.ac.uk/miccai99
- Journal of Computer Aided Surgery, http://jws-edcc.interscience.wiley.com/cas
- ACM Transactions on Human Computer Interaction (TOCHI), http://issl.cs.byu.edu/TOCHI/TOCHI.html

Haptic Rendering

Diego C. Ruspini
Robotics Laboratory
Stanford University
Stanford, CA 94305-9010
ruspini@cs.stanford.edu

Talk Focus (Rendering Graphic Models)

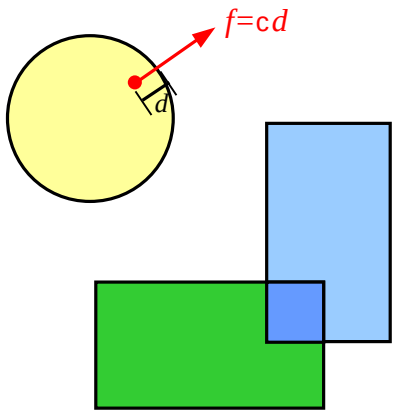


- Scene graph composed of large numbers of graphic primitives
  - surface representation
  - polygons, lines, points
  - overlapping, intersecting and gaps between objects
- Surface Properties
  - Shading, Texture
  - Friction

Outline

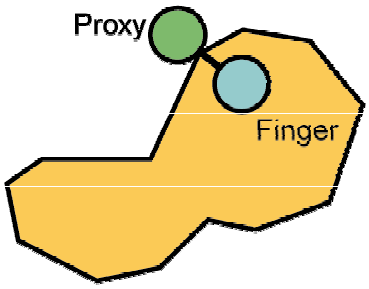
- Penalty Based Methods
- Constraint Based Methods
  - Shading
  - Friction
  - Texture
- Control Architecture
- Conclusion

Penalty Based Methods



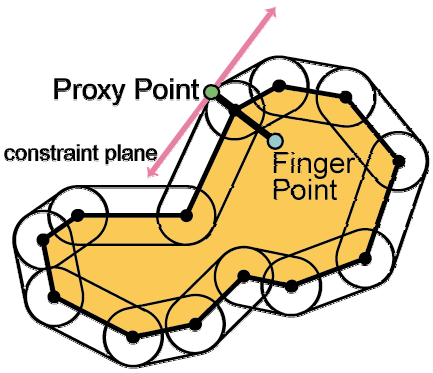
- Force proportional to penetration distance
- Pros
  - Easy to implement
  - sphere, cube, torus...
- Cons
  - thin objects (pop-through)
  - combining objects to form more complex models

Constraint Based Methods



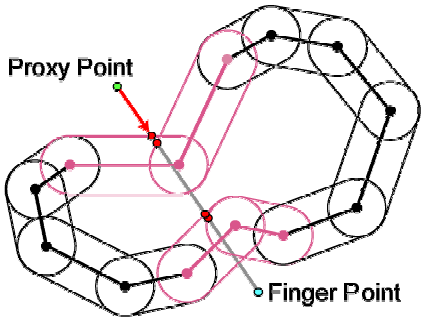
- A representative object
  - constrained by obstacles
  - attached to user
  - by virtual spring
- An idea with many names
  - god-object
  - virtual proxy
  - haptic point
  - SCP
  - IHIP...

Virtual Proxy Basic How To



- Finite size
- No topology needed
- Configuration Space
- Constraint Planes
- Two stage solution
  - Move linearly to goal
  - Update goal position

Move Stage



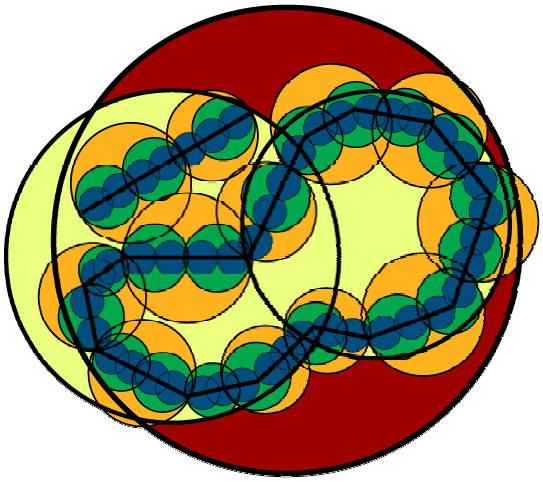
- Move linearly to goal position
- Stop at first contact
- Naive test
  - compare all primitives against path of proxy
- Better test
  - exploit coherence to reduce number of low-level tests

High Level Pruning

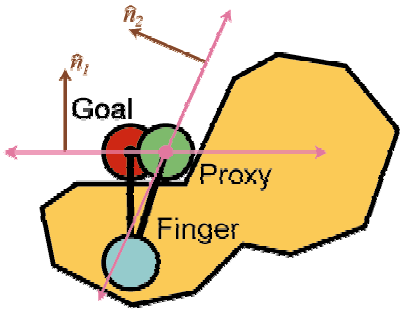
Bounding Sphere Hierarchy

- Many methods to prune scene graph have been proposed for haptic applications
  - Oct-trees
  - Volume Slabs
  - Boundary Space Partitions
  - Axis Aligned Bounding Boxes
  - Object Aligned Bounding Boxes
  - Boundary Sphere Hierarchies

BSH Example



Update Goal Configuration



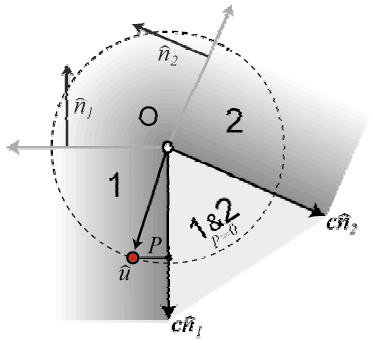
- Given local constraints find goal configuration that minimizes distance to user position
- $\min \mathrm{D} = \|\mathrm{x} - \mathrm{u}\|$ s.t.

$$\hat{n}_1^T x \geq 0$$

$$\vdots$$

$$\hat{n}_m^T x \geq 0$$

Update Dual

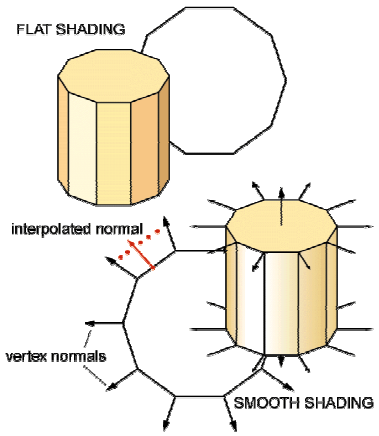


- minimize potential energy between the user and proxy position
- $\min \mathrm{P} = \|\mathrm{p} - \hat{\mathrm{u}}\|$ where

$$\mathrm{p} = \sum_{i=1}^{m} (-c\hat{n}_i) w_i + (\vec{0}) w_0$$

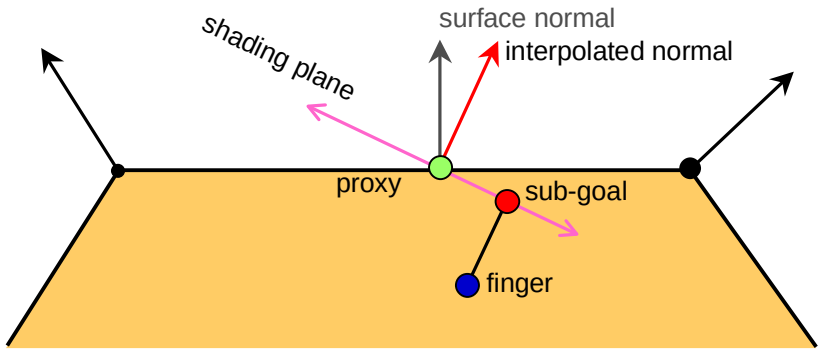
$$\sum_{i=0}^{m} w_i = 1, \ \forall w_i \geq 0$$

Haptic Shading

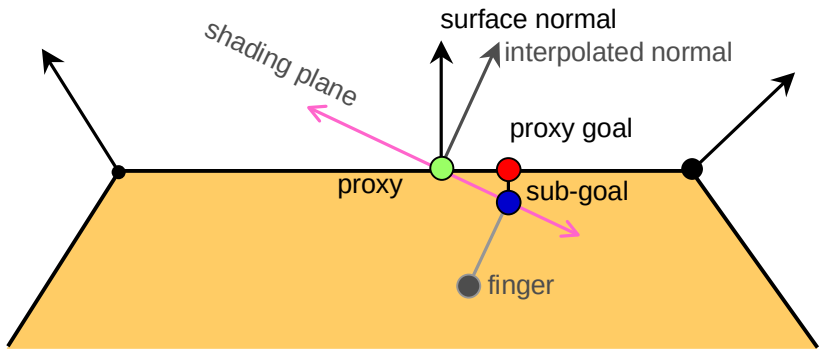


- Graphic Shading
  - eliminate color discontinuities
  - Gouraud, Phong
  - vertex surface normals
- Haptic Shading
  - eliminate force discontinuities
  - Minsky, Morgenbesser

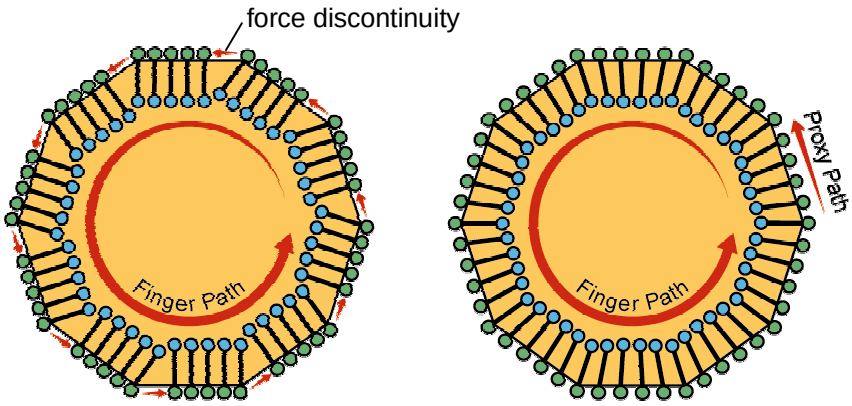
Shading Phase I



Shading Phase II



Shading Example

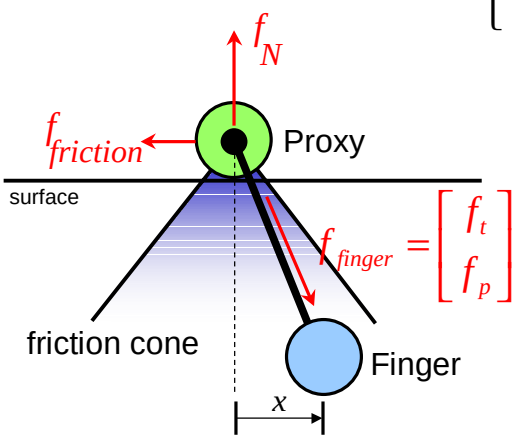


Faceted Cylinder

Shaded Cylinder

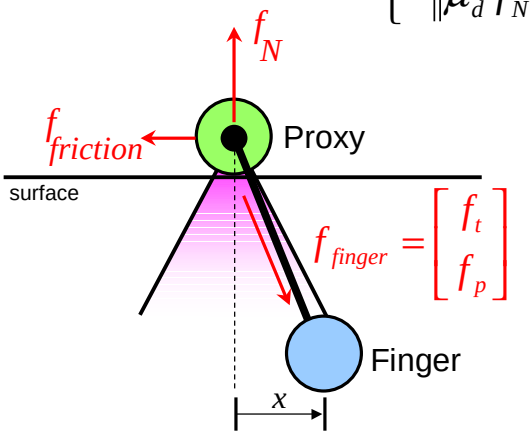
Static Friction

$$f_{friction} = \begin{cases} -f_t & \text{if } \|f_t\| < \|\mu_s f_N\| \\ 0 & \text{otherwise} \end{cases}$$



Dynamic Friction

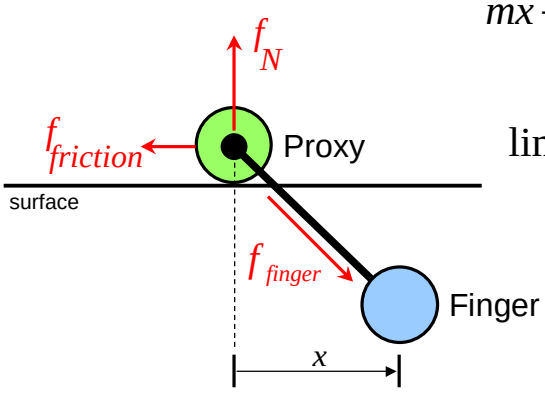
$$f_{friction} = \begin{cases} -f_t & \text{if } \|f_t\| < \|\mu_d f_N\| \\ -\|\mu_d f_N\| \hat{x} & \text{otherwise} \end{cases}$$



Viscous Friction

$$f_{friction} = -b\dot{x}$$

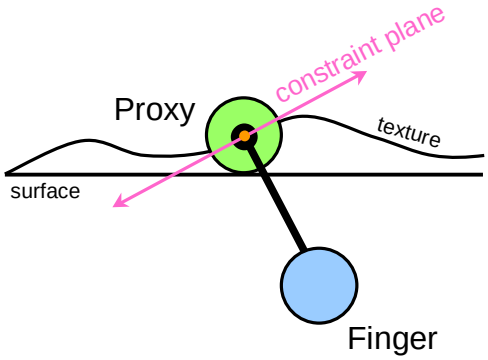
$$m\ddot{x} + b\dot{x} = f_{finger} + f_N$$



$$\lim m \to 0$$

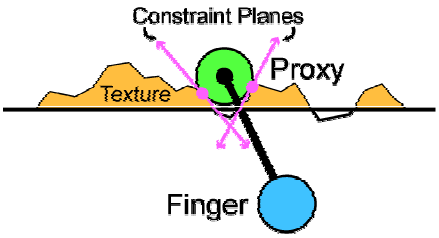
$$\dot{x} = \frac{f_{finger} + f_N}{b}$$

Texture

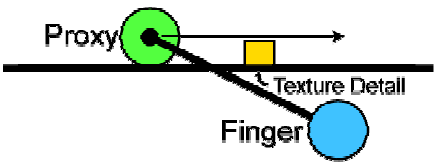


- Image based bump maps
- Compute surface normal displacement
  - Blinn
- Use local surface normal to shade surface

High Fidelity Texture

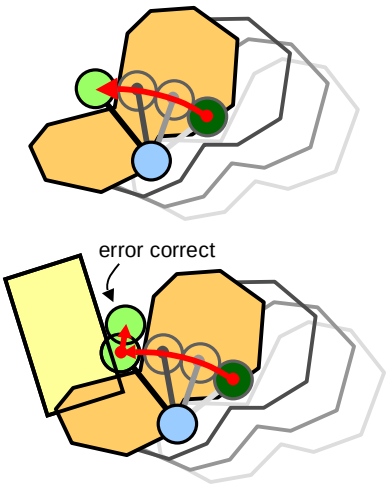


- Allow multiple texture constraints



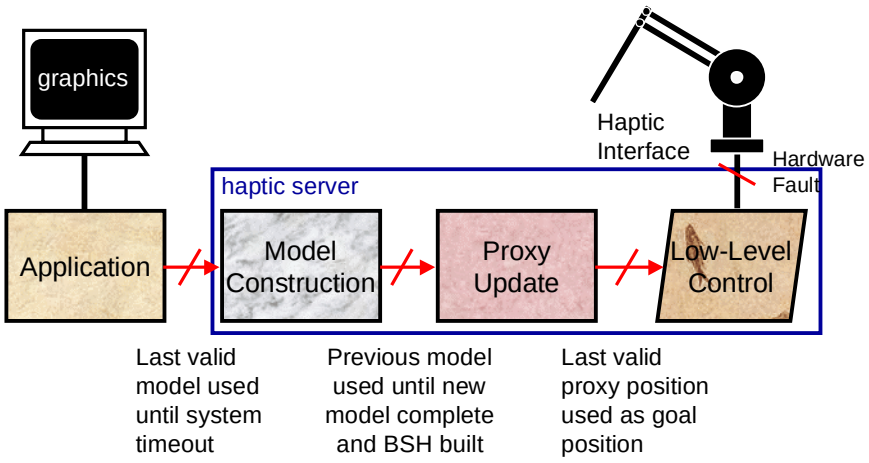
- Check for “obstacles” during move stage to prevent missing of detail

Moving Obstacles

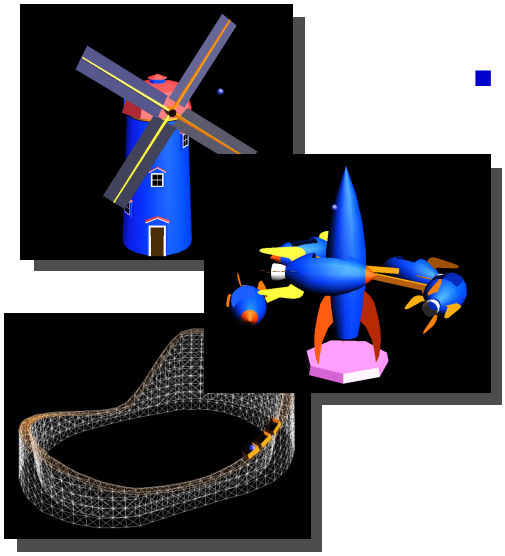


- Single object contact
  - represent proxy and goal position in local frame
- Multi-object contact
  - represent proxy and goal position in frame of obstacle with largest relative velocity
  - $\Delta t < r_{proxy} / v_{max}$
- Force on surface proportional to multiplier weight

Robust Control



Examples/Conclusion



- Constraint Based Methods can effectively display graphic models including:
  - shading
  - texture
  - friction/stiffness
  - dynamics

Extending CB Methods to more complex effectors is still an active topic of research

- line segment(MIT)
- NURB surfaces(Utah)

Haptic Rendering of Graphical Models

Diego Ruspini
Robotics Laboratory
Department of Computer Science,
Stanford University, Stanford, CA 94305-9010
email: ruspini@cs.stanford.edu

Abstract

In this section we will examine the haptic rendering of the polygonal models commonly found in graphic applications. While much of the work described extends to other model representations, such as volumetric or NURB surfaces, these will be discussed in other sections of the course and will only be briefly mentioned here. The haptic rendering of models composed of large number of polygonal primitives is important because these models are the most widely used in creating interactive 3D environments. In this section we will look at how constraint-based methods can be used not only to enforce non-penetration constraints but also demonstrate how these methods can be applied to model properties such as shaded surfaces, friction and texture.

1 Introduction

A haptic interface is a force reflecting device which allows a user to touch, manipulate, create or alter objects in a simulated virtual world. Haptic rendering is the process by which a model specification is taken and appropriate forces are computed to give the illusion of physical contact through the haptic device. In these notes we will focus on haptic rendering of models that represent the surface of the environment using a large numbers of simple polygonal primitives. These bounding surface representations are important because they are the most common representations in use by interactive graphics systems. While many of the concepts presented in this section can be extended to work with other model representations (i.e. volumetric, implicit surfaces,...) these will be addressed in other sections and not explicitly examined here.

While many standards exist to specify the surface of a graphic model, at the lowest level, almost all graphics hardware is only capable of rendering simple polygons, lines and points. All higher order surfaces are decomposed into a set of polygonal patches before being rendered. The advantage of this approach is that the graphics hardware can be made highly optimized for displaying these simple primitives. In building a general purpose haptic rendering system it may be wise to follow the example of the graphics community and focus on a small set of powerful primitives then create a large base of low-level objects most of which will be never used in practice.

Towards this end, in these notes, we will examine some of the methods and algorithms that went into the development of the haptic rendering system "HL." The "HL" system shares many of the same ideas as other advanced haptic rendering systems and therefore forms a good basis for

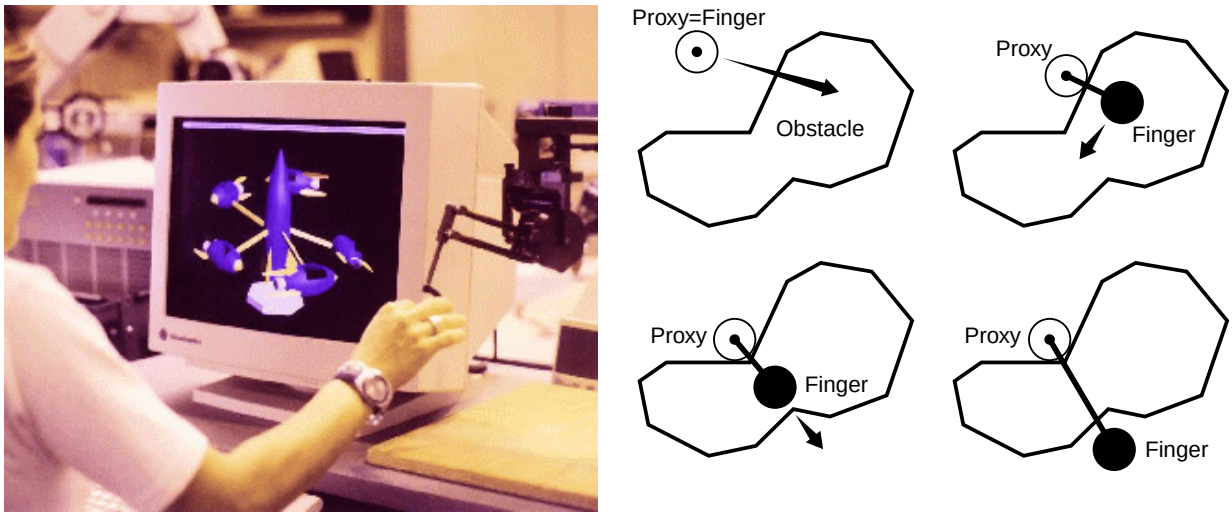


Figure 1: A user is haptically interacting with a dynamic virtual environment (left). The sensation of contact is created by applying forces through the haptic device to move the user's finger to the location of the constrained representative object *(the proxy)*. The virtual proxy moves to locally minimize the distance to the user's finger position subject to the constraints in the environment (right)

examining how constraint-based haptic rendering can be accomplished. In addition to rendering polygonal geometry we will also look at how other graphic information can be used to augment the haptic sensation.

2 Penalty Based Methods

Haptic systems have been around for a number of years. Early haptic rendering systems modeled surface contacts by generating a repulsive force proportional to the amount of penetration into an obstacle. While these penalty based methods, worked well to model simple obstacles, such as planes or spheres, a number of difficulties are encountered when trying to extend these models to display more complex environments.(Slide 4)

When multiple primitives touch or are allowed to intersect it is often difficult to determine what the appropriate restoration force should be. Simply adding the penetration force from each object can result in the creation of large forces that could potentially cause damage to the haptic device or injury to the user. In addition the penetration distance and direction into an obstacle is not always uniquely defined. As a user presses into an obstacle at some point the user's position will be nearer to a surface other then the one into which he or she originally penetrated. When this "pop-through" occurs the user will be actively pushed through the object, resulting in a unrealistic and usually undesirable sensation. Lastly, small or thin objects may not have a sufficient internal volume to create the constraint forces required to prevent the probe from passing through the obstacle.

3 Constraint Based Methods

Another approach first proposed by Zilles et. al [36] presents an alternative method that does not depend on determining the penetration distance into an obstacle. In constraint based methods a

representative object substitutes in the virtual environment for the physical finger or probe. The representative object can be viewed as if connected to the user's real finger by a stiff spring. As the user moves his/her finger in the workspace of the haptic device he/she may pass into or through one or more of the virtual obstacles. The representative object, however, is stopped by the obstacles and quickly moves to a position that minimizes its distance to the user's finger position subject to the constraints in the environment. The haptic device is used to generate the forces of the virtual spring which appear to the user as the constraint forces caused by contact with a real environment [5].

This representative object has been given many names (god-object, haptic point, ideal haptic interface point, surface contact point...) and has been used in a wide variety of systems. Each system differs in the way the environment is defined and how the update procedure is performed but most share some basic similarities. The update rate of the constrained object position must be very high ($> 1000Hz$) in order to achieve realistic high-fidelity force response. In each update loop the current position of the haptic device is found and the location of some representative effector point is computed. Some method of collision detection is utilized to determine if it is possible to move the representative object toward the effector configuration. The representative object is moved as far as possible by some simple motion and from there a new potential direction of motion is found. At the end of the servo loop the error between the user's position and the representative object is used to generate forces on the haptic device. This in effect reduces the error by physically moving the user's position to the configuration of the representative object.

In the system we will describe this representative object is designated as a *proxy* [6]. The virtual proxy was the first approach to look at rendering the "polygon soup" type models that are the most common in graphic applications. The virtual proxy is modeled as a finite massless sphere, and no topology of the objects in the environment is required making it applicable to dynamic environments and scene graphs containing many moving or intersecting obstacles. The virtual proxy framework was also extended to correctly and robustly utilize additional graphical information such as shading normals, friction and texture. An example of virtual proxy moving in the environment is illustrated in figure 1.

In the next section we will look at how the proxy's position is updated during each servo loop. In section 5 a method for simulated smooth curved surfaces using the information normally available for Phong or Gouraud shading is described. Methods to render static, dynamic and viscous friction are described in section 6. Texture is introduced in section 6.3. A short summary of how fast collision detection can be achieved and how dynamic models are introduced is described in sections 7 and 8. The conclusion shows some examples from the current system and describes some future directions.

4 Updating the Proxy Position

From an algorithmic point of view it can be easily seen that the motion of the proxy is very similar to that of a robot reactively moving towards a goal (the user's finger) under the influence of an artificial potential field [18]. When unobstructed, the proxy moves directly towards the goal. If the proxy encounters an obstacle, direct motion is not possible, but the proxy may still be able to reduce the distance to the goal by moving along one or more of the constraint surfaces. When the proxy is unable to further decrease its distance to the goal, it stops at the local minimum configuration.

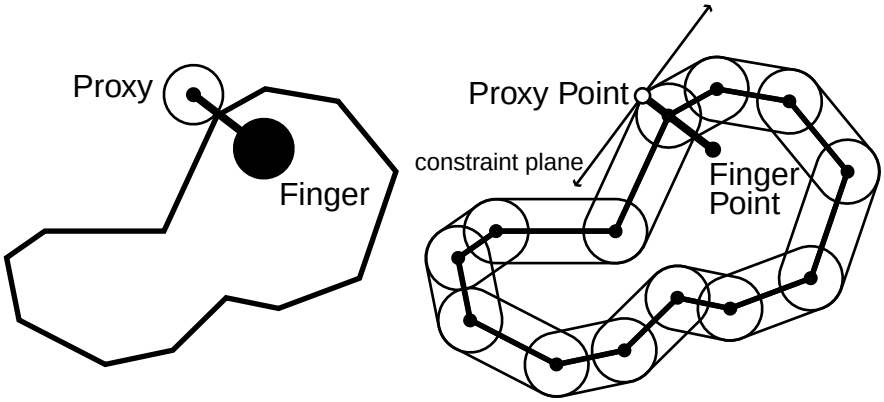


Figure 2: Actual and Configuration Space Obstacle

Many of the concepts used in haptic display have their origins in robotics applications. For this discussion we will model the proxy as a smooth massless sphere. The radius of the proxy is selected both to be large enough to be easily visible for graphic display and to prevent it from falling through small gaps that may exist between the polygonal patches representing the surface of the obstacle. These gaps are common, and checking and fixing a model to eliminate "leaks" is in general very computationally expensive. At each time step the position of the proxy is changed to reduce its distance to the user's finger position subject to the constraints in the environment. Each iteration is divided into two stages: *move* and *update*.

4.1 Moving the Proxy [7]

In the *move* stage the volume swept by the virtual proxy, as it moves along a linear path towards its goal, is checked to see if it penetrates any of the primitives in the environment. The initial goal configuration is the user's finger position but will change as dictated in the *update* stage. If the proxy's path does not collide with any obstacles, the proxy is moved directly to the goal. Otherwise the proxy is moved until it makes contact with the first primitive or primitives along the path.

A naive comparison of the proxy's path with each primitive in the environment, would be unable to achieve a sufficiently high update rate (for haptic display) on any but the most simple models. A common technique to reduce the number of low-level comparisons that must be made is to surround each obstacle with a hierarchy of bounding volumes. In our system a bounding sphere hierarchy is used and is described in detail in section 7. Many other methods of high-level collision detection have also been proposed [14]. All these techniques exploit spatial or temporal coherence to quickly eliminate primitives from the environment that lie too far from the proxy's path to effect its motion. Primitives that are not eliminated by the high-level pruning technique must be checked individually to see if they intersect the proxy's path. This can be accomplished efficiently with algorithm's such as Gilbert's [12] or Lin-Canny [21] which can quickly compute the distance between two convex polyhedron. The low-level test is also similar to the ray intersection test used in ray-tracing applications for graphics. In our system we use Gilbert's algorithm since it requires very little preprocessing and can be used in other parts of the haptic rendering process as we will see below.

Once the point of contact has been found, the proxy's position is updated to this new configuration. At this point direct movement to the goal is no longer possible, but it may still be possible

to reduce the distance to the user's finger position. In the *update* phase this new goal configuration is found.

4.2 Updating the Goal Position 10

If the proxy is currently at the user's finger position no further work is required. In general, however, the proxy will not be exactly at the goal configuration and a new direction, constrained by the contact surfaces, must be found to further decrease the distance to the goal. The available free space around the proxy can be effectively modeled by examining its configuration space [20]. In this space the proxy is represented as a point identifying the center of the proxy. The primitives in contact with the proxy are mapped to *configuration space obstacles* (C-obstacles), consisting of all points within one proxy radius of the original obstacles. For each primitive a unique constraint plane tangent to the configuration space surface and going through the proxy position can then be defined. Each constraint plane limits the potential motion of the proxy to the half-space above the plane. In addition the user's finger position will always be situated beneath the constraint plane. An example of the configuration space, C-obstacles, and the proxy constraint planes is shown in Figure fig:configurationspace.

The intersection of all such half-spaces defines a convex, unbounded polygon which represents locally all points reachable by direct linear motion from the current proxy position. The new goal configuration is the point in the free-space polyhedron nearest to the user's finger position.

The gilbert distance algorithm [12], used during collision detection (Section 7), can efficiently find the minimum distance between two convex bounded polyhedra. In addition to distance, the algorithm, will also find the nearest point on each body. Each polyhedra is defined as the convex hull of a set of points $P = p_1, \cdots, p_n$ and $Q = q_1, \cdots, q_m$. On completion the algorithm will return a set of weights w such that

$$p_{nearest} = \sum_{i=1}^{max(n,m)} p_i w_i, \quad q_{nearest} = \sum_{i=1}^{max(n,m)} q_i w_i \qquad \sum_{i=1}^{max(n,m)} w_i = 1 \quad w_i \geq 0 \quad \forall i. \tag{1}$$

Given that a polyhedra can contain only one point, finding the distance between a polyhedra and a single point is trivial.

As described above the problem can be stated as:

$$\begin{aligned} \text{minimize} \quad & D(x) = \|x - u\| \quad \text{s.t.} \\ & \hat{n}_1^T x \geq d_1, \\ & \hat{n}_2^T x \geq d_2, \\ & \vdots \\ & \hat{n}_m^T x \geq d_m. \end{aligned} \tag{2}$$

where x is the desired goal configuration, u is the current position of the user's finger, and $\hat{n}_i^T x = d_i, 0 \leq i \leq m$ are the equations for the m constraint planes in contact with the proxy. Here we will assume all $\hat{n}_i$ are unit normals.

The specification of our free-space polyhedron, is not in a form that can be directly exploited by the Gilbert algorithm. The polyhedron is un-bounded and the limits of its extent are defined by

the intersection of all the constraint planes, not the vertices of its convex hull. There does exist, however, a dual relationship between this free-space polyhedron and another polyhedron that does satisfy the requirements of the algorithm [11].

Without loss of generality we will assume that the proxy is centered at the origin and that the user's finger position is located at a point one unit away from the proxy position. The frame of reference and unit of measure can be changed if this is not the case. All the constraint planes go through a common point (the proxy position = the origin) and can therefore be defined by their surface normals $\hat{n}_i^T x = 0, 0 \leq i \leq m$.

In the 3D case the proxy's position can be locally constrained by at most three constraint surfaces, all other contact surfaces can be considered redundant. We will call the planes associated with these constraint surfaces the set of *active* planes. All other constraint planes will be considered *inactive*. At first we will assume an oracle has identified, out of all the original planes, which planes will belong to the *active* set. We will designate these planes by their surface normals $n_{a_i}, 0 \leq i \leq m' \leq 3$. Later we will see that the oracle is not necessary to solve the problem.

Considering only the *active* planes and rewriting equation 2 in matrix notation the constraint problem can be written as:

$$\begin{aligned} \text{minimize} \quad & D(x) = \frac{1}{2}(x-u)^T(x-u) \quad \text{s.t.} \\ & N^T x = 0, \end{aligned} \tag{3}$$

where $N = \left[\hat{n}_{a_1} \dots \hat{n}_{a_{m'}}\right], 0 \leq m' \leq 3$ is the matrix containing normals of the *active* constraint planes. Introducing the m' multipliers $\lambda = [\lambda_1 \cdots \lambda_{m'}]$, the Lagrangian for equation 3 becomes

$$L = \tfrac{1}{2}(x-u)^T(x-u) + \lambda^T(N^T x). \tag{4}$$

The minimum configuration is found where the derivatives of L are zero. Taking the partial with respect to x we obtain:

$$\partial L/\partial x \quad = \quad (x-u) \quad + \quad N\lambda \quad = 0. \tag{5}$$

Solving for x we obtain a relation between the solution x and the Lagrange multipliers λ.

$$x = u - N\lambda \tag{6}$$

Substituting for x in equation 4 we obtain a dual for our original constraint equation 3:

$$\begin{aligned} \text{maximize} \quad -P'(x) &= \tfrac{1}{2}(-N\lambda)^T(-N\lambda) + \lambda^T N^T(u - N\lambda) \\ &= \tfrac{1}{2}\lambda^T N^T N\lambda + \lambda^T N^T u - \lambda^T N^T N\lambda \\ &= -\tfrac{1}{2}\lambda^T N^T N\lambda + \lambda^T N^T u \\ &= -\tfrac{1}{2}\left(\lambda^T N^T N\lambda - 2\lambda^T N^T u + u^T u\right) + \tfrac{1}{2}u^T u \\ &= -\tfrac{1}{2}\left(N\lambda - u\right)^T\left(N\lambda - u\right) + \tfrac{1}{2}u^T u. \end{aligned} \tag{7}$$

Rewriting equation 7 as a minimization and noting that u is a unit normal ($u^T u = 1$) the dual solution can be found to be equivalent to the solution of:

$$\text{minimize} \quad P(x) = \|N\lambda - u\|. \tag{8}$$

In this equation $P(x)$ can be thought of as representing the potential energy of the system. The solution x (that minimizes the distance to the user's position) is the configuration that minimizes the potential energy stored in the virtual spring that exists between the user and the proxy.

To solve this equation using Gilbert's algorithm we will at first make the following substitutions. First we will define a space S' such that:

$$S' = \left[\begin{smallmatrix}0\\0\\0\end{smallmatrix}\middle|\, c_{a_1}\hat{n}_{a_1} \,\middle|\cdots\middle|\, c_{a_{m'}}\hat{n}_{a_{m'}}\right] \tag{9}$$

where $c_{a_i}, 0 \leq i \leq m'$ is a constant such that:

$$c_i = \frac{1}{u^T \hat{n}_i} \tag{10}$$

Next we can define a set of $m' + 1$ weights w' such that:

$$w' = \left[w_0 | \frac{1}{c_{a_1}}\lambda_0 | \ldots | \frac{1}{c_{a_{m'}}}\lambda_{m'}\right]^T \tag{11}$$

where $\lambda_i, 0 \leq i \leq m'$ are the terms from the vector λ and w_0 is a new constraint variable whose value we will be define later.

Recalling that $N = \left[\hat{n}_{a_1} \ldots \hat{n}_{a_{m'}}\right]$ it is trivial to show that $S'w' = N\lambda$. Equation 8 can now rewritten as:

$$\text{minimize} \quad P(x) = \|S'w' - u\|. \tag{12}$$

The solution to equation 12 is the point nearest to u in the space spanned by $S'w'$. If the solution we desire is contained in the convex hull created by the columns of S' then the solution can be found by invoking Gilbert' algorithm. To prove that the solution lies in this space we must show that $\sum_{0 \leq i \leq m'} w_i = 1$ and $w_i \geq 0, \forall w_i, 0 \leq i \leq m'$ where w_i is the i^{th} term of vector w.

From the original problem statement $x_p = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$ satisfies the constraints of the system (it is the current valid proxy position) with $D(x_p) = 1$. Given that the solution x must be nearer or at least the same distance as the current configuration we see that $(u - x)^T(u - x) \leq \|u - x\| \leq \|u - x_p\| \leq 1$. Noting from equation 6 that $u - x = N\lambda$ and our original requirement that planes

on the *active* set satisfy $n_i x = 0$, we see:

$$\begin{aligned}
(u-x)^T(u-x) &= (u-x)^T(N\lambda) \\
&= (u-x)^T(S'w') \\
&= (u-x)^T\left(0w_0 + \frac{1}{u^T\hat{n}_{a_1}}\hat{n}_{a_1}w_1 + \cdots + \frac{1}{u^T\hat{n}_{a_{m'}}}\hat{n}_{a_{m'}}w_{m'}\right) \\
&= \left(\frac{(u-x)^T\hat{n}_{a_1}}{u^T\hat{n}_{a_1}}\right)w_1 + \cdots + \left(\frac{(u-x)^T\hat{n}_{a_{m'}}}{u^T\hat{n}_{a_{m'}}}\right)w_{m'} \\
&= \left(\frac{u^T\hat{n}_{a_1} - x^T\hat{n}_{a_1}}{u^T\hat{n}_{a_1}}\right)w_1 + \cdots + \left(\frac{u^T\hat{n}_{a_{m'}} - x^T\hat{n}_{a_{m'}}}{u^T\hat{n}_{a_{m'}}}\right)w_{m'} \\
&= \left(\frac{u^T\hat{n}_{a_1} - 0}{u^T\hat{n}_{a_1}}\right)w_1 + \cdots + \left(\frac{u^T\hat{n}_{a_{m'}} - 0}{u^T\hat{n}_{a_{m'}}}\right)w_{m'} \\
&= (1)\,w_1 + \cdots + (1)\,w_{m'} \\
&= w_1 + \cdots + w_{m'} \leq 1
\end{aligned} \tag{13}$$

Thus the first constraint is satisfied by setting the unconstrained variable w_0 equal to:

$$w_0 = 1 - \sum_{i=1}^{m'} w_i \tag{14}$$

While it is trivial to show that $w_0 > 0$ proving non-negativity for the other elements of w is more problematic. The normals of the constraint planes may be redundant, permitting an infinite number of solutions, some of which may have negative weights. However only the minimal set of planes for which all the weights are positive will be considered as candidate *active* sets. To see the reason for this distinction note that the force exerted by the virtual spring on the proxy is given by the equation:

$$f = k_s(u-x) = k_s(N\lambda), \tag{15}$$

where k_s is some positive spring constant. The individual force applied by a given constraint plane to oppose the motion of the proxy is given by

$$f_i = -k_s(\hat{n}_i\lambda_i). \tag{16}$$

The constraint surface can only push, not pull, on the proxy. Therefore, the force normal to the surface must be non-negative ($n_i^T f_i = -k_s\lambda_i = -k_s c_i w_i \geq 0$). As $u^t n_i \leq 0$ (the user's position is below the constraint plane by definition) we see that $c_i \leq 0$ and therefore $w_i \geq 0$.

Having shown that the desired solution lies in the convex hull of the space defined by columns of S', it is now easy extend the result so that we no longer require prior knowledge of which constraint planes belong to the *active* set. The intersection of all the half-spaces defined by the constraint planes of the original problem is represented in the dual by the union of the convex hull for all possible sets of planes. The *active* set is defined by the polytope whose distance is the smallest to the user's position.

We know have a means to find the new goal position efficiently. Eliminating our assumption that the finger position is a unit length away from the current proxy position. Let $\hat{u} = fracu\|u\|$.

Use Gilbert's algorithm to find the nearest point between $\hat{u}$ and a polyhedra defined by the $m+1$ of the dual space:

$$S = \left[\begin{smallmatrix} 0 \\ 0 \\ 0 \end{smallmatrix} \middle| c_1 \hat{n}_1 \middle| \cdots \middle| c_m \hat{n}_m \right] \tag{17}$$

The goal displacement can be found by:

$$x = \|u\|(u - Sw) \tag{18}$$

where w is the vector of weights that are returned by the distance algorithm. The non-zero elements of w define that set of *active* constraints. The displacement x can be added to the current proxy position to define the next goal configuration. In addition, the force applied by the user on each constraint plane can be found by equation 16, after scaling, to be:

$$f_i = (k_s \|u\| c_i w_i) \hat{n}_i. \tag{19}$$

An example of this dual relation is illustrated in Figure 3. In this example configuration the proxy position is constrained by two constraint planes $\hat{n}_1$ and $\hat{n}_2$. These constraints map to a dual space triangle defined by the origin and the points along the negative normal directions of $\hat{n}_1$ and $\hat{n}_2$. As can be see in the illustration the distance P closest to the finger position direction $\hat{u}$ is proportional to the distance that the goal configuration is away from the current proxy position. As the finger is moved around the proxy position the relationship is maintained. When the user's finger is inside the triangle the distance to the hull is zero and corresponding to the configurations where the proxy is completely constrained.

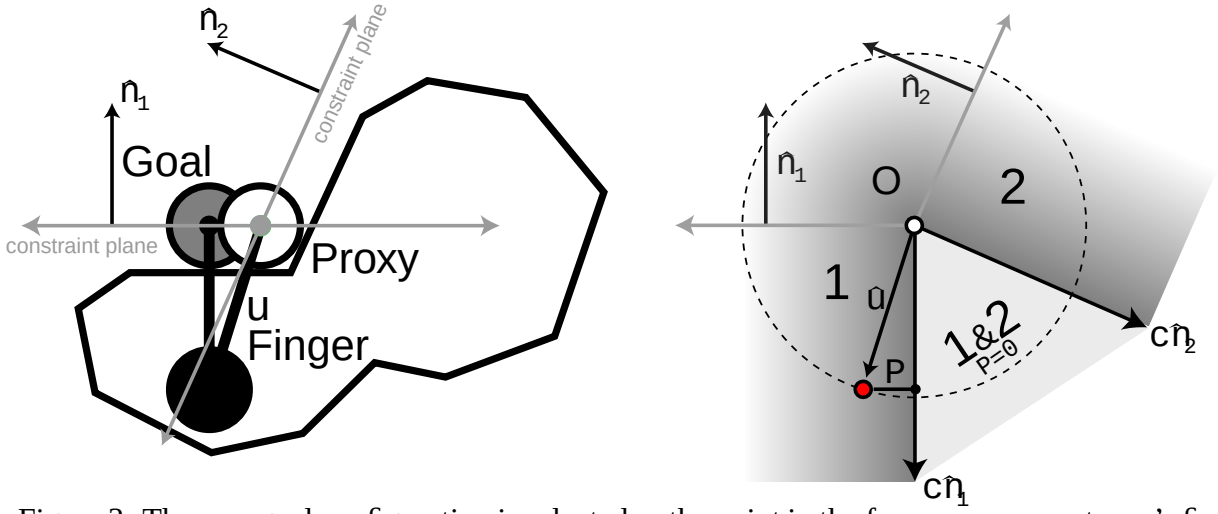


Figure 3: The new goal configuration is selected as the point in the free-space nearest user's finger position (left). Its equivalent dual space representation (right) which represents the change in potential caused by moving from the current configuration.

5 Haptic Shading [12]

As described above the proxy's position is selected to minimize its distance to the user's finger position, subject to the constraints in the environment. In many cases, however, the movement

of the proxy can be altered to create a variety of other useful haptic effects. An alternative minimization is to use information found in many models to allow regular polygonal surfaces to be perceived as if they were constructed out of curved continuous surfaces. As is illustrated in figure 4, in many graphic models, surface normals are defined at the vertices of a polygonal mesh which correspond to the surface normals of an underlying curved surface. To draw a given polygon the graphics hardware interpolates the normals [27] or a corresponding color value [15] for each pixel on the surface. The lighting calculations are performed using the interpolated surface normal information instead of surface normal of the polygon. This has the effect of eliminating abrupt surface color changes between polygon boundaries and giving the appearance of a curved continuous surface. The drawn surface is however still composed of individual polygonal surfaces allowing fast graphic rendering on dedicated hardware.

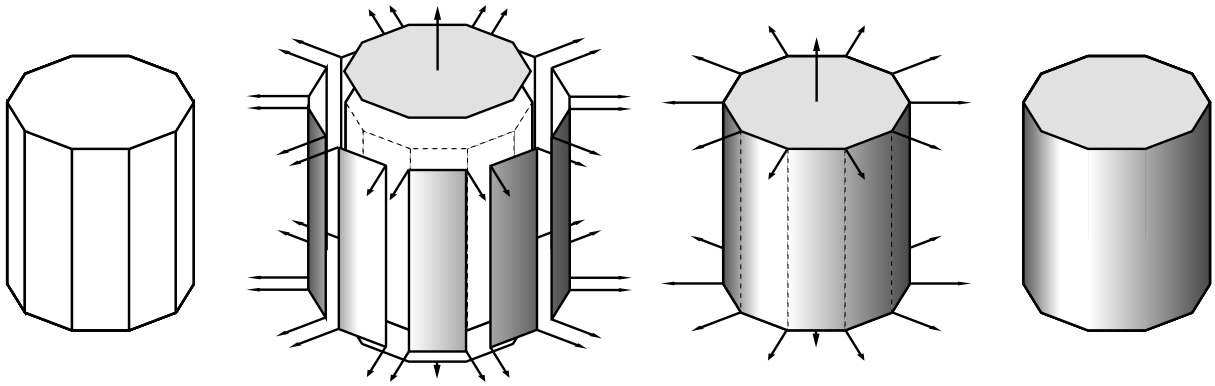


Figure 4: In most graphic systems curved objects are modeled as a set of flat polygonal patches (left). To achieve the appearance of a continuous surface, surface normals are defined on the vertices of each patch (center). The color or lighting model are interpolated over the surface to produce a continuously shaded surface (right). Shaded Cylinder

For haptics these given vertex surface normals can be used to give the sensation the user is touching a continuous, non-faceted surface. This modeling makes use of a haptic illusion first described by Minsky et al. [24]. Minsky was able to haptically display three dimensional height fields on a two degree of freedom planar haptic display. While unable to apply forces in the z (height) direction the illusion of a three dimensional surface was created by applying tangential forces proportional to the slope of the field. The bases for the illusion derives from the disparity between a humans force and position differentiation capabilities. While humans are able to distinguish small force changes, they are relatively incapable noticing small position disparities. By combining generated tangential forces with the normal force created by the physical constraints (in z) of the 2D haptic device an appropriate contact force can be applied to the user's finger. The position of the user's hand in z, however, remains fixed.

Morgenbesser and Srinivasan [25] where the first to try to use this illusion to shade virtual models. In their solution the direction of the normal force is changed while retaining the magnitude caused by the penetration of the original object. Their work, however, required that the topology of the surface be known, limiting its applicability when the environment contained intersecting or moving obstacles. In addition, contact with multiple constraint surface was not considered. Such a case would occur if a user, for example, were following the crevasse created around

the contact region of two side by side shaded cylinders. An alternative approach is to use an alternative minimization to determine the best goal position for the proxy. Since this approach only alters the position of the proxy and not directly the forces applied to the user, stable performance is much easier to guarantee.

When contact occurs, with a polygonal surface containing vertex defined normals, a new local surface normal is calculated by interpolating the normals from the vertices of the polygon. This process is very similar to the interpolation done in graphics but has a few caveats which will be discussed in section 5.1. Once the interpolated normal is known it can be used to define a constraint plane going through the current proxy position.

The haptic shading method proceeds in two passes. In the first pass the new goal solution is found as in the *update* stage described in section 4 [13]. In this pass, however, the interpolated constraint plane is used instead of original for any contact surface containing user supplied normals. This new sub-goal can be thought of as the desired goal configuration of the underlying curved model. This goal position may, however, violate the constraints of the original polygonal geometry since it may lie above or below the true object surface. Instead the *update* procedure is called again but with the original (non-interpolated) constraint planes. The goal configuration generated in the first pass substitutes for the user's finger position [14]. This two pass approach has the effect of finding the nearest valid configuration to the minimal configuration as defined by the interpolated surface normals.

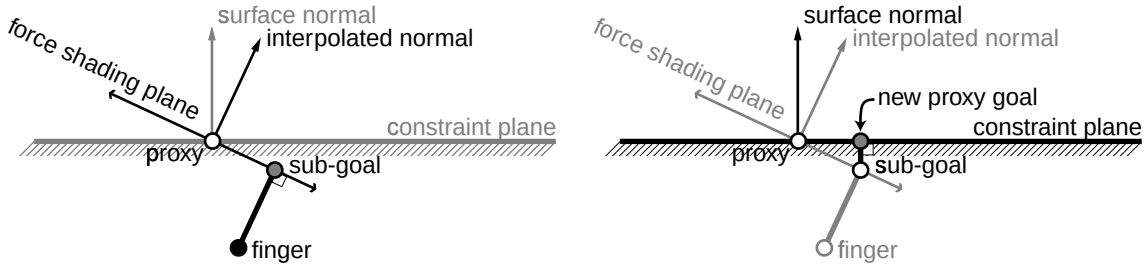


Figure 5: Two pass haptic shading with specified normals

An example of this approach is shown in Figure 5. Note, that after the first pass, the goal position lies below the surface of the object. After the second pass a valid proxy goal on the surface of the original obstacle is found. This goal is to the right of the goal position that would have been found if shading were not applied. If during the next *move* stage no obstacle is encountered, the proxy will move to this configuration and a force pulling the user's finger to right will be applied as would be expected from a object having the surface normal illustrated.

If the sub-goal configuration, after the first pass, is above all the true constraint planes, the sub-goal is first projected back onto the nearest true constraint plane. This ensures that the new sub-goal will always be on the object surface and that surface effects like friction and texture will be handled correctly.

The difference between a haptically shaded surface, a flat surface and the true curved surface is illustrated in Figure 6. In all the figures the difference between the user's position and the position of the proxy are shown as the user's finger follows a circular counter-clockwise path around the object. As seen in Figure 6(a), a strong discontinuity occurs when the proxy reaches each edge of this ten-sided polygonal approximation of a circular obstacle. This results in a force discontinuity

which gives the user the impression of crossing over and edge. In Figure 6(b), surface normals have been specified on vertices of the obstacle. The resulting movement of the proxy shows that the resultant force is always perpendicular to the interpolated surface just as in the case of the true circular object illustrated in Figure 6(c) [15]. The affect of this minimization is to eliminate the large instantaneous changes in force that normally occur at polygon boundaries resulting in a surface that feels smooth and continuous. The discrimination abilities of humans are insufficient to detect the small positional differences between the polygonal and underlying curved surface.

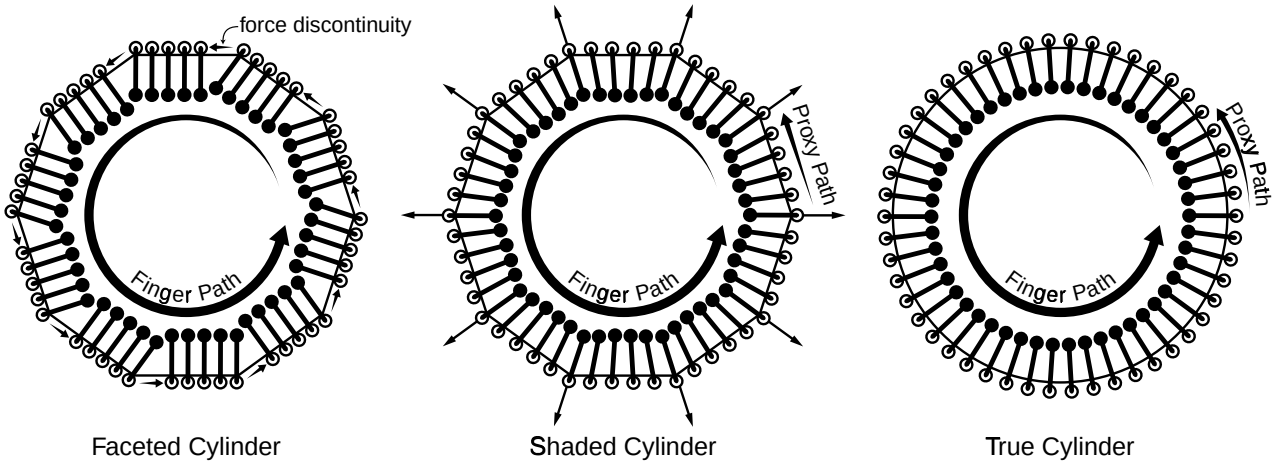


Figure 6: Haptic shading (center) eliminates the force discontinuities associated with moving along a faceted cylindrical surface (left). Although the path of the finger and proxy differ from that of a true cylinder (right) a humans position discrimination ability is insufficient to distinguish the tactile differences between the two displays.

5.1 Determining Shaded Surface Normals

For the shading algorithm described above, the desired surface normal for the shading constraint plane must be found by interpolating its value from the normals defined on the vertices of the primitive. When the contact point is on the surface of the polygon the weights used for the interpolation can be obtained from the collision detection algorithm. Gilbert's distance algorithm returns returns the nearest point on the surface as a weighed sum of a set of vertices on the polygon. These same weights can be used to find the shading normal, see Gilbert et. al [12] for more information. As the contact point may lie on either side of the polygonal primitive a check should be made to ensure that the interpolated surface normal points away from the obstacle. The interpolated normal $\hat{n}$ should be inverted if $\hat{n}' \cdot \hat{n} > 0$ where $\hat{n}$ is the outward normal of the original primitive.

While the finding the interpolated normal for surface contact is fairly straight forward, special consideration needs to be given if the proxy is in contact with one of the edges or vertices of the polygon. As is illustrated in Figure 5.1 the shaded constraint surface is found on the configuration space obstacle and not on the original primitive. On the surface the interpolated surface normal can be mapped to the top and bottom surfaces of the configuration space obstacles, as illustrated in Figure 5.1(b). It is unclear, however, what mapping should be used for points on the surface outside this region. Extrapolating the surface normals will result in boundary values which depend

on all the vertices of the polygon making it difficult to create patches that will form a single continuous surface when placed together. Using the same surface values as the nearest edge or vertex will lead to large differences between the interpolated and true surface normal and will create a singularity where the top and bottom surface meet. Interpolating around the angle formed by the edge will also result in interfering shading planes if the edge is shared by two shaded polygons representing a continuous surface.

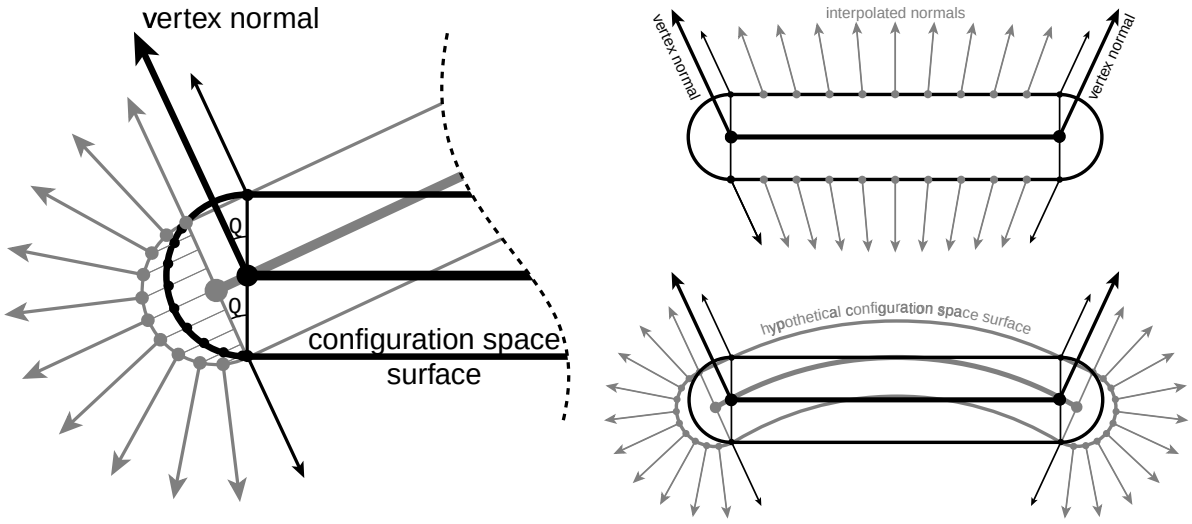


Figure 7: The surface normals of a hypothetical configuration space obstacle are projected onto the true configuration space surface to define the shading normals used to simulate a curved surface.

One approach that does not suffer the problems with the previously mentioned interpolation schemes is to attempt to model the surface normals that would be created if the surface was rotated. If contact is made with the boundary of a configuration space obstacle a hypothetical configuration space surface is created which is tilted so that its upper and lower surface match those of the nearest interpolated edge or vertex normals. An example of this is illustrated in Figure 5.1(a). The configuration space normals of this hypothetical surface are projected onto the actual configuration space boundary too define the shaded surface normals for the configuration space obstacle. Some example configurations are shaded normals are illustrated in Figure 5.1(c). This mapping has an important property in that patches, which share a common edge and have identical surface normals defined on the vertices of this edge, will match at their configuration space boundary. This will ensure that the transition across the surface boundary will feel smooth to the user.

6 Surface Properties

Several researchers [2, 6, 22, 32, 33] have proposed methods to simulate static, dynamic, viscous friction and texture. These methods worked by introducing additional force to simulated the forces of friction by the contact surface and often depended on estimates of the finger's velocity which made stability of the solution very difficult to guarantee. All these effects, however, can be created by restricting or changing the motion of the proxy. This results in a controller that is much more stable and easier to control.

6.1 Static Friction [16]

Static friction (stiction) is particularly simple to model within the virtual proxy framework. The force exerted on the proxy by the user can be estimated by the equation $f = k_p(p - v)$, where p is the position of the proxy, v is the position of the finger and k_p is the proportional gain of the haptic controller. For a given constraint plane, let f_n and f_t be the components of the force on the proxy normal and tangential to the constraint plane, respectively. If the given constraint surface has a static friction parameter μ_s, then the proxy is in static contact if $\|f_t\| \leq \|f_n\|$, i.e., the user's position is in the friction cone of the surface. An example of such a configuration is shown in figure 8(left). When any constraint surface is in static contact with the proxy, the proxy's position is prevented from changing by making the new sub-goal position equal to the current proxy position.

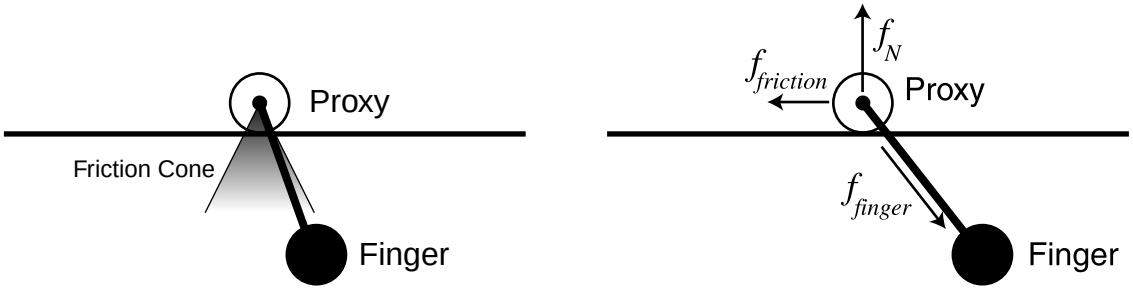


Figure 8: Static friction can be simulated by not permitting proxy movement if the user's finger is in a given friction cone (left). Viscous and dynamic friction can be modeled by constraining the motion of the proxy subject to the forces applied to it(right).

6.2 Viscous and Dynamic Friction [17] [18]

Viscous and dynamic friction can be modeled by looking at a simplified set of equations for the motion of the proxy. As is illustrated in Figure 8(right) the equations of motion for the proxy can be written as:

$$m\ddot{x} + b\dot{x} = f_{finger} + f_N - \mu f_N, \tag{20}$$

where x is the position of the proxy, m is its mass, b is the viscous damping term, and f_{finger},f_N, $-\mu f_N$ are the force on the proxy created by the user's finger, the surface constraint, and the drag caused by dynamic friction respectively. Because the mass of the proxy can be considered as being very small equation 20 can be observed as $m \longrightarrow 0$. When the mass of the proxy is zero the body quickly reaches its saturation velocity. In dynamic equilibrium, the velocity of the proxy is given by

$$\dot{x} = \frac{f_{finger} + f_N - \mu f_N}{b}. \tag{21}$$

This limit can be used to bound the amount that the proxy can travel in one clock cycle. When multiple constraint surfaces exist, the lowest velocity bound is taken as the limit of the proxy's movement. In the event that the maximum velocity is negative, then the dynamic friction term is sufficient to resist all movement and the proxy's position is not changed. If $b = 0$ no viscous

term exists and the maximum velocity is not bounded. Since this approach does not require the estimation of the user's finger velocity, from a finite set of encoder values, this approach is not susceptible to the errors found in other approaches.

6.3 Texture [19]

Image mapped texture is often used in graphics to create richer more realistic environments. As with graphics, texture can be applied to create higher fidelity scenes then can be realistically created using polygonal surfaces alone. An image-based texture map can be used to modulate any of the surface properties described in section 6. In addition the force shaded constraint planes can be modified in a manner similar to bump mapping introduced by Blinn [5] for computer graphics. For this application the contact point weights, used for shading, are used to interpolate a texture coordinate from coordinates defined on the vertices of the polygon. The texture coordinates map to an image-based texture and are converted to a displacements to the surface original or shaded normal as in Blinn. Once the texture normal is found it is used as the same as the shading normal in section 4. An example of this is is illustrated in Figure 9. As with shading, the texture does not lift the proxy off the surface of the original obstacle but displaces the goal tangential to the configuration space surface.

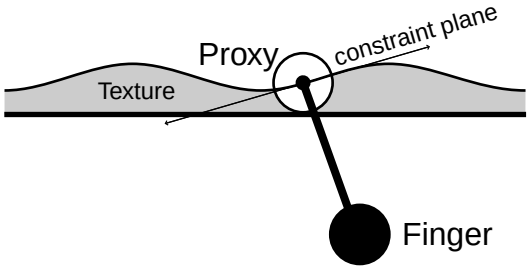


Figure 9: An image based texture is used to alter the constraint plane to create the sensation of bumps on the virtual surface

While imaged mapped texture is very similar to haptic shading some important differences exist. Texture is typically used to model higher frequency information then used in shading. In fact, in shading the primary goal is to eliminate the high frequency discontinuities at the surface boundaries. This necessitates several changes in-order to capture this high level detail and avoid missing important features that describe the surface. First, during the *move* stage of the proxy update loop the path of the proxy on the surface should be mapped onto the texture space and the values encountered on the texture surface should be checked to see if proxy's position should be constrained [20]. An example of this is illustrated in Figure 10(left). If the whole path is not examined a small detail may be missed.

Secondly, unlike bump-mapping for computer graphics it may be desirable to allow multiple constraint planes to be specified at a given location [20]. As is illustrated in Figure 10(right) where at a given point the proxy may be in contact with multiple surfaces. In this case it may be desirable to allow multiple texture images to be specified for a surface. One constraint plane is created for each image. In our system selection of appropriate textures images to model a given surface is currently left to the user.

These techniques are useful for modeling basic textures but much work still needs to be done.

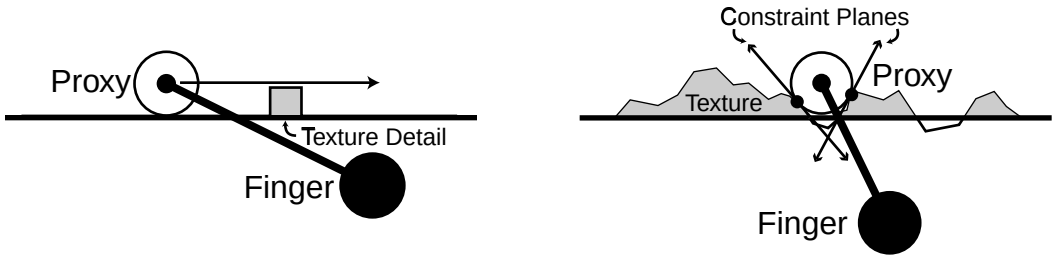


Figure 10: The entire path of the proxy must be checked to ensure small textural detail is not missed (left). Likewise multiple constraint planes may need to be defined to represent the constraints on the user's position as specified by the texture (right).

7 Collision Detection [8] [9]

Because the environment is normally constructed from a large number of primitives, a naive test based on checking if each primitive is in the path of the proxy would be prohibitively expensive. Instead a hierarchical bounding representation for the object can be constructed to take advantage of the spatial coherence inherent in the environment. The bounding representation, used by our system, is similar to that first proposed by Quinlan [28]. This approach represents a good example of how high-level pruning structure is constructed.

The 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 can be 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. A cut-away view showing the leaf nodes (yellow) and bounding sphere hierarchy for a typical model is illustrated in Figure 11. Note that a node is not required to fully contain all the descendant internal nodes, only the descendant leaf nodes.

In Quinlan's approach, 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 the bounding box already computed earlier. The radius is taken to be just large enough to contain all the descendant 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 $O(n\lg n)$ execution time, where n is the number of leaf spheres. Once constructed the time required to determine which primitives may lie in the proxy's path is only $O(\lg n)$.

The sphere hierarchy is used to prune the number of low-level checks that need to be performed but is not used to determine the exact contact point. If the proxy's path intersects one of the leaf nodes of the hierarchy then the primitive attached to that leaf is checked to see if it intersects the

Figure 11: Bounding Sphere Hierarchy of a cat model

path of the proxy. A cache is maintained to avoid calling the low-level check multiple times for the same primitive during the same iteration. This is possible since several leaf nodes may cover a single primitive. In addition, some spatial coherence information used by the low-level distance algorithm can be kept in the cache to reduce the computation time between successive calls.

8 Dynamics [21]

Our previous discussion has been limited to the rendering of static environments. To create an engaging virtual world, the user must be able to manipulate and dynamically interact with the virtual objects. In general a mechanical system can be described by a configuration space vector $q = [q_1 \dots q_n]^T$, where n is the number of DOF of the system. The forward dynamics equations of motion of such a system can be used to obtained the configuration space accelerations of the system. These equations have a general from that can be written as:

$$\ddot{q} = M(q)^{-1}(\Gamma - b(q, \dot{q}) - g(q)), \tag{22}$$

where $M(q)$ is the mass matrix, $b(q, q)$ the centrifugal coriolis vector, $g(q)$ the gravity force vector and Γ is the vector representing the internal and external torques applied to the system either through internal actuation or external forces applied by the environment.

When a collision occurs, between the proxy and an object(s) in the environment, a force is applied to the user simulating a contact with the surface. In a dynamic environment an equal and opposite force f_c is applied at the contact point(s) which may induce accelerations on the virtual system. The corresponding joint torque vector is given by

$$\Gamma_{ext} = J_i^T f_i, \tag{23}$$

where J_i is the Jacobian of the contact point i, such that the velocity v_i of the contact point is given by $v_i = J_i\dot{\Theta}$. The contact force f_i caused by contact of the proxy with the environment can be compute from the multipliers found in the *update* stage as defined by equation 19

Note that this force is in general not sufficient to prevent penetration between the proxy and objects in the environment as this equation does not incorporate the internal constraints of the proxy or other objects. A more complete solution to computing the contact forces for rigid body simulation can be found in [30]. This simplified model, however, is sufficient for simulating the interactions found in most haptic environments. Once the joint space accelerations are known, the equations of motion for the system can be integrated, from a given initial joint space configuration and velocity, to obtain the motion for the entire system over time.

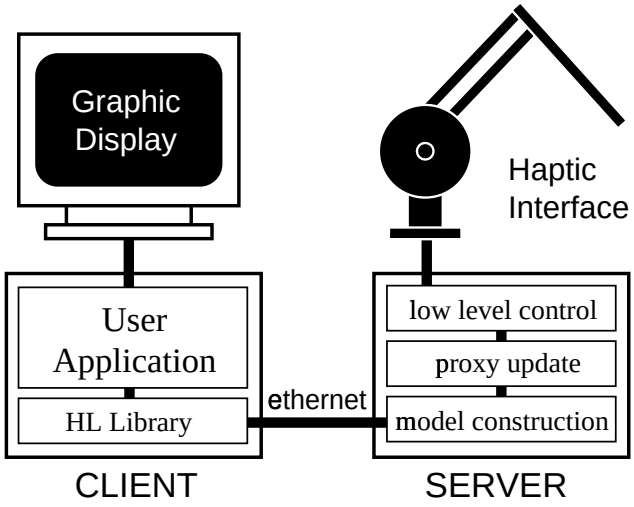


Figure 12: System architecture

9 Stability and Control 22

All the effect presented in the previous section were created solely by changing the proxy configuration. This reduces the job of the haptic controller to attempting to reduce the error between the proxy position and the haptic device. The position control of a mechanical system is a task that has been discussed extensively in the robotics literature. In our current implementation we rely on a simple operational space proportional derivative (PD) controller [19]. As all the modeling effects are achieved by the movement of the proxy, controller gains and other parameters can be set solely by considering the mechanical properties of the haptic device. The stability properties of these types of controllers are well known and can be made quite robust if a sufficiently high update rate can be maintained.

Because the size of a model may be unknown at run-time, it is important to construct haptic render in such a way that haptic display degrades gracefully and safely as the complexity of the environment is increased. One approach is to separate the application, model construction, proxy update and the low level control tasks into separate processes. This will help insure that if there is a delay in one sub-system the problem will not effect overall performance and safety of the system. In the “HL” system illustrated in Figure fig:arch the scene graph is sent to the haptic server on a separate machine by the application program. If a failure occurs in the application program the current scene graph can continue to be used until a timeout occurs and the haptic rendering process can be shutdown gracefully. The haptic server can itself be divided into separate tasks to increase the level of robustness and safety.

The computation of the bounding sphere hierarchy (BSH), while fast, is not real-time since it is a function of the number of primitives in the object. As the transmission and computation of a BSH for an object cannot be guaranteed to be completed in one servo tick it is important to keep the previous definition of the object structure until such time as the new definition is available. The robustness of the system is increased because while the addition of an object in a scene might be delay it will never be only partially defined.

As is the case with the construction of the BSH the update of the proxy position is also a function of the complexity of the environment. In this case, thankfully, $O(lgn)$ instead of $O(nlgn)$. As such it still can not be made real-time. There does exist, however, a point which is in the free space of the obstacles which can be used when no new position is available. This point is the last computed position of the proxy. If the control and update loops are separated the haptic controller can be commanded at a fixed rate to use the last computed proxy position. Thus the stability of the controller is maintained while the fidelity of the haptic display degrades gracefully as the complexity of the environment is increased.

It remains to show the movement of the proxy is stable. As seen in section 4 the basic *move/update* loop can only decrease the distance to the user's finger. It can therefore be shown that the update loop will add no energy to the user/haptic system. Likewise the static, dynamic and viscous friction properties only restrict the motion of the proxy and are thus inherently stable. Shading and texture can increase the distance between the user's finger and the proxy. This increase implies that the surface is active and can add energy to the user/haptic system. In most graphic models the interpolated and true surface normals typically differ by less then 30°. In these cases the added energy is very small, and is not noticed by the user. In our test on typical models the motion was always stable although there do exist contrived examples where unstable behavior is possible. Lastly, energy stored in a virtual dynamic system can be transfered to the user through contact. If the system being modeled is inherently stable then the entire system will be stable. Nevertheless the masses and inertias of the simulated system should be selected so as not to be so large that they may damage the haptic device or the user, and care should be taken to ensure the motion of simulated system is bounded.

10 Applications [23]

The intuitive nature of haptic interaction makes it well suited for a wide range of applications. For instance, haptics can be used to train a surgeon to perform an operation without the cost and difficulties of training on animals or cadavers. In another area a haptic system can be used to allow an animator to specify the movement of a 3D model. The animator can feel the joint limits of the model and feel the penetration constraints imposed by the environment. In mechanical design an engineer can apply force and interact with a model in a physically intuitive manner. Other applications can not even yet be imagined but it is hoped their development will be spurred by the low-level work presented here.

Figure 13 illustrates some of the virtual environments that can be modeled by our system. On the upper left a micro-mechanical sensor is modeled. The user if free to push on the test mass to see how the system responds to his/her input. The user can also use the probe to check clearances and ensure that the system will behave as expected. The size, mass, and time parameters of the system are scaled to allow intuitive interactions with the model. In other models in Figure 13 such as the windmill, roller-coaster or carousel, the size and mass of the system is reduced so that the

user can easily interact with an object which would be difficult to interact with in reality. Other object like the crank and the teapot are rendered full size. These models illustrate the numerous possibilities for using haptics to interact with virtual systems.

In tests and the examples below the client computer was a SGI Indigo2 running IRIX 6.2. The server was a 200MHz Pentium Pro running Linux 2.0.2. Communication between computers was made through a standard TCP/IP ethernet connection. The haptic device employed was a ground based PHANToM manipulator. The server produced stable results with position gains over 1800 Newtons/meter on models containing as many as 24000 polygonal primitives. The proxy update loop computation time was approximately $O(\lg n)$ with the number n of polygons in the model.

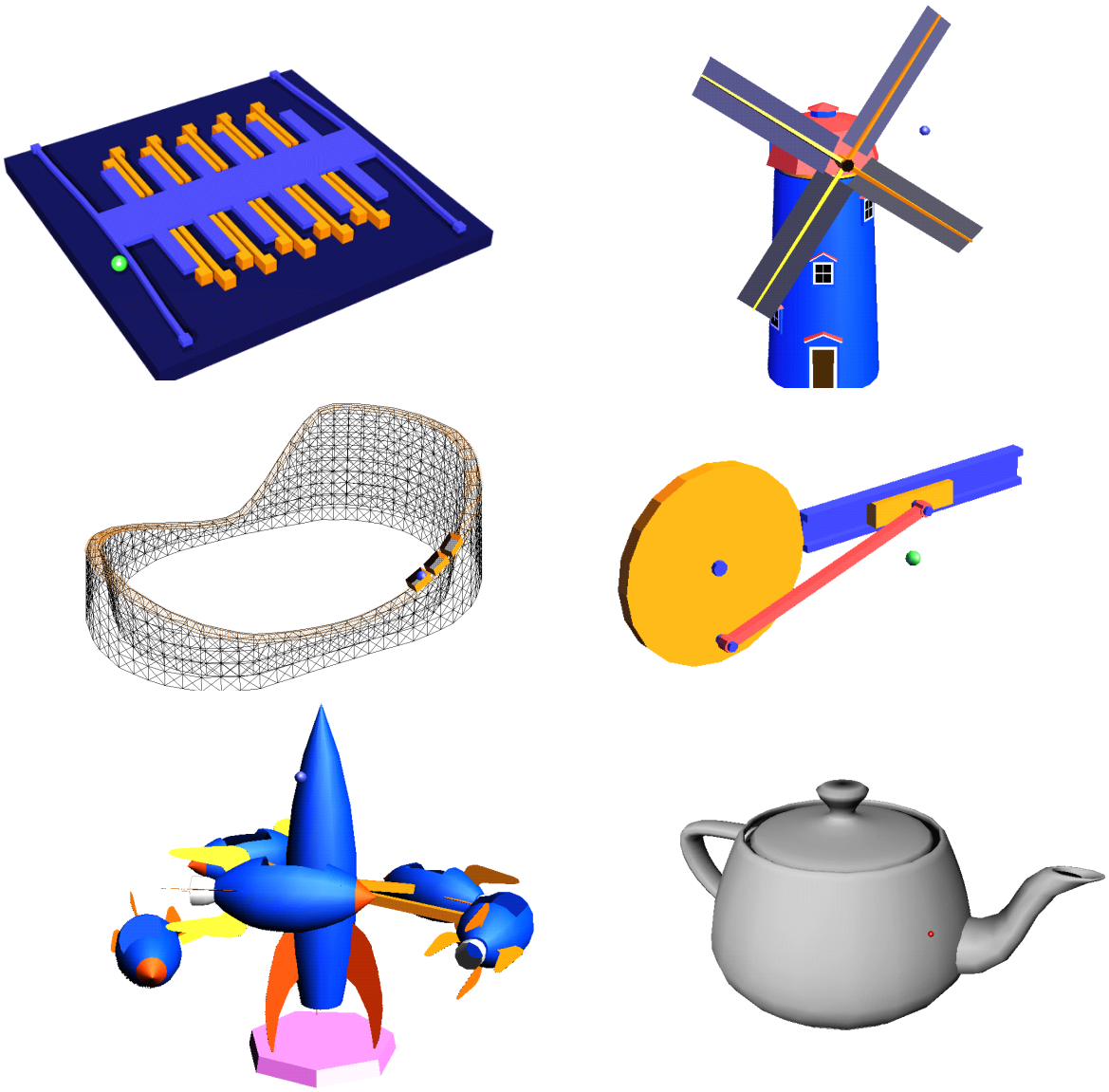


Figure 13: Many types of haptics environments can be modeled by our system from the simulation of a small micro-sensor to a large roller-coaster. Other models include a windmill, a crank, a rocket carousel or the famous Cadwell teapot.

11 Conclusion

The techniques we have described were used to model a variety of virtual models, see Figure 13. As computational power continues to increase the size and complexity of the models that can be simulated will continue to grow. Haptics by allowing a user to interact intuitively with a model can greatly improve the efficiency in designing and evaluating new systems and designs. We are currently investigating methods to allow more complex systems to be modeled and allow interaction through more complex articulated effectors.

References

[1] Avila, R., Sobierajski, S., "A Haptic Interaction Method for Volume Visualization," *Visualization '96 Proceedings*, October 1996.

[2] Adachi, Y., Kumano, T., Ogino K., "Intermediate Representation for Stiff Virtual Objects." *Proc. IEEE Virtual Reality Annual International Symposium '95*, (March 1995), pp. 203-210.

[3] Baraff, D., "Analytical Methods for Dynamic Simulation of non-penetrating Rigid Bodies," *SIGGRAPH 89 Proceedings*, (August 1989), pp. 223-232.

[4] Baraff, D., "Fast Contact Force Computation for Nonpenetrating Rigid Bodies," *SIGGRAPH 94 Proceedings*, (August 1994), pp. 23-34.

[5] Blinn, J., "Simulation of Wrikled Surfaces," *SIGGRAPH 89 Proceedings*, (August 1978), pp. 286-292.

[6] Buttolo, P., Kung, D., Hannaford, B., "Manipulation in Real Virtual and Remote Environments," *Proc. IEEE Conference on Systsms, Man and Cybernetics*, (August 1990), pp. 177-185.

[7] K. Chang, "Efficient Dynamic Control and Simulation of Robotic Mechanisms," internal report.

[8] Craig, J., "Introduction to Robotics Mechanics and Control," *Addison-Wesley*, 1989.

[9] Featherstone, R., *Robot Dynamics Algorithms.* Kluwer, 1987.

[10] Finch, M., Chi, V., Taylor, R. M. II, Falvo, M., Washburn, S., Superfine, R., "Surface Modification Tools in a Virtual Environment Interface to a Scanning Probe Microscope," *Proc. 1995 Symposium on Interactive 3D Graphics*, (April 1995), pp. 13-18.

[11] H. Goldstein, "Classical Mechanics," Addison-Wesley Publishing Company Inc., 1980.

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

[13] Gill, P., Hannarling, S., Murray, W., Saunders, M., Wright, M., "User's Guide to LLSOL," *Stanford University Technical Report SOL86-1*, January 1986.

[14] Gottschalk, S., Lin, M. C., Manocha D., "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *SIGGRAPH 96 Proceedings*, (August 1996), pp. 171-180.

[15] Gouraud, H., "Continuous Shading of Curved Surfaces," *IEEE Transactions on Computers*, C-20(6):pp 623-629, June 1971.

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

[17] T. Kane, *Dynamics: Theory and Applications*, McGraw-Hill, 1985.

[18] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robot," *International Journal of Robotics Research*, 5(1), pp. 90-98.

[19] Khatib, O., "A Unified Approach to Motion and Force Control of Robotic Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation*, Vol 3., No 1., 1987.

[20] Latombe, Jean-Claude, "Robot Motion Planning," Kluwer Academic Publishing, 1991, pp. 58-152.

[21] Lin, M., Canny, J. F., "A Fast Algorithm for Incremental Distance Calculation," *International Conference on Robotics and Automation*, (May 1991), pp. 1008-1014.

[22] Mark, W. R., Randolph, S. C., Finch, M., Van Verth, J. M., Taylor, R. M. II, "Adding Force Feedback to Graphics Systems: Issues and Solutions," *SIGGRAPH 96 Proceedings*, (August 1996), pp. 447-452.

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

[24] Minsky, M. D. R., "Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display," PhD thesis, MIT, June 1995.

[25] Morgenbesser, H. B., "Force Shading for Haptic Shape Perception in Haptic Virtual Environments." M.Eng. thesis, MIT, September 1995.

[26] Ouh-Young, M., "Force Display in Molecular Docking," *Ph.D. Dissertation*, University of North Carolina at Chapel Hill, UNC-CH CS TR90-004, February, 1990.

[27] Phong, B. T., "Illumination for Computer Generated Pictures." *Communications of the ACM*, 18(6),(June 1975), pp. 311-317.

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

[29] Ruspini, D., Kolarov, K., Khatib, O., “The Haptic Display of Complex Graphical Environments.” SIGGRAPH 97 Proceedings, (August 1997), pp. 345-352.

[30] Ruspini, D., Khatib, O., “Collision/Contact Models for the Dynamic Simulation of Complex Environments,” *Workshop on the Dynamic Simulation, IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, IROS ’97, Genoble, France, 1997.

[31] Ruspini, D., Kolarov, K., Khatib, O., “Graphical and Haptic Manipulation of 3D Objects.” *First PHANToM User’s Group Workshop*, September 27-30, 1996.

[32] Salcudean, S. E., Vlaar, T. D., “On the Emulation of Stiff Walls and Static Friction with a Magnetically Levitated Input/Output Device,” ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems, in *Dynamics Systems and Control*, (April 1995), pp. 123-130.

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

[34] Srinivasan, M. A., Beauregard, G. L., Brock, D. L., “The Impact of Visual Information of the Haptic Perception of Stiffness in Virtual Environments,” *ASME Winter Annual Meeting*, November 1996.

[35] Taylor, R., Robinett, W., Chi, V., Brooks, F., Wright, W., Williams, R., Snyder, E., “The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope,” *Proceedings of SIGGRAPH 93*, August 1993.

[36] C. Zilles, J. Salisbury, “Constraint-based God-object Method for Haptic Display.” ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1994, *Dynamic Systems and Control 1994*, vol. 1, pp. 146-150.

Haptic Application Issues

Dan Staples
SensAble Technologies

DSTAPLES@SENSABLE.COM

My name is Dan Staples and I manage the software development efforts at SensAble Technologies. I hope that I can bring some unique perspective to the area of commercial applications development, as we provide both a Haptics SDK and are introducing our first “end user” application here at the show. Implementing an end user application has been an interesting experience -- we found and navigated around many potholes. But it gave us a wealth of experience. I hope to share with you some of those issues and some of the solutions -- as well as stimulate your thought on the subject.

Also a quick disclaimer. Although there are clearly interesting 2D haptic devices in the world, the 3D world presents a more interesting set of challenges. As such, this talk which focuses on application issues, focuses on 3Das there are many more issues with 3D applications.

==========

Contact me at:

Dan Staples

Vice President, Software Development

SensAble Technologies

www.sensable.com

dstaples@sensable.com

617-621-0150

Products

PHANTOM Haptic Interface

GHOST Software Development Kit

FreeForm 3D Touch Clay Modeling

Haptic Application Issues

- Visual ≠ Haptic
- 2D ≠ 3D

Surprisingly, the issues fall into two pretty neat bins. Although there are lots of issues, they center around two main things -- 1) Visual paradigms do not always translate to Haptic paradigms and 2) A new level of 3D interaction brings with it a new set of questions.

If you just remember these two things, you will be well on your way to making sure you don't hit the potholes:

1)Never assume that something that works visually will work haptically

and

2)Always consider how the 3D interaction paradigm provides new opportunities (and challenges!) for you and your users.

Visual ≠ Haptic

- Not all meshes are created equal
- Zoom does not really make things bigger
- Z is not infinite

There are really three things that stand out to me in the realm of not assuming visual things work well haptically.

First, display meshes have been optimized over the years for visual purposes. And these optimizations, while great for visuals, have a tendency to work against the goals of haptics.

Second, zooming in and out in a graphics system is usually done by moving the camera. As I'll explain in a second, this is not usually sufficient in a haptically aware system.

Finally, the pseudo-world of the monitor allows us to depict infinite Z depth. Of course the real world is less forgiving.

Each of these is covered in a bit more detail on the following pages...

Mesh suitability issues

- Gaps & Fall-thru
- Back-face culling

A mentioned previously, not all meshes are created equal. Assumptions made for visual purposes are not necessarily valid in a touch-enabled system. The first few items on this and the following page are good examples of this.

First, most graphic or CAD systems tessellate surfaces individually, with no attempt at connecting the vertices of the meshes topologically where two surfaces adjoin. (This is true even for solid models). This makes tessellation significantly faster and easier. Of course the issue this creates is that gaps exist between the individual surface meshes. And while its technically correct to “fall through” these gaps when touching the object, users don’t want the technically correct answer -- they want simulated reality. So the haptic system must find ways of dealing with these issues.

Another important issue is back-face culling. Typically solid modeling systems will cull out those triangles whose normals face away from the user since these can’t contribute to the visual scene. However, these triangles are still critical for haptics -- imagine if you felt around the back of the part and fell inside it! So in implementation, the haptic system must be able to connect far enough up the graphics pipeline to precede back-face culling.

Mesh suitability issues

- Level of Detail
- Size & Subsetting

Another mesh related issue is level of detail. As graphic systems have become more sophisticated, vendors have come to realize that not all details need to be displayed at all zoom levels. As such, when zoomed out significantly, frequently a more coarse tessellation is used, significantly improving performance. The trick when 3D touch enabling is to either ensure that you can always get the right detail level from the pipeline, or go the true what you see is what you touch route, where indistinct display leads to indistinct feel. These are some of the trade-offs that application developers must consider.

A final concern is just one of pure horsepower and bandwidth. Visual meshes can be extraordinarily large. Despite recent advances, most haptic algorithms are not capable of dealing performantly with meshes of these sizes. As such, some method of subsetting the visual mesh is required. Implicit methods, like taking cues from the viewing frustum and explicit methods, like dragging around a proxy for the haptic space have both been used with some success.

Zoom is not scale

- Moving the camera
  - Only magnifies visuals
- Scaling the object/scene/workspace
  - To touch what you see (at appropriate level of detail)

One of the slides that Thomas Massie showed at last year's presentation was how ten grooves per centimeter in a plate might feel like a texture. And if you simply put the camera closer to them, but don't "zoom" them haptically, they still feel like a texture. Is that the right thing? Probably not. In most cases when you see more detail you want to be able to feel more detail too. On the other hand is it right to scale one for one? That is, should you grow the visuals and the "size" of the feel the same amount? This is what we chose for our FreeForm application because high detail is extremely important. But there is no one right answer. The application has to tune how much to scale the feel and how much to move the camera to achieve the desired effect. (And of course they interact, so you have to coordinate their effects on one another).

Z is not infinite

- In graphics, Z can extend infinitely
- In haptics, the device has a physical Z-limit
  - When objects are scaled, they simply don't fit
  - Which portion of graphic Z is touchable?
  - What UI is best for changing touchability?

Another assumption that those of us from the graphics world are used to is that Z is infinite. You can clip it if you want, but if you want to see the whole z-depth, you certainly can. This of course is not true with haptics. The device has physical limits. Depending on the model device you are using it can vary some, but its probably not a lot bigger than a breadbox. So when you scale up the objects to be able to feel them more precisely, they start bulging out of your physical workspace. The application must define a paradigm for defining what is touchable. In our application we chose to automatically set it such that the user can feel what is closest to him, but may not be able to feel what's farther away. But of course you have to allow an override for this.

There is no right or wrong answer (actually, there are lots of wrong answers -- and a few right ones). The application has to understand its end user and decide what will make the most sense to him or her.

Visual ≠ Haptic

- Not all meshes are created equal
- Zoom does not really make things bigger
- Z is not infinite

To summarize to this point, one key thing to keep in front of you at all times is that assumptions that are made graphically don't always hold haptically. The biggies are right here -- mesh suitability, needing to properly scale the feel, and deciding what's touchable when. Of course there are other issues than these, but hopefully this gives you a good head start.

2D ≠ 3D

- 3D Navigation
- 3D Buttons
- 3D Helpers

The second area of general issues is technically not haptics per se, but is critically important to making a sane haptics application. That is, for many this is the first time that they will be navigating a system in full 3 dimensions. This has an immediate coolness factor, but can definitely lead to unnecessary complexity if you don't rein yourself in.

The 3 principal areas to consider are the complexities of 3D navigation, whether to use 3D buttons or not, and the proper use of 3D helpers. I'll explain each of these in a bit more detail.

3D Navigation

- 3D cursor
- Depth cues
- Select in 3D or 2D project

First, cursors in most development environments are 2D -- they help you guide the mouse pointer around on a 2D plane. In 3D, you have the immediate need to be able to understand what z depth you are at. And while touch helps this immensely, if you're not touching anything, its not acceptable to be "lost in space". Probably the most effective solution to this problem is Stereo viewing -- and that works really well. However, two lesser graphic helpers can go a long ways. 1) Use a 3D cursor that gets larger and smaller depending on depth (like a real object does) and/or 2) Provide shadow casting, laser beam, or the like that projects onto other objects to give a sense of location in 3D space.

A second issue is selecting elements. The traditional 2D graphics select is nice since its very easy to point and shoot. It of course has the downside of being ambiguous when objects overlap. Selecting in full 3D is much less ambiguous, but is quite difficult ergonomically if you don't use haptics to help somewhat. Consider for example selecting a line in 3D -- you can go round and round it several times before homing in on it. The proper use of haptic helpers, which are covered in following pages, can make this a lot easier.

3D Buttons

- “Real” push buttons or MFC?
- Ergonomic considerations
- VR or Engineering?

Another important UI consideration is how to present the user interface. The technology exists today to make it totally immersive, with the feel of real 3D buttons. But what is best for the user? Its physically easier to get over a 2D location and click it than it is to push a button with the device (since now your whole wrist and upper arm is involved). And yet, the latter is clearly much more realistic.

The answer to how to proceed here largely depends on the type of application being pursued. Its most expedient for both the developer and the end user to use the UI framework that other applications use -- MFC for example. And if the application is about pure productivity, this is probably the way to go. If on the other hand, the application is principally about being immersed in a 3D environment and providing maximum realism, then the 3D “real” push buttons are probably the best route.

3D Helpers

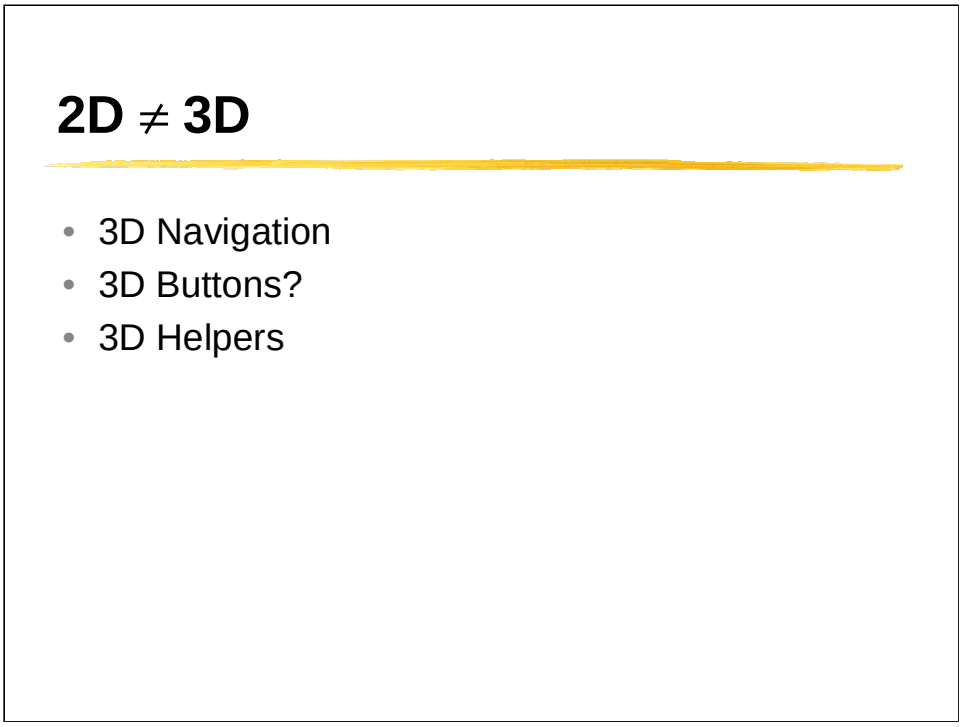
- Pushing the user around ☺
  - Locking to planes/lines/points
  - Attracting the device to gravity wells
  - Get creative (and then make sure its useful!)

I affectionately call this pushing the user around. But it's a fitting title -- haptics gives all of us the advantage of being able to physically guide the user -- this is simply not possible without haptics. Only through the application of a 2 way device, can we push the user towards his goals. Here are a couple of useful examples of this…

The first is locking and/or blocking. Its possible to physically lock the user to particular geometry be it visually displayed or not. For example, you can constrain the user to only move along a particular line, the normal to a surface at a point for example. Or you can lock the device to a particular XYZ to facilitate rotating something about a specific point. You can also define blocking behaviors, such as "keep out of this region". The possibilities are really endless.

You can also use attraction to your and the users advantage. For example, when the user is near a particular point of interest, you can actually pull him onto it when he gets in close proximity. This is a really nice effect that can add real benefit.

And speaking of real benefit, this is where you need to watch out. Its very easy to become so enamored with these effects that you forget you are supposed to do something useful with them. So get creative, but then make sure you're really boosting productivity and not just cool for coolness sake.



So whether you are adding 3D touch to your application or building a native 3D touch application from the ground up, just as you couldn't make the same assumptions for haptics you did for graphics, you have to deal with the fact that 3D is not 2D. Don't get carried away -- just recognize that a 3D world opens interesting possibilities and brings new complexities. Hopefully, the above 3 areas will keep you focused ensuring you have covered the right bases.

Conclusions

- Haptics & 3D interaction create the door
- You have to open it and guide users through it!
  - Beware of the graphic assumptions troll
  - Use the haptic advantage to slay outdated UI concepts
  - Don’t be seduced by the siren song of too much coolness
- Keep your user in front of you at all times

Haptics and 3D interaction open up an exciting new world. But its those of us who write software applications that need to explore that world and guide our users through it. There are lots of pitfalls that, if not properly thought through, we can drag our users directly into. So how do we avoid that?

First, remember that the optimizations for graphics systems don’t usually apply in the haptic world. Carefully consider how and where you tap into the graphics pipeline to ensure you have a cohesive data set.

Second, don’t just bolt haptics onto an existing application. The real benefits come from re-thinking old UI and using haptics to make it better. Consider how users are handicapped by existing UI paradigms and use both 3D navigation and force feedback to improve their productivity.

At the same time, don’t be seduced by the siren song -- there’s lots of cool UI that isn’t productive. Strike the right balance for your application.

And finally, the thing that helps you balance all this out are your end users. If you know them well and always keep them in front of you, you will rarely go wrong.

Volume Haptics

Ricardo S. Avila

General Electric
Corporate Research & Development
Niskayuna, New York 12309

Haptics: From Basic Principles to Advanced Applications

SIGGRAPH 1999

Volume Haptics

Ricardo S. Avila
General Electric Corporate R & D
1 Research Circle
Niskayuna, NY, 12309

Phone: (518) 387-6632
Fax: (518) 387-6981
Email: avila@crd.ge.com

Outline

Volume Representations
Haptic Rendering
Volume Modification
Visual Rendering
Integration Issues
Applications

This section of the course is concerned with the use of volumetric representations for haptic interaction. The talk begins with a basic introduction to **volume representations** followed by three important areas associated with building haptic applications that utilize volumes. The first of these three areas, **haptic rendering**, covers techniques for calculating forces from volumes. This is followed by a **volume modification** section which describes a method that allows a user to modify an object and feel the results interactively. A **visual rendering** technique that supports haptic interaction is presented next. The three areas are then pulled together in a section on **integration issues**. Finally, the talk ends with a description of several example **applications** of these techniques. Most of this course material is drawn from the Visualization ‘96 paper, “A Haptic Interaction Method for Volume Visualization,” by Ricardo S. Avila and Lisa M. Sobierajski.

Outline

- ✔ **Volume Representations**
- Haptic Rendering
- Volume Modification
- Visual Rendering
- Integration Issues
- Applications

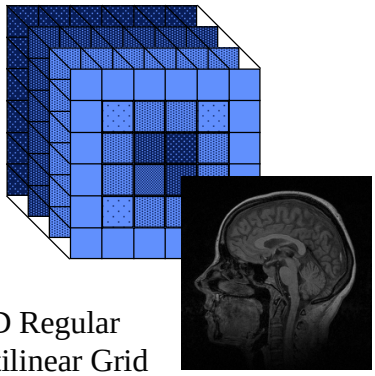
This section briefly describes volume representations, the general area of volume visualization, and a basic motivation for volume haptics.

Volume Representations

Collection of 3D Primitives
- Voxel/Cube Cells
- Tetrahedral Cells

Grid Organization
- Regular
- Irregular

Internal Data
- Scalar (density)
- Vector (flow)



3D Regular
Rectilinear Grid

Volume representations can exist in a wide variety of forms, but they all utilize a collection of primitives in at least three dimensions. The most common form consists of a collection of data samples arranged on a regular rectilinear grid. Other 3D grid organizations (e.g. curvilinear, irregular) can also used to construct volumes, but for the purposes of this talk we will concentrate on volumes constructed on a regular rectilinear grid. Each sample location, often called a voxel, contains some form of data. Typically this is a single scalar value such as a material density, but it may consist of a vector quantity or even a collection of several scalar and vector quantities. The illustration on the right demonstrates a common approach to constructing a volume. A series of images is used to specify the internal data inside the volume. In this case, an MR image is used and therefore each voxel contains a scalar MR quantity at the voxel location in the volume. The stack of MR images and the distance between adjacent voxels along each axis specifies the geometry and internal values of the volume, but only at the precise voxel locations.

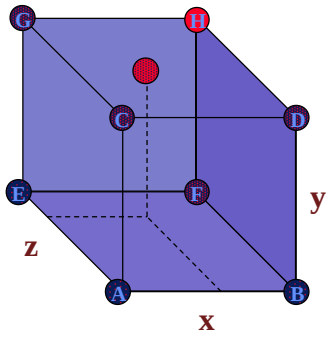
Interpolation

Smooth Scalar Field

- Trilinear/Tricubic Interpolation

Trilinear Interpolation

$$
\begin{aligned}
D(x,y,z) = {} & A(1-x)(1-y)(1-z) + B(x)(1-y)(1-z) + \\
& C(1-x)(y)(1-z) + D(x)(y)(1-z) + \\
& E(1-x)(1-y)(z) + F(x)(1-y)(z) + \\
& G(1-x)(y)(z) + H(x)(y)(z)
\end{aligned}
$$

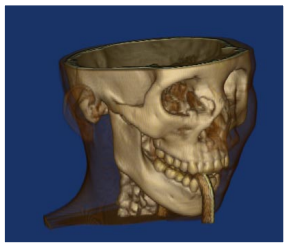


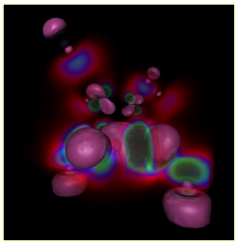
In order to evaluate the scalar value in between voxels an interpolation scheme is used, the simplest form being nearest neighbor interpolation. A volume that uses nearest neighbor interpolation is essentially a collection of three dimensional boxes organized on a regular 3D grid where each box has a homogeneous value. This discrete form of a volume representation will not be of much use to us for haptic interaction since a smooth scalar field is usually necessary.

Trilinear interpolation provides a computationally efficient method for producing a smooth scalar field. In essence it interpolates scalar quantities linearly along each dimension. Although the resulting scalar field only exhibits C0 continuity, it is suitable for haptic applications. Higher order interpolation methods may also be used, such as tricubic interpolation, with a much higher computational expense. All of the methods described in this talk utilize trilinear interpolation.

Volume Data

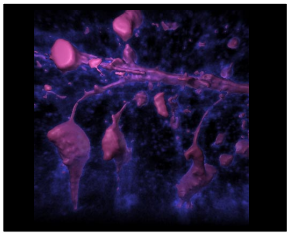
Computed Tomography





Simulation

Confocal Microscopy



Scan Conversion

Volume data can originate from a variety of sources. Some of the most common are medical (top left) and biological (bottom left) scanning devices. These devices typically scan a single image at a time, stepping through space acquiring one image plane at a time to produce a volume. In addition, physical simulations often produce volumetric data such as the high potential iron protein shown above (top right). Finally, standard geometry can be scan converted into a volume representation such as the vase shown above (bottom right). Over the years a large amount of research has been conducted on the visual rendering of volume data. This area of research is known as volume visualization.

Volume Visualization

Volume Rendering

- Ray Casting
  - Light Reflection Functions [Blinn 82]
  - Volume Rendering [Drebin et. al. 88]
  - Display of Surfaces [Levoy 88]
- Splatting
  - Footprint Evaluation [Westover 90]

Polygonal Isosurfaces

- Marching Cubes [Lorensen, Cline 87]

Volume visualization techniques have been explored for a number of years yielding many noteworthy publications. An approach to visualizing volumes known as volume rendering treats the volume as a translucent object allowing the viewer to see its internal structure easily. Ray casting techniques have been used extensively to produce high quality volume rendered images. These techniques [2,3,6] cast rays from image space into the data or object space. Alternatively, Westover's splatting technique[11] projects information from object space to image space. Another approach to visualizing volumes is to extract polygonal isosurfaces from the volume which are then rendered using standard graphics hardware. Marching cubes [7]achieves this task in a quick and efficient manner. A polygonal representation created in this manner may also be used for haptics by using techniques for computing forces from polygonal geometry. Many of the concepts and techniques employed in the area of volume visualization are of potential use when we explore the area of volume haptics.

Motivation

Haptics with Volumes

- Beyond Surface Models
- Locality of Reference
- Simple Modification

Volume Visualization with Haptics

- Enhance Understanding
- Intuitive Interaction
- Fast Manipulation

I would like to motivate two main areas in this presentation. First, the general area of haptics can benefit from using volumetric representations. Volume representations permit physical interaction with not just the surface properties of an object, but also internal and potentially inhomogeneous characteristics. In addition, haptic interaction generally requires an extremely fast object sampling process. One benefit of a volume representation is the ability to partition the world into discrete units. This high locality of reference permits applications to sample the volume for local intersection and force information in constant time. Volumes, which are essentially three dimensional images, can be modified using simple filtering tools. These modification techniques make possible forms of interaction that are difficult to obtain with a purely geometric representation.

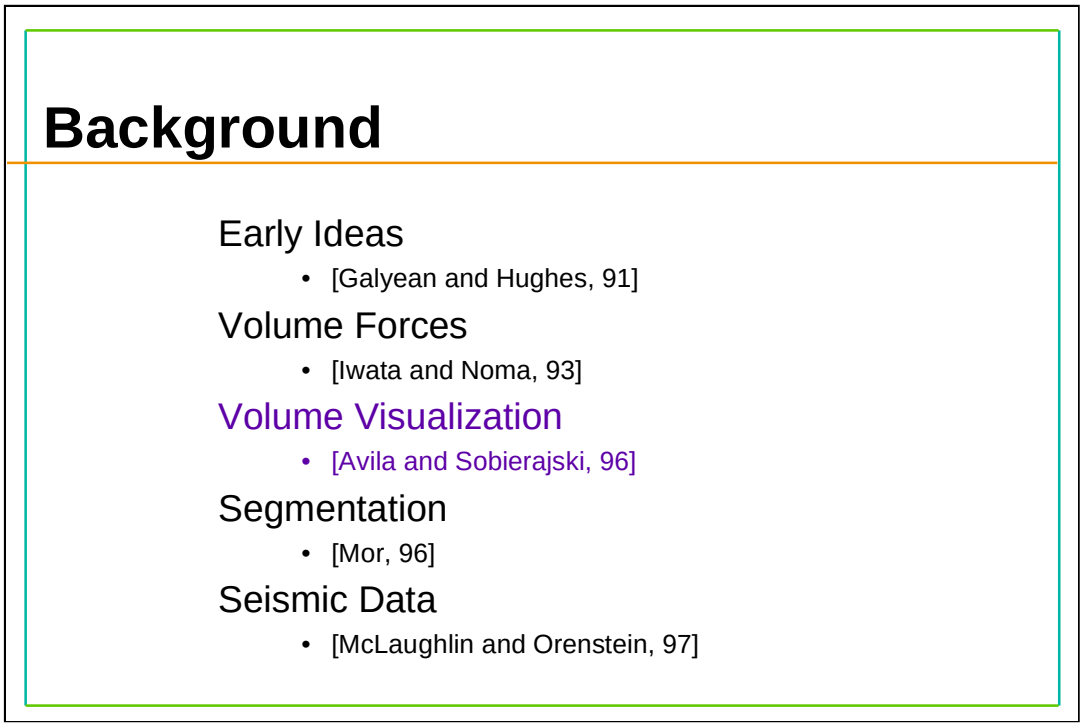
I would also like to motivate the use of haptics as a useful technique in the area of volume visualization. This volume rendered image of a CT scan of a human foot does a pretty good job of conveying the outer skin and bone structure present in the medical data.

Outline

- Volume Representations
- ✔ **Haptic Rendering**
- Volume Modification
- Visual Rendering
- Integration Issues
- Applications

However, there are several features that are difficult to convey using purely visual methods. For example, haptics can aid in the understanding of the thin boundaries between adjacent bones. Also, because volume rendering techniques produce images with translucent boundaries, they often produce images which contain regions where the structure of the data is difficult to understand. Haptics allows the user to investigate and understand the structure of these regions in an intuitive manner. Haptic interaction also benefits manipulating volumes. Performing measurements and selecting regions can be achieved quickly and intuitively with haptic interaction techniques.

The computation of forces from volume representations is the focus of this section on haptic rendering.



There has been relatively little work in the area of haptic rendering of volumetric representations. A 1991 Siggraph paper by Galyean and Hughes [4] described some basic ideas on the use of force-feedback and volumes. Iwata and Noma published the first paper that described haptic rendering techniques specifically for volumetric representations [5]. Avila and Sobierajski investigated haptic volume rendering techniques and provided information on how to integrate to a visualization system [1]. Mor described techniques for working with segmented medical data, such as an MR knee [9]. More recently McLaughlin and Orenstein have investigated volumetric haptic techniques for exploring seismic data [8]. This talk will concentrate on the Visualization ‘96 paper by Avila and Sobierajski.

Haptic Rendering

Goal

- To Compute Forces From a Volume

Requirements

- First Priority : Force Refresh Rate >= 1Khz
- Consistent Rate (not average)
- Smooth Transitions
- Allow Modifications to Object
- Compliment Visual Rendering
- Work in View Coordinate System
- Convey Internal Information

The goal of haptic rendering is to compute forces from a volumetric representation. Given the requirement to maintain a consistent haptic refresh rate of 1Khz or greater, this task is given top priority in a haptics application. The forces should be generated with smooth force transitions and we do not want to prevent the object from changing during interaction. In addition, the forces generated should be able to be consistent with the visual information provided to the user. Another important requirement is that the haptics workspace is in the visual coordinate system of the user. Therefore, we will use the view coordinate system for haptic interaction. Finally, since volumes contain internal density information we would like to make sure that we have the means to convey this information to the user. This is done visually with volume rendering and can also be done with haptic rendering.

Haptic Surface Rendering

Collision Detection

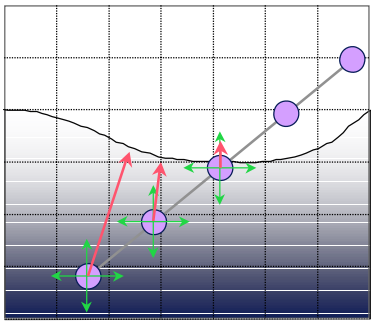
- Point Contact Model
- Check: (scalar value >= threshold)

Force Direction

- Estimate Normal
- Central Differences

Force Magnitude

- Need Penetration Distance
- Estimation Based on Scalar Field
- Utilize Transfer Functions



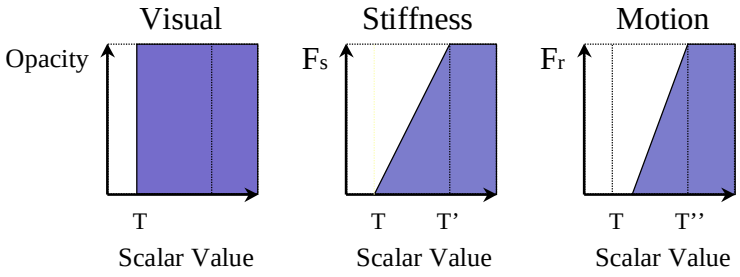
A common visualization technique is to render an isosurface present within a volume. An isosurface is defined by specifying a threshold value. Scalar values equal to or above the threshold are considered part of the object. We achieved haptic rendering of an isosurface by using a simple point contact model. A collision occurs when the haptic pointer enters the object and in this case that means that the scalar value at that point is above the threshold. Next we must determine the direction and magnitude for the contact force. The direction should be normal to the isosurface and we will estimate this normal using a central differencing technique. Ideally we would like to know the penetration distance of our haptic pointer into the isosurface. One could use a god object (as described by Zilles [12]) here and simply compute the distance between the surface entry point and the haptic pointer. We took a different approach and used the scalar field as an indicator of penetration distance. This technique utilizes transfer functions to compute force magnitudes as we explore the volume below the isosurface.

Haptic Surface Rendering

Force Terms

- Ambient : (constant)
- Stiffness : (penetration, N)
- Motion Retarding : (velocity, density)

F = **A** + **N***Fs(d) - **V***Fr(d)



The transfer function for a visual rendering of an isosurface is shown on the bottom left. Scalar values below T have zero opacity (zero density) and above T have complete opacity (full density).

When computing the force at a point in a volume we will sum three basic terms. This vector sum consists of an ambient, stiffness, and motion retarding term. The ambient or constant term is used to provide a global, constant force on the haptic pointer. The stiffness term will depend on the penetration distance into the isosurface. The magnitude of this vector is determined by a lookup in the stiffness transfer function Fs. As we move the haptic pointer inside the object it is desirable to feel the density below the surface. Therefore we have a motion retarding term that also maps scalar values to force magnitudes. In this case, however, the direction of the motion retarding force Fr opposes the user's motion.

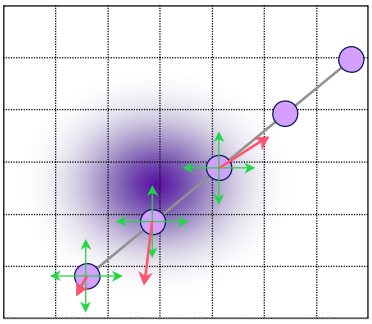
This force model allows the user to explore an isosurface and feel the properties of the scalar field below the surface.

Haptic Volume Rendering

Collision Detection
- Point Contact Model
- Check: (Opacity > 0)

Force Direction
- Estimate Normal
- Central Differences

Force Magnitude
- Based on Scalar Field
- Utilize Transfer Functions



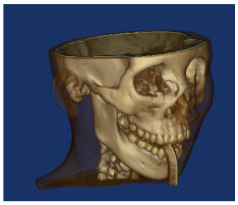
Volume rendering assigns partial densities to locations in the volume thereby creating a fuzzy or translucent object. We will compute forces for this type of model using techniques that are similar to the ones used for isosurfaces. Again we will utilize a point contact model and check to see if our haptic pointer has made contact with the model. However, now we will check to see if the opacity of the model (which is computed using a lookup table based on the scalar value) is above zero. If so, we will again compute the direction of the force by using central differences and the magnitude by using transfer functions. In essence, we are feeling the opacities of a volume rendered object.

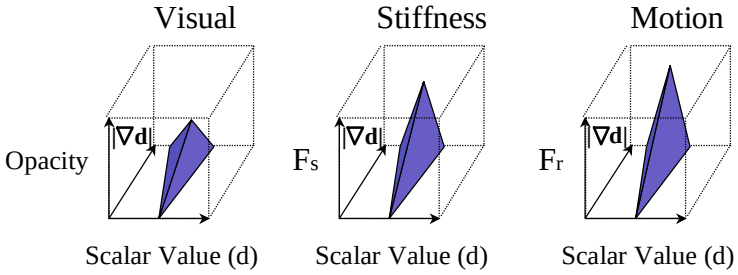
Haptic Volume Rendering

Force Terms

- Ambient (constant)
- Stiffness (N, density, gradient magnitude)
- Motion Retarding (velocity, density, gradient magnitude)

$\mathbf{F} = \mathbf{A} + \mathbf{N}*Fs(d, |\nabla d|) - \mathbf{V}*Fr(d, |\nabla d|)$





A visual volume rendered transfer function can get a little more complicated than the isosurface case. Here we are using a two-dimensional transfer function to map scalar values and gradient magnitudes to opacities. The image above (upper right) shows a volume rendering of this type. Alternatively, we can drop the gradient magnitude dimension of the transfer functions and simply map scalar values to opacities.

As with haptic surface rendering, we will compute the force at a location in the volume using ambient, stiffness, and motion retarding terms. The transfer functions for the stiffness and motion retarding terms are based on the same terms used to compute opacities. Clearly, this need not be the case, but doing so provides the user with a clear correspondence between visual and force interaction.

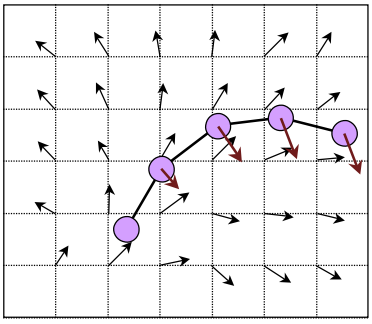
Haptic Flow Rendering

Force Magnitude & Direction

- Point/Particle Model
- Flow Field Influences User
- User Can Modify Path

Utility

- Flow Overloads Visual Displays
- Provides Intuitive Interaction



Another area where haptics is of use is as an aid to understanding flow data. We can consider the haptic pointer as a particle being influenced by the flow. In addition, the user may modify the path of the particle by exerting force on it. This type of interaction is useful because conveying 3D or greater flow information to a user is a difficult task. Visual rendering techniques for flow data often confuse the viewer because they project too much information on a two-dimensional image. Haptics allows the user to interactively explore the flow field in an intuitive manner.

Outline

- Volume Representations
- Haptic Rendering
- ✔ **Volume Modification**
- Visual Rendering
- Integration Issues
- Applications

Next we will explore a technique for modifying a volume model during haptic interaction.

Volume Modification

Goal

- To Interactively Modify a Model

Requirements

- Second Priority
- Feel Changes as They Happen
- Need Smooth Transitions
- Modify Density, Color, and Material Properties

The principal goal here is to allow the user to interactively modify a volumetric model. This implies that the user should be able to feel and see the modification as it is being performed. While this is an important task, its priority is second to haptic rendering. However, we must ensure that modifications occur smoothly and without distracting force artifacts. Also note that for some applications we may want to modify more that just the shape of an object. Therefore we made sure that we could modify the density, RGB color, and material properties of an object.

Volume Modification

Voxel Representation		
Property	Type	Bytes
Density	Scalar	1
Gradient Direction	Encoded Vector	2
Gradient Magnitude	Scalar	1
Color	RGB Vector	3
Material Properties	Scalar Index	1

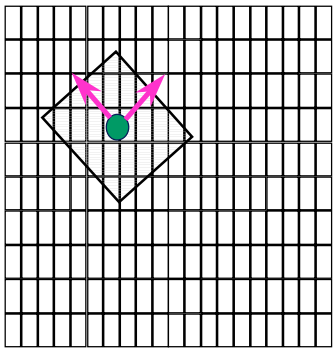
In our implementation we store quite a bit of information per voxel. The original scalar value is stored as a single byte. The gradient direction is precomputed at each location in the volume and stored in 2 bytes and the gradient direction is stored in another byte. This is useful for accelerating visual volume rendering. These values are not used for haptic rendering since they do not have the precision necessary for haptic interaction. An RGB color is also stored in each voxel. This is useful for painting applications. Finally, all other properties are captured in a material property index which occupies 1 byte. It is important to note that we have great flexibility in what can be stored at each voxel location. If isosurface feeling and shape modification is all that is needed in an application then the only quantity necessary is the 1 byte density.

In our implementation 1 byte was used to store the density. If possible, at least 16 bits should be allocated to this quantity.

Volume Modification

Filter Extent
- Arbitrary Size
- Symmetry Buys Speed

Apply Filter to Volume
- Density
- Color
- Material Properties



We chose to modify volume representations with a 3D filtering operation. The filter's position and orientation are defined by the haptic pointer and may be of any size or shape. Restricting the filter to a sphere allows us to ignore orientation and gain some speed. The filter may be applied to any of the quantities stored in the voxel. Modification of density and RGB values will change the shape and color of the object. Changing the material properties could be used to modify the object's specular appearance or the physical stiffness of the object.

Volume Modification

Blend new value A with voxel value V_{old} :

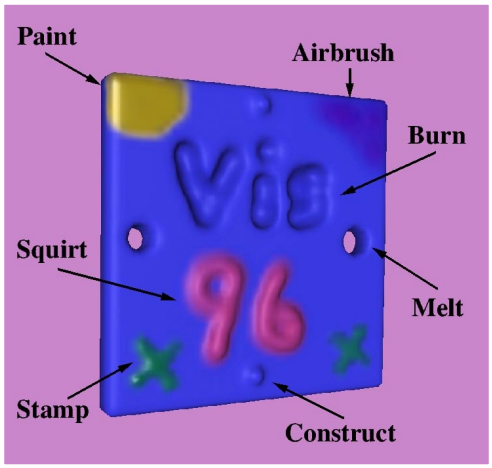
$$V_{new} = A * F_x + V_{old} * (1-F_x)$$

0.0	0.1	0.3	0.1	0.0
0.1	0.5	0.7	0.5	0.1
0.3	0.7	1.0	0.7	0.3
0.1	0.5	0.7	0.5	0.1
0.0	0.1	0.3	0.1	0.0

Filter

The application of the filter is performed by blending the new desired values with the old values in the volume. A simple linear combination of these values using blending weights provides for smooth changes of the volumetric object. To remove material from the volume the new value A must be lower than the isosurface value. To add or deposit material the value must be higher than the isosurface value. This technique can be done very fast with small filter extents in the range of 3-5 voxels cubed. It also allows the user to feel the volume as it is changing.

Modification Tools



This image shows some of the possibilities for modification with a volumetric approach. We took a volume which contained a thin slab and performed various modifications to it while physically feeling it. We performed ray casting to view the isosurface and haptic surface rendering to feel it. We modified the color of the object (airbrush) in the upper right corner. We removed material (melt) on the right side and added material (construct) on the bottom. We used an asymmetric kernel to add material (stamp) on the lower left. We painted (paint) the upper left corner by changing its material property to be a specular material and changed its color to yellow. In the middle we removed material and blended in a black color (burn) to write “Vis”. Finally, we added material and changed the color to pink to write “96”

Outline

- Volume Representations
- Haptic Rendering
- Volume Modification
- ✔ **Visual Rendering**
- Integration Issues
- Applications

The visual rendering of a volumetric model is an important aspect of an application that utilizes haptics.

Visual Rendering

Goal

- To Render a Volume in a Haptics Application

Requirements

- Lowest Priority
- Handle Isosurfaces and Volume Rendering
- Visual Appearance Consistent with Forces

The goal is to produce an image of the volume without getting in the way of haptic rendering or volume modification. This element of an application receives lowest priority since we can accept when an image is not up to date and has some artifacts. It is more important that we achieve fast haptic rendering. We would like to handle both isosurfaces and translucent volume rendered objects and be able to provide a visual appearance that is consistent with physical interaction. In some cases one may want the visual and haptic interaction to differ significantly and these techniques are general enough to handle that.

Ray Casting

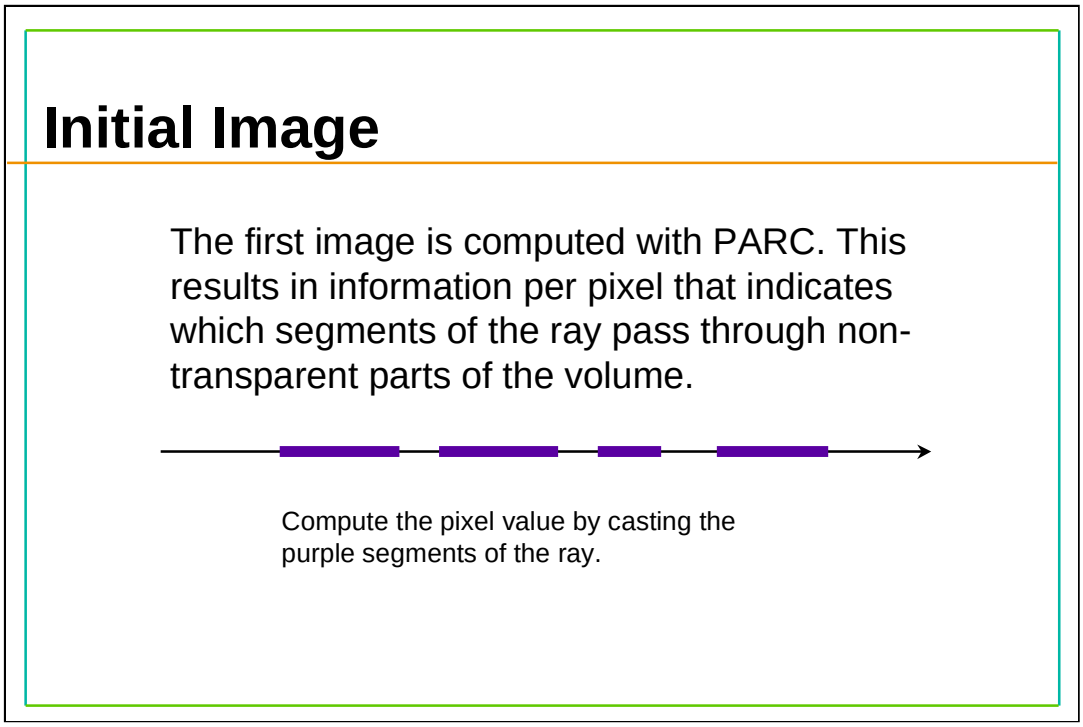
Advantages

- Sharp Surfaces or Translucent Objects
- Work is Easily Partitioned
- Incremental Image Updates Possible

Disadvantages

- Software Algorithm Requires CPU Processing

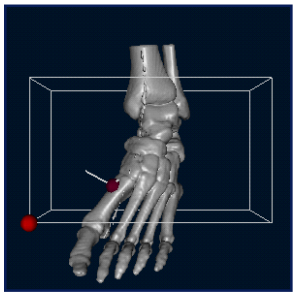
We will use a ray casting approach to render the volume data. Other approaches such as marching cubes can be used to extract and display an isosurface in the volume. Some advantages to a ray casting approach include the ability to handle both isosurfaces and translucent objects, the computation of an image is easily partitioned into small image regions, and when a fixed viewpoint is acceptable, the image can be easily updated in regions where local changes occur. In addition, a ray caster is easily parallelized and can scale nicely to powerful multi-processor computing platforms. The main disadvantage to a ray casting approach is the fact that it is computed on a machine's CPU. This software approach means that the rendering technique must compete for CPU and memory resources with the haptic and modification components of an application.



We use a fixed view ray casting technique as our rendering engine during haptic interaction. The volume may be rotated and repositioned until the moment when the user wishes to physically interact with it. We use the ray casting acceleration approach known as Polygon Assisted Ray Casting (PARC) [10] to skip over empty regions of the volume while the user is interactively rotating, translating and zooming the volume. Once the user has selected the fixed viewpoint for haptic interaction, we use a software variation of PARC that does not use the graphics hardware for acceleration. The first step is to compute the non-empty regions along the ray for each pixel in the image. The information is saved during the ray casting of the initial fixed viewpoint image, and will be used when we need to update pixel values.

Display Update

- Refresh the display with the volume rendered image
- Intermix geometry to indicate the position and orientation of the haptic pointer, and the bounds of the haptic space
- Image is displayed at 20-30 times/second



During haptic interaction, the image on the screen is generally updated about 20 to 30 times per second. If our visual update rate drops below 10Hz, the “jump” in the image is too distracting for effective interaction. At each image update we need to refresh the ray cast image, and intermix a geometric representation of the haptic pointer and, if desired, the bounds of the haptic space into this image. We refresh the ray cast image by copying both the RGB and Z values of our current working image into the frame buffer. We then use the graphics hardware to draw a polygonal representation of the position and orientation of the haptic pointer, and possibly a wireframe outline of our haptic workspace. This method works well for opaque isosurface rendering, but for translucent rendering we do not have a single Z value to store in the frame buffer. Since we do not have the computational resources to accurately blend the geometric haptic pointer with the translucent volume rendered image, we can instead select an opacity threshold at which to capture a Z value. The Z value is then the distance along the ray at which the opacity first exceeded this threshold.

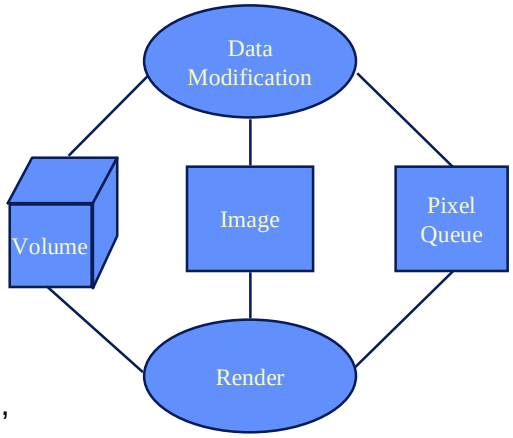
Image Update

During Data Modification

The pixels affected by the modification are marked and placed in the pixel queue. The segment information for each pixel is updated.

During Rendering

A pixel is removed from the pixel queue and updated. The new pixel value is stored in the image, and the pixel is unmarked.



If data modification is not being performed, we would simple need to refresh the RGB and Z values in the frame buffer with the initial fixed viewpoint image for each image update, then intermix the geometry in the correct location. If data modification is occurring, then we will need to update the initial image to reflect this modification. Each time a modification operation is performed, the region of the volume modified is projected into screen space to determine the pixels in the image that are affected. These pixels are added to the pixel queue indicating that they require updating. They are also marked in the image indicating that they have an update pending. This allows us to avoid adding the same pixel to the queue twice. In addition, the segment information indicating non-empty regions along the ray is updated if necessary since additional material may have been deposited in the volume. At every iteration of the haptic loop, a small number of pixels are selected from the pixel queue for updating. The ray casting method uses the segment information to avoid doing work in empty regions of the volume. The new pixel value is stored in the image, and the pixel is unmarked.

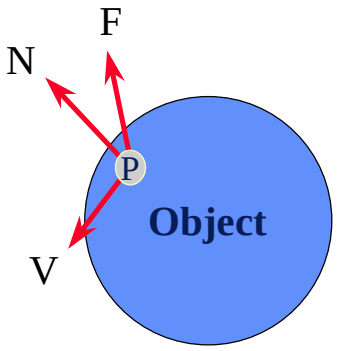
Outline

- Volume Representations
- Haptic Rendering
- Volume Modification
- Visual Rendering
- ✔ **Integration Issues**
- Applications

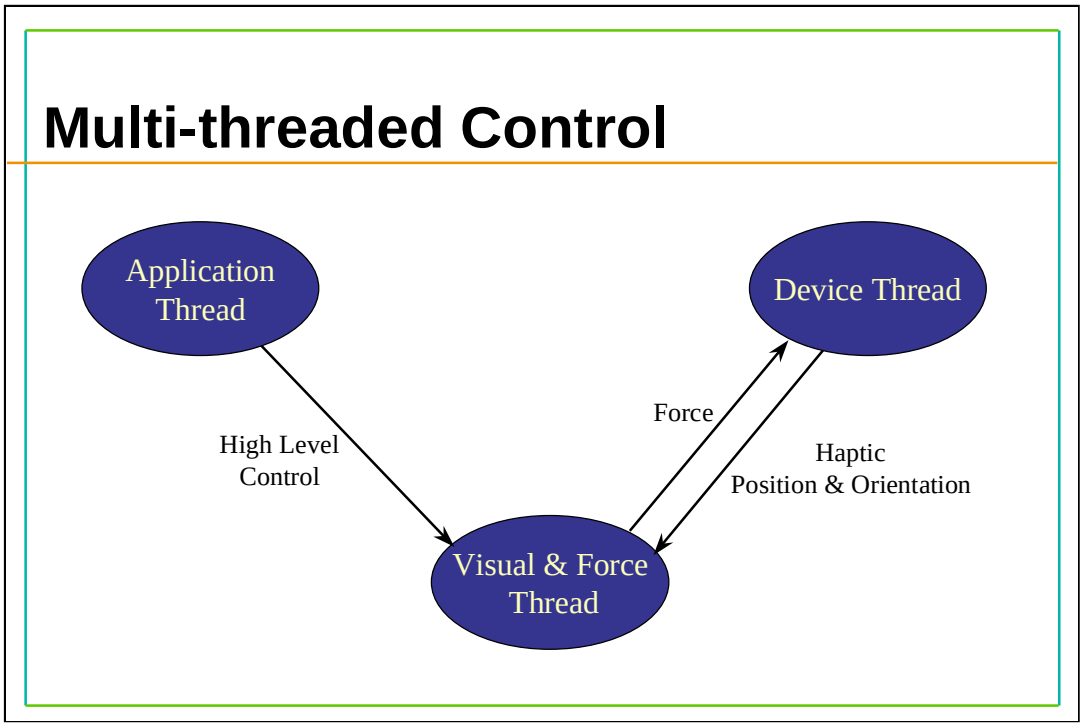
The next section covers some of the issues we dealt with when putting all the components of a haptics application together.

Hard Coded Control

- Loop (2 - 5 Khz)
  - Get Position and Velocity
  - Calculate and Apply Force
    - Test for contact
    - Calculate normal, distance
  - Modify Object (20 Hz)
    - Apply filter to voxel properties
  - Render Pixels
  - Update Image (20 Hz)



There are several possibilities when it comes to the control of a haptics application. We chose to create a hard-coded control loop that runs at 2-5 Khz on an R4400 and precisely specifies the timing of each part of our code. This loop begins with obtaining the position and velocity of the haptic pointer. This information is first used in a set of routines that calculate forces from the volume. Forces are calculated every time the control loop is executed. Once forces have been calculated and sent to the haptic device we check to see if it is time to allow modifications to the object. Modifications to the object can occur 20 times per second. Here is where we apply a filter to the appropriate information stored in the volume. A small number of rays (typically 2) are cast at every loop iteration to capture any changes that have been made in the volume. Finally, we check to see if it is time for an image update. We allow this to happen 20 times per second. This very careful control of the application loop ensures that our application priorities are met. It is also important to add code into the loop that does some safety checks. If our forces or velocity exceeds a low threshold we shut down force feedback.



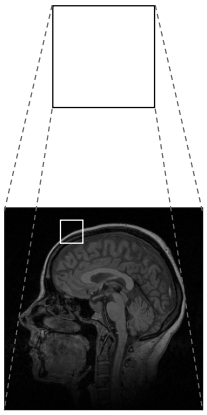
An alternative is to break up the application across several threads. There are several ways to break up the application. We have been using the following approach. An application thread controls the main application loop, user interface, and standard keyboard and mouse input. A visual and force thread is responsible for rendering and modeling physical interaction. This thread then communicates with a dedicated haptic device thread that is responsible for sending forces to the haptic device and obtaining the state of the device. The device thread always maintains a 1Khz or greater force refresh rate. This is significantly different than the hard-coded control loop since each of these threads operates asynchronously.

Workspace Scaling & Panning

Workspace Limitation
- To Scale Mapping Often Not Possible

Stability Issue
- Stationary Ability (~1mm)

Scaling & Panning
- Scale Virtual Workspace (1mm < 1/10 voxel)
- Pressing on Workspace Walls Pans
- Panning Velocity Proportional to Force



Real Workspace

Virtual Workspace

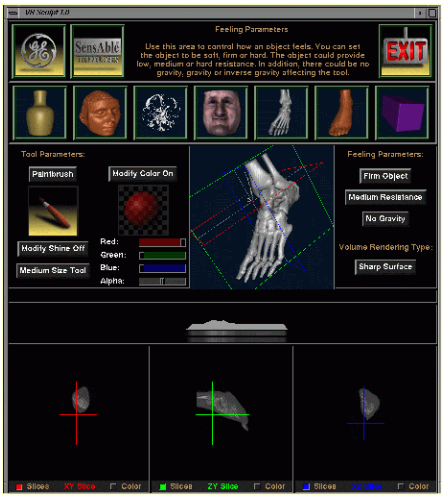
A problem that often arises with haptic applications is that the real haptic workspace is usually smaller than we would like. For instance, interaction with a model of a full human body would require a workspace larger than haptic devices can provide. The simple solution to this is to scale the virtual world to fit in the workspace of the device. The trouble with this approach is that we can only hold our hands steady within about a 1mm radius. If due to scaling that 1mm distance maps to 2 centimeters across a human model, my instability at 1mm will be magnified to 2 centimeters. At each iteration of the haptic control loop the position of the haptic pointer may jump a large distance and can produce some distracting forces depending on the spatial frequency of the volume. A solution to this problem is to reduce the mapping so that a 1mm haptic workspace distance maps to a small distance in voxels. But this means that only a small region of the volume will fit in the haptic workspace. This is addressed by placing a box around the workspace and letting the user pan this box through the volume by pressing on the walls of the box. The harder the user presses the faster the box moves.

Outline

- Volume Representations
- Haptic Rendering
- Volume Modification
- Visual Rendering
- Integration Issues
- ✔ **Applications**

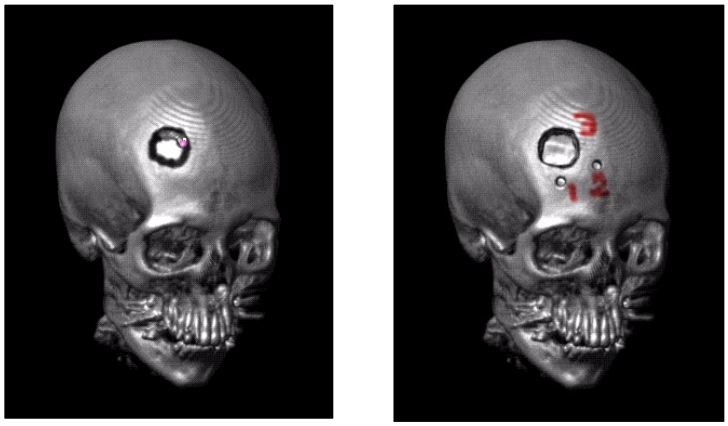
This section describes the application of the previously described techniques in the areas of volume visualization and modeling.

VRSculpt



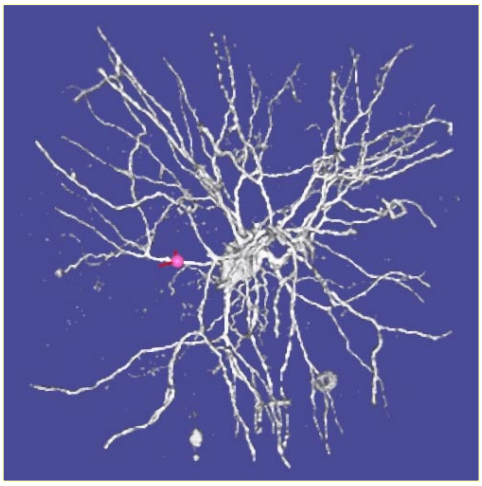
VRSculpt is an application we have developed that is used to demonstrate the volume haptics techniques. It allows a user to see, feel and modify volumes as either an isosurface representation or a translucent volume rendered representation. This application will be running during the live demonstrations at the end of this course.

Medical Application



This example illustrates the use of volume rendering and haptics to interactively explore and modify medical data. This 256x256x225 CT scan of a human head was segmented and volume rendered to highlight bone densities. A tool (shown as a small pink ball) was used to feel the structure of the bone in areas which would be difficult to visually explore. The image on the left shows the tool being used to cut away the skull while tracing over a previously painted black circle. The right image shows the result after three cutting operations were complete. Each cut was numbered directly on the volumetric data using a painting operation.

Scientific Application

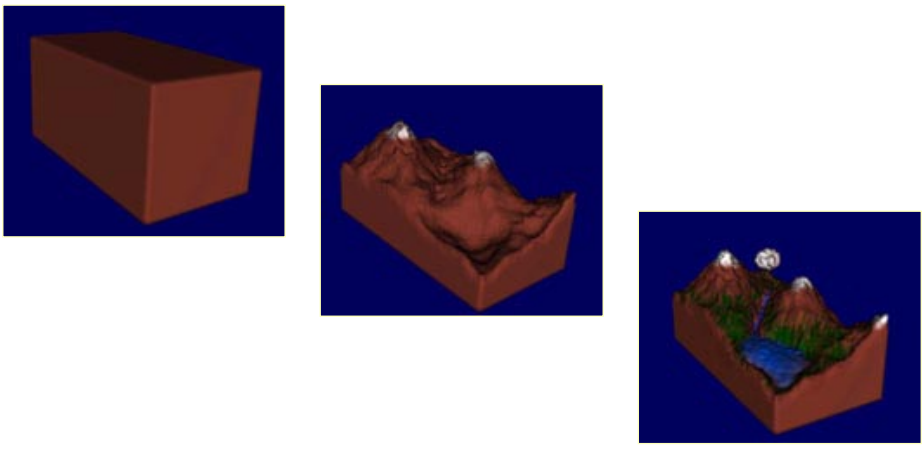


The visual exploration of a complex 3D data set, such as this confocal scan of a lateral geniculate nucleus (LGN) neuron, can be enhanced through the use of haptics. In this example a user is able to feel the structure of the cell and follow dendrites through complicated winding paths. A gentle attracting force was used to follow the dendrites since repelling forces make dendrite tracking difficult in areas where the dendrite changes direction often.

3D Sculpting Application

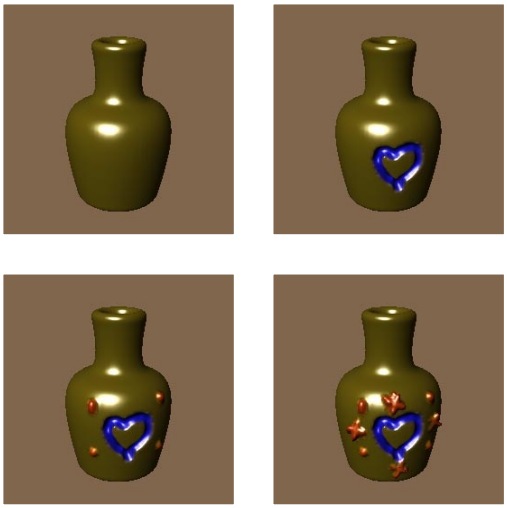
This is an example of volume modeling where an object is created from an "empty" 66x66x66 volume. Using a large tool, dense brown material was added to the volume to form the trunk of the tree. A smaller tool was then used to add the green leaves to the tree. Less dense white material was added to form the clouds in this scene. Small carving tools and painting tools were then used to add detail to the scene, for example, the hollowed hole in the tree trunk. This scene was volume rendered using a compositing technique in order to capture the translucent volumetric objects such as the clouds.

3D Sculpting Application



The following terrain images show the transformation of a 67x127x67 block of material into a volumetric terrain. The material is partially translucent, and therefore a volume rendering method was used to generate the images. Haptic interaction was utilized to sculpt, add scalar value, and paint the volumetric object. The left image represents the initial block of material, while the middle image shows the result of carving out mountains and valleys. In the right image, water has been added to the valleys, grass and trees have been painted on the hills, and a cloud was placed in the sky.

Modeling Application



The above four images demonstrate the ease with which a generic volumetric model can be customized. The initial model, shown in the image on the upper left, contains 115x101x75 data samples. The first step in this transformation is to carve a heart into the surface of the vase while applying blue paint. Next, round red bulges are added to the surface. Finally, red X's are placed on the vase.

More Information

www.crd.ge.com/esl/cgsp/projects/haptics

More information on the subject of haptic volume visualization can be found at the web site above. This site has images and MPEG animations of the examples described in this course.

References

[1] R.S. Avila and L.M. Sobierajski, "A Haptic Interaction Method for Volume Visualization," *Visualization '96 Proceedings*, pp. 197-204 (October 1996).

[2] J. Blinn, "Light Reflectance Functions for Simulating Clouds and Dusty Surfaces," *Computer Graphics*, **16**(3), pp. 21-29 (July 1982).

[3] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graphics*, **22**(4), pp. 64-75 (August 1988).

[4] T.A. Galyean and J.F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique," *Computer Graphics*, **25**(4), pp. 267-274 (July 1991).

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

[6] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics & Applications*, **8**(3), pp. 29-37 (May 1988).

References

[7] W. Lorensen and H. Cline, “Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm,” *Computer Graphics*, **21**(4), pp 163-169 (July 1987).

[8] J.P. McLaughlin and B.J. Orenstein, “Haptic Rendering of 3D Seismic Data,” *The Second PHANToM User’s Group Workshop*, (October 1997).

[9] A. Mor, “Interacting with 3-Dimensional Medical Data,” *The First PHANToM User’s Group Workshop*, (September 1996).

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

[11] L. Westover, “Footprint Evaluation for Volume Rendering,” *Computer Graphics*, **24**(4), pp. 367-376 (August 1990).

[12] C. Zilles and K. Salisbury, "A Constraint-based God-object Method For Haptic Display," *Proceedings of the International Conference on Intelligent Robots and Systems*, (August 1995).

Assembly and Path Planning

Ricardo S. Avila

General Electric
Corporate Research & Development
Niskayuna, New York 12309

Industrial Applications

Maintenance & Assembly Analysis

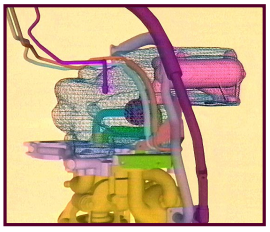
- Must Be Considered During Design
- Removal Path Planning
- Assembly Sequencing

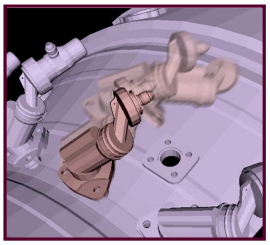
Training

- Hands-on Training is Necessary
- Physical Parts Expensive
- Numerous Product Configurations

Tools

- 3D Visualization-based Applications
- Complex User Interfaces

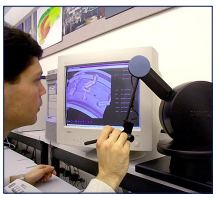




Haptic Environments

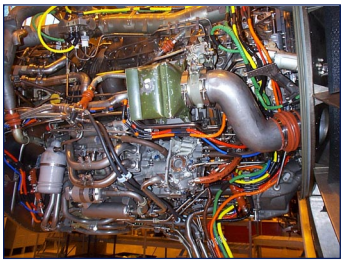
Why Haptics?

- Increased Productivity
- Cost Effective
- Complete Monitoring
- Intuitive and Therefore Accessible



Issues

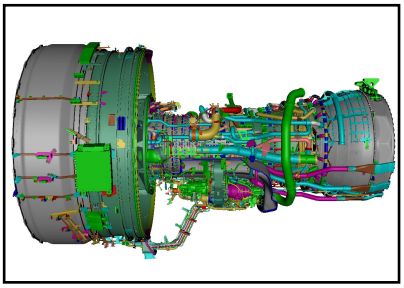
- Collision Detection
- Force Calculations
- Haptic Devices
- Are We There Yet?



Collision Detection

Numerous Techniques Exist

- Volume-based
- OBB
- Sphere tree



Issues

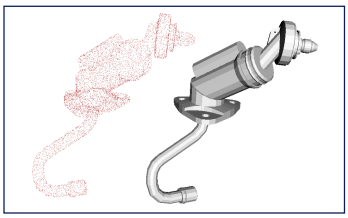
- Data Size
  - 10K Polygon Moving Object
  - 100K Polygon Stationary Objects
- Collision Time
  - .001 seconds
- Feature Size
  - 2 meters down to .001 meters

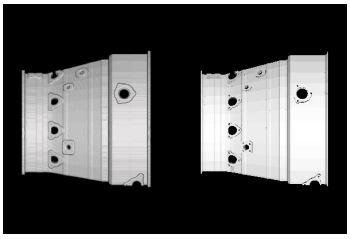
Volume Approach

Stationary Objects
- Build a Distance Volume
- Compress Representation

Moving Object
- Point Sample Polygonal Surface

Algorithm
- Sample the Volume
- Calculate Scalar Value
- Estimate Penetration Distance
- Calculate Force Contribution



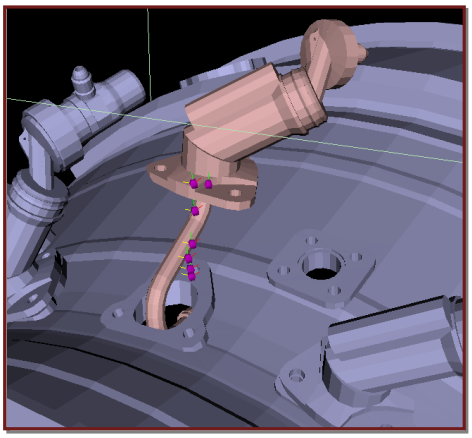


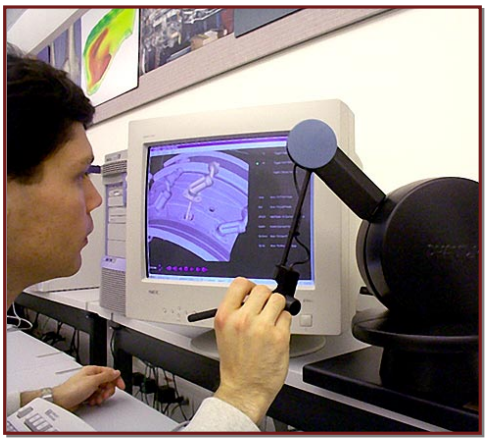
Devices

3 DOF Device
- 3 DOF Device, 6 DOF Problem
  - Translation or Rotation
  - One leads, one follows
- Suitable for Some Problems
- Motion is not Always Intuitive

6 DOF Device
- Intuitive Interaction
- Training Workspace
- Safety

Maintainability





Video

A Haptic Interaction Method for Volume Visualization

Ricardo S. Avila and Lisa M. Sobierajski

GE Corporate Research & Development
Schenectady, New York 12345

Abstract

Volume visualization techniques typically provide support for visual exploration of data, however additional information can be conveyed by allowing a user to see as well as feel virtual objects. In this paper we present a haptic interaction method that is suitable for both volume visualization and modeling applications. Point contact forces are computed directly from the volume data and are consistent with the isosurface and volume rendering methods, providing a strong correspondence between visual and haptic feedback. Virtual tools are simulated by applying three-dimensional filters to some properties of the data within the extent of the tool, and interactive visual feedback rates are obtained by using an accelerated ray casting method. This haptic interaction method was implemented using a PHANToM haptic interface.

1. Introduction

Traditional methods for visualizing volumetric data rely almost entirely on our powerful sense of vision to convey information. While this has proven quite effective for most visualization tasks, it remains worthwhile to investigate the benefit of augmenting these visualization methods with information obtained through other sensory channels. In particular, our sense of touch, in combination with our kinesthetic sense, is capable of supplying a large amount of information about the structure, location, and material properties of objects [6]. The study of the many issues related to interaction with an environment through the sense of touch is known as haptics.

Haptic rendering is the name given to the process of feeling virtual objects [13]. This involves tactile feedback for sensing properties such as surface texture, and kinesthetic feedback for sensing the shape and size of objects. Traditional methods for producing convincing haptic renderings have mainly utilized scenes comprised of geometric primitives such as polygons, spheres, and surface patches. These investigations have generally focused on simulating realistic interactions with static and dynamic collections of geometric objects given the capabilities and limitations of haptic devices. Although the benefits of haptic rendering volume data have been recognized [8], this area of research has not yet been fully explored.

Haptic interaction has been successfully applied to simulate specific tasks in several application areas. In molecular docking studies, a robotic arm was used to supply molecular interaction forces [3]. In another application, a haptic device was used as a nanomanipulator for a scanning tunneling microscope [16], enabling scientists to manipulate individual atoms on a surface. A medical planning and training system [4] has also been developed which simulates knee palpation through the use of visual and haptic feedback.

One goal of the work presented in this paper is to develop a haptic interaction method for use in a volume visualization system. There are several reasons to pursue the addition of haptic cues to volume visualization. The use of a force feedback device during visualization is a natural output method for interactively conveying complex information to the user. This is particularly useful when the user attempts to precisely locate a feature within a volume, or to understand the spatial arrangement of complex three-dimensional structures.

A second goal of this haptic interaction method is to allow the haptic device to be used for input as well as output. The position and orientation of the haptic device could be used to simulate a virtual tool that could modify local properties of the volume dataset. For example, this modification ability can be employed during data exploration to alter visibility of one structure to allow visual access to another. Data modification can also be used for three-dimensional annotation of scientific data. In addition, the field of volume graphics [9] can benefit from a haptic data modification method that allows for interactive, virtual volume modeling [17].

This paper is organized into eight sections. An overview of the haptic interaction method is given in Section 2. The volume data representation used for visual and haptic interaction is covered in Section 3. We discuss our haptic rendering method in Section 4, while the corresponding volume rendering method is described in Section 5. Techniques and tools for data modification are covered in Section 6. Our implementation, and some results of this method are given in Section 7, while Section 8 concludes the paper with a discussion of future work.

2. System Overview

We identified several major requirements that must be met by a haptic visualization method in order to provide meaningful force feedback and data modification capabilities.

- **Constant haptic refresh rate:** Large variations in the rate at which forces are updated can produce distracting tactile artifacts.
- **Fast force calculations:** Complex force computations would reduce the haptic refresh rate and would therefore decrease the amount of processing time available for rendering and data modification.
- **Fast, incremental rendering:** Interactive render rates are necessary for visual feedback of the haptic pointer location and data modification, and the time cost of rendering must be amortized over a number of force feedback iterations to maintain a consistent haptic refresh rate.
- **Fast data modification:** Interactive data modification rates are required for both visual and force feedback.
- **Consistent haptic and volume rendering:** Volume rendering and haptic rendering should be consistent. A structure which appears amorphous should also feel amorphous**.**

To satisfy these five major requirements, we made some assumptions about the force feedback, rendering, and data modification computations that could occur during haptic interaction. First, the force feedback and data modification calculations are restricted to using only a local area of data values. In addition, viewing, lighting, and global material properties are fixed during haptic interaction, and a local data modification operation must effect only a small region of the displayed image.

Based on the major requirements of the system, and the assumptions that were made, we developed the haptic visualization method illustrated in Figure 1. The haptic interaction loop begins after an initial image of the scene has been computed. The first step in the interaction loop is to obtain the current position and orientation of the haptic pointer from the physical device. We will refer to the virtual counterpart of this physical pointer device as the "tool", since it will often be used, for example, as a virtual scalpel, chisel, or paintbrush. If we determine that a data modification operation is necessary at this time, the modification computation is performed, and the volume database is updated. In addition, the pixel status buffer is updated to indicate which pixels of the image have been affected by this modification operation. Once this optional modification step is complete, the current force is computed and supplied to the haptic device. During the rendering phase, some small number of the pixels that require updating are rendered using a ray casting method. Finally, if it is time to refresh the physical display device, the current image is copied to the screen and the graphics hardware is used to render a geometric object that indicates the size, position, and orientation of the current tool.

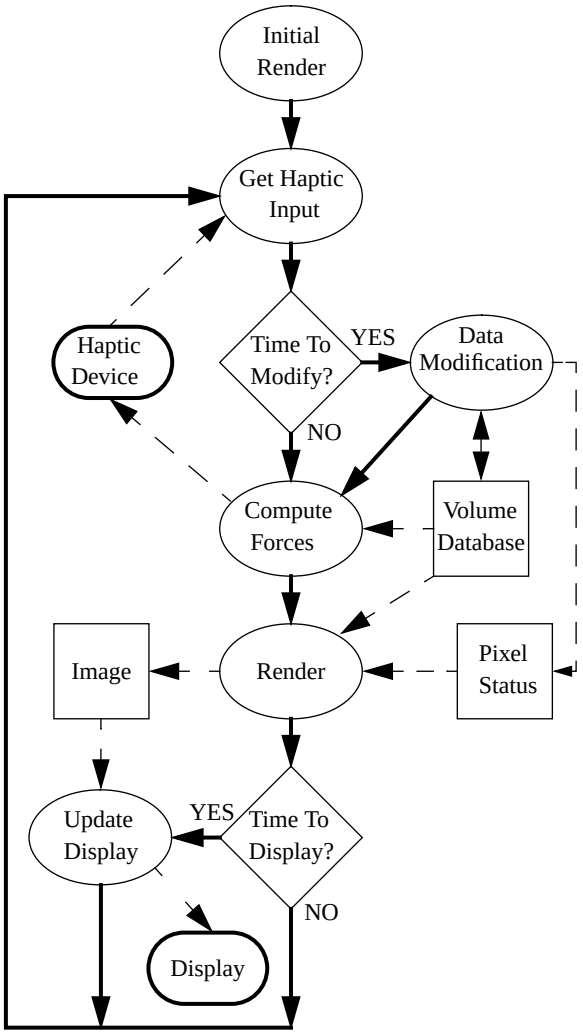


Figure 1: An overview of the haptic visualization method. Solid lines indicate control flow while dashed lines show data flow.

The data modification operation does not occur during every iteration of the haptic interaction loop. Instead, a timer indicating elapsed time since the previous modification is consulted during each iteration. If this elapsed time exceeds some threshold, the modification operation is performed. The main reason for limiting the rate at which data modification occurs is that we are maintaining a haptic refresh rate of 1 to 5 KHz. Therefore, there is only a small amount of computational time left over after the force calculation in each iteration. Increasing the rate of data modification would decrease the amount of time available to update the pixels of the image affected by the modification.

The rate at which the physical display device is refreshed is also limited by an elapsed time threshold. In this case, a 30Hz limit is imposed since a refresh rate much greater than this is unnecessary.

3. Data Representation

A volume is represented as a 3D rectilinear array of volume elements, or voxels, each specifying a set of scalar properties at a discrete grid location. An interpolation function is used to produce a continuous scalar field for each property. This is critical for producing smooth volume and haptic rendering.

In order to meet the requirements of the system, the contents of each voxel must contain a large number of physical properties. This includes a scalar value for density, values for material classification and shading properties, as well as values for mechanical properties such as stiffness, and viscosity. In addition, it is often desirable to precompute and store values that do not often change such as density gradients and partial shading results for each voxel. Given unlimited memory resources, each voxel would contain high precision storage for each of these parameters.

One possibility that we considered was to store the volume in a space-efficient, hierarchical data structure such as an octree, thereby reducing storage requirements in empty areas of the volume. There are two problems with this approach. First, maintaining the requirement of a consistent haptic refresh rate is far easier when the time required to read and modify voxel values is constant. Second, in many cases data modification operations will lead to datasets that can no longer be stored efficiently in hierarchical data structures.

When defining the values stored in each voxel in the rectilinear grid, we considered interactive rendering rates to have highest priority, since that is a direct requirement of the system. The ability to render large datasets was given the next highest priority, and property modification flexibility was considered last. From these priorities, we obtained the definition of a voxel requiring 8 bytes as shown in Table 1. Three bytes are allocated for gradient magnitude and direction in order to save time during volume rendering. A 24 bit color, rather than a color LUT, is stored within each voxel to ensure compositing and painting operations on volumes include fine details. A collection of material properties is indexed through a look up table. An entry in the table specifies additional characteristics such as material classification and shading parameters. Haptic properties such as stiffness and viscosity may also be assigned to an index.

Material opacity is generally stored in a table indexed by the LUT index, the density value, or both the density and gradient magnitude values. Material classification, shading properties, and haptic properties are typically obtained through a segmentation process applied to the density values.

Table 1: Voxel Representation

Property	Type	Size (bytes)
Density	Scalar	1
Gradient Direction	Encoded Unit Vector	2
Gradient Magnitude	Scalar	1
Color	R,G,B	3
Material Properties	LUT Index	1

4. Haptic Rendering

The system allows for the exploration and modification of both isosurface and translucent volumes. The forces generated can either be constructed to approximate a realistic feel of a virtual object or to convey meaningful structural information for data exploration purposes. In the case where the user wishes to explore internal structures of a rigid body, such as bone, it is desirable to produce contact forces and visual characteristics which are inconsistent with bone, but that allow the user to penetrate the bone to feel and see internal structure.

The force equations are based on two principal requirements. First, the interaction forces must be calculated fast enough to be used within an interactive system. Typically, force update rates of 1-5 KHz are generated for this system. Second, the forces imparted to the user should be consistent with the rendering of the volumetric object.

In order to meet the speed requirement and since the haptic device we used can only handle translation forces, the force calculation is simplified to a point contact. This has been shown to be a reasonable simplification for many tasks [12]. The general equation we used for feeling an object using a point contact model is:

$$\vec{F} = \vec{A} + R(\vec{V}) + S(\vec{N})$$

and is illustrated in Figure 2. The force $\vec{F}$ supplied to the user located at position P and moving in direction $\vec{V}$ is equal to the vector sum of an ambient force $\vec{A}$, a motion retarding force $R(\vec{V})$, and a stiffness force normal to the object $S(\vec{N})$.

The ambient force is the sum of all forces acting on the tool that are independent of the volumetric data itself. Some forces such as gravitational or buoyant forces are independent of the tool position while other forces such as synthetic guide forces, which aid the user during interactive volume modification, are dependent on position. For example, a virtual plane perpendicular to a surface can be used as a guide when attempting to cut a straight line. The ambient force would be used to guide the tool back to the plane.

The motion retarding force is proportional to velocity and can be used to represent a viscous force. The last term captures the stiffness of the object and is always in the direction of local gradient. When simulating interaction on rigid surfaces, which are generally governed by Hooke's law, this term can then be set to a linear force function in the direction of the surface normal and proportional to the penetration distance of point P.

This general equation for force feedback is the basis for calculating forces which are consistent with different types of rendering methods. The forces generated are not intended to be a realistic simulation of interacting with materials. Rather, the intent is to convey additional information to the user about the data being explored.

The display of volume data requires a segmentation step in order to determine the visual appearance of the projected volume. In a similar manner, we introduce a segmentation step which produces tactile properties to the volume. In order to ensure consistency between visual and haptic rendering, the transfer functions used for the assignment of visual and tactile properties are similar.

4.1 Haptic Volume Rendering

When rendering a translucent volume we employ a gradient magnitude segmentation method [11] in order to assign opacities. The segmentation method specifies an opacity transfer function $\alpha = t_\alpha(d, |\nabla d|)$, where the opacity value α at a sample location is defined by both the material density and the magnitude of the density gradient at that location. The function t_α can be specified in a number of different ways. As was done in [10], we compute α values by multiplying a density transfer function by a gradient magnitude transfer function. In order to keep volume and haptic rendering consistent, force transfer functions t_r and t_s are constructed which are similar to t_α, but produce force magnitudes rather than opacities. The retarding and stiffness force functions for haptic volume rendering become:

$$R(\vec{V}) = -\vec{V} t_r(d, |\nabla d|)$$

$$S(\vec{N}) = \frac{\vec{N}}{|\vec{N}|} t_s(d, |\nabla d|)$$

The normal vector $\vec{N}$ is computed using central differences. We found that a linear correspondence between the visual transfer function and the haptic transfer functions produced an intuitive force response, as in:

$$t_r(d, |\nabla d|) = C_1 t_\alpha(d, |\nabla d|) + C_2$$

$$t_s(d, |\nabla d|) = C_3 t_\alpha(d, |\nabla d|)$$

Essentially, the more opaque a material, the greater its stiffness and motion retarding properties. The stiffness function has an implied zero additive constant to ensure that the initial contact with an object starts from a zero force.

Other mappings of the opacity transfer function may be suitable depending on the type of forces required. For instance, an exponentially increasing opacity transfer function may be translated into a linear force response through the use of a logarithmic function.

If our intent was to simulate a realistic haptic and visual rendering of a volume, then the segmentation of the volume into all relevant material characteristics would be necessary. The properties in the new representation would replace the approximations found in the visual and haptic transfer functions.

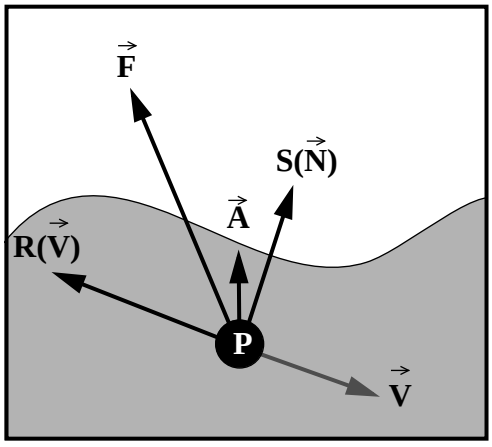


Figure 2: Forces acting on a haptic sensing point P which is moving at a velocity $\vec{V}$.

4.2 Haptic Isosurface Rendering

The fast and robust calculation of stiffness and motion retarding forces is essential when interacting with volumetric isosurfaces. Unfortunately, the stiffness computation requires that the penetration distance of the tool below the isosurface is available at every location in the volume. While it is possible to precompute the distance to an isosurface for every voxel in the volume, we decided to investigate techniques for approximating stiffness and retarding forces based only on the density field. There are two reasons for this. First, the system allows for the interactive modification of the volume. Creating a new distance map for the volume would be prohibitive. Second, for small penetration distances, the density field itself can give a reasonable approximation of the distance to an isosurface.

Similar to volume rendering, the retarding and stiffness force functions used to feel a surface are dependent on transfer functions:

$$R(\vec{V}) = -\vec{V} f_r(d)$$

$$S(\vec{N}) = \frac{\vec{N}}{|\vec{N}|} f_s(d)$$

Here the density d is used as an indicator of penetration distance in the thin shell between the isosurface density values d_i and d_j, where $d_i < d_j$. The function $f_r(d)$ maps density values into retarding force magnitudes while $f_s(d)$ maps density value into stiffness force magnitudes. We set these functions to:

$$f_r(d) = \begin{cases} (d_i < d \le d_j)\ C_4(d - d_i) + C_5 \\ otherwise \quad 0 \end{cases}$$

$$f_s(d) = \begin{cases} (d_i < d \le d_j)\ C_6 \dfrac{(d - d_i)}{(d_j - d_i)} \\ otherwise \quad 0 \end{cases}$$

The retarding force is set to a linear function proportional to the difference in density above d_i. Similar to haptic volume rendering, the coefficients C_4, C_5, and C_6 specify a linear mapping from density values to force magnitudes. The stiffness force varies from zero to C_6 depending linearly on where the value d lies between d_i and d_j. This can be viewed as a penetrable shell model with viscous internal properties. A nice property of this model is that it allows the user to feel subsurface structure when the density and normal vector change below the surface.

5. Rendering

In order to provide fast rendering of isosurfaces and translucent volumes, we use a volumetric ray tracing method [15] to generate images of the volumetric data. This method is flexible since it allows for the rendering of multiple independent, possibly overlapping volumetric objects. If the viewing position is fixed and global effects are ignored during the haptic interaction loop, then data modification operations correspond to local image updates within the image-space extent of the modified region of the volume.

When modeling a solid object, an isosurface representation is a natural choice for both force feedback and rendering. To produce high quality images, the ray tracer computes the analytical intersection of the ray with the isosurface as defined by the interpolation function, and a central differences technique is employed to estimate surface normals. When modeling amorphous objects such as smoke and clouds, a volumetric feedback equation and rendering method are required. Images are generated by sampling material properties along a ray, and compositing them to produce a final pixel intensity value.

Using a standard ray tracing method, the computation required to update even a small region of the image (typically 30^2 to 50^2 pixels for a 512^2 image) may be too slow for interactive object modification. Therefore an acceleration method must be employed to achieve interactive frame update rates. In this haptic interaction method, the Polygon Assisted Ray Casting (PARC) method [14] is employed to avoid casting rays through empty regions of the volume. This acceleration method relies on a projection of a geometric approximation of the volume to avoid segments of rays that pass through empty regions of the volume. We use the standard hardware projection method to render the full geometric approximation when the initial image is generated. The geometric approximation is updated whenever the scalar field of a volume is altered by a data modification operation. A software projection method is then used to update only the affected pixels with the new geometric approximation.

When data modification occurs, flags are set to indicate which pixels must be recomputed. During each iteration of the haptic interaction loop, some small number of pixels are updated (typically less than 10), and the flags for these pixels are cleared. When the display device is refreshed, the current image may contain some pixels which have not yet been updated, producing results similar to those obtained by frameless rendering techniques [2]. Typically these effects are not noticed since the tool obscures the region of the volume being modified for a few frames. By the time this region becomes visible in the image, these pixels have been updated.

The current image is stored in the image buffer, with a color, opacity, and depth value stored for each pixel location. The depth value indicates the distance from the image plane at which the ray cast through that pixel accumulated an opacity value greater that V_α. When the display device requires refreshing during the haptic interaction loop, this image is copied into the color and depth components of the framebuffer. Fast projection of the tool is then obtained through the use of graphics hardware. For opaque surfaces, opacity values stored in the pixels are binary, therefore $V_\alpha \neq 0$ is equivalent to $V_\alpha = 1$, and this method correctly combines the ray traced image of the volume with the projection of the tool. When the tool is within a translucent volume, the combination provides only an approximate image since the opacity value stored in a pixel represents the final opacity of all accumulated samples, and the distance value represents the location along the ray at which the accumulated opacity reached V_α. In our experience, adequate images are generally produced with $V_\alpha \approx 0.25$. If more accurate combined images are required during the modification of internal features in a translucent volume, they can be generated at the cost of memory or computation time.

6. Data Modification

The data modification component of our haptic visualization method is an optional component that can be used to modify any local property of the data stored in the volume database. A local property is one that affects only

a small region of the volume, and therefore will cause only a small region of the image to require updating. As described in Section 3, three modifiable values are stored for each data sample: material density, color, and an index value. Density and color are independent data properties. Stiffness, material classification, and material shading parameters such as opacity and ambient, diffuse, and specular reflectivity are dependent properties stored in look-up tables that are indexed by either the index value, or the density value. The values in the look-up tables cannot be modified since this would result in global changes in the appearance of the volume. Instead, the index value or density value used to index the look-up table is modified.

Independent data properties can be modified by setting them to a constant value, or by applying a filter to the current values. The index value defining dependent data properties is generally only modified using the constant value method unless there is a linear relationship between this index value and all the properties represented by this value. For the independent color values, we could use the constant value method to "paint" an object red by setting the color property at the tool location, or some small region centered at the tool location, to red. In addition, we could define the material properties of the paint by also setting the index value, which would change all properties dependent on the index value simultaneously.

With the filter method, we could "melt" an object by updating density values in a small region around the tool location according to $d_i = (1-\alpha)d_{i-1}$, where d_i is the new density value, d_{i-1} is the current density value, and α is obtained by sampling a spherical filter with a Gaussian distribution centered at the tool location. In contrast to melting, we can "construct" an object using $d_i = \alpha D + (1-\alpha)d_{i-1}$, where D is the density of the material that we are adding to the volume. Note that melting is just a special case of constructing with $D = 0$. In fact, constructing will appear like melting whenever the opacity of d_i is less than the opacity of the density d_{i-1} that it is replacing.

The two modification methods described above can lead to a wide variety of tools by varying the constant value or filter type, and the data properties that are modified. The most difficult part of defining a new tool is selecting a tool name, since there are many possible virtual tools that do not have physical counterparts. In our system, a modification operation is defined by providing the modification method, the extent of modification, the properties affected, and any necessary filter and constant values. A tool is composed of one or more modification operations, with the condition that no property is affected by more than one modification operation in a tool. Figure 3 illustrates the use of some common tools on a volumetric wall. These tools are described briefly in Table 2, where the first column indicates a tool name, the second column lists the modification method used for each operation, the third column defines the properties that are modified for each operation, and the last column describes the modification process. Specific constants for the instance of the tools shown in Figure 3 are given in parenthesis in the last column.

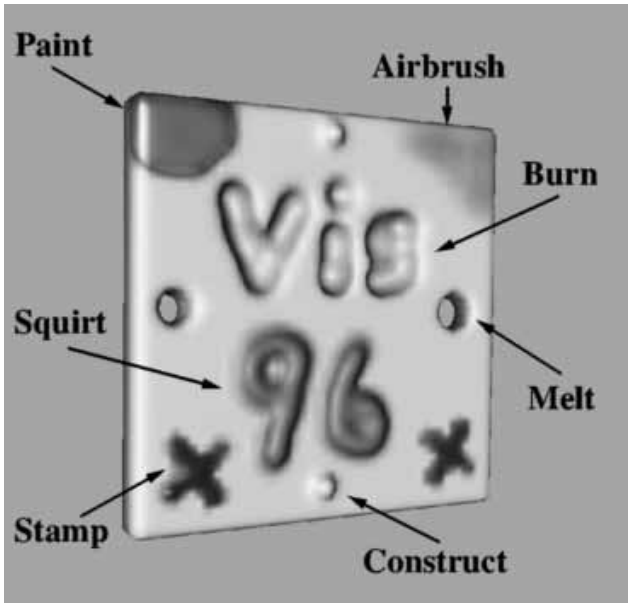


Figure 3: An example of various tools applied to a volumetric wall.

Table 2: Common Tools

Tool Name	Method	Property Affected	Description
Melt	Filter	Density	remove density
Construct	Filter	Density	add density (63% dense)
Burn	Filter	Density	remove density
	Filter	Color	blend in black
Squirt	Filter	Density	add density (63% dense)
	Filter	Color	add color (red)
Stamp	Filter	Density	add shape (cross-hair filter)
	Filter	Color	add color (green)
	Constant	Index	set material (shiny)
Paint	Constant	Color	set color (yellow)
	Constant	Index	set material (shiny)
Airbrush	Filter	Color	blend color (purple)
	Constant	Index	set material (dull)

7. Implementation and Results

The methods discussed in this paper were implemented on a Silicon Graphics Indigo^2 Extreme workstation with a 200 MHz R4400 processor and 96 MB of RAM. A 1.5x PHANToM haptic interface [12] was used to provide force-feedback. The PHANToM connected directly to the workstation through an EISA bus.

The haptic interaction loop described in Section 2 was implemented as a single process, and was responsible for computing forces, modifying the volume, and rendering. We typically visualized volumetric objects using a 512^2 image, with data modification rates of 10Hz, image update rates of 20Hz, and force update rates of 5KHz. Faster data modification rates are possible, but are not practical due to the precision limitations imposed by representing density and the components of color as 8 bit values. The force update rate is higher than the minimum required rate of 1KHz, and could potentially be reduced to 3KHz to provide more processing power for rendering in a larger image.

Special care was taken to ensure that the system would not produce unsafe forces. At the start of a haptic session, the system computes forces but does not apply them until a force within an acceptable range is calculated. This prevents a user from starting haptic feedback in an area of high force magnitude. In addition, all forces are checked against a maximum force threshold. If this value is exceeded, the system shuts down haptic interaction.

Another concern that we faced was the ergonomics of the system. During haptic interaction, the tight link between visual and tactile feedback can often trick a user into believing that she is holding a real tool, and interacting with a physical object. This illusion is shattered as soon as the user attempts to rest her hand against the object for greater control of small movements. We have found that providing places for the user to rest her elbow and wrist can help to maintain this illusion of physical reality. These resting places are also necessary to help reduce fatigue and muscle strain that could be caused by prolonged use of a haptic device.

Figure 4 illustrates the use of the system in understanding a complex set of dendrites emanating from a lateral geniculate nucleus (LGN) cell. This $256 \times 256 \times 195$ voxel LGN cell was scanned with a confocal microscope, and volume and haptic rendered as discussed in Section 4.1. The ability to feel as well as see the dendrites provides a large amount of additional information when visualizing this data. It proved useful to determine the path of intertwined dendrites using haptic feedback. However, a problem was encountered when feeling the path of dendrites since it was easy to slip off the dendrite and have to feel your way back to resume exploration. One solution to this problem was to invert the mapping of opacities to force magnitudes such that the high opacities attracted the tool. This made following winding dendrites a far easier task.

Another example of where haptic interaction can be used to explore scientific data is shown in Figure 5. A $152 \times 261 \times 220$ CT data set of the Visible Human's foot was segmented and visualized as a skin surface and a bone surface. Initially, the bone surface is completely contained within the skin, and is therefore not visible in the image. An "invisibility" tool is used to set the LUT indices on a portion of the skin surface to a value that represents an invisible material, revealing the inner bone structure for visual inspection. Since the density values have not been modified, force feedback is still provided for the transparent skin regions. With a strong input force, the user can "poke" through the skin surface to explore internal features.

The ability to feel and modify a volume rendered object was used to annotate and cut a $256 \times 256 \times 225$ voxel CT head shown in Figures 6 and 7. Figure 6 shows the tool cutting into the surface of the skull, revealing interior regions of bone. Prior to cutting, a black circle was traced on the surface as a cutting guide. The forces assigned to the skull were selected to be rigid, providing a sensation similar to bone. Figure 7 shows the skull after the removal of the cut out section. Additional punctures and annotations were also placed on the surface.

Figure 8 shows a volumetric scene created from an empty volume with a resolution of 68^3 voxels. Several construction and painting tools were used, and the image was generated using a volume rendering technique. This haptic interaction method can form the basis of a volume modeling [17,5] or painting [1,7] system that would allow for the creation, modification, and rendering of both solid and amorphous objects.

8. Future Work

We have found that the integration of haptic interaction into a scientific visualization process can lead to a better understanding of complex, three-dimensional data, and can result in more natural, intuitive methods for modifying this data. One limitation that we encountered during our work was the lack of rotational forces, since the PHANToM device provides only three degrees of translational force feedback. An area of future research that we would like to explore is the extension of our force feedback equations and data modification operations for haptic devices that provide six degrees of freedom in both input and output.

We found it difficult to work with high frequency data using the force equations presented in this paper. When the density field changes from empty space to dense object within one or two voxels, it is difficult to provide force feedback without producing unwanted vibrations. We would like to investigate methods for reducing these vibrations on high frequency data. This may include limit-

ing the speed at which the user can move through these regions, providing spherical rather than point contact, or maintaining auxiliary buffers that indicate the distance to a surface.

Physically realistic haptic interaction is another area that requires further investigation. This includes the ability to obtain, store, and modify material properties that define how an object reacts to an applied force. We would also like to investigate methods for quickly computing forces for more realistic tools. For example, an artist could feel the bristles of the brush interact with the object as he paints.

Acknowledgements

The volumetric wall data set shown in Figure 3 is courtesy of Sidney Wang. The LGN cell shown in Figure 4 is courtesy of Barry Burbach. The CT data for Figure 5 is courtesy of the National Library of Medicine's Visible Human Project. The CT data shown in Figures 6 and 7 is courtesy of North Carolina Memorial Hospital. We would like to thank Thomas Massie for providing valuable feedback on the use of the PHANToM.

References

1. M. Agrawala, A. Beers, and M. Levoy, "3D Painting on Scanned Surfaces," *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pp. 145-150 (April 1995).
2. G. Bishop, H. Fuchs, L. McMillan, and E.J. Scher Zagier, "Frameless Rendering: Double Buffering Considered Harmful," *Proceedings of SIGGRAPH '94*, pp. 175-176 (July 1994).
3. F.P. Brooks, P. M. Ouh-Young, J.J Batter, P.J. Kilpatrick, "Project GROPE:Haptic Displays for Scientific Visualization," *Proceedings of SIGGRAPH '90*, pp. 177-186 (August 1990).
4. G. Burdea, N. Langrana, K. Lange, D. Gomez, and S. Deshpande, "Dynamic Force Feedback in a Virtual Knee Palpation," *Journal Of Artificial Intelligence in Medicine*, **6**, pp. 321-333 (1994).
5. T. A. Galyean and J.F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique," *Computer Graphics*, **25**(4), pp. 267-274, (July 1991).
6. S.R. Geiger, I. Darian-Smith, J.M. Brookhart, and V.B. Mountcastle, "The Nervous System," *Handbook of Physiology*, Volume III, Section 1, pp. 739-788, (1984).
7. P. Hanrahan and P. Haeberli, "Direct WYSIWYG Painting and Texturing on 3D Shapes," *Proceedings of SIGGRAPH '90*, pp. 215-223, (August 1990).
8. H. Iwata and H. Noma, "Volume Haptization," *IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pp. 16-23 (October 1993).
9. A. Kaufman, R. Yagel, and D. Cohen, "Volume Graphics," *IEEE Computer* **26**(7), pp. 51-64 (July 1993).
10. P. Lacroute and M. Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation", *Computer Graphics (Proc. SIGGRAPH)*, pp. 451-457 (July 1994).
11. M. Levoy, "Display of Surfaces from Volume Data", *IEEE Computer Graphics & Applications*, **8**(3) pp. 29-37 (May 1988).
12. 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).
13. 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).
14. L.M. Sobierajski and R.S. Avila, "A Hardware Acceleration Method for Volumetric Ray Tracing," *Visualization '95 Proceedings*, pp. 27-34 (October 1995).
15. L.M. Sobierajski and A.E. Kaufman, "Volumetric Ray Tracing," *1994 Symposium on Volume Visualization*, pp. 11-18 (October 1994).
16. R.M. Taylor, W. Robinett, V.L. Chi, F.P. Brooks, W.V. Wright, R.S. Williams, and E.J. Snyder, "The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope," *Proceedings of SIGGRAPH '93*, pp. 127-134 (August 1993).
17. S. Wang and A. Kaufman, "Volume Sculpting," *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pp. 151-156 (April 1995).

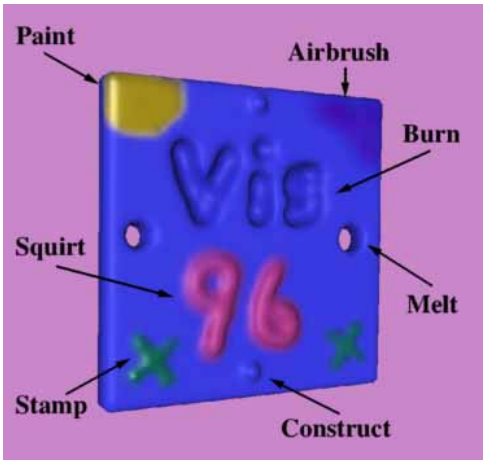


Figure 3: An example of various tools applied to a volumetric wall.

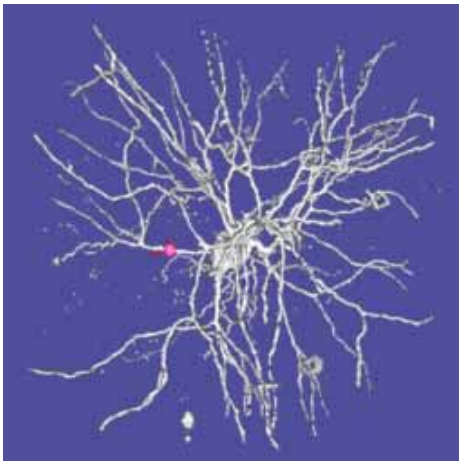


Figure 4: The complex dendritic paths of an LGN cell are explored through visual and force feedback.

Figure 5: An invisibility tool is used to reveal the bone structure within the Visible Human's foot.

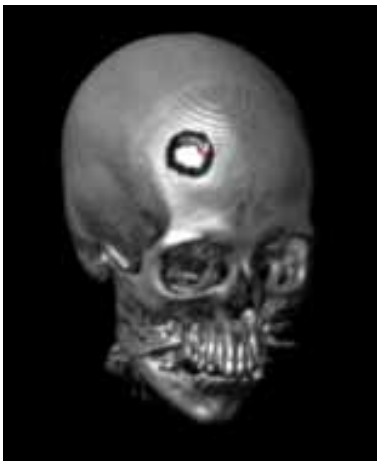


Figure 6: Haptic interaction on a volume rendered CT scan of a human head. A circle was painted on the skull surface and then a cut operation was initiated.

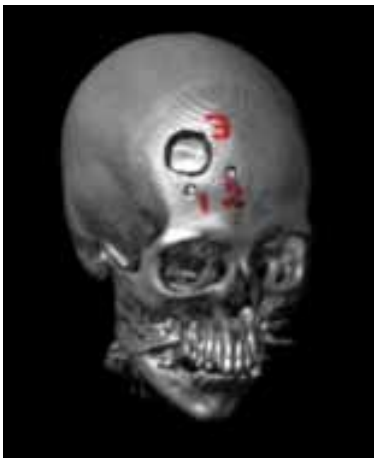


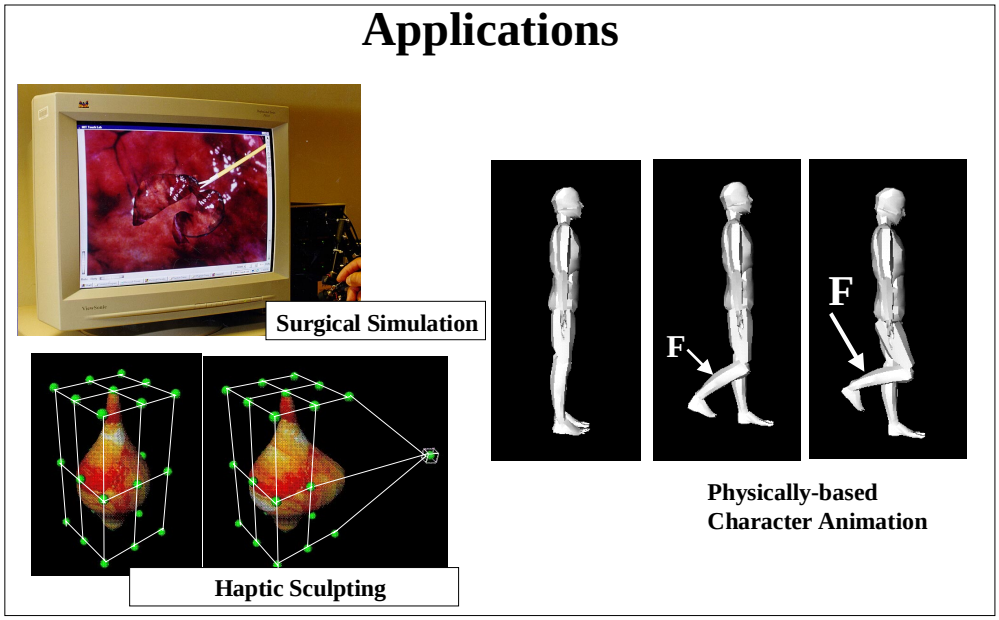
Figure 7: The skull section was removed and several drilling and painting operations were performed.

Figure 8: Haptic modeling tools are used to construct a volume rendered tree and clouds.

Force-Reflecting Deformable Objects for Virtual Environments

Cagatay Basdogan, Ph.D.
Research Laboratory of Electronics
Room: 36-758, MIT, 02139
basdogan@mit.edu

Graphical display of deformable objects has been extensively studied in computer graphics. With the addition of haptic feedback, deformable objects gain a new characteristic. Now, our models should not only estimate the direction and the amount of deformation of each node but also the magnitude and direction of interaction forces that will be reflected to the user via a haptic device. This tutorial note discusses the modeling and programming principles of force-reflecting deformable objects.



Applications:

• Surgical simulators are currently being developed at many research centers and companies to train doctors and residents with new surgical devices and techniques. Conveying to the surgeon the touch and force sensations with the use of haptic interfaces is an important component of a simulator. Force-reflecting deformable models in various fidelities need to be developed to simulate the behavior of soft tissues when they are manipulated with surgical instruments. The developed algorithms should deal directly with geometry of anatomical organs, surface and compliance characteristics of tissues, and the estimation of appropriate reaction forces to convey to the user a feeling of touch and force sensations.

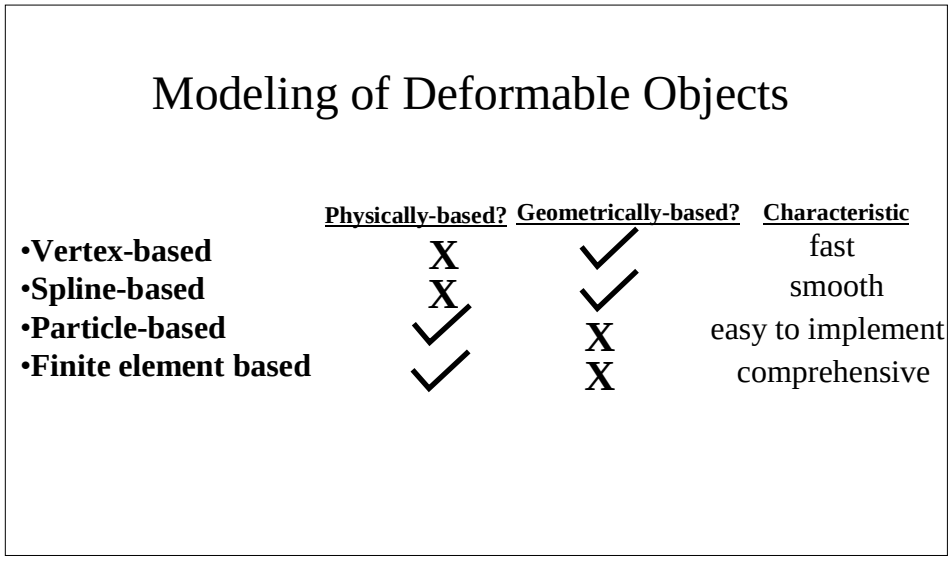
• 3D modeling of deformable objects using NURBS or FFD are well known concepts in CAD. With the addition of force feedback, the interactions will be more intuitive and physically based. For example, various constraints can be implemented naturally using force feedback.

• An animator can intuitively deform the body parts of a 3D character using a haptic device. For example, an animator can use force cues to decide on how much the knee of a 3D character should be flexed at each time frame to make its locomotion more realistic.

• Mechanistic interactions between the melted materials and the manufacturing tools can be studied in virtual environments. For example, an extrusion process can be simulated to better understand the behavior of materials under certain external loads.

Desired properties of force-reflecting deformable models

- reflect stable forces
- display smooth deformations
- handle various boundary conditions and constraints
- display “physically-based” behavior

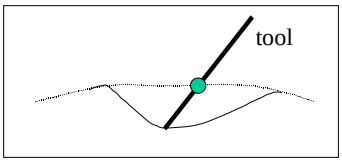


One way to categorize the deformation techniques is according to the approach followed by the researchers to deform the surfaces: *geometric* or *physically-based* deformations. In geometric deformations, the object or the surrounding space is deformed based purely on geometric manipulations. In general, the user manipulates vertices or control points that surround the 3D object to modify the shape of the object. On the other hand, physically-based deformation techniques aim to model the physics involved in the motion and dynamics of interactions. Models simulate physical behavior of objects under the affect of external and internal forces. Geometric-based deformation techniques are faster, and are relatively easier to implement. But they do not simulate the underlying mechanics of deformations. Hence, the emphasis is on visual display and the goal is to make deformations appear smoother to the end-user. Sophisticated physically based models, although necessary for simulating the dynamics of realistic interactions, are not suitable for fully interactive, real-time simulation of multiple objects in virtual environments due to the current limitations in computational power.

Modeling of Deformable Objects

Vertex-based :

$$\text{Depth} = a_0 + a_2 (\text{Radial Distance})^2$$

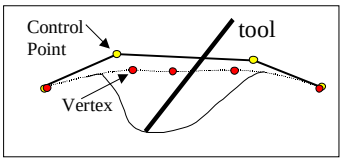


Spline-based :

$$Q(u,v,w) = \sum_i \sum_j \sum_k P_{ijk} B_i(u) B_j(v) B_k(w)$$

$$\Delta P = (B^T B)^{-1} B^T \Delta Q$$

$$Q_{new} = B (P + \Delta P)$$

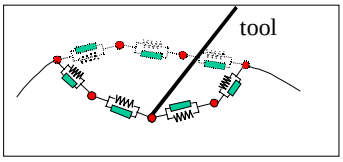


Particle-based :

$$F = ma$$

- F_{spring}
- $F_{damping}$
- $F_{gravity}$

$$a(t + \Delta t) = F/m$$
$$v(t + \Delta t) = v(t) + \Delta t\, a(t + \Delta t)$$
$$p(t + \Delta t) = p(t) + \Delta t\, v(t + \Delta t)$$



Geometric and *physically-based* deformation techniques can be sub-grouped as follows:

A. Geometric-based Deformation Models
   - *Vertex-based:* The vertices of the object are manipulated to display the visual deformations.
   - *Spline-based:* Instead of directly transforming the vertices of the object, control points are assigned to a group of vertices and are manipulated to achieve smoother deformations.

B. Physically-based Deformation Models
   - *Particle-based:* Particle systems consists of a set of point masses, connected to each other through a network of springs and dampers, moving under the influence of internal and external forces. In this model, each particle is represented by its own mass, position, velocity, and acceleration.
   - *Finite Element based:* The volume occupied by the object is divided into finite elements, properties of each element is formulated and the elements are assembled together to study the deformation states for the given loads.

For geomeric-based models, we assume that the user will define his/her own force interaction model. The force model will depend on the deformation model. For example, a set of linear/nonlinear springs can be considered between the home and

deformed positions of nodes to compute the direction and magnitude of the force vector that will be reflected to the user. In physically-based modeling, the model automatically computes the magnitude and direction of forces applied to each node.

- *Vertex-based*: A region of the object surface in the close vicinity of the collision point (or the nearest surface point) can be locally deformed. In order to deform object, we translate all of the vertices within a certain distance (called the *radius of influence*) of the collision point, along the direction of the haptic stylus. For example, the magnitude of translation can be determined using a simple second order polynomial. The degree and the coefficients of the polynomial define the shape of the deformations. For example, if a second degree polynomial with no linear deformation term is assumed ($a_1 = 0$), then the deformation function takes the following form

$$\text{Depth} = a_0 + a_2 (\textit{Radial Distance})^2$$

where, $a_0 = \text{AP}$ and $a_2 = -\text{AP} / (\textit{radius of influence})^2$. The vector AP is constructed from the coordinates of the stylus tip and the contact point. The *radial distance* is the distance of each neighboring vertex, within the radius of influence, to the collision point.

- *Spline-Based:* Sederberg and Parry (1986) suggested a free-form deformation (FFD) technique for deforming the space that encloses the object. FFD enables the user to interactively modify the object shape by repositioning the lattice of control points that surround the 3D object. Any point within the lattice is defined as:

$$Q(u,v,w) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} P_{ijk} B_i(u) B_j(v) B_k(w)$$

or, in matrix form

$$Q = BP \qquad (*)$$

where, P_{ijk} are the control points, and $B_i(u)$, $B_j(v)$, $B_k(w)$ are known as the third degree Bernstein polynomials or Bezier basis functions. Hsu et al. (1992) suggested a method for direct manipulation of free-form surfaces. In this method, control points are moved such that the resulting surface smoothly reaches its intended position by means of a least squares solution. Assume that a single point of the 3D object is translated an amount of ΔQ and moved to a new location ($Q + \Delta Q$), then Eq. (*) can be rewritten in the following form:

$$(Q + \Delta Q)_{1\times 3} = B_{1\times 64} (P + \Delta P)_{64\times 3} \qquad (**)$$

where, ΔQ and ΔP represent the changes in the position of object point and the control points (recall from Eq. (5) that there are 64 control points). Eq. (**) reduces to:

$$\Delta Q_{1\times 3} = B_{1\times 64}\, \Delta P_{64\times 3}$$

Now, the goal is to calculate the change in the control points for a given ΔQ. This can be achieved through the use of pseudoinverse solution:

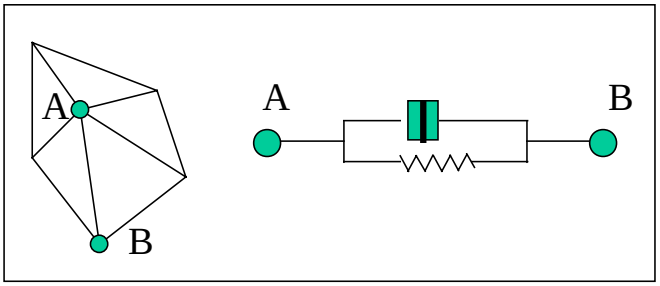
$$\Delta P = (B^T B)^{-1} B^T \Delta Q$$

Once the changes in the positions of control points are known, the deformed positions of the object can be calculated from $Q_{new} = B(P + \Delta P)$.

Suggested References:

1. Basdogan C., Ho, C., Srinivasan, M.A., Small, S., Dawson, S., 1998, "Force interactions in laparoscopic simulations: haptic rendering of soft tissues" *Medicine Meets Virtual Reality (MMVR'6) Conference*, pp. 385-391, San Diego, CA, January 19-22.
2. Dachille, F., Qin, H., Kaufman, A., El-sana J., 1999, "Haptic sculpting of dynamic surfaces", *submitted to the 1999 Symposium on Interactive 3D Graphics.*
3. Hsu W.M. Hughes J.F., Kaufman H., 1992, "Direct Manipulation of Free-Form Deformations", *Computer Graphics (Proceedings of the SIGGRAPH),* Vol. 26, No.2, pp. 177-184.
4. Edwards, J., Luecke, G., 1996, "Physically based models for use in a force feedback virtual environment", Japan/USA Symposium on Flexible Automation, ASME 1996, pp. 221-228.

- *Particle-Based:* Particle systems (also known as mass-spring models) consists of a set of point masses, connected to each other through a network of springs and dampers, moving under the influence of internal and external forces (see figure below).



Each vertex (i.e. node) of the 3D object has a mass and is connected to its neighbors with springs and dampers, moving under the influence of internal and external forces.

The total force applied on each particle can decomposed into *spring, gravitational,* and *dissipative forces.*

$$F_{total} = \left\{ \begin{array}{l} F_{spring} = \sum k(l - l_0) \\ F_{gravitational} = mg \\ F_{dissipative} = -bv_{i,j} \end{array} \right\}_{i^{th} particle} \qquad (***)$$

Then, the acceleration, velocity, and position of each particle can be updated using the Euler integration method.

$$\begin{aligned} a_{t+\Delta t} &= F_{total} / m \\ v_{t+\Delta t} &= v_t + \Delta t \; a_{t+\Delta t} \qquad (****) \\ p_{t+\Delta t} &= p_t + \Delta t \; v_{t+\Delta t} \end{aligned}$$

Particle systems have been extensively used in computer graphics to simulate the behavior of clothes and fluid flow. This technique is simple to implement since the developer does not need to construct the equations of motion explicitly. Moreover, it is physically-based since it can model the viscoelastic behavior of deformable objects.

Suggested References:

1. Cover S.A., Ezquerra N.F., O'Brien J., Rowe R., Gadacz T., Palm E., 1993, "Interactively Deformable Models for Surgery Simulation", *IEEE Computer Graphic and Applications*, November, pp. 65-78.
2. Ng, H., Grimsdale, R., 1996, "Computer Graphics Techniques for Modeling Cloth", *IEEE Computer Graphic and Applications*, September, pp. 28-41.
3. Joukhadar A., Laugier, C., 1995, "Fast Dynamic Simulation of Rigid and Deformable Objects", IEEE/ICRA, pp. 305-310.
4. Witkin A., Barraff D., Kass M., Tutorial notes on " *An Introduction to Physically-Based Modeling*", SIGGRAPH'98.
5. Lee, Y., Terzopoulos, D., Waters, K., 1995, "Realistic Modeling for Facial Animation", *(Proceedings of the SIGGRAPH)*, pp. 55-62.
6. N. Swarup, "Haptic Interaction with Deformable Objects Using Real-Time Dynamic Simulation", M.S. Thesis, Mechanical Engineering Department, Massachusetts Institute of Technology, (1995).

- *FEM-Based:* The finite element models come in various forms and the selected model depends on the type of loading, element, and shape functions. We do not present a model in here due to the limited space, but the following references are quite helpful in developing finite element models for simulating deformable objects.

Suggested References:

1. Rao, S. S., 1988, "The finite element method in engineering", Pergamon Press, NY.
2. Zeinkiewicz, O.C., 1979, "The finite lement method", McGraw-Hill, New Delhi.
3. Bathe, K., 1996, "Finite Element Procedures", Prentice Hall, New Jersey.
4. Bro-Nielsen, M., Cotin, S., "Real-time Volumetric Deformable Models for Surgery Simulation using Finite Lements and Condensation", EUROGRAPHICS'96, Vol. 15, No. 3, pp. 57-66.

Constraints

Examples:
• a node is fixed in 3D space
• a node is constrained to stay on a path
• curvature constraint
• constant volume

Implementation:
1) Particle-based models
a) Penalty
b) Lagrange multipliers
2) FEM

Constraints:
So far, we have discussed various modeling techniques for simulating force-reflecting deformable objects. To control the deformations and to make the simulations more realistic, constraints have to be implemented into models. Several types of contraints can be mentioned:

- a node is fixed in 3D space
- a group of nodes has to follow a path
- curvature constraints has to be specified for modifying free form surfaces
- volume of the object has to be kept constant

1) *Implementation of constraints to particle-based models:*

Many techniques have been suggested to implement constraints. We briefly discuss only two of them in here: (a) penalty methods and (b) Lagrange multipliers (Interested readers may find the details of these techniques in the suggested references). In general, the constrained force estimated through (a) penalty or (b) Lagrange multiplier technique is added to the unconstrained force computed through equation (***). Then, the total force ($F_{total} = F_{constrained} + F_{unconstrained}$) is inserted into Eq. (****) to update the accelaration, velocity and position of each particle.

a) Penalty methods

Calculate the constrained force using the following formulation:

$$F^{constrained}{}_i = (-k_s G - k_d \dot{G}) J$$

where, G(u) is your constraint function that has to be satisfied, u reperesents your nodal displacements, k_s and k_d are spring and damping coefficients that can be adjusted to satisfy constraints, and J is jacobian ($J = \partial G / \partial u_i$). For example, if we want to fix a node (i^{th} node) in 3D space, we define G(u) as G(u) = u_i . Then, $F^{constrained}{}_i = (-k_s u_i - k_d \dot{u}_i)$.

b) Lagrange multipliers

First, solve the following equation for λ and then insert the solution into $F_{constrained} = j^T \lambda$ to estimate the constrained force.

$$JM^{-1}J^T \lambda = -\dot{J}\dot{u} - JM^{-1}F_{unconstrained} - k_s G - k_d \dot{G}$$

where, M is the diagonal mass matrix, λ is a vector of Lagrange multipliers.

Suggested Readings:

1. Witkin, A., 1997, "An Introduction to Physically Based Modeling: Constrained Dynamics, SIGGRAPH'97 Tutorial Notes.
2. Platt J., 1992, "A Generalization of Dynamic Constraints", Graphical Models and Image Processing", Vol. 54, No. 6, pp. 516-525.
3. Promayon, E., Baconnier, P., Puech, C., 1996, "Physically-Based Deformations Constrained in Displacements and Volume", EUROGRAPHICS'96, vol. 15, No. 3, pp. 155-164.

2) *Implementation of constraints to FEM:*

Here, we include a pseudocode that describes how to implement simple boundary conditions to your FEM. Interested readers may find the details of these techniques in suggested references.

F = K.U (original equation)

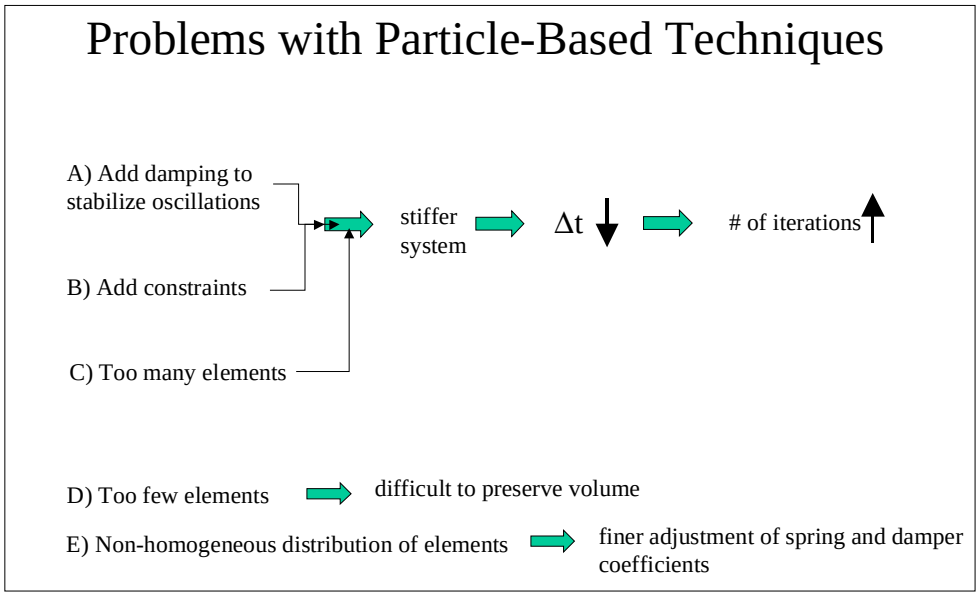
FF = KK .U (create a copy of the original system)

```
for j = 1: nn_constrained                    % loop through constraints
        id = bcdof(j);                       % get the degree of freedom for constraint
        val = bcval(j);                      % get the constrained value
        for i = 1: nn                        % loop through equations of system
                FF(i) = FF(i) – val*KK(i,id);
                KK(id,i) = 0;
                KK(i,id) = 0;
        end
        KK(id,id) = 1;
        FF(id) = val;
end
```

Once you obtained the modified matrices (KK and FF), solve the modified system for the unknown nodal displacements ($U = KK^{-1}FF$). Then, insert the computed nodal displacements into the original equation (F = KU) to find the applied forces at the nodes. Note that the computations can be simplified if the interactions are point-based only.

Suggested Readings:

1. Huebner, K, Thornton, E., Byrom, T., 1995, "The finite element method for engineers", John Wiley & Sons, Inc.
2. Kwon, Y.W., Bang, H., 1997, "The finite element method using Matlab", CRC Press.



Some of the problems associated with particle-based systems are listed below

- A damping term needs to be considered to bring the system into a global equilibrium. Increasing damping makes the system *stiffer*. This is going to force us to take shorter time steps to achieve stability.
- Adding multiple constraints leads to a stiffer system. We may need to reduce the "elasticity" of the system to control the deformations. This is, again, going to force us to take shorter time steps to achieve stability.
- Uneven distribution of vertices (nodes) of the 3D model may easily generate unstable interaction forces and non-smooth graphical deformations.
- Note that explicit integration schemes are conditionally stable (see the work done by Barraf and Witkin on implicit techniques, "Large steps in Cloth animation", SIGGRAPH'98)

The following solutions can be proposed for these problems:

- Taking variable time step to improve the stability
- Considering local deformations to reduce the stability problems
- Remeshing or automatic refinement to reduce the stability problems and to make the deformations appear smoother
- Controlling deformation and/or force update rate ($\beta = \frac{\Delta l}{l_0} or \frac{\Delta F}{F_0}$). For example, if ($\beta < \beta_{critical}$) then set $\beta = \beta_{critical}$

Problems with FEM Techniques

A)Change in topology ⇨ Re-meshing

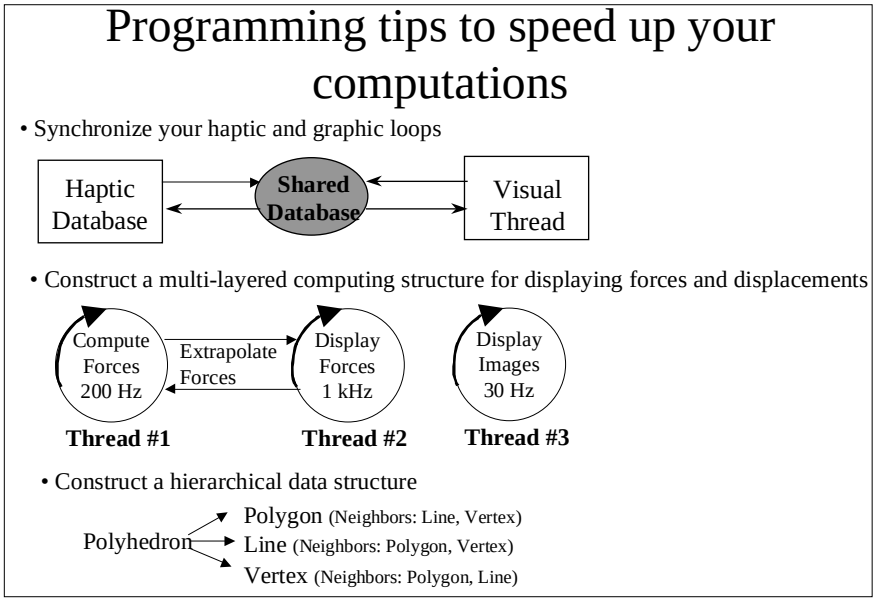
B) Computationally very expensive to perform dynamic analysis

C) Matrix singularities

D) Memory allocation

$$F = KU \qquad U = K^{-1}F$$

In general, finite element models are comphrensive and well suited for accurate computation of deformations. However, it is difficult achive a real-time performance using FEM. Moreover, the addition of haptic feedback increases the complexity of the problem. To achive real-time rendering rates, K^{-1} can be pre-computed and static condensation (i.e. eliminating unwanted degrees of freedom) technique can be implemented. However, the precomputation of K^{-1} is a problem if the topology of object permenantly changes during the interaction. For example, if an object is sliced or cut, it has to be remeshed and the stiffness matrix has to be updated. Moreover, taking the inverse of the K matrix is not trouble free and singularities may occur. Finally, the entries of the matrices need to be allocated wisely to save from the memory.



- *Synchronize your haptic and graphic loops:* Software integration of visual and haptic modalities was achieved in an efficient manner by creating a hierarchical database for geometrical properties of objects and by programming with multi-threading techniques. In our simulations, visual and haptic servo loops were separated to achieve faster rendering rates. When displaying visual images, it is known that the update rate should be around 30 Hz to appear continuous. On the other hand, to create a satisfying haptic display, the update rates for sending the force commands to the haptic interface needs to be about 1000 Hz. In order to create a VE that satisfies both requirements and optimally use the CPU power of a computer, the visual and the haptic servo loops need to be separated. That is, we run two loops at the same time, with the graphic loop updated at 30 Hz and the haptic loop updated at 1000 Hz. Since there are two loops running at the same time, there is always a chance a conflict occurs in accessing the shared memory. For example, in the case of simulating deformable objects, changes in geometry require frequent updates of visual and haptic databases in real-time. This will cause a problem if one loop is writing data to the memory and the other loop is reading from there. In order to avoid this situation, we need to synchronize the two loops. The easiest way to synchronize two loops is to create a Boolean flag. When one loop wants to access the shared data, it should check the flag first to see if the data is being accessed by the other loop. If the flag indicates that the shared memory is not being used, the loop can access the data and the flag is set to indicate that the shared memory is currently being used. If the flag indicates that the data is being used by the other loop, the loop waits until the

other loop is done. When one loop finishes its operations with the shared memory, it sets the flag back to normal to let other loop access the data.

• *Construct a multi-layered computational architecture:* Although separating haptic and graphic loops, using a client-server model as described in the previous paragraph, is helpful in improving update rates, it may not be sufficient in certain situations. In a typical client-server model for haptic rendering of 3D objects, haptic thread is usually designated as the client and the model computations are performed in this thread. However, physically-based modeling techniques for displaying forces and deformations are computationally expensive and the haptic update rate may drop below the requirement. For example, a real-time dynamic analysis of force-reflecting deformable objects using finite-element techniques is quite difficult with the available computational power. To overcome this difficulty, we suggest a layer between the "computation" and the "display" modules. In this layer, forces can be extrapolated based on the previous force values and their rate of change. Based on this approach, forces can be computed at 200 Hz using a finite element technique, extrapolated in between the computation cycles, and displayed to the user at 1 kHz.

•*Construct a data structure for primitive hierarchy:* We use polygonal models in our simulations. We separate each polyhedron into three types of primitives: vertices, lines (i.e. edges), and polygons. In our database, each primitive has a pointer to its neighboring primitives. The primitive hierarchy helps us to quickly access the neighbors of the primitive when it is necessary. For example, when a simulated tool contacts a primitive of an object in the current loop to modify its coordinates, we know that, in the next loop, the model can only affect the primitives that are in the close neighborhood of the contacted primitive. Neighborhood hirerchy is helpful in simulating force-reflecting deformable objects. For example, forces due to inertial effects can be transferred to all nodes by propagating radially through the neighboring primitives from the contact point.

Modeling tips to speed up your computations

You may consider

- deforming your objects locally
- taking advantage of single point interactions
- condensing your matrices in FEM
- pre-computation of matrices, unit displacements, etc.
- transforming your coordinates to modal coordinates
- decoupling your force and deformation model

Number of computations is significantly important in simulating force-reflecting deformable objects in virtual environments. Most of the time, the developer needs to reduce the # of computations or to make simplifications in the model in order to achieve real-time rendering rates. Here, we suggest a few tips in this regard:

1) *deforming your objects locally*

r = |vertex[i].coord - Collision Point |;
if (r < $R_{deformation}$)
vertex[i].frozen = yes;

2) *taking advantage of single point interactions*

a) "if the force is applied to a single node" in FEM

$$U = K_i^{-1} F_i$$

where, "i" is the i-th column of K^{-1} matrix and i-th entry of force vector.

b) "if the force is applied to a single node" in FFD model (refer to spline-based modeling section)

For single point manipulation, the solution reduces to the following simple form

$$\Delta P = \frac{B^T}{\sum_i (b_i)^2} \Delta Q$$

where, b_i's are the elements of the B matrix.

3) *condensing your matrices in FEM*

a) e.g.: construct your matrix using volume elements, but solve the equations for surface elements

b) e.g.: eliminate unwanted degrees of freedom such as rotational dof.

$$\begin{bmatrix} K_{MM} & K_{MS} \\ K_{SM} & K_{SS} \end{bmatrix} \begin{bmatrix} U_M \\ U_S \end{bmatrix} = \begin{bmatrix} F_M \\ F_S \end{bmatrix}$$

$$\begin{aligned} \left[\overline{K}\right]\left[U_M\right] &= \left[\overline{F}_M\right] \\ \left[\overline{K}\right] &= \{K_{MM} - K_{MS} K_{MM}^{-1} K_{SM}\} \\ \left[\overline{F}_M\right] &= \{F_M - K_{MS} K_{SS}^{-1} F_S\} \end{aligned}$$

where, the subscript M denotes the masters and the subscript S denotes the slaves which are to be eliminated.

4) *pre-computation in FEM*

a) compute K^{-1} in advance
b) compute displacements for a given unit force at each node and then apply superposition

5) *transforming your coordinates to modal coordinates (dynamic analysis)*

If you would like to simulate the inertial properties of force-reflecting deformable objects, it may be worthwhile to consider a modal analysis. In modal analysis, you transfer your coordinates to modal coordinates to decouple your differential equations. This will enable you to obtain the explicit form of the governing equation for each node. Moreover, you can also reduce the dimension of the system by picking the most significant modes and re-arranging your mass, damping, and stiffness matrices (i.e. modal reduction).

a) transforming to modal coordinates (coupled diff. Eqs. -> uncoupled diff. Eqs.)

$$M\ddot{U} + B\dot{U} + KU = F$$
$$U(t) = \Phi X(t)$$
$$\Phi^T M\Phi\ddot{X} + \Phi^T B\Phi\dot{X} + \Phi^T K\Phi X = \Phi^T F$$
$$\ddot{X} + \Phi^T B\Phi\dot{X} + \Omega^2 X = \Phi^T F$$

where,

$\Phi = [\varphi_1, \varphi_2, \varphi_3, ..., \varphi_n]$ $\quad \Omega^2 = diag(\omega_i^2)$

Φ is a modal matrix whose columns are eigenvectors of ($M^{-1}K$) and Ω^2 is a diagonal matrix which stores the eigenvalues on its diagonals. Observe that $\Phi^T B\Phi$ is not a diagonal matrix (i.e. we still have a system of uncoupled differential equations). Assume that damping matrix is propotional to mass and stiffness matrices ($B = \alpha M + \beta K$, where α and β are constants. Then the equations take the following form:

$$\ddot{X} + \Delta\dot{X} + \Omega^2 X = \Phi^T F \qquad \text{(a set of uncoupled diff. eqs.)}$$

where, $\Delta = \text{diag}(2\omega_i\zeta_i)$ and ζ is modal damping factor.

b) modal reduction (eliminate high frequency modes)

This technique involves the selection of dominant modes and elimination of high frequency modes. To achieve this, the eigenvalues of the system are listed in incerasing order, and then the columns of the ϕ matrix are rearranged according to this order to construct a reduced order system. Note that the first six nodes of the eigen-matrix represent the rigid body modes.

$$\Phi = [\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, ..., \varphi_n]$$

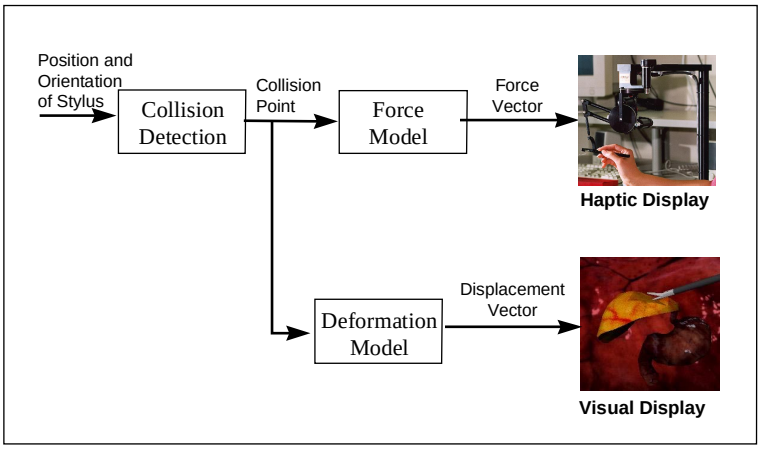
Suggested Readings:

1. Bathe, K., 1996, "Finite Element Procedures", Prentice Hall, New Jersey.
2. Shabana, A., 1996, "Theory of Vibration", Springer-Verlag.
3. Pentland, A., Williams, J., 1989, "Good Vibrations: Modal Dyanmics for Graphics and Animation", SIGGRAPH Proceedings, Vol. 23, No. 3, pp.215-222.

6) *loose coupling of force and displacement models:*

One can loosely couple the deformation model with the force model to simulate the nonlinear material characteristics of deformable objects. To implement this idea, the information returned by the collision-detection module (collision point and depth of penetration) can be independently used by "deformation" and "force" models. This technique, for example, can be used to simulate the "nonlinear force" characteristics of soft tissues. Since the developed tissue models for simulation purposes are usually linear, nonlinear force-displacement characteristics of organs would not be simulated using these models. However, a nonlinear force profile constructed using the experimental

measurements can be used to reflect nonlinear forces to the user while a smooth deformation profile is displayed graphically using the FFD technique.



The concept of loosely coupling force and displacement models. To implement this idea, we independently use the collision point and depth of penetration in deformation and force models following the detection of collision.

Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

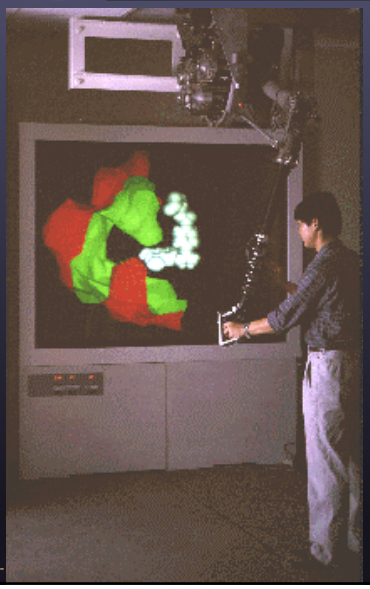
Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

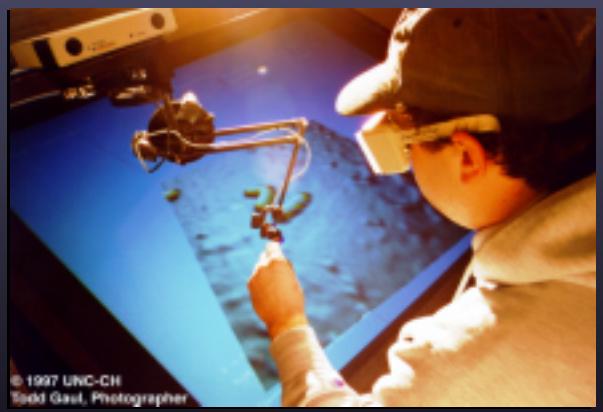
University of North Carolina at Chapel Hill

Department of Computer Science

Russell M. Taylor II

Haptics: From Basic Principles to Advanced Applications





Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

Overview

- Discussion of three applications
  - –Teaching Force Fields
  - –Drug Design (simulation)
  - –Microscope control (nanoManipulator)
- Lessons learned from each
- Haptic requirements of each

The Point

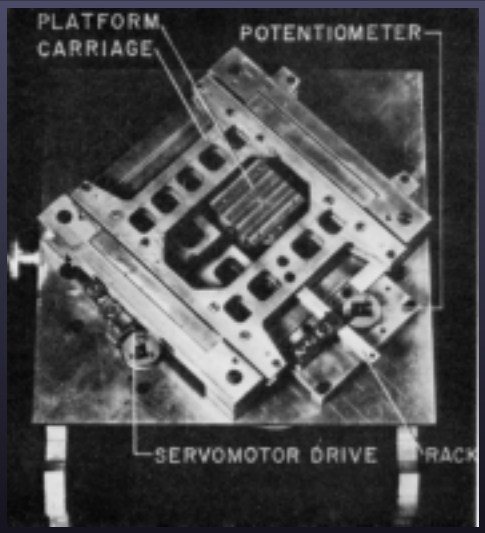
- Answer “What is this good for?” with specifics
  - –Shows what benefit may be gained by adding each technique
  - –Shows which techniques may be useful for a given application
- List particular gains using each technique.

Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

Teaching Physics Force Fields

- Students felt changing forces on test charge, as they moved test charge.
- Interested students gained the most
- Dispelled incorrect preconceptions

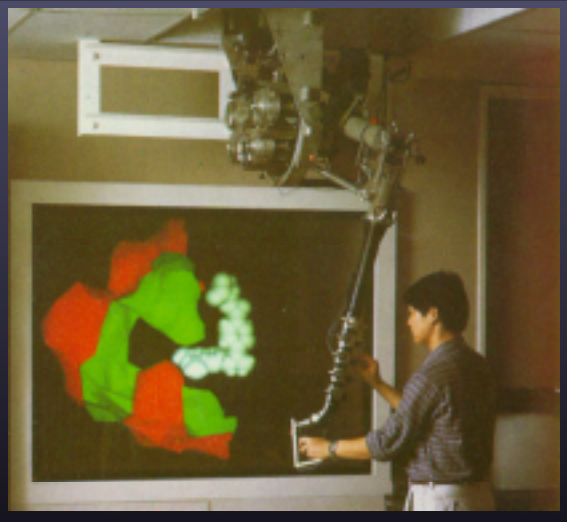
Picture of device



Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

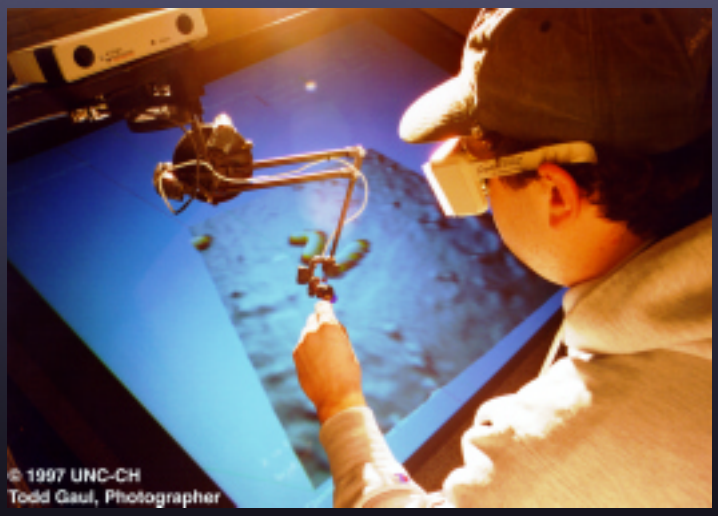
Drug/Protein Docking

- A “Lock and Key” Problem
- Natural use for force-feedback
- Up to a factor of 2 improvement in 6DOF positioning task
- Chemists said they understood more
- 6DOF in and out required



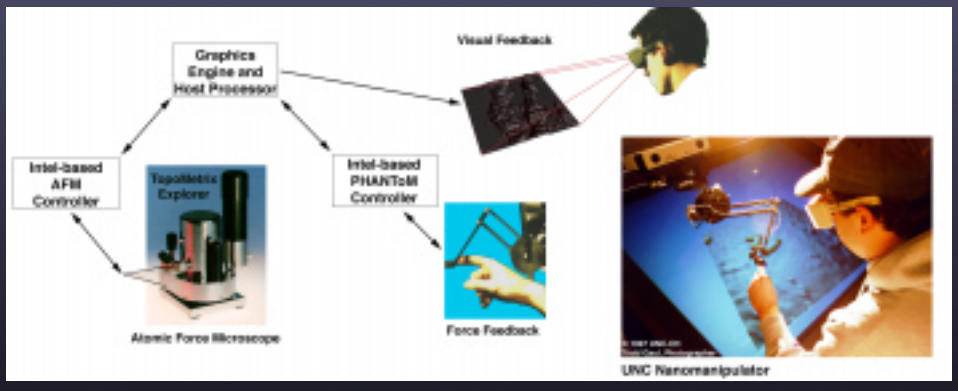
Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

nanoManipulator



RMT - 10/97 - Slide L 9

nanoManipulator System Diagram



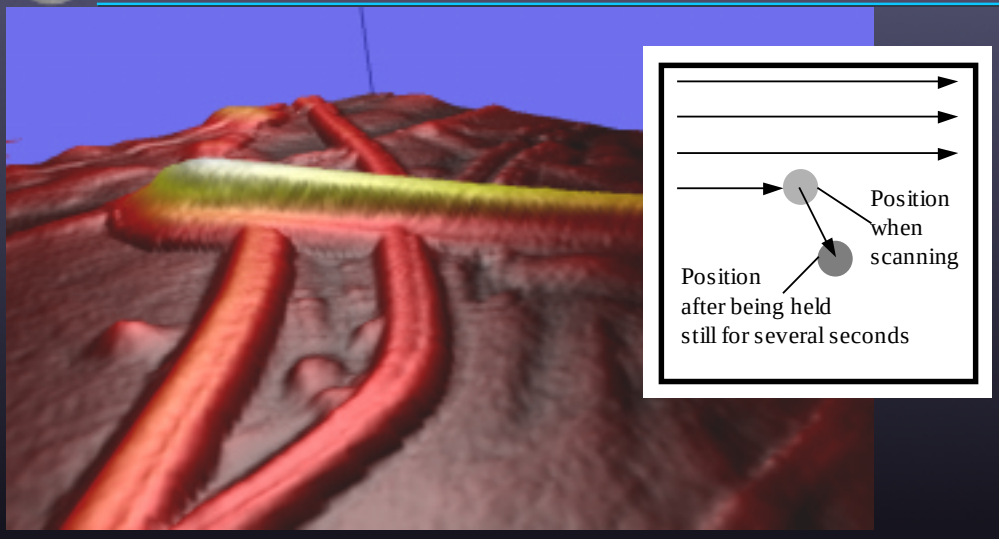
RMT - 10/97 - Slide L 10

Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

nanoManipulator FF gain

- Touch lets you get to the right spot
- The map is not the territory
- Result: Peeling apart two carbon nanotubes underneath a third

Finding the right spot



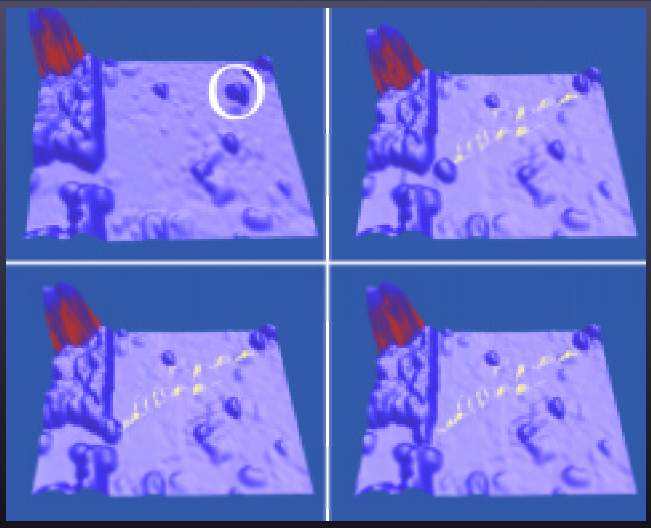
Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

nanoManipulator FF gain

- You can't see what you're doing
- You *can* feel what you're doing

- Results:
  - nanoWorld Cup
  - thin gold ring

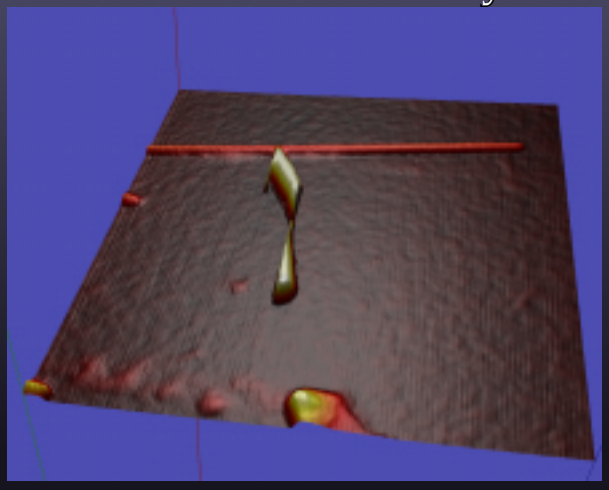
Finding the right path



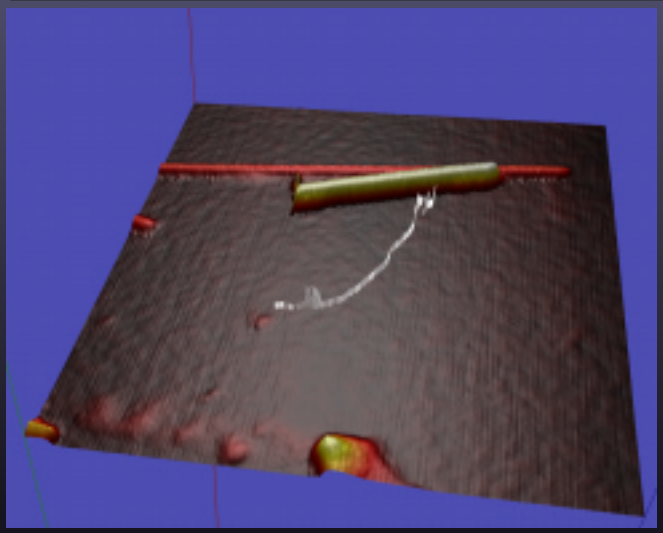
Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

nanoManipulator FF gain

Observation modifies the system



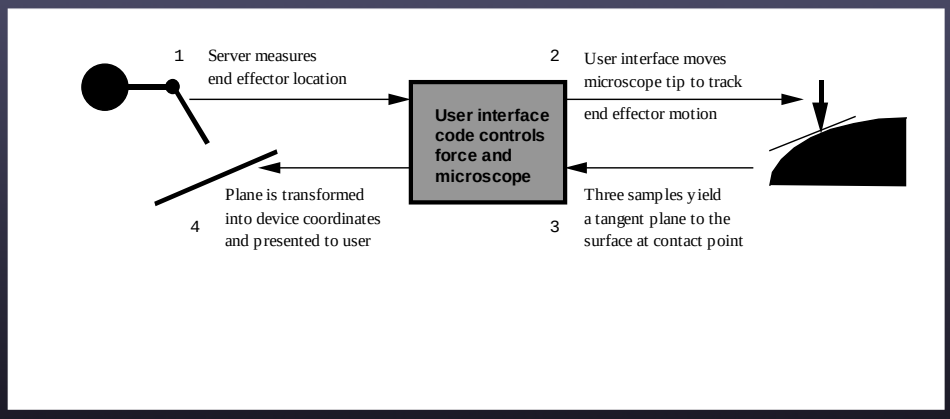
A light touch



Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

nanoManipulator requirements

- Controlling a single tip: 3DOF out
- 6DOF in used for positioning, advanced control
- Network link to main application
- Force loop and application loop operate at their own rates



Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

Summary

- Dispelled misunderstandings in force fields
- Improved task performance in 6DOF positioning task
- nanoManipulator
  - Finding the right point
  - Control during modification
  - "Haptic imaging"

Summary

- Match application requirements
  - 2DOF in/out for force fields
  - 6DOF in/out for Docker
  - 6DOF in, 3DOF out for nM
  - Separate force, graphics and microscope loops

Scientific Applications of Force Feedback: Molecular Simulation and Microscope Control

Further Information

- http://www.cs.unc.edu/Research/force/
- http://www.cs.unc.edu/Research/graphics/GRIP/
- http://www.cs.unc.edu/Research/nano/

Acknowledgements

Support: National Institutes of Health National Center for Research Resources, National Science Foundation.

Past and present FF students: William Mark, Scott Randolph, Mark Finch, Chris DiMattia, Brian Grant, Jun Chen, Renee Maheshwari and Adam Seeger

Collaborators: Frederick P. Brooks, Jr., William V. Wright, Mary Whitton, Vernon L. Chi, Sean Washburn and Richard Superfine

Haptics: From Basic Principles to Advanced Applications

Thomas Massie

SensAble Technologies, Inc.

215 First Street

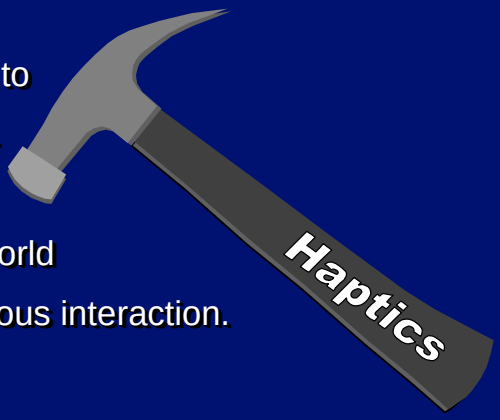
Cambridge, MA 02142

SIGGRAPH Los Angeles

Haptics: Where is it useful?

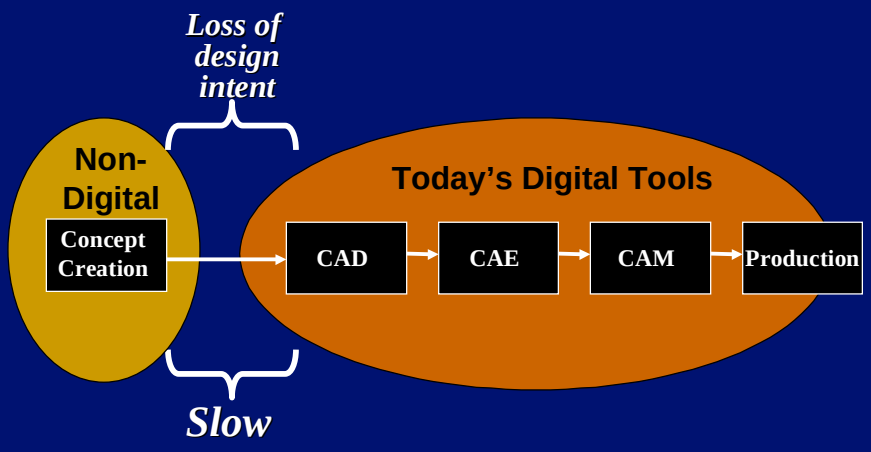
1. Massive amounts of 3D data to interpret, navigate, and change.

2. Analog task in the physical world requires or benefits from dexterous interaction.



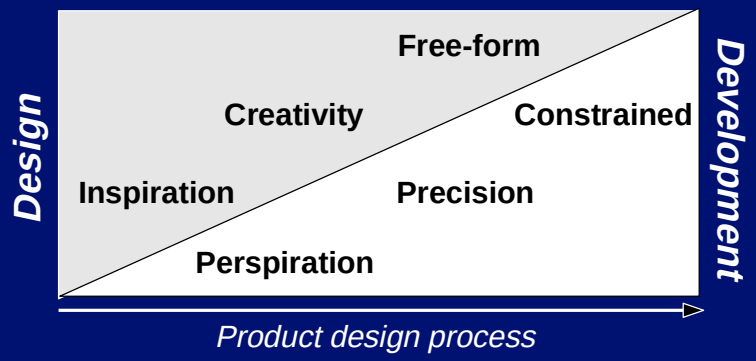
SIGGRAPH Los Angeles

Opportunity to Improve Today's 3D Modeling Process



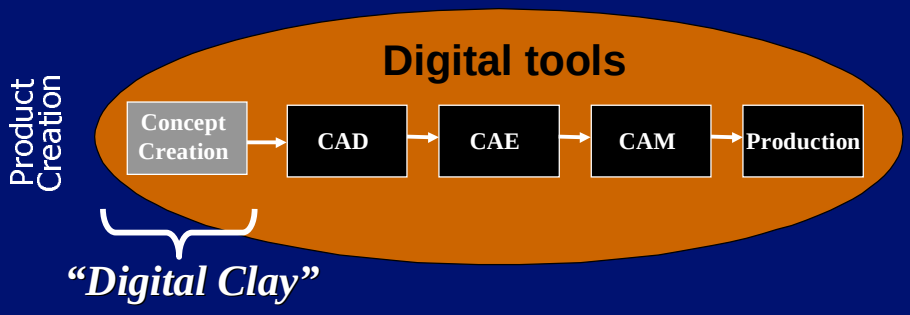
SIGGRAPH Los Angeles

Why Does Conceptual Modeling Still Use Physical Media?



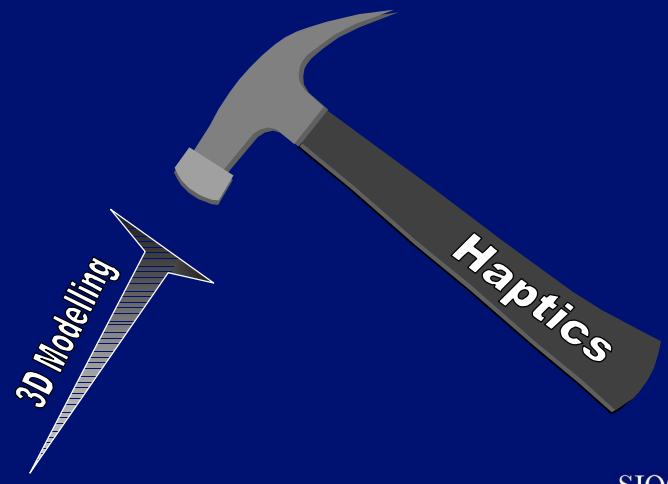
SIGGRAPH Los Angeles

A Commercial Application for Haptics: Digital Sculpting!



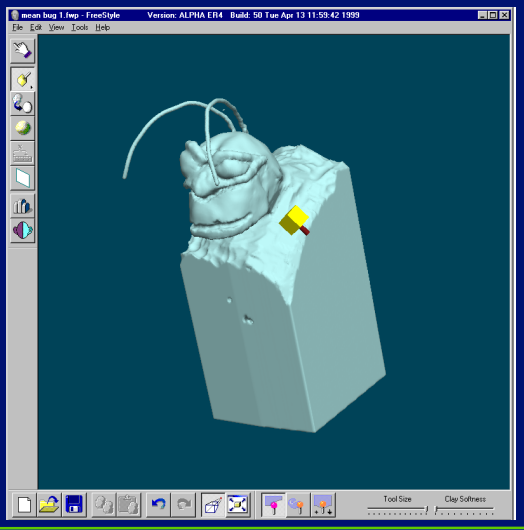
SIGGRAPH Los Angeles

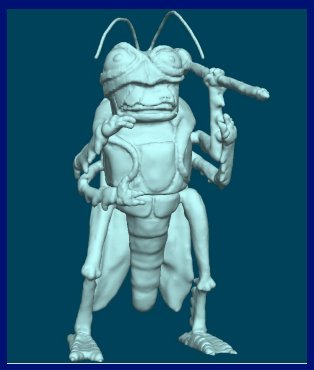
This Hammer Has Found A Nail!



SIGGRAPH Los Angeles

A Commercial Application for Haptics: Digital Sculpting!





SIGGRAPH Los Angeles

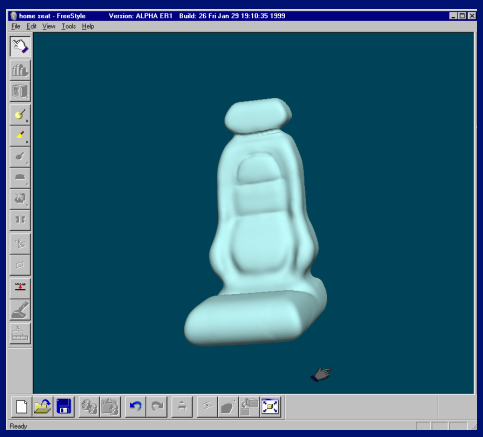
Why is the Sense of Touch Crucial for Sculpting?

5 Reasons:

- Navigating in 3D - the “Z dimension”
- Resolving visual ambiguities
- Continuous, modulated, pressure sensitive control
- The model guides model creation
- Sculpting is spontaneous, not programmed or algorithmic

SIGGRAPH Los Angeles

Digital Clay - An Open Invitation to Sculpt



SIGGRAPH
Los Angeles