

# **RECENT ADVANCES IN HAPTIC RENDERING AND APPLICATIONS**

Edited by Ming C. Lin and Miguel A. Otaduy

Copyright © 2005



# ORGANIZERS

Ming C. Lin  
Department of Computer Science  
University of North Carolina  
Chapel Hill, NC 27599-3175  
lin@cs.unc.edu  
PHO: 919-962-1974  
FAX: 919-962-1799

Miguel Otaduy  
Department of Computer Science  
Institute of Scientific Computing  
ETH Zentrum  
CH - 8092 Zürich  
otaduy@inf.ethz.ch

# LECTURERS

- **Elaine Cohen, University of Utah**  
[cohen@cs.utah.edu](mailto:cohen@cs.utah.edu)
- **David Johnson, University of Utah**  
[dejohnso@cs.utah.edu](mailto:dejohnso@cs.utah.edu)
- **Roberta Klatzky, Carnegie Mellon University**  
[klatzky@cmu.edu](mailto:klatzky@cmu.edu)
- **Ming Lin, University of North Carolina at Chapel Hill**  
[lin@cs.unc.edu](mailto:lin@cs.unc.edu)
- **Bill McNeely, Boeing Research**  
[bill.mcneely@boeing.com](mailto:bill.mcneely@boeing.com)
- **Miguel Otaduy, ETH Zurich**  
[otaduy@inf.ethz.ch](mailto:otaduy@inf.ethz.ch)
- **Dinesh Pai, Rutgers University**  
[dpai@cs.rutgers.edu](mailto:dpai@cs.rutgers.edu)
- **Ken Salisbury, Stanford University & Intuitive Surgery, Inc**  
[jks@robotics.stanford.edu](mailto:jks@robotics.stanford.edu)
- **Hong Tan, Purdue University**  
[hongtan@purdue.edu](mailto:hongtan@purdue.edu)
- **Russell Taylor, University of North Carolina at Chapel Hill**  
[taylorr@cs.unc.edu](mailto:taylorr@cs.unc.edu)



# BIOGRAPHICAL SKETCHES OF LECTURERS

**Elaine Cohen** is a professor of computer science at the University of Utah. She is co-head of the Geometric Design and Computation Project and co-author of *Geometric Modeling with Splines: An Introduction* (A. K. Peters, 2001). Dr. Cohen has focused her research in computer graphics, geometric modeling, and manufacturing, with emphasis on complex sculptured models represented using NURBS (Non-Uniform Rational B-splines) and NURBS-features. Results in manufacturing research have been focused on automating process planning, automatic toolpath generation for models having many surfaces, optimizing both within and across manufacturing stages and fixture automation. She has also been working on issues related to telepresence and design collaborations in virtual environments. Recent research has produced algorithms for determining both visibility and accessibility of one object by another. Computation of such information is necessary for manufacturing, assembly planning, graphics, and virtual environments. Research in haptics has been focused on developing new approaches to solving geometric computations such as fast and accurate contact and tracking algorithms for sculptured models and while research in haptics systems has focused on realistic force feedback in distributed haptic systems for complex mechanical models. Dr. Cohen was the 2001 recipient of the University of Utah Distinguished Research Award and is a member of the Computer Science and Telecommunications Board of the National Academies.

Website: <http://www.cs.utah.edu/~cohen/>

**David Johnson** is a research scientist at the University of Utah, School of Computing, where he also received his PhD. His primary research interest is in distance algorithms for haptic rendering, motivated by the needs of virtual prototyping applications. He has given invited talks at Ford Motor Company, Intel Research, and Institute for Mathematics and its Applications on these topics. In addition, he was a speaker at a Solid Modeling 2001 tutorial on haptic rendering.

Website: <http://www.cs.utah.edu/~dejohnso/>

**Roberta Klatzky** is a Professor of Psychology at Carnegie Mellon University, where she also holds faculty appointments in the Center for the Neural Basis of Cognition and the Human-Computer Interaction Institute. She served as Head of Psychology from 1993 - 2003. She received a B.S. in mathematics from the University of Michigan and a Ph.D. in experimental psychology from Stanford University. Klatzky's research interests are in human perception and cognition, with special emphasis on haptic perception and spatial cognition. She has done extensive research on haptic and visual object recognition, human navigation

under visual and nonvisual guidance, and motor planning. Her work has application to haptic interfaces, navigation aids for the blind, exploratory robotics, teleoperation, and virtual environments. She also has interests in medical decision-making. Professor Klatzky is the author of over 150 articles and chapters, and she has authored or edited 4 books. Klatzky has been a member of the National Research Council's Committee on Human Factors and Committee on Techniques for Enhancing Human Performance, as well as other working groups of the NRC. Her service to scientific societies includes chairing the governing board of the Psychonomics Society and the Psychology Section of AAAS, as well as serving as Treasurer of the American Psychological Society and being on the advisory board of the International Association for the Study of Attention and Performance. She is a member of several editorial boards and a past associate editor of the journal *Memory and Cognition*.

Website: <http://www.psy.cmu.edu/faculty/klatzky/>

**Ming Lin** received her B.S., M.S., Ph.D. degrees in Electrical Engineering and Computer Science all from the University of California, Berkeley. She is currently a full professor in the Computer Science Department at the University of North Carolina (UNC), Chapel Hill. She received the NSF Young Faculty Career Award in 1995, Honda Research Initiation Award in 1997, UNC/IBM Junior Faculty Development Award in 1999, and UNC Hettleman Award for Scholarly Achievements in 2002. Her research interests include haptics, physically-based modeling, robotics, and geometric computing. She has served as a program committee member for many leading conferences on virtual reality, computer graphics, robotics and computational geometry. She was the general chair and/or the program chair of the First ACM Workshop on Applied Computational Geometry, 1999 ACM Symposium on Solid Modeling and Applications, the Workshop on Intelligent Human Augmentation and Virtual Environments 2002, ACM SIGGRAPH/EG Symposium on Computer Animation 2003, and ACM Workshop on General-Purpose Computing on GPUs 2004, Computer Animation and Social Agents 2005, and Eurographics 2005. She also serves on the Steering Committee of ACM SIGGRAPH/EG Symposium on Computer Animation. She is an associated editor and a guest editor of several journals and magazines, including *IEEE Transactions on Visualization and Computer Graphics*, *International Journal on Computational Geometry and Applications*, *IEEE Computer Graphics and Applications*, and *ACM Computing Reviews*. She also edited the book "*Applied Computation Geometry*." She has given several lectures and invited presentations at SIGGRAPH, GDC, and many other international conferences.

Website: <http://www.cs.unc.edu/~lin/> and <http://gamma.cs.unc.edu/>

**Bill McNeely** obtained a Ph.D. in physics from Caltech in 1971 and for the next 6 years pursued physics research in Hamburg, Germany. In 1977 he began working for Boeing Computer Services in Seattle, Washington, developing advanced computer graphics for engineering applications. From 1981 to 1988 he served as chief engineer at startup company TriVector Inc., creating software products for technical illustration. In 1988 he founded the software consulting company McNeely & Associates. In 1989 he returned to Boeing, where he now holds the position of Technical Fellow. At Boeing he has pursued research in high-performance computer graphics, haptic simulation, and information assurance.

Website: [\*http://www.boeing.com/phantom/\*](http://www.boeing.com/phantom/)

**Miguel Otaduy** received his PhD in Computer Science in 2004 from the University of North Carolina, Chapel Hill, supported by fellowships from the Government of the Basque Country and UNC Computer Science Alumni. He received a bachelor's degree in electrical engineering from University of Mondragon (Spain) in 2000. Between 1995 and 2000, he was a research assistant at the research lab Ikerlan. In the summer of 2003 he worked at Immersion Medical. His research areas include haptic rendering, physically-based simulation, collision detection, and geometric modeling. He and Lin introduced the novel concept of sensation preserving simplification for haptic rendering in their SIGGRAPH 2003 paper and proposed the first 6-DOF haptic texture rendering algorithm. He is currently a research associate at ETH Zurich.

Website: [\*http://graphics.ethz.ch/~otmiguel/\*](http://graphics.ethz.ch/~otmiguel/)

**Dinesh Pai** is a Professor in the Department of Computer Science at Rutgers University. Previously, he was a Professor at the University of British Columbia and a fellow of the BC Advanced Systems Institute. He received his Ph.D. from Cornell University. His research spans the areas of graphics, robotics, and human-computer interaction. His current work is in Human Simulation and Multisensory Computation; the latter includes multisensory simulation (integrating graphics, haptics, and auditory displays) and multisensory modeling (based on measurement of shape, motion, reflectance, sounds, and contact forces). He is the author of over 85 refereed publications, including 25 journal papers and 4 SIGGRAPH papers. He has served on 6 editorial boards and on the program committees of all major conferences in Robotics and Graphics, and is a program chair for the 2004 Symposium on Computer Animation. He organized the SIGGRAPH2001 Panel "Newton's Nightmare: Reality meets Faux Physics," and co-taught the SIGGRAPH 2003 course on "Physics-Based Sound Synthesis for Graphics and Interactive Systems."

Website: [\*http://www.cs.rutgers.edu/~dpai/\*](http://www.cs.rutgers.edu/~dpai/)

**Kenneth Salisbury** is well known for his seminal contributions to the fields of robotics, haptics, and robotically assisted surgery, and for his successes in facilitating transfer of these technologies. Professor Salisbury received his Ph.D. in Mechanical Engineering from Stanford University in 1982. From 1982 to 1999 he served at MIT as Principal Research Scientist in Mechanical Engineering and as a member of the Artificial Intelligence Laboratory. Among the projects with which he has been associated are the Stanford-JPL Robot Hand, the JPL Force Reflecting Hand Controller, the MIT-WAM arm, and the Black Falcon Surgical Robot. His work with haptic interface technology led to the founding of SensAble Technologies Inc., producers of the PHANTOM haptic interface and FreeForm software. In 1997 he became the Scientific Adviser to Intuitive Surgical in Mountain View, where his efforts focused on the development of dexterity-enhancing telerobotic systems for surgeons. In the Fall of 1999, he joined the faculty at Stanford in the departments of Computer Science and Surgery, where his research now focuses on human-centered robotics, collaborative computer-mediated haptics, and surgical simulation. He currently serves on the National Science Foundation's Advisory Council for Robotics and Human Augmentation, as Scientific Adviser to Intuitive Surgical, Inc., and as Technical Adviser to Robotic Ventures, Inc.

Website: [\*http://www.robotics.stanford.edu/~jks/\*](http://www.robotics.stanford.edu/~jks/)

**Hong Tan** is founder and director of the Haptic Interface Research Laboratory at Purdue University. In 1986, she received a Bachelor's degree in Biomedical Engineering (with a minor in Computer Science) from Shanghai Jiao Tong University. She earned her Master and Doctorate degrees, both in Electrical Engineering and Computer Science, from the Massachusetts Institute of Technology (MIT) in 1988 and 1996, respectively. From 1991 to 1993, she was a Research Associate at the Tufts University School of Medicine. From 1996 to 1998, she was a Research Scientist at the MIT Media Laboratory. In 1998, she joined the faculty at Purdue's School of Electrical and Computer Engineering as an Assistant Professor, and in 2003 received tenure and promotion to an Associate Professor. Since 2002, she has held a courtesy appointment at Purdue's School of Mechanical Engineering for her contribution to the Perception-Based Engineering program. Her broad research interests in the area of haptic interfaces include psychophysics, haptic rendering, and distributed contact sensing. She is a recipient of National Science Foundation's Early Faculty Development (CAREER) Award (2000-2004). She has served on numerous conference program committees, and is currently co-organizer (with Blake Hannaford) of International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (2003-2005).

Website: [\*http://www.ece.purdue.edu/~hongtan/\*](http://www.ece.purdue.edu/~hongtan/)

**Russell Taylor** is a Research Associate Professor of Computer Science, Physics & Astronomy, and Materials Science at the University of North Carolina at Chapel Hill. He has spent his 10-year career building scientific visualizations and interactive computer-graphics applications to help scientists better understand their instrumentation and data. He was the Principal Investigator on UNC's NIH National Research Resource on Computer Graphics for Molecular Studies and Microscopy and is now Co-director of UNC's NanoScale Science Research Group ([www.cs.unc.edu/Research/nano](http://www.cs.unc.edu/Research/nano)). For the past two years, Russ has taught a "Visualization in the Sciences" course to computer science, physics, chemistry, statistics, and materials science graduate students. The course is based in large part on the textbook "Information Visualization: Perception for Design" ([www.cs.unc.edu/Courses/comp290-069](http://www.cs.unc.edu/Courses/comp290-069)). Dr. Taylor has been a contributor to several past SIGGRAPH courses.

Website: <http://www.cs.unc.edu/~taylorr/>



# COURSE SYLLABUS

This course is designed to cover the psychophysics, design guideline, and fundamental haptic rendering algorithms, e.g. 3 degree-of-freedom (DOF) force-only display, 6-DOF force-and-torque display, or vibrotactile display for wearable haptics, and their applications in virtual prototyping, medical procedures, scientific visualization, 3D modeling & CAD/CAM, digital sculpting and other creative processes. We have assembled an excellent team of researchers and developers from both academia and industry to cover topics on fundamental algorithm design and novel applications.

## **FUNDAMENTALS IN HAPTIC RENDERING**

- *Haptic Perception & Design Guidelines* (Roberta Klatzky)
- *Haptic Display of Sculptured Surfaces and Models* (Elaine Cohen & David Johnson)
- *6-DOF Haptics* (*Voxel-sampling*: Bill McNeely, *Multiresolution*: Miguel Otaduy, *Spatialized Normal Cone*: David Johnson)
- *Texture Rendering* (*Perceptual parameters*: Hong Tan and *Force model*: Miguel Otaduy)
- *Modeling of Deformable Objects* (Dinesh Pai)
- *Wearable Haptics* (Hong Tan)

## **APPLICATIONS**

- *Virtual Prototyping* (Bill McNeely)
- *Medical Applications* (Kenneth Salisbury)
- *Scientific Visualization* (Russell Taylor)
- *CAD/CAM & Model Design* (Elaine Cohen)
- *Haptic Painting & Digital Sculpting* (Ming Lin)
- *Reality-Based Modeling for Multimodal Display* (Dinesh Pai)

## **PRE-REQUISITES**

This course is for programmers and researchers who have done some implementation of 3D graphics and want to learn more about how to incorporate recent advances in haptic rendering with their 3D graphics applications or virtual environments. Familiarity with basic 3D graphics, geometric operation and elementary physics is highly recommended.

## **INTENDED AUDIENCE**

Researchers who have background in computer graphics and want to learn how to add haptic interaction to simulated environments and those who are working in VR and various applications ranging from digital sculpting, medical training, scientific visualization, CAD/CAM, rapid prototyping, engineering design, education and training, painting, digital sculpting, etc.



# COURSE SCHEDULE

- 8:30am      **Introduction and Overview** (Ming Lin & Miguel Otaduy)
1. Definition of some very basic terminology
  2. Brief overview of a graphics+haptics system (HW/SW/control)
  3. Roadmap for the course
  4. Introduction of the speakers

## **SESSION I: DESIGN GUIDELINES AND BASIC POINT-BASED TECHNIQUES**

- 8:45am      **Haptic Perception & Design Guidelines** (Roberta Klatzky)
1. Sensory aspects of haptics
    - 1.1 mechanoreceptor function
    - 1.2 other receptors: thermal, pain
    - 1.3 cutaneous vs. kinesthetic components of haptic sense
  2. Psychophysical aspects of haptics
    - 2.1 haptic features
    - 2.2 link between exploration and haptic properties
  3. Complementary functions of haptics and vision
    - 3.1 material vs. geometric properties
    - 3.2 differential accessibility of these properties
    - 3.3 haptic/visual integration
  4. Issues for design
    - 4.1 force feedback vs. array feedback
    - 4.2 need for psychophysical input in developing & evaluating haptic feedback devices
    - 4.3 challenges for technology

- 9:30am      **Basics of 3-DOF Haptic Display** (Ken Salisbury & Ming Lin)
1. Geometric representations: point-based, polygonal, NURBS, etc.
  2. Force models: penalty-based, god-objects, virtual proxy, and so on
  3. Fast proximity queries and collision response, e.g. H-Collide
  4. Friction, texture rendering, force shading, etc.
  5. Multi-threading for haptic display

## **SESSION II: 6-DOF HAPTIC RENDERING FOR OBJECT-OBJECT INTERACTION**

- 10:00am      **Introduction to 6-DOF Haptic Display** (Bill McNeely)  
Brief introduction to 6-DOF haptic rendering & issues
- 10:15am      **BREAK**
- 10:30am      **6-DOF Haptic Display using Voxel Sampling** (Bill McNeely)  
**& Applications to Virtual Prototyping**
1. The Voxmap PointShell (VPS) approaches
    - 1.1 overview
    - 1.2 comparison with other approaches
    - 1.3 enhancements: distance fields, geometric awareness, temporal coherence
  2. Considerations in large-scale haptic simulation
    - 2.1 importance of minimum 10Hz graphics frame rate
    - 2.2 dynamic pre-fetching of voxel data
  3. Applications to virtual prototyping
    - 3.1 haptic-enabled FlyThru
    - 3.2 Spaceball quasi-haptic interface
- 10:55am      **Sensation Preserving Simplification for 6-DOF Haptic Display** (Miguel Otaduy)
1. Needs for multiresolution approaches in 6-DOF haptic rendering of complex interactions
  2. Contact Levels of Detail (C-LOD)
    - 2.1 definition of C-LOD
    - 2.2 creation of simplification hierarchy
  3. Collision detection and C-LOD selection
    - 3.1 definition of error metrics
    - 3.2 on-the-fly LOD selection
    - 3.3 accelerated collision queries
- 11:20am      **6-DOF Haptic Display of Sculptured Surfaces** (David Johnson)
1. Need for multiple point tracking
  2. Normal cones to solve collinearity condition, pt-surface, surface-surface
  3. Hybrid systems with local updates

### **SESSION III: HAPTIC RENDERING OF HIGHER-ORDER PRIMITIVES**

- 11:45am      **3-DOF Haptic Display of Sculptured Surfaces** (Elaine Cohen)
1. Introduce sculptured models, NURBS background, trimmed models
  2. Equations of distance, extremal distance, collinearity condition

3. Local tracking, wall models, etc for sculptured models
4. Tracking on trimmed NURBs models, in particular CAD/CAM models

12:15pm      **Q & A** (All Morning Speakers)

12:15am      **LUNCH**

## **SESSION IV: RENDERING OF TEXTURES AND DEFORMABLE SURFACES**

- 1:45pm      **Wearable Vibrotactile Haptic Displays** (Hong Tan)  
Brief introduction of wearable haptic displays originally developed for sensory substitution; discussion on the types of information that can be successfully conveyed by array-based wearable vibrotactile displays.
- 2:00pm      **Toward Realistic Haptic Rendering of Textures** (Hong Tan)  
Recent work on assessing the perceptual quality of haptically rendered surface textures. Emphasis will be placed on a quantitative parameter space for realistic haptic texture rendering, and on the types of perceptual instabilities commonly encountered and their sources.
- 2:20pm      **Haptic Rendering of Textured Surfaces** (Miguel Otaduy)  
  1. Overview of 3-DoF haptic texture rendering methods.
  2. A force model for 6-DoF haptic texture rendering
  3. GPU-based penetration depth computation
- 2:45pm      **Modeling of Deformable Objects** (Dinesh Pai)  
Force rendering of deformations – Basics of Elasticity; Numerical methods for BVPs (FEM,BEM); Precomputed Green's functions; Capacitance Matrix Algorithms; Cosserat models.

## **SESSION V: NOVEL APPLICATIONS**

- 3:10pm      **Reality-based Modeling for Multimodal Display** (Dinesh Pai)  
  1. Estimation theory; Contact force measurements; Visual measurements of deformation; Sound measurements;
  2. Case Study A: automated measurement with ACME;
  3. Case Study B: interactive measurement with HAVEN.

- 3:30pm      **BREAK**
- 3:45pm      **Haptics and Medicine** (Kenneth Salisbury)
1. Tissue Modeling
  2. Topological changes on deformable models: cutting, suturing, etc.
  3. Haptic interaction methods
  4. Simulation-based training, skills assessment and planning, etc.
- 4:20pm      **Applications in Scientific Visualization** (Russell Taylor)
1. Benefits of haptics for scientific visualization
  2. Haptic display of force fields
    - 2.1 for training Physics students
    - 2.2 for molecular docking
    - 2.3 for comprehending flows
  3. Haptic display of simulated models
    - 3.1 for molecular dynamics
    - 3.2 for electronics diagnosis training
    - 3.3 for medical training
  4. Haptic display of data:
    - 4.1 remote micromaching.
    - 4.2 display multiple data sets?
  5. Haptic control of instrumentation, e.g. scanned-probe microscopes
- 4:50pm      **Physically-based Haptic Painting & Interaction with Fluid Media** (Ming Lin)
1. Modeling of 3D deformable virtual brushes & viscous paint media
  2. Haptic rendering of brush-canvas interaction
  3. Haptic display of interaction with fluid media (e.g. oil paint)
- 5:15pm      **Q & A and Conclusion** (All Speakers)

# TABLE OF CONTENTS

## PREFACE

## PART I – Tutorial/Survey Chapters

<b>1. Introduction to Haptic Rendering .....</b>	<b>A3</b>
<i>by Miguel A. Otaduy and Ming C. Lin</i>	
<b>2. A Framework for Fast and Accurate Collision Detection for Haptic Interaction .....</b>	<b>A34</b>
<i>by Arthur Gregory, Ming C. Lin, Stefan Gottschalk, and Russell Taylor</i>	
<b>3. Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling .....</b>	<b>A42</b>
<i>by William A. McNeely, Kevin D. Puterbaugh, and James J. Troy</i>	
<b>4. Advances in Voxel-Based 6-DOF Haptic Rendering .....</b>	<b>A50</b>
<i>by William A. McNeely, Kevin D. Puterbaugh, and James J. Troy</i>	
<b>5. Sensation Preserving Simplification for Haptic Rendering .....</b>	<b>A72</b>
<i>by Miguel A. Otaduy and Ming C. Lin</i>	
<b>6. A Haptic System for Virtual Prototyping of Polygonal Models .....</b>	<b>A84</b>
<i>by David E. Johnson, Peter Willemsen, and Elaine Cohen</i>	
<b>7. Direct Haptic Rendering of Complex Trimmed NURBS Models .....</b>	<b>A89</b>
<i>by Thomas V. Thompson II and Elaine Cohen</i>	
<b>8. Haptic Rendering of Surface-to-Surface Sculpted Model Interaction .....</b>	<b>A97</b>
<i>by Donald D. Nelson, David E. Johnson, and Elaine Cohen</i>	
<b>9. Tactual Displays for Sensory Substitution and Wearable Computers .....</b>	<b>A105</b>
<i>by Hong Z. Tan and Alex Pentland</i>	
<b>10. Toward Realistic Haptic Rendering of Surface Textures .....</b>	<b>A125</b>
<i>by Seungmoon Choi and Hong Z. Tan</i>	

<b>11. Haptic Display of Interaction between Textured Models .....</b>	<b>A133</b>
<i>by Miguel A. Otaduy, Nitin Jain, Avneesh Sud, and Ming C. Lin</i>	
<b>12. A Unified Treatment of Elastostatic Contact Simulation for Real Time Haptics .....</b>	<b>A141</b>
<i>by Doug L. James and Dinesh K. Pai</i>	
<b>13. Scanning Physical Interaction Behavior of 3D Objects .....</b>	<b>A154</b>
<i>by Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, Som H. Yau</i>	
<b>14. The AHI: An Audio and Haptic Interface for Contact Interactions .....</b>	<b>A164</b>
<i>by Derek DiFilippo and Dinesh K. Pai</i>	
<b>15. Haptics for Scientific Visualization .....</b>	<b>A174</b>
<i>by Russell M. Taylor II</i>	
<b>16. DAB: Interactive Haptic Painting with 3D Virtual Brushes .....</b>	<b>A180</b>
<i>by Bill Baxter, Vincent Scheib, Ming C. Lin, and Dinesh Manocha</i>	
<b>17. ArtNova: Touch-Enabled 3D Model Design .....</b>	<b>A188</b>
<i>by Mark Foskey, Miguel A. Otaduy, and Ming C. Lin</i>	

## **PART II – Presentation Slides**

- 1. Haptics: Introduction and Overview .....B3**  
*by Ming C. Lin and Miguel A. Otaduy*
- 2. Haptic Perception and Implications for Design .....B5**  
*by Roberta Klatzky*
- 3. Introduction to 3-DoF Haptic Rendering .....B12**  
*by Ming C. Lin*
- 4. Introduction to 6-DoF Haptic Display .....B18**  
*by Bill McNeely*
- 5. Voxel Sampling for Six-DoF Haptic Rendering .....B19**  
*by Bill McNeely*
- 6. Sensation Preserving Simplification  
for 6-DoF Haptic Display .....B24**  
*by Miguel A. Otaduy*
- 7. Haptic Rendering of Polygonal Models  
Using Local Minimum Distances .....B37**  
*by David E. Johnson (Joint work with Elaine Cohen  
and Pete Willemsen)*
- 8. Haptic Rendering of Sculptured Models .....B45**  
*by Elaine Cohen (Joint work with Tom Thompson,  
Don Nelson and David Johnson)*
- 9. Wearable Vibrotactile Displays .....B49**  
*by Hong Z. Tan*
- 10. Towards Realistic Haptic Rendering of Surface Texture .....B52**  
*by Seungmoon Choi and Hong Z. Tan*
- 11. Haptic Rendering of Textured Surfaces .....B56**  
*by Miguel A. Otaduy*
- 12. Modeling Deformable Objects for Haptics .....B68**  
*by Dinesh K. Pai (Mostly joint work with Doug James)*

<b>13. Reality-based Modeling for Haptics and Multimodal Displays .....</b>	<b>B73</b>
<i>by Dinesh K. Pai</i>	
<b>14. Applications in Scientific Visualization .....</b>	<b>B80</b>
<i>by Russell M. Taylor II</i>	
<b>15. Haptic Interaction with Fluid Media .....</b>	<b>B87</b>
<i>by William Baxter and Ming C. Lin</i>	



# PREFACE

To date, most human–computer interactive systems have focused primarily on the graphical rendering of visual information and, to a lesser extent, on the display of auditory information. Among all senses, the human haptic system provides unique and bidirectional communication between humans and their physical environment. Extending the frontier of visual computing, haptic interfaces, or force feedback devices, have the potential to increase the quality of human-computer interaction by accommodating the sense of touch. They provide an attractive augmentation to visual display and enhance the level of understanding of complex data sets. They have been effectively used for a number of applications including molecular docking, manipulation of nano-materials, surgical training, virtual prototyping and digital sculpting.

Compared with visual and auditory display, haptic rendering has extremely demanding computational requirements. In order to maintain a stable system while displaying smooth and realistic forces and torques, haptic update rates of 1 KHz or more are typically used. Haptics presents many new challenges to researchers and developers in computer graphics and interactive techniques. Some of the critical issues include the development of novel data structures to encode shape and material properties, as well as new techniques for data processing, information analysis, physical modeling, and haptic visualization.

This course will examine some of the latest developments on haptic rendering and applications, while looking forward to exciting future research in this area. We will present novel haptic rendering algorithms and innovative applications that take advantage of haptic interaction sensory modality. Specifically we will discuss different rendering techniques for various geometric representations (e.g. point-based, volumetric, polygonal, multiresolution, NURBS, distance fields, etc) and physical properties (rigid bodies, deformable models, fluid medium, etc), as well as textured surfaces and full-body interaction (e.g. wearable haptics). We will also show how psychophysics of touch can provide the foundational design guidelines for developing perceptually driven force models and discuss issues to consider in validating new rendering techniques and evaluating haptic interfaces.

In addition, we will also look at different approaches to design touch-enabled interfaces for various applications, ranging from medical training, model design and maintainability analysis for virtual prototyping, scientific visualization, 3D painting and mesh editing, to data acquisition for multi-modal display. These recent advances indicate promising potentials that haptic interfaces together with interactive 3D graphics can offer a faster and more natural way of interacting with virtual environments and complex datasets in diverse applications.

*Ming C. Lin and Miguel A. Otaduy*



# **RECENT ADVANCES IN HAPTIC RENDERING AND APPLICATIONS**

## **Supplementary Course Notes**

### **PART I – Tutorial and Survey Chapters**

**Ming C. Lin**

**University of North Carolina at Chapel Hill**

**Miguel A. Otaduy**

**ETH Zurich**



# Introduction to Haptic Rendering

Miguel A. Otaduy   Ming C. Lin

Department of Computer Science  
University of North Carolina at Chapel Hill

## 1 Why Haptic Rendering?

For a long time, human beings have dreamed of a virtual world where it is possible to interact with synthetic entities as if they were real. To date, the advances in computer graphics allow us to *see* virtual objects and avatars, to *hear* them, to *move* them, and to *touch* them. It has been shown that the ability to touch virtual objects increases the sense of presence in virtual environments [Insko 2001].

Haptic rendering offers important applicability in engineering and medical training tasks. In this chapter we introduce the concept of haptic rendering, and we briefly describe some of the basic techniques and applications. In the first section we define some terminology, discuss the evolution of the research in haptic rendering, and introduce practical applications.

### 1.1 Definitions

The term *haptic* (from the Greek *haptesthai*, meaning “to touch”) is the adjective used to describe something relating to or based on the sense of touch. Haptic is to touching as visual is to seeing and as auditory is to hearing [Fisher et al. 2004].

As described by Klatzky and Lederman [Klatzky and Lederman 2003], touch is one of the main avenues of sensation, and it can be divided into cutaneous, kinesthetic, and haptic systems, based on the underlying neural inputs. The cutaneous system employs receptors embedded in the skin, while the kinesthetic system employs receptors located in muscles, tendons, and joints. The haptic sensory system employs both cutaneous and kinesthetic receptors, but it differs in the sense that it is associated with an active procedure. Touch becomes active when the sensory inputs are combined with controlled body motion. For example, cutaneous touch becomes active when we explore a surface or grasp an object, while kinesthetic touch becomes active when we manipulate an object and touch other objects with it.

*Haptic rendering* is defined as the process of computing and generating forces in response to user interactions with virtual objects [Salisbury et al. 1995]. Several haptic rendering algorithms consider the paradigm of touching virtual objects with a single contact point. Rendering algorithms that follow this description are called 3-DoF haptic rendering algorithms, because a point in 3D has only three DoFs. Other haptic rendering algorithms deal with the problem of rendering the forces and torques arising from the interaction of two virtual objects. This problem is called 6-DoF haptic rendering, because the grasped object has six DoFs (position and orientation in 3D), and the haptic feedback comprises 3D force and torque. When we eat with a fork, write with a pen, or open a lock with a key, we are moving an object in 3D, and we feel the interaction with other objects. This is, in essence, 6-DoF object manipulation with force-and-torque feedback. Fig. 1 shows an example of a user experiencing haptic rendering. When we manipulate an object and touch other objects with it, we perceive cutaneous feedback as the result of grasping, and kinesthetic feedback as the result of contact between objects.

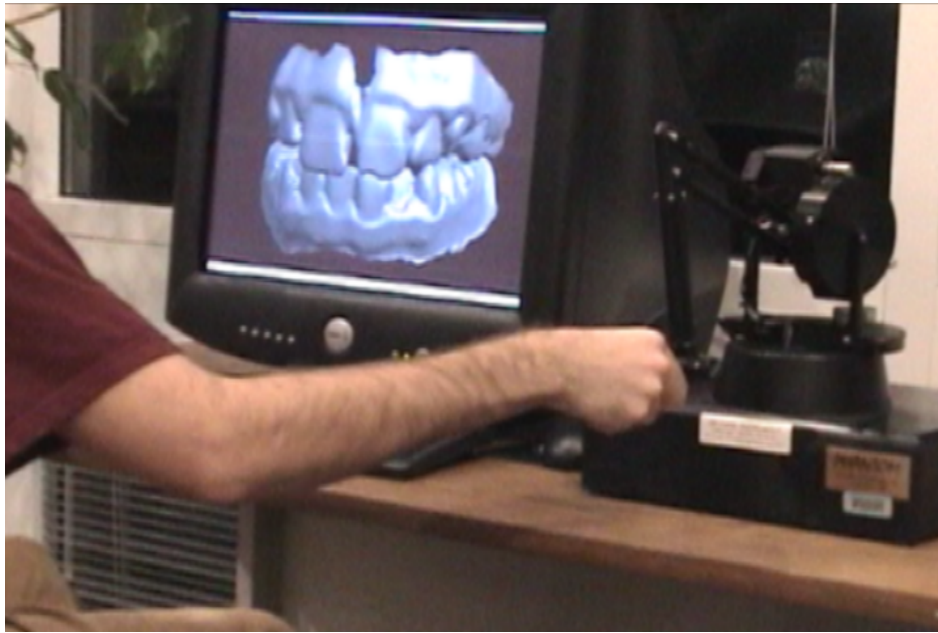


Figure 1: **Example of Haptic Rendering.** *A person manipulates a virtual jaw using a haptic device (shown on the right of the image), and the interaction between jaws is displayed both visually and haptically.*

## 1.2 From Telerobotics to Haptic Rendering

In 1965, Ivan Sutherland [Sutherland 1965] proposed a multimodal display that would incorporate haptic feedback into the interaction with virtual worlds. Before that, haptic feedback had already been used mainly in two applications: flight simulators and master-slave robotic teleoperation. The early teleoperator systems had mechanical linkages between the master and the slave. But, in 1954, Goertz and Thompson [Goertz and Thompson 1954] developed an electrical servomechanism that received feedback signals from sensors mounted on the slave and applied forces to the master, thus producing haptic feedback.

From there, haptic interfaces evolved in multiple directions, but there were two major breakthroughs. The first breakthrough was the idea of substituting the slave robot by a simulated system, in which forces were computed using physically based simulations. The GROPE project at the University of North Carolina at Chapel Hill [Brooks, Jr. et al. 1990], lasting from 1967 to 1990, was the first one to address the synthesis of force feedback from simulated interactions. In particular, the aim of the project was to perform real-time simulation of 3D molecular-docking forces. The second breakthrough was the advent of computer-based Cartesian control for teleoperator systems [Bejczy and Salisbury 1980], enabling a separation of the kinematic configurations of the master and the slave. Later, Cartesian control was applied to the manipulation of simulated slave robots [Kim and Bejczy 1991].

Those first haptic systems were able to simulate the interaction of simple virtual objects only. Perhaps the first project to target computation of forces in the interaction with objects with rich geometric information was Minsky's *Sandpaper* [Minsky et al. 1990]. Minsky et al. developed a planar force feedback system that allowed the exploration of textures. A few years after Minsky's work, Zilles and Salisbury presented an algorithm for 3-DoF haptic rendering of polygonal models [Zilles and Salisbury 1995]. Almost in parallel with Zilles and Salisbury's work, Massie and Salisbury [Massie and Salisbury 1994] designed the PHANToM, a stylus-based haptic interface that was later commercialized and has become one of the most commonly used force-feedback devices. But in the late '90s, research in haptic rendering revived one of the problems that first inspired virtual force feedback: 6-DoF haptic rendering or, in other words, grasping of a virtual object and synthesis of kinesthetic feedback of the interaction between this object and its environment.

Research in the field of haptics in the last 35 years has covered many more areas than what we have summarized here. [Burdea 1996] gives a general survey of the field of haptics and [McLaughlin et al. 2002] discuss current research topics.

### 1.3 Haptic Rendering for Virtual Manipulation

Certain professional activities, such as training for high-risk operations or pre-production prototype testing, can benefit greatly from simulated reproductions. The fidelity of the simulated reproductions depends, among other factors, on the similarity of the behaviors of real and virtual objects. In the real world, solid objects cannot interpenetrate. Contact forces can be interpreted mathematically as constraint forces imposed by penetration constraints. However, unless penetration constraints are explicitly imposed, virtual objects are free to penetrate each other in virtual environments. Indeed, one of the most disconcerting experiences in virtual environments is to pass through virtual objects [Insko et al. 2001; Slater and Usoh 1993]. Virtual environments require the simulation of non-penetrating rigid body dynamics, and this problem has been extensively explored in the robotics and computer graphics literature [Baraff 1992; Mirtich 1996].

It has been shown that being able to touch physical replicas of virtual objects (a technique known as *passive haptics* [Insko 2001]) increases the sense of presence in virtual environments. This conclusion can probably be generalized to the case of synthetic cutaneous feedback of the interaction with virtual objects. As reported by Brooks et al. [Brooks, Jr. et al. 1990], kinesthetic feedback radically improved situation awareness in virtual 3D molecular docking. Kinesthetic feedback has proved to enhance task performance in applications such as telerobotic object assembly [Hill and Salisbury 1977], virtual object assembly [Unger et al. 2002], and virtual molecular docking [Ouh-Young 1990]. In particular, task completion time is shorter with kinesthetic feedback in docking operations but not in repositioning operations.

To summarize, haptic rendering is especially useful in particular examples of training for high-risk operations or pre-production prototype testing activities that involve intensive object manipulation and interaction with the environment. Such examples include minimally invasive or endoscopic surgery [Edmond et al. 1997; Hayward et al. 1998] and virtual prototyping for assembly and maintainability assessment [McNeely et al. 1999; Chen 1999; Andriot 2002; Wan and McNeely 2003]. Force feedback becomes particularly important and useful in situations with limited visual feedback.

### 1.4 3-DoF and 6-DoF Haptic Rendering

Much of the existing work in haptic rendering has focused on 3-DoF haptic rendering [Zilles and Salisbury 1995; Ruspini et al. 1997; Thompson et al. 1997; Gregory et al. 1999; Ho et al. 1999]. Given a virtual object  $A$  and the 3D position of a point  $\mathbf{p}$  governed by an input device, 3-DoF haptic rendering can be summarized as finding a contact point  $\mathbf{p}'$  constrained to the surface of  $A$ . The contact force will be computed as a function of  $\mathbf{p}$  and  $\mathbf{p}'$ . In a dynamic setting, and assuming that  $A$  is a polyhedron with  $n$  triangles, the problem of finding  $\mathbf{p}'$  has an  $O(n)$  worst-case complexity. Using spatial partitioning strategies and exploiting motion coherence, however, the complexity becomes  $O(1)$  in many practical situations [Gregory et al. 1999].

This reduced complexity has made 3-DoF haptic rendering an attractive solution for many applications with virtual haptic feedback, such as: sculpting and deformation [Dachille et al. 1999; Gregory et al. 2000a; McDonnell et al. 2001], painting [Johnson et al. 1999; Gregory et al. 2000a; Foskey et al. 2002], volume visualization [Avila and Sobierajski 1996], nanomanipulation [Taylor et al. 1993], and training for diverse surgical operations [Kuhnapfel et al. 1997; Gibson et al. 1997]. In each of these applications, the interaction between the subject and the virtual objects is sufficiently captured by a point-surface contact model.

In 6-DoF manipulation and exploration, however, when a subject grasps an object and touches other objects in the environment, the interaction generally cannot be modeled by a point-surface contact. One reason is the existence of multiple contacts that impose multiple simultaneous non-penetration constraints

on the grasped object. In a simple 6-DoF manipulation example, such as the insertion of a peg in a hole, the grasped object (i.e., the peg) collides at multiple points with the rest of the scene (i.e., the walls of the hole and the surrounding surface). This contact configuration cannot be modeled as a point-object contact. Another reason is that the grasped object presents six DoFs, 3D translation and rotation, as opposed to the three DoFs of a point. The feasible trajectories of the peg are embedded in a 6-dimensional space with translational and rotational constraints, that cannot be captured with three DoFs.

Note that some cases of object-object interaction have been modeled in practice by ray-surface contact [Basdogan et al. 1997]. In particular, several surgical procedures are performed with 4-DoF tools (e.g., laparoscopy), and this constraint has been exploited in training simulators with haptic feedback [Çavuşoğlu et al. 2002]. Nevertheless, these approximations are valid only in a limited number of situations and cannot capture full 6-DoF object manipulation.

## 2 The Challenges

Haptic rendering is in essence an interactive activity, and its realization is mostly handicapped by two conflicting challenges: high required update rates and the computational cost. In this section we outline the computational pipeline of haptic rendering, and we discuss associated challenges.

### 2.1 Haptic Rendering Pipeline

Haptic rendering comprises two main tasks. One of them is the computation of the position and/or orientation of the virtual probe grasped by the user. The other one is the computation of contact force and/or torque that are fed back to the user. The existing methods for haptic rendering can be classified into two large groups based on their overall pipelines.

In *direct rendering* methods [Nelson et al. 1999; Gregory et al. 2000b; Kim et al. 2003; Johnson and Willemsen 2003; Johnson and Willemsen 2004], the position and/or orientation of the haptic device are applied directly to the grasped probe. Collision detection is performed between the grasped probe and the virtual objects, and collision response is applied to the grasped probe as a function of object separation or penetration depth. The resulting contact force and/or torque are directly fed back to the user.

In *virtual coupling* methods [Chang and Colgate 1997; Berkelman 1999; McNeely et al. 1999; Ruspini and Khatib 2000; Wan and McNeely 2003], the position and/or orientation of the haptic device are set as goals for the grasped probe, and a virtual viscoelastic coupling [Colgate et al. 1995] produces a force that attracts the grasped probe to its goals. Collision detection and response are performed between the grasped probe and the virtual objects. The coupling force and/or torque are combined with the collision response in order to compute the position and/or orientation of the grasped probe. The same coupling force and/or torque are fed back to the user.

In Sec. 8, I describe the different existing methods for 6-DoF haptic rendering in more detail, and I discuss their advantages and disadvantages. Also, as explained in more detail in Sec. 4, there are two major types of haptic devices, and for each type of device the rendering pipeline presents slight variations. Impedance-type devices read the position and orientation of the handle of the device and control the force and torque applied to the user. Admittance-type devices read the force and torque applied by the user and control the position and orientation of the handle of the device.

### 2.2 Force Update Rate

The ultimate goal of haptic rendering is to provide force feedback to the user. This goal is achieved by controlling the handle of the haptic device, which is in fact the end-effector of a robotic manipulator. When the user holds the handle, he or she experiences kinesthetic feedback. The entire haptic rendering system



is regarded as a mechanical impedance that sets a transformation between the position and velocity of the handle of the device and the applied force.

The quality of haptic rendering can be measured in terms of the dynamic range of impedances that can be simulated in a stable manner [Colgate and Brown 1994]. When the user moves the haptic device in free space, the perceived impedance should be very low (i.e., small force), and when the grasped virtual object touches other rigid objects, the perceived impedance should be high (i.e., high stiffness and/or damping of the constraint). The quality of haptic rendering can also be measured in terms of the responsiveness of the simulation [Brooks, Jr. et al. 1990; Berkelman 1999]. In free-space motion the grasped probe should respond quickly to the motion of the user. Similarly, when the grasped probe collides with a virtual wall, the user should stop quickly, in response to the motion constraint.

With impedance-type devices, virtual walls are implemented as large stiffness values in the simulation. In haptic rendering, the user is part of a closed-loop sampled dynamic system [Colgate and Schenkel 1994], along with the device and the virtual environment, and the existence of sampling and latency phenomena can induce unstable behavior under large stiffness values. System instability is directly perceived by the user in the form of disturbing oscillations. A key factor for achieving a high dynamic range of impedances (i.e., stiff virtual walls) while ensuring stable rendering is the computation of feedback forces at a high update rate [Colgate and Schenkel 1994; Colgate and Brown 1994]. Brooks et al. [Brooks, Jr. et al. 1990] reported that, in the rendering of textured surfaces, users were able to perceive performance differences at force update rates between 500Hz and 1kHz.

A more detailed description of the stability issues involved in the synthesis of force feedback, and a description of related work, are given in Sec. 4. Although here we have focused on impedance-type haptic devices, similar conclusions can be drawn for admittance-type devices (See [Adams and Hannaford 1998] and Sec. 4).

### 2.3 Contact Determination

The computation of non-penetrating rigid-body dynamics of the grasped probe and, ultimately, synthesis of haptic feedback require a model of collision response. Forces between the virtual objects must be computed from contact information. Determining whether two virtual objects collide (i.e., intersect) is not enough, and additional information, such as penetration distance, contact points, contact normals, and so forth, need to be computed. Contact determination describes the operation of obtaining the contact information necessary for collision response [Baraff 1992].

For two interacting virtual objects, collision response can be computed as a function of object separation, with worst-case cost  $O(mn)$ , or penetration depth, with a complexity bound of  $\Omega(m^3n^3)$ . But collision response can also be applied at multiple *contacts* simultaneously. Given two objects  $A$  and  $B$  with  $m$  and  $n$  triangles respectively, contacts can be defined as pairs of intersecting triangles or pairs of triangles inside a distance tolerance. The number of pairs of intersecting triangles is  $O(mn)$  in worst-case pathological cases, and the number of pairs of triangles inside a tolerance can be  $O(mn)$  in practical cases. In Sec. 5, we discuss in more detail existing techniques for determining the contact information.

The cost of contact determination depends largely on factors such as the convexity of the interacting objects or the contact configuration. There is no direct connection between the polygonal complexity of the objects and the cost of contact determination but, as a reference, existing exact collision detection methods can barely execute contact queries for force feedback between pairs of objects with 1,000 triangles in complex contact scenarios [Kim et al. 2003] at force update rates of 1kHz.

Contact determination becomes particularly expensive in the interaction between textured surfaces. Studies have been done on the highest texture resolution that can be perceived through cutaneous touch, but there are no clear results regarding the highest resolution that can be perceived kinesthetically through an intermediate object. It is known that, in the latter case, texture-induced roughness perception is encoded in vibratory motion [Klatzky and Lederman 2002]. Psychophysics researchers report that 1mm textures

are clearly perceivable, and perceived roughness appears to be even greater with finer textures [Lederman et al. 2000]. Based on Shannon’s sampling theorem, a  $10\text{cm} \times 10\text{cm}$  plate with a sinusoidal texture of  $1\text{mm}$  in orthogonal directions is barely correctly sampled with 40,000 vertices. This measure gives an idea of the triangulation density required for capturing texture information of complex textured objects. Note that the triangulation density may grow by orders of magnitude if the textures are not sinusoidal and/or if information about normals and curvatures is also needed.

### 3 Psychophysics of Haptics

In the design of contact determination algorithms for haptic rendering, it is crucial to understand the psychophysics of touch and to account for perceptual factors. The structure and behavior of human touch have been studied extensively in the field of psychology. The topics analyzed by researchers include characterization of sensory phenomena as well as cognitive and memory processes.

Haptic perception of physical properties includes a first step of stimulus registration and communication to the thalamus, followed by a second step of higher-level processing. Perceptual measures can be originated by individual mechanoreceptors but also by the integration of inputs from populations of different sensory units [Klatzky and Lederman 2003]. Klatzky and Lederman [Klatzky and Lederman 2003] discuss object and surface properties that are perceived through the sense of touch (e.g., texture, hardness, and weight) and divide them between geometric and material properties. They also analyze active exploratory procedures (e.g., lateral motion, pressure, or unsupported holding) typically conducted by subjects in order to capture information about the different properties.

Knowing the exploratory procedure(s) associated with a particular object or surface property, researchers have studied the influence of various parameters on the accuracy and magnitude of sensory outputs. Perceptual studies on tactile feature detection and identification, as well as studies on texture or roughness perception are of particular interest for haptic rendering. In this section we summarize existing research on perception of surface features and perception of roughness, and then we discuss issues associated with the interaction of visual and haptic modalities.

#### 3.1 Perception of Surface Features

Klatzky and Lederman describe two different exploratory procedures followed by subjects in order to capture shape attributes and identify features and objects. In *haptic glance* [Klatzky and Lederman 1995], subjects extract information from a brief haptic exposure of the object surface. Then they perform higher-level processing for determining the identity of the object or other attributes. In *contour following* [Klatzky and Lederman 2003], subjects create a spatiotemporal map of surface attributes, such as curvature, that serves as the pattern for feature identification. Contact determination algorithms attempt to describe the geometric interaction between virtual objects. The instantaneous nature of haptic glance [Klatzky and Lederman 1995] makes it strongly dependent on purely geometric attributes, unlike the temporal dependency of contour following.

Klatzky and Lederman [Klatzky and Lederman 1995] conducted experiments in which subjects were instructed to identify objects from brief cutaneous exposures (i.e., haptic glances). Subjects had an advance hypothesis of the nature of the object. The purpose of the study was to discover how, and how well, subjects identify objects from brief contact. According to Klatzky and Lederman, during haptic glance a subject has access to three pieces of information: roughness, compliance, and local features. Roughness and compliance are material properties that can be extracted from lower-level processing, while local features can lead to object identification by feature matching during higher-level processing. In the experiments, highest identification accuracy was achieved with small objects, whose *shapes* fit on a fingertip. Klatzky and Lederman concluded that large contact area helped in the identification of textures or patterns, although it was better to have a stimulus of a size comparable to or just slightly smaller than that of the contact area

for the identification of geometric surface features. The experiments conducted by Klatzky and Lederman posit an interesting relation between feature size and contact area during cutaneous perception.

Okamura and Cutkosky [Okamura and Cutkosky 1999; Okamura and Cutkosky 2001] analyzed feature detection in robotic exploration, which can be regarded as a case of object-object interaction. They characterized geometric surface features based on the ratios of their curvatures to the radii of the robotic fingertips acquiring the surface data. They observed that a larger fingertip, which provides a larger contact area, can miss small geometric features. To summarize, the studies by Klatzky and Lederman [Klatzky and Lederman 1995] and Okamura and Cutkosky [Okamura and Cutkosky 1999; Okamura and Cutkosky 2001] lead to the observation that human haptic perception of the existence of a geometric surface feature depends on the ratio between the contact area and the size of the feature, not the absolute size of the feature itself. This observation has driven the design of multiresolution contact determination algorithms for haptic rendering [Otaduy and Lin 2003b].

### 3.2 Perception of Texture and Roughness

Klatzky and Lederman [Klatzky and Lederman 2003] describe a textured surface as a surface with protuberant elements arising from a relatively homogeneous substrate. Interaction with a textured surface results in perception of roughness. Existing research on the psychophysics of texture perception indicates a clear dichotomy of exploratory procedures: (a) perception of texture with the bare skin, and (b) perception through an intermediate (rigid) object, a probe.

Most of the research efforts have been directed towards the characterization of cutaneous perception of textures. Katz [Katz 1989] suggested that roughness is perceived through a combination of spatial and vibratory codes during direct interaction with the skin. More recent evidence demonstrates that static pressure distribution plays a dominant role in perception of coarse textures (features larger than 1mm) [Lederman 1974; Connor and Johnson 1992], but motion-induced vibration is necessary for perceiving fine textures [LaMotte and Srinivasan 1991; Hollins and Risner 2000].

As pointed out by Klatzky and Lederman [Klatzky and Lederman 2002], in object-object interaction roughness is encoded in vibratory motion transmitted to the subject. In the last few years, Klatzky and Lederman have directed experiments that analyze the influence of several factors on roughness perception through a rigid probe. Klatzky et al. [Klatzky et al. 2003] distinguished three types of factors that may affect the perceived magnitude of roughness: interobject physical interaction, skin- and limb-induced filtering prior to cutaneous and kinesthetic perception, and higher-level factors such as efferent commands. The design of contact determination and collision response algorithms for haptic texture rendering is mostly concerned with factors related to the physical interaction between objects: object geometry [Lederman et al. 2000; Klatzky et al. 2003], applied force [Lederman et al. 2000], and exploratory speed [Lederman et al. 1999; Klatzky et al. 2003]. The influence of these factors has been addressed in the design of haptic texture rendering algorithms [Otaduy et al. 2004].

The experiments conducted by Klatzky and Lederman to characterize roughness perception [Klatzky and Lederman 2002] used a common setup: subjects explored a textured plate with a probe with a spherical tip, and then they reported a subjective measure of roughness. Plates of jittered raised dots were used, and the mean frequency of dot distribution was one of the variables in the experiments. The resulting data was analyzed by plotting subjective roughness values vs. dot interspacing in logarithmic graphs.

Klatzky and Lederman [Klatzky and Lederman 1999] compared graphs of roughness vs. texture spacing (a) with finger exploration and (b) with a rigid probe. They concluded that, in the range of their data, roughness functions were best fit by linear approximations in finger exploration and by quadratic approximations in probe-based exploration. In other words, when perceived through a rigid spherical probe, roughness initially increases as texture spacing increases, but, after reaching a maximum roughness value, it decreases again. Based on this finding, the influence of other factors on roughness perception can be characterized by the maximum value of roughness and the value of texture spacing at which this maximum

takes place.

Lederman et al. [Lederman et al. 2000] demonstrated that the diameter of the spherical probe plays a crucial role in the maximum value of perceived roughness and the location of the maximum. The roughness peak is higher for smaller probes, and it occurs at smaller texture spacing values. Lederman et al. [Lederman et al. 2000] also studied the influence of the applied normal force during exploration. Roughness is higher for larger force, but the influence on the location of the peak is negligible. The effect of exploratory speed was studied by Lederman et al. [Lederman et al. 1999]. They found that the peak of roughness occurs at larger texture spacing for higher speed. Also, with higher speed, textured plates feel smoother at small texture spacing, and rougher at large spacing values. The studies reflected that speed has a stronger effect in passive interaction than in active interaction.

### 3.3 Haptic and Visual Cross-modal Interaction

Haptic rendering is often presented along with visual display. Therefore, it is important to understand the issues involved in cross-modal interaction. Klatzky and Lederman [Klatzky and Lederman 2003] discuss aspects of visual and haptic cross-modal integration from two perspectives: attention and dominance.

Spence et al. [Spence et al. 2000] have studied how visual and tactile cues can influence a subject's attention. Their conclusions are that visual and tactile cues are treated together in a single attentional mechanism, and wrong attention cues can affect perception negatively.

Sensory dominance is usually studied by analyzing perceptual discrepancies in situations where cross-modal integration yields a unitary perceptual response. One example of relevance for this dissertation is the detection of object collision. During object manipulation, humans determine whether two objects are in contact based on a combination of visual and haptic cues. Early studies of sensory dominance seemed to point to a strong dominance of visual cues over haptic cues [Rock and Victor 1964], but in the last decades psychologists agree that sensory inputs are weighted based on their statistical reliability or relative appropriateness, measured in terms of accuracy, precision, and cue availability [Heller et al. 1999; Ernst and Banks 2001; Klatzky and Lederman 2003].

The design of contact determination algorithms can also benefit from existing studies on the visual perception of collisions in computer animations. O'Sullivan and her colleagues [O'Sullivan et al. 1999; O'Sullivan and Dingliana 2001; O'Sullivan et al. 2003] have investigated different factors affecting visual collision perception, including eccentricity, separation, distractors, causality, and accuracy of simulation results. Basing their work on a model of human visual perception validated by psychophysical experiments, they demonstrated the feasibility of using these factors for scheduling interruptible collision detection among large numbers of visually homogeneous objects.

## 4 Stability and Control Theory Applied to Haptic Rendering

In haptic rendering, the human user is part of the dynamic system, along with the haptic device and the computer implementation of the virtual environment. The complete human-in-the-loop system can be regarded as a sampled-data system [Colgate and Schenkel 1994], with a continuous component (the user and the device) and a discrete one (the implementation of the virtual environment and the device controller). Stability becomes a crucial feature, because instabilities in the system can produce oscillations that distort the perception of the virtual environment, or uncontrolled motion of the device that can even hurt the user. In Sec. 2.2, we have briefly discussed the importance of stability for haptic rendering, and we have introduced the effect of the force update rate on stability. In this section we review and discuss in more detail existing work in control theory related to stability analysis of haptic rendering.

## 4.1 Mechanical Impedance Control

The concept of mechanical impedance extends the notion of electrical impedance and refers to the quotient between force and velocity. Hogan [Hogan 1985] introduced the idea of impedance control for contact tasks in manipulation. Earlier techniques controlled contact force, robot velocity, or both, but Hogan suggested controlling directly the mechanical impedance, which governs the dynamic properties of the system. When the end effector of a robot touches a rigid surface, it suffers a drastic change of mechanical impedance, from low impedance in free space, to high impedance during contact. This phenomenon imposes serious difficulties on earlier control techniques, inducing instabilities.

The function of a haptic device is to display the feedback force of a virtual world to a human user. Haptic devices present control challenges very similar to those of manipulators for contact tasks. As introduced in Sec. 2.1, there are two major ways of controlling a haptic device: impedance control and admittance control. In impedance control, the user moves the device, and the controller produces a force dependent on the interaction in the virtual world. In admittance control, the user applies a force to the device, and the controller moves the device according to the virtual interaction.

In both impedance and admittance control, high control gains can induce instabilities. In impedance control, instabilities may arise in the simulation of stiff virtual surfaces. The device must react with large changes in force to small changes in the position. Conversely, in admittance control, rendering a stiff virtual surface is not a challenging problem, because it is implemented as a low controller gain. In admittance control, however, instabilities may arise during free-space motion in the virtual world, because the device must move at high velocities under small applied forces, or when the device rests on a stiff physical surface. Impedance and admittance control can therefore be regarded as complementary control techniques, best suited for opposite applications. Following the unifying framework presented by Adams and Hannaford [Adams and Hannaford 1998], Contact determination and force computation algorithms are often independent of the control strategy.

## 4.2 Stable Rendering of Virtual Walls

Since the introduction of impedance control by Hogan [Hogan 1985], the analysis of the stability of haptic devices and haptic rendering algorithms has focused on the problem of rendering stiff virtual walls. This was known to be a complex problem at early stages of research in haptic rendering [Kilpatrick 1976], but impedance control simplified the analysis, because a virtual wall can be modeled easily using stiffness and viscosity parameters.

Ouh-Young [Ouh-Young 1990] created a discrete model of the Argonne ARM and the human arm and analyzed the influence of force update rate on the stability and responsiveness of the system. Minsky, Brooks, et al. [Minsky et al. 1990; Brooks, Jr. et al. 1990] observed that update rates as high as 500Hz or 1kHz might be necessary in order to achieve stability.

Colgate and Brown [Colgate and Brown 1994] coined the term Z-Width for describing the range of mechanical impedances that a haptic device can render while guaranteeing stability. They concluded that physical dissipation is essential for achieving stability and that the maximum achievable virtual stiffness is proportional to the update rate. They also analyzed the influence of position sensors and quantization, and concluded that sensor resolution must be maximized and the velocity signal must be filtered.

Almost in parallel, Salcudean and Vlaar [Salcudean and Vlaar 1994] studied haptic rendering of virtual walls, and techniques for improving the fidelity of the rendering. They compared a continuous model of a virtual wall with a discrete model that accounts for differentiation of the position signal. The continuous model is unconditionally stable, but this is not true for the discrete model. Moreover, in the discrete model fast damping of contact oscillations is possible only with rather low contact stiffness and, as indicated by Colgate and Brown too [Colgate and Brown 1994], this value of stiffness is proportional to the update rate. Salcudean and Vlaar proposed the addition of braking pulses, proportional to collision velocity, for improving the perception of virtual walls.

### 4.3 Passivity and Virtual Coupling

A subsystem is *passive* if it does not add energy to the global system. Passivity is a powerful tool for analyzing stability of coupled systems, because the coupled system obtained from two passive subsystems is always stable. Colgate and his colleagues were the first to apply passivity criteria to the analysis of stability in haptic rendering of virtual walls [Colgate et al. 1993]. Passivity-based analysis has enabled separate study of the behavior of the human subsystem, the haptic device, and the virtual environment in force-feedback systems.

#### 4.3.1 Human Sensing and Control Bandwidths

Hogan discovered that the human neuromuscular system exhibits externally simple, springlike behavior [Hogan 1986]. This finding implies that the human arm holding a haptic device can be regarded as a passive subsystem, and the stability analysis can focus on the haptic device and the virtual environment.

Note that human limbs are not passive in all conditions, but the bandwidth at which a subject can perform active motions is very low compared to the frequencies at which stability problems may arise. Some authors [Shimoga 1992; Burdea 1996] report that the bandwidth at which humans can perform controlled actions with the hand or fingers is between 5 and 10Hz. On the other hand, sensing bandwidth can be as high as 20 to 30Hz for proprioception, 400Hz for tactile sensing, and 5 to 10kHz for roughness perception.

#### 4.3.2 Passivity of Virtual Walls

Colgate and Schenkel [Colgate and Schenkel 1994] observed that the oscillations perceived by a haptic user during system instability are a result of active behavior of the force-feedback system. This active behavior is a consequence of time delay and loss of information inherent in sampled-data systems, as suggested by others before [Brooks, Jr. et al. 1990]. Colgate and Schenkel formulated passivity conditions in haptic rendering of a virtual wall. For that analysis, they modeled the virtual wall as a viscoelastic unilateral constraint, and they accounted for the continuous dynamics of the haptic device, sampling of the position signal, discrete differentiation for obtaining velocity, and a zero-order hold of the output force. They reached a sufficient condition for passivity that relates the stiffness  $K$  and damping  $B$  of the virtual wall, the inherent damping  $b$  of the device, and the sampling period  $T$ :

$$b > \frac{KT}{2} + B. \quad (1)$$

#### 4.3.3 Stability of Non-linear Virtual Environments

After deriving stability conditions for rendering virtual walls modeled as unilateral linear constraints, Colgate and his colleagues considered more complex environments [Colgate et al. 1995]. A general virtual environment is non-linear, and it presents multiple and variable constraints. Their approach enforces a discrete-time passive implementation of the virtual environment and sets a multidimensional viscoelastic *virtual coupling* between the virtual environment and the haptic display. In this way, the stability of the system is guaranteed as long as the virtual coupling is itself passive, and this condition can be analyzed using the same techniques as those used for virtual walls [Colgate and Schenkel 1994]. As a result of Colgate's virtual coupling [Colgate et al. 1995], the complexity of the problem was shifted towards designing a passive solution of virtual world dynamics. As noted by Colgate et al. [Colgate et al. 1995], one possible way to enforce passivity in rigid body dynamics simulation is to use implicit integration with penalty methods.

Adams and Hannaford [Adams and Hannaford 1998] provided a framework for analyzing stability with admittance-type and impedance-type haptic devices. They derived stability conditions for coupled systems based on network theory. They also extended the concept of virtual coupling to admittance-type devices.

Miller et al. [Miller et al. 1990] extended Colgate’s passivity analysis techniques, relaxing the requirement of passive virtual environments but enforcing *cyclo-passivity* of the complete system. Hannaford and his colleagues [Hannaford et al. 2002] investigated the use of adaptive controllers instead of the traditional fixed-value virtual couplings. They designed passivity observers and passivity controllers for dissipating the excess of energy generated by the virtual environment.

#### 4.4 Multirate Approximation Techniques

Multirate approximation techniques, though simple, have been successful in improving the stability and responsiveness of haptic rendering systems. The idea is to perform a full update of the virtual environment at a low frequency (limited by computational resources and the complexity of the system) and to use a simplified approximation for performing high-frequency updates of force feedback.

Adachi [Adachi et al. 1995] proposed an *intermediate representation* for haptic display of complex polygonal objects. In a slow collision detection thread, he computed a plane that served as a unilateral constraint in the force-feedback thread. This technique was later adapted by Mark et al. [Mark et al. 1996], who interpolated the intermediate representation between updates. This approach enables higher stiffness values than approaches that compute the feedback force values at the rate imposed by collision detection. More recently, a similar multirate approach has been followed by many authors for haptic interaction with deformable models [Astley and Hayward 1998; Çavuşoğlu and Tendick 2000; Duriez et al. 2004]. Ellis et al. [Ellis et al. 1997] produce higher-quality rendering by upsampling directly the output force values.

### 5 Collision Detection

Collision detection has received much attention in robotics, computational geometry, and computer graphics. Some researchers have investigated the problem of interference detection as a mechanism for indicating whether object configurations are valid or not. Others have tackled the problems of computing separation or penetration distances, with the objective of applying collision response in simulated environments. The existing work on collision detection can be classified based on the types of models handled: 2-manifold polyhedral models, polygon soups, curved surfaces, etc. In this section we focus on collision detection for polyhedral models. The vast majority of the algorithms used in practice proceed in two steps: first they cull large portions of the objects that are not in close proximity, using spatial partitioning, hierarchical techniques, or visibility-based properties, and then they perform primitive-level tests.

In this section, we first describe the problems of interference detection and computation of separation distance between polyhedra, with an emphasis on algorithms specialized for convex polyhedra. Then we survey algorithms for the computation of penetration depth, the use of hierarchical techniques, and multiresolution collision detection. we conclude the section by covering briefly the use of graphics processors for collision detection and the topic of continuous collision detection. For more information on collision detection, please refer to surveys on the topic [Lin and Gottschalk 1998; Klosowski et al. 1998; Lin and Manocha 2004].

#### 5.1 Proximity Queries Between Convex Polyhedra

The property of convexity has been exploited in algorithms with sublinear cost for detecting interference or computing the closest distance between two polyhedra. Detecting whether two convex polyhedra intersect can be posed as a linear programming problem, searching for the coefficients of a separating plane. Well-known linear programming algorithms [Seidel 1990] can run in expected linear time due to the low dimensionality of the problem.

The separation distance between two polyhedra  $A$  and  $B$  is equal to the distance from the origin to the Minkowski sum of  $A$  and  $-B$  [Cameron and Culley 1986]. This property was exploited by Gilbert

et al. [Gilbert et al. 1988] in order to design a convex optimization algorithm (known as GJK) for computing the separation distance between convex polyhedra, with linear-time performance in practice. Cameron [Cameron 1997] modified the GJK algorithm to exploit motion coherence in the initialization of the convex optimization at every frame for dynamic problems, achieving nearly constant running-time in practice.

Lin and Canny [Lin and Canny 1991; Lin 1993] designed an algorithm for computing separation distance by tracking the closest features between convex polyhedra. Their algorithm “walks” on the surfaces of the polyhedra until it finds two features that lie on each other’s Voronoi region. Exploiting motion coherence and geometric locality, *Voronoi marching* runs in nearly constant time per frame. Mirtich [Mirtich 1998b] later improved the robustness of this algorithm.

Given polyhedra  $A$  and  $B$  with  $m$  and  $n$  polygons respectively, Dobkin and Kirkpatrick [Dobkin and Kirkpatrick 1990] proposed an algorithm for interference detection with  $O(\log m \log n)$  time complexity that uses hierarchical representations of the polyhedra. Others have also exploited the use of hierarchical convex representations along with temporal coherence in order to accelerate queries in dynamic scenes. Guibas et al. [Guibas et al. 1999] employ the inner hierarchies suggested by Dobkin and Kirkpatrick, but they perform faster multilevel walking. Ehmann and Lin [Ehmann and Lin 2000] employ a modified version of Dobkin and Kirkpatrick’s outer hierarchies, computed using simplification techniques, along with a multilevel implementation of Lin and Canny’s Voronoi marching [Lin and Canny 1991].

## 5.2 Penetration Depth

The penetration depth between two intersecting polyhedra  $A$  and  $B$  is defined as the minimum translational distance required for separating them. For intersecting polyhedra, the origin is contained in the Minkowski sum of  $A$  and  $-B$ , and the penetration depth is equal to the minimum distance from the origin to the surface of the Minkowski sum. The computation of penetration depth can be  $\Omega(m^3 n^3)$  for general polyhedra [Dobkin et al. 1993].

Many researchers have restricted the computation of penetration depth to convex polyhedra. In computational geometry, Dobkin et al. [Dobkin et al. 1993] presented an algorithm for computing directional penetration depth, while Agarwal et al. [Agarwal et al. 2000] introduced a randomized algorithm for computing the penetration depth between convex polyhedra. Cameron [Cameron 1997] extended the GJK algorithm [Gilbert et al. 1988] to compute bounds of the penetration depth, and van den Bergen [van den Bergen 2001] furthered his work. Kim et al. [Kim et al. 2002a] presented an algorithm that computes a locally optimal solution of the penetration depth by walking on the surface of the Minkowski sum.

The fastest algorithms for computation of penetration depth between arbitrary polyhedra take advantage of discretization. Fisher and Lin [Fisher and Lin 2001] estimate penetration depth using distance fields computed with fast marching level-sets. Hoff et al. [Hoff et al. 2001] presented an image-based algorithm implemented on graphics hardware. On the other hand, Kim et al. [Kim et al. 2002b] presented an algorithm that decomposes the polyhedra into convex patches, computes the Minkowski sums of pairwise patches, and then uses an image-based technique in order to find the minimum distance from the origin to the surface of the Minkowski sums.

## 5.3 Hierarchical Collision Detection

The algorithms for collision detection between convex polyhedra are not directly applicable to non-convex polyhedra or models described as polygon soups. Brute force checking of all triangle pairs, however, is usually unnecessary. Collision detection between general models achieves large speed-ups by using hierarchical culling or spatial partitioning techniques that restrict the primitive-level tests. Over the last decade, bounding volume hierarchies (BVH) have proved successful in the acceleration of collision detection for dynamic scenes of rigid bodies. For an extensive description and analysis of the use of BVHs for collision detection, please refer to Gottschalk’s PhD dissertation [Gottschalk 2000].



Assuming that an object is described by a set of triangles  $T$ , a BVH is a tree of BVs, where each BV  $C_i$  bounds a cluster of triangles  $T_i \in T$ . The clusters bounded by the children of  $C_i$  constitute a partition of  $T_i$ . The effectiveness of a BVH is conditioned by ensuring that the branching factor of the tree is  $O(1)$  and that the size of the leaf clusters is also  $O(1)$ . Often, the leaf BVs bound only one triangle. A BVH may be created in a top-down manner, by successive partitioning of clusters, or in a bottom-up manner, by using merging operations.

In order to perform interference detection using BVHs, two objects are queried by recursively traversing their BVHs in tandem. Each recursive step tests whether a pair of BVs  $a$  and  $b$ , one from each hierarchy, overlap. If  $a$  and  $b$  do not overlap, the recursion branch is terminated. Otherwise, if they overlap, the algorithm is applied recursively to their children. If  $a$  and  $b$  are both leaf nodes, the triangles within them are tested directly. This process can be generalized to other types of proximity queries as well.

One determining factor in the design of a BVH is the selection of the type of BV. Often there is a trade-off among the tightness of the BV (and therefore the culling efficiency), the cost of the collision test between two BVs, and the dynamic update of the BV (relevant for deformable models). Some of the common BVs, sorted approximately according to increasing query time, are: spheres [Quinlan 1994; Hubbard 1994], axis-aligned bounding boxes (AABB) [Beckmann et al. 1990], oriented bounding boxes (OBB) [Gottschalk et al. 1996],  $k$ -discrete-orientation polytopes (k-DOP) [Klosowski et al. 1998], convex hulls [Ehmann and Lin 2001], and swept sphere volumes (SSV) [Larsen et al. 2000]. BVHs of rigid bodies can be computed as a preprocessing step, but deformable models require a bottom-up update of the BVs after each deformation. Recently, James and Pai [James and Pai 2004] have presented the BD-tree, a variant of the sphere-tree data structure [Quinlan 1994] that can be updated in a fast top-down manner if the deformations are described by a small number of parameters.

## 5.4 Multiresolution Collision Detection

Multiresolution analysis of a function decomposes the function into a basic low-resolution representation and a set of detail terms at increasing resolutions. Wavelets provide a mathematical framework for defining multiresolution analysis [Stollnitz et al. 1996].

Multiresolution representations of triangles meshes have drawn important attention in computer graphics. They have been defined in two major ways: following the mathematical framework of wavelets and subdivision surfaces [Lounsbery et al. 1997; Eck et al. 1995] or following level-of-detail (LOD) simplification techniques (please refer to [Luebke et al. 2002] for a survey on the topic). LOD techniques present the advantage of being applicable to arbitrary meshes, but they lack a well-defined metric of resolution. They construct the multiresolution representations starting from full-resolution meshes and applying sequences of local simplification operations. LOD techniques can be divided into those that produce a discrete set of representations (static LODs), and those that produce continuously adaptive representations (dynamic LODs). Multiresolution or LOD techniques have been used in applications such as view-dependent rendering [Hoppe 1997; Luebke and Erikson 1997], interactive editing of meshes [Zorin et al. 1997], or real-time deformations [Debunne et al. 2001]. The idea behind multiresolution techniques is to select the resolution or LOD of the representation in an adaptive manner based on perceptual parameters, availability of computational resources, and so forth.

Multiresolution collision detection refers to the execution of approximate collision detection queries using adaptive object representations. Hubbard [Hubbard 1994] introduced the idea of using sphere-trees [Quinlan 1994] for multiresolution collision detection, refining the BVHs in a breadth-first manner until the time allocated for collision detection expires. In a sphere-tree each level of the BVH can be regarded as an implicit approximation of the given mesh, by defining the surface as a union of spheres. Unlike LOD techniques, in which simplification operations minimize surface deviation, sphere-trees add extraneous “bumpiness” to the surface, and this characteristic can hurt collision response.

O’Sullivan and Dingliana [O’Sullivan and Dingliana 2001] have incorporated perceptual parameters into

the refinement of sphere-trees. They insert pairs of spheres that test positive for collision in a priority queue sorted according to perceptual metrics (e.g., local relative velocity, distance to the viewer, etc.). In this way the adaptive refinement focuses on areas of the objects where errors are most noticeable.

The use of multiresolution representations for haptic rendering has also been investigated by several researchers. Pai and Reissel [Pai and Reissel 1997] investigated the use of multiresolution image curves for 2D haptic interaction. El-Sana and Varshney [El-Sana and Varshney 2000] applied LOD techniques to 3-DoF haptic rendering. They created a multiresolution representation of the haptically rendered object as a preprocessing step and, at runtime, they represented the object at high resolution near the probe point and at low resolution further away. Their approach does not extend naturally to the interaction between two objects, since multiple disjoint contacts can occur simultaneously at widely varying locations without much spatial coherence. Otaduy and Lin [Otaduy and Lin 2003b; Otaduy and Lin 2003a] introduced *contact levels of detail*, dual hierarchical representations for multiresolution collision detection, and they applied them to 6-DoF haptic rendering, producing a sensation-preserving simplified rendering.

## 5.5 Other Techniques for Collision Detection

We briefly cover two additional topics with potential applicability in haptic rendering: the use of graphics processors for collision detection, and continuous collision detection.

### 5.5.1 Use of Graphics Processors for Collision Detection

The processing capability of GPUs is growing at a rate higher than Moore's law [Govindaraju et al. 2003], and this circumstance has generated an increasing use of GPUs for general-purpose computation, including collision detection. Rasterization hardware enables high performance of image-based collision detection algorithms. Hoff et al. [Hoff et al. 2001] presented an algorithm for estimating penetration depth between deformable polygons using distance fields computed on graphics hardware. Others have formulated collision detection queries as visibility problems. Lombardo et al. [Lombardo et al. 1999] intersected a complex object against a simpler one using the view frustum and clipping planes, and they detected intersecting triangles by exploiting OpenGL capabilities. More recently, Govindaraju et al. [Govindaraju et al. 2003] have designed an algorithm that performs series of visibility queries and achieves fast culling of non-intersecting primitives in  $N$ -body problems with nonrigid motion.

### 5.5.2 Continuous Collision Detection

Continuous collision detection refers to a temporal formulation of the collision detection problem. The collision query attempts to find intersecting triangles and the time of intersection. Redon et al. [Redon et al. 2002] proposed an algorithm that assumes an arbitrary interframe rigid motion and incorporates the temporal dimension in OBB-trees using interval arithmetic. Continuous collision detection offers potential applicability to haptic rendering because it may enable constraint-based simulations without expensive backtracking operations used for computing the time of first collision.

## 6 Rigid Body Simulation

Computation of the motion of a rigid body consists of solving a set of ordinary differential equations (ODEs). The most common way to describe the motion of a rigid body is by means of the Newton-Euler equations, which define the time derivatives of the linear momentum,  $\mathbf{P}$ , and angular momentum,  $\mathbf{L}$ , as a function of external force  $\mathbf{F}$  and torque  $\mathbf{T}$ :

$$\begin{aligned}\mathbf{F}(t) &= \dot{\mathbf{P}}(t) = m \ddot{\mathbf{x}}(t), \\ \mathbf{T}(t) &= \dot{\mathbf{L}}(t) = \omega(t) \times (M\omega(t)) + M\dot{\omega}(t).\end{aligned}\tag{2}$$

As shown in the equations, momentum derivatives can be expressed in terms of the linear acceleration of the center of mass  $\ddot{\mathbf{x}}$ , the angular velocity  $\omega$ , the mass of the body  $m$ , and the mass matrix  $M$ . The complexity of rigid body simulation lies in the computation of force and torque resulting from contacts between bodies. Research in the field of rigid body simulation has revolved around different methods for computing contact forces and the resulting accelerations and velocities, ranging from approximate methods that consider each contact independently (such as penalty-based methods) to analytic methods that account concurrently for all non-penetration constraints. Important efforts have been devoted to capturing friction forces as well.

In this section we briefly describe the main methods for solving the motion of colliding rigid bodies, focusing on their applicability to haptic rendering. For further information, please refer to Baraff's or Mirtich's dissertations [Baraff 1992; Mirtich 1996], SIGGRAPH course notes on the topic [Baraff and Witkin 2001], or recent work by Stewart and Trinkle [Stewart and Trinkle 2000]. In the last few years, especially in the field of computer graphics, attention has been drawn towards the problem of simulating the interaction of many rigid bodies [Mirtich 2000; Milenkovic and Schmidl 2001; Guendelman et al. 2003]. For haptic rendering, however, one is mostly concerned with the dynamics of the object grasped by the user; therefore the interaction of many rigid objects is not discussed here.

## 6.1 Penalty-Based Methods

When two objects touch or collide, collision response must be applied to prevent object interpenetration. One method for implementing collision response is the insertion of stiff springs at the points of contact [Moore and Wilhelms 1988]. This method is inspired by the fact that, when objects collide, small deformations take place at the region of contact, and these deformations can be modeled with springs, even if the objects are geometrically rigid.

Given two intersecting objects  $A$  and  $B$ , penalty-based collision response requires the definition of a contact point  $\mathbf{p}$ , a contact normal  $\mathbf{n}$  and a penetration depth  $\delta$ . The penalty-based spring force and torque applied to object  $A$  are defined as follows:

$$\begin{aligned}\mathbf{F}_A &= -f(\delta)\mathbf{n}, \\ \mathbf{T}_A &= (\mathbf{p} - \mathbf{c}_A) \times \mathbf{F}_A,\end{aligned}\tag{3}$$

where  $\mathbf{c}_A$  is the center of mass of  $A$ . Opposite force and torque are applied to object  $B$ . The function  $f$  could be a linear function defined by a constant stiffness  $k$  or a more complicated non-linear function. It could also contain a viscous term, dependent on the derivative of the penetration depth.

The basic formulation of penalty methods can be modified slightly in order to introduce repulsive forces between objects, by inserting contact springs when the objects come closer than a distance tolerance  $d$ . In this way, object interpenetration occurs less frequently. The addition of a tolerance has two major advantages: the possibility of using penalty methods in applications that do not allow object interpenetration, and a reduction of the cost of collision detection. As noted in Sec. 5, computation of penetration depth is notably more costly than computation of separation distance.

Penalty-based methods offer several attractive properties: the force model is local to each contact and computationally simple, object interpenetration is inherently allowed, and contact determination needs to be performed only once per simulation frame. This last property makes penalty-based methods best suited for interactive applications with fixed time steps, such as haptic rendering [McNeely et al. 1999; Kim et al. 2003; Johnson and Willemsen 2003; Otaduy and Lin 2005] and games [Wu 2000; Larsen 2001]. But

penalty-based methods also have some disadvantages. There is no direct control over physical parameters, such as the coefficient of restitution. Non-penetration constraints are enforced by means of very high contact stiffness, and this circumstance leads to instability problems if numerical integration is executed using fast, explicit methods. The solution of penalty-based simulation using implicit integration, however, enhances stability in the presence of high contact stiffness [Wu 2000; Larsen 2001; Otaduy and Lin 2005].

Friction effects can be incorporated into penalty-based methods by means of localized force models that consider each contact point independently. Most local friction methods propose different force models for static or dynamic situations [Karnopp 1985; Hayward and Armstrong 2000]. Static friction is modeled by fixing adhesion points on the surfaces of the colliding objects and setting tangential springs between the contact points and the adhesion points. If the elastic friction force becomes larger than a threshold determined by the normal force and the friction coefficient, the system switches to dynamic mode. In the dynamic mode, the adhesion point follows the contact point. The system returns to static mode if the velocity falls under a certain threshold.

So far, we have analyzed contact determination and collision response as two separate problems, but the output of the contact determination step has a strong influence on the smoothness of collision response and, as a result, on the stability of numerical integration. As pointed out by Larsen [Larsen 2001], when a new contact point is added, the associated spring must be unstretched. In other words, the penetration depth value must be zero initially and must grow smoothly. The existence of geometry-driven discontinuities is an inherent problem of penalty-based simulations with fixed time steps. Some authors [Hasegawa and Sato 2004] have proposed sampling the intersection volume to avoid geometric discontinuities in the application of penalty-based methods to rigid body simulation and haptic rendering, but this approach is applicable only to very simple objects.

## 6.2 Constraint-Based Simulation

Constraint-based methods for the simulation of rigid body dynamics handle all concurrent contacts in a single computational problem and attempt to find contact forces that produce physically and geometrically valid motions. Specifically, they integrate the Newton-Euler equations of motion (see Eq. 2), subject to geometric constraints that prevent object interpenetration. The numerical integration of Newton-Euler equations must be interrupted before objects interpenetrate. At a collision event, object velocities and accelerations must be altered, so that non-penetration constraints are not violated and numerical integration can be restarted. One must first compute contact impulses that produce constraint-valid velocities. Then, one must compute contact forces that produce valid accelerations.

The relative normal accelerations  $\mathbf{a}$  at the points of contact can be expressed as linear combinations of the contact forces  $\mathbf{F}$  (with constant matrix  $A$  and vector  $\mathbf{b}$ ). Moreover, one can impose non-penetration constraints on the accelerations and non-attraction constraints on the forces:

$$\begin{aligned} \mathbf{a} &= A\mathbf{F} + \mathbf{b}, \\ \mathbf{a} &\geq 0, \quad \mathbf{F} \geq 0. \end{aligned} \tag{4}$$

Baraff [Baraff 1989] pioneered the application of constraint-based approaches to rigid body simulation in computer graphics. He posed constrained rigid body dynamics simulation as a quadratic programming problem on the contact forces, and he proposed a fast, heuristic-based solution for the frictionless case. He defined a quadratic cost function based on the fact that contact forces occur only at contact points that are not moving apart:

$$\min (\mathbf{F}^T \mathbf{a}) = \min (\mathbf{F}^T A\mathbf{F} + \mathbf{F}^T \mathbf{b}). \tag{5}$$

The quadratic cost function suggested by Baraff indicates that either the normal acceleration or the contact force should be 0 at a resting contact. As indicated by Cottle et al. [Cottle et al. 1992], this

condition can be formulated as a linear complementarity problem (LCP). Baraff [Baraff 1991; Baraff 1992] added dynamic friction to the formulation of the problem and suggested approaches for static friction, as well as a solution following an algorithm by Lemke [Lemke 1965] with expected polynomial cost in the number of constraints. Earlier, Lötstedt had studied the problem of rigid body dynamics with friction in the formulation of the LCP [Lötstedt 1984]. Later, Baraff himself [Baraff 1994] adapted an algorithm by Cottle and Dantzig [Cottle and Dantzig 1968] for solving frictionless LCPs to the friction case, and achieved linear-time performance in practice.

Stewart and Trinkle [Stewart and Trinkle 1996] presented an implicit LCP formulation of constraint-based problems. Unlike previous algorithms, which enforced the constraints only at the beginning of each time step, their algorithm solves for contact impulses that also enforce the constraints at the end of the time step. This formulation eliminates the need to locate collision events, but it increases the number of constraints to be handled, and it is unclear how it behaves with complex objects.

Stewart and Trinkle [Stewart and Trinkle 1996] mention the existence of geometry-driven discontinuities, similar to the ones appearing with penalty methods, in their implicit formulation of the LCP. After numerical integration of object position and velocities, new non-penetration constraints are computed. If numerical integration is not interrupted at collision events, the newly computed non-penetration constraints may not hold. Constraint violation may produce unrealistically high contact impulses and object velocities in the next time step. This phenomenon is equivalent to the effect of prestretched penalty-based springs described by Larsen [Larsen 2001]. Stewart and Trinkle suggest solving a non-linear complementarity problem, with additional cost involved.

If numerical integration is interrupted at collision events, the effects of geometry-driven discontinuities can be alleviated by capturing all the contact points that bound the contact region. Baraff [Baraff 1989] considers polygonal contact regions between polyhedral models and defines contact constraints at the vertices that bound the polygonal regions. Similarly, Mirtich [Mirtich 1998a] describes polygonal contact areas as combinations of edge-edge and vertex-face contacts.

### 6.3 Impulse-Based Dynamics

Mirtich [Mirtich and Canny 1995; Mirtich 1996] presented a method for handling collisions in rigid body dynamics simulation based solely on the application of impulses to the objects. In situations of resting, sliding, or rolling contact, constraint forces are replaced by trains of impulses. Mirtich defined a collision matrix that relates contact impulse to the change in relative velocity at the contact. His algorithm decomposes the collision event into two separate processes: compression and restitution. Each process is parameterized separately, and numerical integration is performed in order to compute the velocities after the collision. The parameterization of the collision event enables the addition of a friction model to instantaneous collisions.

The time-stepping engine of impulse-based dynamics is analogous to the one in constraint-based dynamics: numerical integration must be interrupted before interpenetration occurs, and valid velocities must be computed. One of the problems of impulse-based dynamics emerges during inelastic collisions from the fact that accelerations are not recomputed. The energy loss induced by a train of inelastic collisions reduces the time between collisions and increases the cost of simulation per frame. In order to handle this problem, Mirtich suggested the addition of unrealistic, but visually imperceptible, energy to the system when the microcollisions become too frequent. As has been pointed out by Mirtich, impulse-based approaches are best suited for simulations that are collision-intensive, with multiple, different impacts occurring frequently.

## 7 Haptic Texture Rendering

Although haptic rendering of textures was one of the first tackled problems [Minsky et al. 1990], it has been mostly limited to the interaction between a probe point and a textured surface. We begin this section with

a description of Minsky’s pioneering algorithm for rendering textures on the plane [Minsky 1995]. Then we discuss rendering of textures on 3D surfaces, covering basic 3-DoF haptic rendering, height-field-based methods, and probabilistic methods.

### 7.1 Rendering Textures on the Plane

Minsky [Minsky 1995] developed the *Sandpaper* system for 2-DoF haptic rendering of textures on a planar surface. Her system was built around a force model for computing 2D forces from texture height field information. Following energy-based arguments, her force model synthesizes a force  $\mathbf{F}$  in 2D based on the gradient of the texture height field  $h$  at the location of the probe:

$$\mathbf{F} = -k\nabla h. \quad (6)$$

Minsky also analyzed qualitatively and quantitatively roughness perception and the believability of the proposed force model. One of the main conclusions of her work is to establish her initial hypothesis, that texture information can be conveyed by displaying forces tangential to the contact surface. This hypothesis was later exploited for rendering textured 3D surfaces [Ho et al. 1999].

### 7.2 3-Degree-of-Freedom Haptic Rendering

As described in Sec. 1.4, 3-DoF haptic rendering methods compute feedback force as a function of the separation between the probe point controlled with the haptic device and a contact point constrained to the surface of the haptically rendered object. Early 3-DoF haptic rendering methods set the contact point as the point on the surface of the object closest to the probe point. As has been addressed by Zilles and Salisbury [Zilles and Salisbury 1995], these methods lead to force discontinuities and possible “pop-through” problems, in which the contact point jumps between opposing sides of the object. Instead, Zilles and Salisbury proposed the *god-object* method, which defines the computation of the contact point as a constrained optimization problem. The contact point is located at a minimum distance from the probe point, but its interframe trajectory is constrained by the surface. Zilles and Salisbury solve the position of the contact point using Lagrange multipliers, once they define the set of active constraints.

Ruspini et al. [Ruspini et al. 1997] followed a similar approach. They modeled the contact point as a sphere of small radius and solved the optimization problem in the configuration space. Ruspini and his colleagues also added other effects, such as force shading for rounding of corners (by modifying the normals of constraint planes), or friction (by adding dynamic behavior to the contact point).

### 7.3 Methods Based on Height Fields

High-resolution surface geometry can be represented by a parameterized coarse mesh along with texture images storing detailed height field or displacement field information, similarly to the common approach of texture mapping in computer graphics [Catmull 1974]. Constraint-based 3-DoF haptic rendering methods determine a unique contact point on the surface of the rendered object. Usually, the mesh representation used for determining the contact point is rather coarse and does not capture high-frequency texture. Nevertheless, the parametric coordinates of the contact point can be used for accessing surface texture information from texture images.

Ho et al. [Ho et al. 1999] introduced a technique similar to bump mapping [Blinn 1978] that alters the surface normal based on the gradient of the texture height field. A combination of the original and refined normals is used for computing the direction of the feedback force.

Techniques for haptic texture rendering based on a single contact point can capture geometric properties of only one object and are not suitable for simulating full interaction between two surfaces. The geometric interaction between two surfaces is not limited to, and cannot be described by, a pair of contact points.

Moreover, the local kinematics of the contact between two surfaces include rotational degrees of freedom, which are not captured by point-based methods.

Ho et al. [Ho et al. 1999] indicate that a high height field gradient can induce system instability. Along a similar direction, Choi and Tan [Choi and Tan 2003b; Choi and Tan 2003a] have studied the influence of collision detection and penetration depth computation on 3-DoF haptic texture rendering. Discontinuities in the output of collision detection are perceived by the user, a phenomenon that they describe as *aliveness*. This phenomenon is a possible problem in 6-DoF haptic rendering too.

## 7.4 Probabilistic Methods

Some researchers have exploited statistical properties of surfaces for computing texture-induced forces that are added to the classic 3-DoF contact forces. Siira and Pai [Siira and Pai 1996] synthesized texture forces according to a Gaussian distribution for generating a sensation of roughness. In order to improve stability, they did not apply texture forces during static contact. Later, Pai et al. [Pai et al. 2001] presented a technique for rendering roughness effects by dynamically modifying the coefficient of friction of a surface. The roughness-related portion of the friction coefficient was computed according to an autoregressive process driven by noise.

Probabilistic methods have proved to be successful for rendering high-frequency roughness effects in point-surface contact. It is also possible, although this approach has yet to be explored, that they could be combined with geometric techniques for synthesizing high-frequency effects in 6-DoF haptic rendering.

# 8 6-Degree-of-Freedom Haptic Rendering

The problem of 6-DoF haptic rendering has been studied by several researchers. As introduced in Sec. 2.1, the existing methods for haptic rendering can be classified into two large groups based on their overall pipelines: *direct rendering* methods and *virtual coupling* methods. Each group of methods presents some advantages and disadvantages. Direct rendering methods are purely geometric, and there is no need to simulate the rigid body dynamics of the grasped object. However, penetration values may be quite large and visually perceptible, and system instability can arise if the force update rate drops below the range of stable values. Virtual coupling methods enable reduced interpenetration, higher stability, and higher control of the displayed stiffness. However, virtual coupling [Colgate et al. 1995] may introduce noticeable filtering, both tactile and visual, and it requires the simulation of rigid body dynamics.

The different 6-DoF haptic rendering methods propose a large variety of options for solving the specific problems of collision detection, collision response, and simulation of rigid body dynamics. In the presence of infinite computational resources, an ideal approach to the problem of 6-DoF haptic rendering would be to compute the position of the grasped object using constraint-based rigid body dynamics simulation [Baraff 1992] and to implement force feedback through virtual coupling. This approach has indeed been followed by some, but it imposes serious limitations on the complexity of the objects and contact configurations that can be handled interactively. We now discuss briefly the different existing methods for 6-DoF haptic rendering, focusing on those that have been applied to moderately complex objects and scenarios.

## 8.1 Direct Haptic Rendering Approaches

Gregory et al. [Gregory et al. 2000b] presented a 6-DoF haptic rendering system that combined collision detection based on convex decomposition of polygonal models [Ehmann and Lin 2001], predictive estimation of penetration depth, and force and torque interpolation. They were able to handle interactively dynamic scenes with several convex objects, as well as pairs of non-convex objects with a few hundred triangles and rather restricted motion. Kim et al. [Kim et al. 2003] exploited convex decomposition for collision detection and incorporated fast, incremental, localized computation of per-contact penetration depth [Kim

et al. 2002a]. In order to improve stability and eliminate the influence of triangulation on the description of the contact manifold, they introduced a contact clustering technique. Their system was able to handle pairs of models with nearly one hundred convex pieces each interactively.

Earlier, Nelson et al. [Nelson et al. 1999] introduced a technique for haptic interaction between pairs of parametric surfaces. Their technique tracks contact points that realize locally maximum penetration depth during surface interpenetration. Tracking contact points, instead of recomputing them for every frame, ensures smooth penetration values, which are used for penalty-based force feedback. The contact points are solved in parametric space, and they are defined as those pairs of points for which their difference vector is collinear with surface normals.

Johnson and Willemssen [Johnson and Willemssen 2003] suggested a technique for polygonal models that defines contact points as those that satisfy a local minimum-distance criterion, according to Nelson’s definition [Nelson et al. 1999]. Johnson and Willemssen exploit this definition in a fast collision culling algorithm, using spatialized normal cone hierarchies [Johnson and Cohen 2001]. The performance of their technique depends on the convexity and triangulation of the models, which affect the number of contact points. Recently, Johnson and Willemssen [Johnson and Willemssen 2004] have incorporated an approximate but fast, incremental contact-point-tracking algorithm that is combined with slower exact collision updates from their previous technique [Johnson and Willemssen 2003]. This algorithm handles models with thousands of triangles at interactive rates, but the forces may suffer discontinuities if the exact update is too slow.

## 8.2 Virtual Coupling with Object Voxelization

In 1999, McNeely et al. [McNeely et al. 1999] presented a system for 6-DoF haptic rendering that employs a discrete collision detection approach and virtual coupling. The system is intended for assembly and maintenance planning applications and assumes that only one of the objects in the scene is dynamic. The surfaces of the scene objects are voxelized, and the grasped object is point-sampled. The collision detection module checks for inclusion of the sample points in the scene voxels, and then a local force model is applied. Hierarchical culling of sample points is possible, but ultimately the computational cost depends on the number of contact points. This system has been integrated in a commercial product, VPS, distributed by Boeing.

McNeely and his colleagues introduced additional features in order to alleviate some of the limitations. Surface objects are voxelized only on the surface, therefore deep penetrations, which can occur if objects collide at high velocities, cannot be handled. They propose pre-contact braking forces, similar to the braking impulses suggested by Salcudean [Salcudean and Vlaar 1994], for reducing the contact velocity of the grasped object and thereby preventing deep penetrations. The existence of multiple contact points produces high stiffness values that can destabilize the simulation of rigid body dynamics. They propose averaging the effects of the different contact points before contact forces are applied to the grasped object, for limiting the stiffness and thereby ensuring stable simulation. The locality of the force model induces force discontinuities when contact points traverse voxel boundaries. They point out that force discontinuities are somewhat filtered by the virtual coupling. Renz et al. [Renz et al. 2001] modified McNeely’s local force model to ensure continuity of the surface across voxel boundaries, but incurring more expensive force computation.

Using the same voxelization and point-sampling approach for collision detection, Wan and McNeely [Wan and McNeely 2003] have proposed a novel solution for computing the position of the grasped object. The early approach by McNeely et al. [McNeely et al. 1999] computed object dynamics by explicit integration of Newton-Euler equations. Instead, Wan and McNeely [Wan and McNeely 2003] presented a purely geometric solution that eliminates the instability problems that can arise due to high contact stiffness. Their algorithm formulates linear approximations of the coupling and contact force and torque in the space of translations and rotations of the grasped object. The state of the object is computed at every



frame by solving for the position of quasi-static equilibrium. Deep penetrations are avoided by formulating the coupling force as a non-linear spring.

### 8.3 Rigid Body Dynamics with Haptic Feedback

Chang and Colgate [Chang and Colgate 1997] proposed a solution to 6-DoF haptic rendering by combining virtual coupling [Colgate et al. 1995] and rigid body simulation based on impulse dynamics [Mirtich 1996]. They found that impulses alone were not efficient in resting contact situations, and in those cases they suggested a combination of impulses and penalty forces. Recently, Constantinescu et al. [Constantinescu et al. 2004] have reached a similar conclusion. As has been addressed by Constantinescu, combining impulses and penalty forces requires a state machine in order to determine the state of the object, but it is not clear how to extend this solution to scenes with many contacts. Both Chang and Constantinescu have tested their implementations only on simple benchmarks.

One of the reasons for the simplicity of Chang and Constantinescu's benchmarks is the cost of collision detection for the simulation of rigid body dynamics. As has been discussed in Sec. 6, impulse- [Mirtich 1996] or constraint-based [Baraff 1992] methods must interrupt the integration before object interpenetration, and this leads to many collision queries per frame. Some researchers have integrated haptic interaction with constraint-based rigid body simulations [Berkelman 1999; Ruspini and Khatib 2000] in scenes with simple geometry.

As indicated in Sec. 6.1, non-penetration constraints can be relaxed using penalty-based methods. McNeely et al. [McNeely et al. 1999] employed penalty methods for rigid body simulation but, as explained earlier, they observed numerical instabilities due to high stiffness values, and large interpenetrations under high impact velocities. Those problems can be tackled with high-stiffness penalty contact forces along with implicit integration, an approach used in interactive rigid body simulations [Wu 2000; Larsen 2001]. Implicit integration requires the evaluation of the Jacobian of the Newton-Euler equations and the solution of a linear system of equations [Baraff and Witkin 1998]. As demonstrated by Otaduy and Lin [Otaduy and Lin 2005], implicit integration can be performed at force update rates under the assumption that only the grasped object is dynamic.

### 8.4 Multiresolution Techniques

The application of 6-DoF haptic rendering algorithms to complex models and complex contact scenarios becomes a challenging issue, due to the inherent cost of collision detection that induces slow force updates. Otaduy and Lin [Otaduy and Lin 2003b] have presented a sensation-preserving simplification technique for 6-DoF haptic rendering of complex polygonal models by selecting contact resolutions adaptively. Otaduy et al. [Otaduy et al. 2004] have also proposed a rendering algorithm for the interaction of textured surfaces. Their work is focused on the acceleration of collision detection and response using level-of-detail representations and texture images.

## References

- ADACHI, Y., KUMANO, T., AND OGINO, K. 1995. Intermediate representation for stiff virtual objects. *Virtual Reality Annual International Symposium*, 203–210.
- ADAMS, R. J., AND HANNAFORD, B. 1998. A two-port framework for the design of unconditionally stable haptic interfaces. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- AGARWAL, P., GUIBAS, L., HAR-PELED, S., RABINOVITCH, A., AND SHARIR, M. 2000. Penetration depth of two convex polytopes in 3d. *Nordic J. Computing* 7, 227–240.
- ANDRIOT, C. 2002. Advances in virtual prototyping. *Clefs CEA Vol. 47, Research and Simulation*.

- ASTLEY, O. R., AND HAYWARD, V. 1998. Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. *Proc. of IEEE International Conference on Robotics and Automation*.
- AVILA, R. S., AND SOBIERAJSKI, L. M. 1996. A haptic interaction method for volume visualization. In *IEEE Visualization '96*, IEEE. ISBN 0-89791-864-9.
- BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. *Proc. of ACM SIGGRAPH*, 43–54.
- BARAFF, D., AND WITKIN, A. 2001. *Physically-Based Modeling*. ACM SIGGRAPH Course Notes.
- BARAFF, D. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, J. Lane, Ed., vol. 23, 223–232.
- BARAFF, D. 1991. Coping with friction for non-penetrating rigid body simulation. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, T. W. Sederberg, Ed., vol. 25, 31–40.
- BARAFF, D. 1992. *Dynamic simulation of non-penetrating rigid body simulation*. PhD thesis, Cornell University.
- BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of SIGGRAPH '94*, A. Glassner, Ed., ACM SIGGRAPH, 23–34. ISBN 0-89791-667-0.
- BASDOGAN, C., HO, C.-H., AND SRINIVASAN, M. A. 1997. A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. *DSC-Vol. 61, Proceedings of the ASME Dynamic Systems and Control Division*, pp. 77–84.
- BECKMANN, N., KRIEGEL, H., SCHNEIDER, R., AND SEEGER, B. 1990. The r\*-tree: An efficient and robust access method for points and rectangles. *Proc. SIGMOD Conf. on Management of Data*, 322–331.
- BEJCZY, A., AND SALISBURY, J. K. 1980. Kinematic coupling between operator and remote manipulator. *Advances in Computer Technology Vol. 1*, 197–211.
- BERKELMAN, P. J. 1999. *Tool-Based Haptic Interaction with Dynamic Physical Simulations Using Lorentz Magnetic Levitation*. PhD thesis, Carnegie Mellon University.
- BLINN, J. F. 1978. Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, 286–292.
- BROOKS, JR., F. P., OUH-YOUNG, M., BATTER, J. J., AND KILPATRICK, P. J. 1990. Project GROPE — Haptic displays for scientific visualization. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, F. Baskett, Ed., vol. 24, 177–185.
- BURDEA, G. 1996. *Force and Touch Feedback for Virtual Reality*. John Wiley and Sons.
- CAMERON, S., AND CULLEY, R. K. 1986. Determining the minimum translational distance between two convex polyhedra. *Proceedings of International Conference on Robotics and Automation*, 591–596.
- CAMERON, S. 1997. Enhancing GJK: Computing minimum and penetration distance between convex polyhedra. *IEEE International Conference on Robotics and Automation*, 3112–3117.
- CATMULL, E. E. 1974. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Dept. of CS, U. of Utah.

- ÇAVUŞOĞLU, M. C., AND TENDICK, F. 2000. Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. *Proc. of IEEE International Conference on Robotics and Automation*, 2458–2465.
- ÇAVUŞOĞLU, M. C., TENDICK, F., AND SASTRY, S. S. 2002. Haptic interfaces to real and virtual surgical environments. In *Touch in Virtual Environments*, M. L. McLaughlin, J. P. Hespanha, and G. S. Sukhatme, Eds. Prentice Hall PTR, Upper Saddle River, NJ, ch. 13, 217–237.
- CHANG, B., AND COLGATE, J. E. 1997. Real-time impulse-based simulation of rigid body systems for haptic display. *Proc. of ASME Dynamic Systems and Control Division*.
- CHEN, E. 1999. Six degree-of-freedom haptic system for desktop virtual prototyping applications. In *Proceedings of the First International Workshop on Virtual Reality and Prototyping*, 97–106.
- CHOI, S., AND TAN, H. Z. 2003. Aliveness: Perceived instability from a passive haptic texture rendering system. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- CHOI, S., AND TAN, H. Z. 2003. An experimental study of perceived instability during haptic texture rendering: Effects of collision detection algorithm. *Proc. of Haptics Symposium*, 197–204.
- COLGATE, J. E., AND BROWN, J. M. 1994. Factors affecting the z-width of a haptic display. *IEEE International Conference on Robotics and Automation*, 3205–3210.
- COLGATE, J. E., AND SCHENKEL, G. G. 1994. Passivity of a class of sampled-data systems: Application to haptic interfaces. *Proc. of American Control Conference*.
- COLGATE, J. E., GRAFING, P. E., STANLEY, M. C., AND SCHENKEL, G. 1993. Implementation of stiff virtual walls in force-reflecting interfaces. *Virtual Reality Annual International Symposium*, 202–207.
- COLGATE, J. E., STANLEY, M. C., AND BROWN, J. M. 1995. Issues in the haptic display of tool use. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 140–145.
- CONNOR, C. E., AND JOHNSON, K. O. 1992. Neural coding of tactile texture: Comparison of spatial and temporal mechanisms for roughness perception. *Journal of Neuroscience* 12, pp. 3414–3426.
- CONSTANTINESCU, D., SALCUDAN, S. E., AND CROFT, E. A. 2004. Impulsive forces for haptic rendering of rigid contact. *Proc. of International Symposium on Robotics*, 1–6.
- COTTLE, R. W., AND DANTZIG, G. B. 1968. Complementarity pivot theory of mathematical programming. *Linear Algebra and its Applications*, 103–125.
- COTTLE, R. W., PANG, J. S., AND STONE, R. E. 1992. The linear complementarity problem. *Academic-Press, Inc.*.
- DACHILLE, F., QIN, H., KAUFMAN, A., AND EL-SANA, J. 1999. Haptic sculpting of dynamic surfaces. *Proc. of ACM Symposium on Interactive 3D Graphics*, 103–110.
- DEBUNNE, G., DESBRUN, M., CANI, M. P., AND BARR, A. H. 2001. Dynamic real-time deformations using space and time adaptive sampling. *Proc. of ACM SIGGRAPH*.
- DOBKIN, D. P., AND KIRKPATRICK, D. G. 1990. Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. 17th Internat. Colloq. Automata Lang. Program.*, Springer-Verlag, vol. 443 of *Lecture Notes in Computer Science*, 400–413.
- DOBKIN, D., HERSHBERGER, J., KIRKPATRICK, D., AND SURI, S. 1993. Computing the intersection-depth of polyhedra. *Algorithmica* 9, 518–533.

- DURIEZ, C., ANDRIOT, C., AND KHEDDAR, A. 2004. A multi-threaded approach for deformable/rigid contacts with haptic feedback. *Proc. of Haptics Symposium*.
- ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERRY, M., AND STUETZLE, W. 1995. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Conference Proceedings*, Addison Wesley, R. Cook, Ed., Annual Conference Series, ACM SIGGRAPH, 173–182. held in Los Angeles, California, 06–11 August 1995.
- EDMOND, C., HESKAMP, D., SLUIS, D., STREDNEY, D., WIET, G., YAGEL, R., WEGHORST, S., OPPENHEIMER, P., MILLER, J., LEVIN, M., AND ROSENBERG, L. 1997. Ent endoscopic surgical simulator. *Proc. of Medicine Meets VR*, 518–528.
- EHMANN, S., AND LIN, M. C. 2000. Accelerated proximity queries between convex polyhedra using multi-level voronoi marching. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2101–2106.
- EHMANN, S., AND LIN, M. C. 2001. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum (Proc. of Eurographics'2001)* 20, 3, 500–510.
- EL-SANA, J., AND VARSHNEY, A. 2000. Continuously-adaptive haptic rendering. *Virtual Environments 2000*, pp. 135–144.
- ELLIS, R. E., SARKAR, N., AND JENKINS, M. A. 1997. Numerical methods for the force reflection of contact. *ASME Transactions on Dynamic Systems, Modeling and Control* 119, 768–774.
- ERNST, M. O., AND BANKS, M. S. 2001. Does vision always dominate haptics? *Touch in Virtual Environments Conference*.
- FISHER, S., AND LIN, M. C. 2001. Fast penetration depth estimation for elastic bodies using deformed distance fields. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 2001*.
- FISHER, B., FELS, S., MACLEAN, K., MUNZNER, T., AND RENSINK, R. 2004. Seeing, hearing and touching: Putting it all together. In *ACM SIGGRAPH course notes*.
- FOSKEY, M., OTADUY, M. A., AND LIN, M. C. 2002. ArtNova: Touch-enabled 3D model design. *Proc. of IEEE Virtual Reality Conference*.
- GIBSON, S., SAMOSKY, J., MOR, A., FYOCK, C., GRIMSON, E., AND KANADE, T. 1997. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback. *First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery (CVMed-MRCAS)*, 368–378.
- GILBERT, E. G., JOHNSON, D. W., AND KEERTHI, S. S. 1988. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE J. Robotics and Automation* vol RA-4, 193–203.
- GOERTZ, R., AND THOMPSON, R. 1954. Electronically controlled manipulator. *Nucleonics*, 46–47.
- GOTTSCHALK, S., LIN, M., AND MANOCHA, D. 1996. OBB-Tree: A hierarchical structure for rapid interference detection. *Proc. of ACM Siggraph'96*, 171–180.
- GOTTSCHALK, S. 2000. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, University of North Carolina. Department of Computer Science.

- GOVINDARAJU, N., REDON, S., LIN, M., AND MANOCHA, D. 2003. CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware. *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 25–32.
- GREGORY, A., LIN, M., GOTTSCHALK, S., AND TAYLOR, R. 1999. H-COLLIDE: A framework for fast and accurate collision detection for haptic interaction. In *Proceedings of Virtual Reality Conference 1999*, 38–45.
- GREGORY, A., EHMANN, S., AND LIN, M. C. 2000. *inTouch*: Interactive multiresolution modeling and 3d painting with a haptic interface. *Proc. of IEEE VR Conference*.
- GREGORY, A., MASCARENHAS, A., EHMANN, S., LIN, M. C., AND MANOCHA, D. 2000. 6-DoF haptic display of polygonal models. *Proc. of IEEE Visualization Conference*.
- GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)* 22, 871–878.
- GUIBAS, L., HSU, D., AND ZHANG, L. 1999. *H-Walk*: Hierarchical distance computation for moving convex bodies. *Proc. of ACM Symposium on Computational Geometry*.
- HANNAFORD, B., RYU, J.-H., AND KIM, Y. S. 2002. Stable control of haptics. In *Touch in Virtual Environments*, M. L. McLaughlin, J. P. Hespanha, and G. S. Sukhatme, Eds. Prentice Hall PTR, Upper Saddle River, NJ, ch. 3, 47–70.
- HASEGAWA, S., AND SATO, M. 2004. Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects. In *Proc. of Eurographics*.
- HAYWARD, V., AND ARMSTRONG, B. 2000. A new computational model of friction applied to haptic rendering. *Experimental Robotics VI*.
- HAYWARD, V., GREGORIO, P., ASTLEY, O., GREENISH, S., AND DOYON, M. 1998. Freedom-7: A high fidelity seven axis haptic device with applications to surgical training. *Experimental Robotics*, 445–456. Lecture Notes in Control and Information Sciences 232.
- HELLER, M. A., CALCATERRA, J. A., GREEN, S. L., AND BROWN, L. 1999. Intersensory conflict between vision and touch: The response modality dominates when precise, attention-riveting judgements are required. *Perception and Psychophysics* 61, pp. 1384–1398.
- HILL, J. W., AND SALISBURY, J. K. 1977. Two measures of performance in a peg-in-hole manipulation task with force feedback. *Thirteenth Annual Conference on Manual Control, MIT*.
- HO, C.-H., BASDOGAN, C., AND SRINIVASAN, M. A. 1999. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence* 8, 5, pp. 477–491.
- HOFF, K., ZAFERAKIS, A., LIN, M., AND MANOCHA, D. 2001. Fast and simple 2d geometric proximity queries using graphics hardware. *Proc. of ACM Symposium on Interactive 3D Graphics*, 145–148.
- HOGAN, N. 1985. Impedance control: An approach to manipulation, part i - theory, part ii - implementation, part iii - applications. *Journal of Dynamic Systems, Measurement and Control* 107, 1–24.
- HOGAN, N. 1986. Multivariable mechanics of the neuromuscular system. *IEEE Annual Conference of the Engineering in Medicine and Biology Society*, 594–598.
- HOLLINS, M., AND RISNER, S. 2000. Evidence for the duplex theory of tactile texture perception. *Perception & Psychophysics* 62, 695–705.

- HOPPE, H. 1997. View dependent refinement of progressive meshes. In *ACM SIGGRAPH Conference Proceedings*, 189–198.
- HUBBARD, P. 1994. *Collision Detection for Interactive Graphics Applications*. PhD thesis, Brown University.
- INSKO, B., MEEHAN, M., WHITTON, M., AND BROOKS, F. 2001. Passive haptics significantly enhances virtual environments. Tech. Rep. 01-010, Department of Computer Science, UNC Chapel Hill.
- INSKO, B. 2001. *Passive Haptics Significantly Enhance Virtual Environments*. PhD thesis, University of North Carolina. Department of Computer Science.
- JAMES, D. L., AND PAI, D. K. 2004. Bd-tree: Output-sensitive collision detection for reduced deformable models. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*.
- JOHNSON, D. E., AND COHEN, E. 2001. Spatialized normal cone hierarchies. *Proc. of ACM Symposium on Interactive 3D Graphics*, pp. 129–134.
- JOHNSON, D. E., AND WILLEMSSEN, P. 2003. Six degree of freedom haptic rendering of complex polygonal models. In *Proc. of Haptics Symposium*.
- JOHNSON, D. E., AND WILLEMSSEN, P. 2004. Accelerated haptic rendering of polygonal models through local descent. *Proc. of Haptics Symposium*.
- JOHNSON, D., THOMPSON II, T. V., KAPLAN, M., NELSON, D., AND COHEN, E. 1999. Painting textures with a haptic interface. *Proceedings of IEEE Virtual Reality Conference*.
- KARNOPP, D. 1985. Computer simulation of stick slip friction in mechanical dynamic systems. *Trans. ASME, Journal of Dynamic Systems, Measurement, and Control*.
- KATZ, D. 1989. *The World of Touch*. Erlbaum, Hillsdale, NJ. L. Krueger, Trans. (Original work published 1925).
- KILPATRICK, P. J. 1976. *The use of a kinesthetic supplement in an interactive graphics system*. Ph.d. thesis, The University of North Carolina at Chapel Hill.
- KIM, W., AND BEJCZY, A. 1991. Graphical displays for operator aid in telemanipulation. *IEEE International Conference on Systems, Man and Cybernetics*.
- KIM, Y. J., LIN, M. C., AND MANOCHA, D. 2002. DEEP: an incremental algorithm for penetration depth computation between convex polytopes. *Proc. of IEEE Conference on Robotics and Automation*, 921–926.
- KIM, Y. J., LIN, M. C., AND MANOCHA, D. 2002. Fast penetration depth computation using rasterization hardware and hierarchical refinement. *Proc. of Workshop on Algorithmic Foundations of Robotics*.
- KIM, Y. J., OTADUY, M. A., LIN, M. C., AND MANOCHA, D. 2003. Six-degree-of-freedom haptic rendering using incremental and localized computations. *Presence* 12, 3, 277–295.
- KLATZKY, R. L., AND LEDERMAN, S. J. 1995. Identifying objects from a haptic glance. *Perception and Psychophysics* 57, pp. 1111–1123.
- KLATZKY, R. L., AND LEDERMAN, S. J. 1999. Tactile roughness perception with a rigid link interposed between skin and surface. *Perception and Psychophysics* 61, pp. 591–607.

- KLATZKY, R. L., AND LEDERMAN, S. J. 2002. Perceiving texture through a probe. In *Touch in Virtual Environments*, M. L. McLaughlin, J. P. Hespanha, and G. S. Sukhatme, Eds. Prentice Hall PTR, Upper Saddle River, NJ, ch. 10, 180–193.
- KLATZKY, R. L., AND LEDERMAN, S. J. 2003. Touch. In *Experimental Psychology*, 147–176. Volume 4 in I.B. Weiner (Editor-in-Chief). Handbook of Psychology.
- KLATZKY, R. L., LEDERMAN, S. J., HAMILTON, C., GRINDLEY, M., AND SWENDSEN, R. H. 2003. Feeling textures through a probe: Effects of probe and surface geometry and exploratory factors. *Perception and Psychophysics* 65(4), pp. 613–631.
- KLOSOWSKI, J., HELD, M., MITCHELL, J., SOWIZRAL, H., AND ZIKAN, K. 1998. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Trans. on Visualization and Computer Graphics* 4, 1, 21–37.
- KUHNAPFEL, U. G., KUHN, C., HUBNER, M., KRUMM, H.-G., MAASS, H., AND NEISIUS, B. 1997. The karlsruhe endoscopic surgery trainer as an example for virtual reality in medical education. *Minimally Invasive Therapy and Allied Technologies Vol. 6*, 122–125.
- LAMOTTE, R. H., AND SRINIVASAN, M. A. 1991. Surface microgeometry: Tactile perception and neural encoding. In *Information Processing in the Somatosensory System*, O. Franzen and J. Westman, Eds. Macmillan Press, London, 49–58.
- LARSEN, E., GOTTSCHALK, S., LIN, M., AND MANOCHA, D. 2000. Distance queries with rectangular swept sphere volumes. *Proc. of IEEE Int. Conference on Robotics and Automation*.
- LARSEN, E. 2001. A robot soccer simulator: A case study for rigid body contact. *Game Developers Conference*.
- LEDERMAN, S. J., KLATZKY, R. L., HAMILTON, C., AND RAMSAY, G. I. 1999. Perceiving roughness via a rigid stylus: Psychophysical effects of exploration speed and mode of touch. *Haptics-e*.
- LEDERMAN, S. J., KLATZKY, R. L., HAMILTON, C., AND GRINDLEY, M. 2000. Perceiving surface roughness through a probe: Effects of applied force and probe diameter. *Proceedings of the ASME DSCD-IMECE*.
- LEDERMAN, S. J. 1974. Tactile roughness of grooved surfaces: The touching process and the effects of macro- and microsurface structure. *Perception and Psychophysics* 16, pp. 385–395.
- LEMKE, C. E. 1965. Bimatrix equilibrium points and mathematical programming. *Management Science* 11, 681–689.
- LIN, M. C., AND CANNY, J. F. 1991. Efficient algorithms for incremental distance computation. In *Proc. IEEE Internat. Conf. Robot. Autom.*, vol. 2, 1008–1014.
- LIN, M., AND GOTTSCHALK, S. 1998. Collision detection between geometric models: A survey. *Proc. of IMA Conference on Mathematics of Surfaces*.
- LIN, M. C., AND MANOCHA, D. 2004. Collision and proximity queries. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, J. E. Goodman and J. O’Rourke, Eds. CRC Press LLC, Boca Raton, FL, ch. 35, 787–807.
- LIN, M. 1993. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley.

- LOMBARDO, J. C., CANI, M.-P., AND NEYRET, F. 1999. Real-time collision detection for virtual surgery. *Proc. of Computer Animation*.
- LÖTSTEDT, P. 1984. Numerical simulation of time-dependent contact friction problems in rigid body mechanics. *SIAM Journal of Scientific Statistical Computing* 5, 370–393.
- LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. 1997. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph.* 16, 1 (Jan.), 34–73.
- LUEBKE, D., AND ERIKSON, C. 1997. View-dependent simplification of arbitrary polygon environments. In *Proc. of ACM SIGGRAPH*.
- LUEBKE, D., REDDY, M., COHEN, J., VARSHNEY, A., WATSON, B., AND HUEBNER, R. 2002. *Level of Detail for 3D Graphics*. Morgan-Kaufmann.
- MARK, W., RANDOLPH, S., FINCH, M., VAN VERTH, J., AND TAYLOR II, R. M. 1996. Adding force feedback to graphics systems: Issues and solutions. In *SIGGRAPH 96 Conference Proceedings*, H. Rushmeier, Ed., Annual Conference Series, 447–452.
- MASSIE, T. M., AND SALISBURY, J. K. 1994. The phantom haptic interface: A device for probing virtual objects. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems 1*, 295–301.
- MCDONNELL, K., QIN, H., AND WLODARCZYK, R. 2001. Virtual clay: A real-time sculpting system with haptic interface. *Proc. of ACM Symposium on Interactive 3D Graphics*, 179–190.
- MCLAUGHLIN, M., HESPANHA, J. P., AND SUKHATME, G. S. 2002. *Touch in Virtual Environments*. Prentice Hall.
- MCNEELY, W., PUTERBAUGH, K., AND TROY, J. 1999. Six degree-of-freedom haptic rendering using voxel sampling. *Proc. of ACM SIGGRAPH*, 401–408.
- MILENKOVIC, V. J., AND SCHMIDL, H. 2001. Optimization-based animation. *SIGGRAPH 01 Conference Proceedings*, 37–46.
- MILLER, B. E., COLGATE, J. E., AND FREEMAN, R. A. 1990. Guaranteed stability of haptic systems with nonlinear virtual environments. *IEEE Transactions on Robotics and Automation* 16, 6, 712–719.
- MINSKY, M., OUH-YOUNG, M., STEELE, O., BROOKS, JR., F. P., AND BEHENSKY, M. 1990. Feeling and seeing: Issues in force display. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, R. Riesenfeld and C. Sequin, Eds., vol. 24, 235–243.
- MINSKY, M. 1995. *Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display*. PhD thesis, Ph.D. Dissertation, Program in Media Arts and Sciences, MIT. Thesis work done at UNC-CH Computer Science.
- MIRTICH, B., AND CANNY, J. 1995. Impulse-based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, ACM Press.
- MIRTICH, B. V. 1996. *Impulse-based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California, Berkeley.
- MIRTICH, B. V. 1998. Rigid body contact: Collision detection to force computation. Tech. Rep. TR98-01, Mitsubishi Electric Research Laboratory.



- MIRTICH, B. 1998. V-Clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics* 17, 3 (July), 177–208.
- MIRTICH, B. 2000. Timewarp rigid body simulation. *SIGGRAPH 00 Conference Proceedings*, 193–200.
- MOORE, M., AND WILHELMS, J. 1988. Collision detection and response for computer animation. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, J. Dill, Ed., vol. 22, 289–298.
- NELSON, D. D., JOHNSON, D. E., AND COHEN, E. 1999. Haptic rendering of surface-to-surface sculpted model interaction. *Proc. of ASME Dynamic Systems and Control Division*.
- OKAMURA, A., AND CUTKOSKY, M. 1999. Haptic exploration of fine surface features. *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2930–2936.
- OKAMURA, A. M., AND CUTKOSKY, M. R. 2001. Feature detection for haptic exploration with robotic fingers. *International Journal of Robotics Research* 20, 12, 925–938.
- O’SULLIVAN, C., AND DINGLIANA, J. 2001. Collisions and perception. *ACM Trans. on Graphics* 20, 3, pp. 151–168.
- O’SULLIVAN, C., RADACH, R., AND COLLINS, S. 1999. A model of collision perception for real-time animation. *Computer Animation and Simulation*, pp. 67–76.
- O’SULLIVAN, C., DINGLIANA, J., GIANG, T., AND KAISER, M. K. 2003. Evaluating the visual fidelity of physically based animations. *Proc. of ACM SIGGRAPH*, 527–536.
- OTADUY, M. A., AND LIN, M. C. 2003. CLODs: Dual hierarchies for multiresolution collision detection. *Eurographics Symposium on Geometry Processing*, 94–101.
- OTADUY, M. A., AND LIN, M. C. 2003. Sensation preserving simplification for haptic rendering. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*, 543–553.
- OTADUY, M. A., AND LIN, M. C. 2005. Stable and responsive six-degree-of-freedom haptic manipulation using implicit integration. *Proc. of World Haptics Conference*.
- OTADUY, M. A., JAIN, N., SUD, A., AND LIN, M. C. 2004. Haptic display of interaction between textured models. *Proc. of IEEE Visualization*, 297–304.
- OUH-YOUNG, M. 1990. *Force Display in Molecular Docking*. PhD thesis, University of North Carolina, Computer Science Department.
- PAI, D. K., AND REISSEL, L. M. 1997. Haptic interaction with multiresolution image curves. *Computer and Graphics* 21, 405–411.
- PAI, D. K., VAN DEN DOEL, K., JAMES, D. L., LANG, J., LLOYD, J. E., RICHMOND, J. L., AND YAU, S. H. 2001. Scanning physical interaction behavior of 3d objects. *Computer Graphics (Proc. of ACM SIGGRAPH)*.
- QUINLAN, S. 1994. Efficient distance computation between non-convex objects. In *Proceedings of International Conference on Robotics and Automation*, 3324–3329.
- REDON, S., KHEDDAR, A., AND COQUILLART, S. 2002. Fast continuous collision detection between rigid bodies. *Proc. of Eurographics (Computer Graphics Forum)*.

- RENZ, M., PREUSCHE, C., PÖTKE, M., KRIEGEL, H.-P., AND HIRZINGER, G. 2001. Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm. *Eurohaptics Conference*.
- ROCK, I., AND VICTOR, J. 1964. Vision and touch: An experimentally created conflict between the two senses. *Science* 143, pp. 594–596.
- RUSPINI, D., AND KHATIB, O. 2000. A framework for multi-contact multi-body dynamic simulation and haptic display. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- RUSPINI, D., KOLAROV, K., AND KHATIB, O. 1997. The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH*, 345–352.
- SALCUDEAN, S. E., AND VLAAR, T. D. 1994. On the emulation of stiff walls and static friction with a magnetically levitated input/output device. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*.
- SALISBURY, K., BROCK, D., MASSIE, T., SWARUP, N., AND ZILLES, C. 1995. Haptic rendering: Programming touch interaction with virtual objects. In *1995 Symposium on Interactive 3D Graphics*, P. Hanrahan and J. Winget, Eds., ACM SIGGRAPH, 123–130. ISBN 0-89791-736-7.
- SEIDEL, R. 1990. Linear programming and convex hulls made easy. In *Proc. 6th Ann. ACM Conf. on Computational Geometry*, 211–215.
- SHIMOGA, K. 1992. Finger force and touch feedback issues in dextrous manipulation. *NASA-CIRSSE International Conference on Intelligent Robotic Systems for Space Exploration*.
- SIIRA, J., AND PAI, D. K. 1996. Haptic textures – a stochastic approach. *Proc. of IEEE International Conference on Robotics and Automation*, 557–562.
- SLATER, M., AND USOH, M. 1993. An experimental exploration of presence in virtual environments. Tech. Rep. 689, Department of Computer Science, University College London.
- SPENCE, C., PAVANI, F., AND DRIVER, J. 2000. Crossmodal links between vision and touch in covert endogenous spatial attention. *Journal of Experimental Psychology: Human Perception and Performance* 26, pp. 1298–1319.
- STEWART, D. E., AND TRINKLE, J. C. 1996. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering* 39, 2673–2691.
- STEWART, D. E., AND TRINKLE, J. C. 2000. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *IEEE International Conference on Robotics and Automation*, 162–169.
- STOLLNITZ, E., DEROSE, T., AND SALESIN, D. 1996. *Wavelets for Computer Graphics: Theory and Applications*. Morgan-Kaufmann.
- SUTHERLAND, I. 1965. The ultimate display. *Proc. of IFIP*, 506–508.
- TAYLOR, R. M., ROBINETT, W., CHII, V., BROOKS, F., AND WRIGHT, W. 1993. The nanomanipulator: A virtual-reality interface for a scanning tunneling microscope. In *Proc. of ACM SIGGRAPH*, 127–134.
- THOMPSON, T. V., JOHNSON, D., AND COHEN, E. 1997. Direct haptic rendering of sculptured models. *Proc. of ACM Symposium on Interactive 3D Graphics*, pp. 167–176.

- UNGER, B. J., NICOLAIDIS, A., BERKELMAN, P. J., THOMPSON, A., LEDERMAN, S. J., KLATZKY, R. L., AND HOLLIS, R. L. 2002. Virtual peg-in-hole performance using a 6-dof magnetic levitation haptic device: Comparison with real forces and with visual guidance alone. *Proc. of Haptics Symposium*, 263–270.
- VAN DEN BERGEN, G. 2001. Proximity queries and penetration depth computation on 3d game objects. *Game Developers Conference*.
- WAN, M., AND MCNEELY, W. A. 2003. Quasi-static approximation for 6 degrees-of-freedom haptic rendering. *Proc. of IEEE Visualization*, 257–262.
- WU, D. 2000. Penalty methods for contact resolution. *Game Developers Conference*.
- ZILLES, C., AND SALISBURY, K. 1995. A constraint-based god object method for haptics display. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems*.
- ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. *Proc. of ACM SIGGRAPH*, 259–268.

# A Framework for Fast and Accurate Collision Detection for Haptic Interaction

Arthur Gregory

Ming C. Lin

Stefan Gottschalk

Russell Taylor

Department of Computer Science

University of North Carolina

Chapel Hill, NC 27599-3175

{gregory,lin,stefan,taylor}@cs.unc.edu

## Abstract

*We present a framework for fast and accurate collision detection for haptic interaction with polygonal models. Given a model, we pre-compute a hybrid hierarchical representation, consisting of uniform grids and trees of tight-fitting oriented bounding box trees (OBB-Trees). At run time, we use hybrid hierarchical representations and exploit frame-to-frame coherence for fast proximity queries. We describe a new overlap test, which is specialized for intersection of a line segment with an oriented bounding box for haptic simulation and takes 6-36 operations excluding transformation costs. The algorithms have been implemented as part of H-COLLIDE and interfaced with a PHANTOM arm and its haptic toolkit, GHOST, and applied to a number of models. As compared to the commercial implementation, we are able to achieve up to 20 times speedup in our experiments and sustain update rates over 1000Hz on a 400MHz Pentium II.*

## 1 Introduction

Virtual environments require natural interaction between interactive computer systems and users. Compared to the presentation of visual and auditory information, methods for haptic display are not as well developed. Haptic rendering as an augmentation to visual display can improve perception and understanding both of force fields and of world models populated in the synthetic environments [6]. It allows users to reach into virtual worlds with a sense of touch, so they can feel and manipulate simulated objects.

Haptic display is often rendered through what is essentially a small robot arm, used in reverse. Such devices are now commercially available for a variety of

configurations (2D, 3D, 6D, specialized for laparoscopy or general-purpose). The system used in this work was a 6DOF-in/3DOF-out SensAble Technologies PHANTOM arm.

“Real-time” graphics applications have display update requirements somewhere between 20 and 30 frames/second. In contrast, the update rate of haptic simulations must be as high as 1000 updates/second in order to maintain a stable system. This rate varies with the spatial frequency and stiffness of displayed forces, and with the speed of motion of the user. Also, the skin is sensitive to vibrations of greater than 500 Hz, so changes in force at even relatively high frequencies are detectable [10].

In order to create a sense of touch between the user’s hand and a virtual object, contact or restoring forces are generated to prevent penetration into this virtual object. This is computed by first detecting if a collision or penetration has occurred, then determining the (projected) contact point on the object surface. Most of the existing algorithms are only sufficient to address the collision detection and contact determination problems for relatively small models consisting of only a few thousand polygons or a few surfaces. Our ultimate goal is to be able to achieve smooth, realistic haptic interaction with CAD models of high complexity (normally consisted of tens of thousands of primitives) for virtual prototyping applications. In addition, we are aiming at designing algorithms that are easily extensible to support a wide range of force-feedback devices (including 6 degree-of-freedom arms) and deformable surfaces.

**Main Contribution:** In this paper we present a framework for fast and accurate collision detection for haptic interaction. It consists of a number of algorithms and a system specialized for computing contact(s) between the probe of the force-feedback device and objects in the virtual environment. To meet the

stringent performance requirements for haptic interaction, we use a *hybrid* approach that specializes many earlier algorithms for this application. Our framework utilizes:

- **Spatial Decomposition:** It decomposes the workspace into uniform grids or cells, implemented as a hash table to efficiently deal with large storage requirements. At runtime, the algorithm can quickly find the cell containing the path swept out by the probe.
- **Bounding Volume Hierarchy based on OBBTrees:** An OBBTree is a bounding volume hierarchy [15] of tight-fitting oriented bounding boxes (OBBs). For each cell consisting of a subset of polygons of the virtual model, we pre-compute an OBBTree. At run-time, most of the computation time is spent in finding collisions between an OBBTree and the path swept out by the tip of the probe between two successive time steps. To optimize this query, we have developed a very fast specialized overlap test between a line segment and an OBB, that takes as few as 6 operations and only 36 arithmetic operations in the worst case, not including the cost of transformation.
- **Frame-to-Frame Coherence:** Typically, there is little movement in the probe position between successive steps. The algorithm utilizes this coherence by caching the contact information from the previous step to perform incremental computations.

The algorithm pre-computes a hybrid hierarchy. Our framework also allows the application program to select only a subset of the approaches listed above.

We have successfully implemented all the algorithms described above, interfaced them with *GHOST* (a commercial haptic library) [34] and used them to find surface contact points between the probe of a PHANTOM arm and large geometric models (composed of tens of thousands of polygons). Their performance varies based on the geometric model, the configuration of the probe relative to the model, machine configuration (e.g. cache and memory size) and the combination of techniques used by our system. The overall approach results in a factor of 2 – 20 speed improvement as compared to earlier algorithms and commercial implementations. For a number of models composed of 5,000 – 80,000 polygons, our system is able to sustain a KHz update rate on a 400M Hz PC.

The results presented in this paper are specialized for a point probe against 3D object collision detection. We conjecture that it can be extended to compute

object-object intersection for a six-degree-of-freedom haptic device.

**Organization:** The rest of the paper is organized in the following manner. Section 2 provides a brief survey of related research. Section 3 describes the system architecture and algorithms used in the design of our system. We discuss the implementation issues in Section 4, present our experimental results and compare their performance with a commercial implementation in Section 5.

## 2 Related Work

Collision detection and contact determination are well-studied problems in computer graphics, computational geometry, robotics and virtual environments. Due to limited space, we refer the readers to [22, 17] for recent surveys. In the ray-tracing literature, the problem of computing fast intersections between a ray and a three-dimensional geometric model has also been extensively studied [1]. While a number of algorithms have been proposed that make use of bounding volume hierarchies, spatial partitioning or frame-to-frame coherence, there is relatively little available on hybrid approaches combining two or more such techniques.

**Bounding Volume Hierarchies:** A number of algorithms based on hierarchical representations have been proposed. The set of bounding volumes include spheres [18, 30], axis-aligned bounding boxes [5, 17], oriented bounding boxes [15, 4], approximation hierarchies based on S-bounds [7], spherical shells [21] and k-dop's [20]. In the close proximity scenarios, hierarchies of oriented bounding boxes (OBBTrees) appear superior to many other bounding volumes [15].

**Spatial Partitioning Approaches:** Some of the simplest algorithms for collision detection are based on spatial decomposition techniques. These algorithms partition the space into uniform or adaptive grids (i.e. volumetric approaches), octrees [33], k-D trees or BSP's [27]. To overcome the problem of large memory requirements for volumetric approaches, some authors [29] have proposed the use of hash tables.

**Utilizing Frame-to-Frame Coherence:** In many simulations, the objects move only a little between successive frames. Many efficient algorithms that utilize frame-to-frame coherence have been proposed for convex polytopes [23, 8, 3]. Cohen et al. [9] have used coherence-based incremental sorting to detect possible pairs of overlapping objects in large environments.

**Research in Haptic Rendering:** Several techniques have been proposed for integrating force feedback with a complete real-time virtual environment to enhance the user's ability to perform interaction tasks

[10, 11, 12, 24, 25, 28, 35]. The commercial haptic toolkit developed by SensAble Technologies, Inc. also has a collision detection library probably using BSP-Trees [32, 34]. Ruspini et al. [31] have presented a haptic interface library “HL” that uses a multi-level control system to effectively simulate contacts with virtual environments. It uses a bounding volume hierarchy based on sphere-trees [30].

Nahvi et al. [26] have designed a haptic display system for manipulating virtual mechanisms derived from a mechanical CAD design. It uses the Sarcos Dexterous Arm Master and Utah’s Alpha\_1 CAD system, with algorithmic support from a tracing algorithm and a minimum distance framework developed by Johnson, Cohen et al. [36, 19]. They utilized a variety of algorithmic toolkits from RAPID [15] to build an OBBTree for each object, Gilbert’s algorithm [14] to find distances between two OBB’s and a tracing algorithm for parametric surfaces. Their system takes about 20 – 150 milliseconds for models composed of 500 – 23,000 triangles on an SGI Indigo2 and 4 milliseconds for models composed of 3 parametric surfaces on a Motorola 68040 microprocessors.

Gibson [13] and Sobierajski [2] have proposed algorithms for object manipulation including haptic interaction with volumetric objects and physically-realistic modeling of object interactions.

### 3 Fast Proximity Queries for Haptic Interaction

In this section, we describe the haptic system setup and algorithmic techniques that are an integral part of the collision detection system framework for haptic interaction, H-COLLIDE.

#### 3.1 Haptic System Architecture

Due to the stringent update requirements for real-time haptic display, we run a special stand-alone haptic server written with the VRPN library (<http://www.cs.unc.edu/Research/nano/manual/vrpn>) on a PC connected to the PHANToM. The client application runs on another machine, which is typically the host for graphical display. Through VRPN, the client application sends the server the description of the scene to be haptically displayed, and the server sends back information such as the position and orientation of the PHANToM probe. The client application can also modify and transform the scene being displayed by the haptic server.

#### 3.2 Algorithm Overview

Given the last and current positions of the PHANToM probe, we need to determine if it has in fact passed through the object’s surface, in order to display the appropriate force. The probe movement is usually small due to the high haptic update rates. This implies that we only need to check a relatively small volume of the workspace for collision detection.

Approaches using spatial partitioning seem to be natural candidates for such situations. For large and complex models, techniques based on uniform or adaptive grids can be implemented more efficiently using hash tables. However, to achieve the desired speed, these approaches still have extremely high storage requirements even when implemented using a hashing scheme.

Despite its better fit to the underlying geometry, the hierarchical bounding volume method based on OBB-Trees may end up traversing trees to great depths to locate the exact contact points for large, complex models. To take advantage of each approach and to avoid some deficiency of each, we propose a hybrid technique.

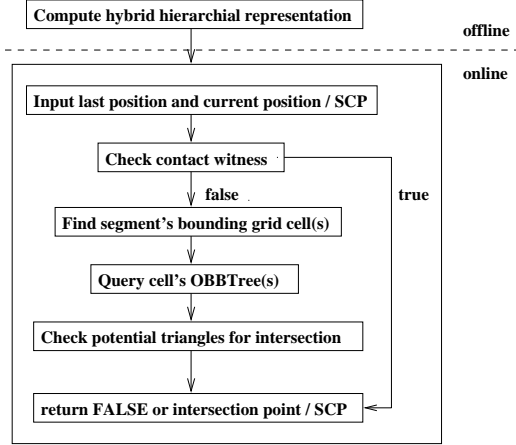
**Hybrid Hierarchical Representation:** Given a virtual environment containing several objects, each composed of tens of thousands of polygons, the algorithm computes a *hybrid hierarchical representation* of the objects as part of the off-line pre-computation. It first partitions the entire virtual workspace into coarse-grain uniform grid cells. Then, for each grid cell containing some primitives of the objects in the virtual world, it computes the OBBTrees for that grid cell and stores the pointer to the associated OBBTrees using a hash table for constant-time proximity queries.

**Specialized Intersection Tests:** The on-line computation of our collision detection system consists of three phases. In the first phase, it identifies “the region of potential contacts” by determining which cells were touched by the probe path, using the precomputed look-up table. In the second phase, it traverses the OBBTree(s) in that cell to determine if collisions have occurred, using the specialized fast overlap test to be described later. In the third phase, if the line segment intersects with an OBB in the leaf node, then it computes the (projected) surface contact point(s) (SCP) using techniques similar to those in [34, 36].

**Frame-to-Frame Coherence:** If in the previous frame the probe of the feedback device was in contact with the surface of the model, we exploit *frame-to-frame coherence* by first checking if the last intersected triangle is still in contact with the probe. If so, we cache this contact witness. Otherwise, we check for collision using hybrid hierarchical representation of the objects.

### 3.3 H-COLLIDE

H-COLLIDE, a framework for fast and accurate collision detection for haptic interaction, is designed based on the hybrid hierarchical representation and the algorithmic techniques described above. Figure 1 shows the system architecture of H-COLLIDE.



**Figure 1.** The System Architecture of H-COLLIDE

### 3.4 Overlap Test based on a Line Segment against an OBBTree

For haptic display using a point probe, we can specialize the algorithm based on OBBTrees by only testing a line segment (representing the path swept out by the probe device between two successive steps) and an OBBTree. (The original algorithm [15] uses a overlap test between a pair of OBBs and can take more than 200 operations per test.) At run time, most of the computation is spent in finding collisions between a line segment and an OBB. To optimize this query, we have developed a very fast overlap test between a line segment and an OBB, that takes as few as 6 operations and only 36 arithmetic operations in the worst case, not including the cost of transformation.

At the first glance, it is tempting to use sophisticated and optimized line clipping algorithms. However, the line-OBB intersection problem for haptic interaction is a simpler one than line clipping and the environment is dynamic and consisting of many OBBs. Next we'll describe this specialized overlap test between a line segment and an oriented bounding box for haptic rendering. Without loss of generality, we will choose the coordinate system centered on and aligned with the box – so the problem is transformed to an overlap test between a segment and a centered axis-aligned bounding box. Our overlap test uses the Separating-

Axis Theorem described in [15], but specialized for a line segment against an OBB.

Specifically, the candidate axes are the three box face normals (which are aligned with the coordinate axes) and their cross-products with the segment's direction vector. With each of these six candidate axes, we project both the box and the segment onto it, and test whether the projection intervals overlap. If the projections are disjoint for any of the six candidate axes, then the segment and the box are disjoint. Otherwise, the segment and the box overlap.

How are the projection intervals computed? Given a direction vector  $v$  of a line through the origin, and a point  $p$ , let the point  $p'$  be the axial projection of  $p$  onto the line. The value  $d_p = v \cdot p / |v|$  is the signed distance of  $p'$  from the origin along the line. Now consider the line segment with midpoint  $m$  and endpoints  $m + w$  and  $m - w$ . The half-length of the line segment is  $|w|$ . The image of the segment under axial projection is the interval centered at

$$d_s = v \cdot m / |v|$$

and with half-length

$$L_s = |w \cdot v| / |v|$$

Given a box centered at the origin, the image of the box under axial projection is an interval with midpoint at the origin.

Furthermore, if the box has thicknesses  $2t^x$ ,  $2t^y$ , and  $2t^z$  along the orthogonal unit directions  $u^x$ ,  $u^y$ , and  $u^z$ , the half-length of the interval is given by

$$L_b = |t^x v \cdot u^x| / |v| + |t^y v \cdot u^y| / |v| + |t^z v \cdot u^z| / |v|$$

With the intervals so expressed, the axis  $v$  is a separating axis if and only if (see Figure 2)

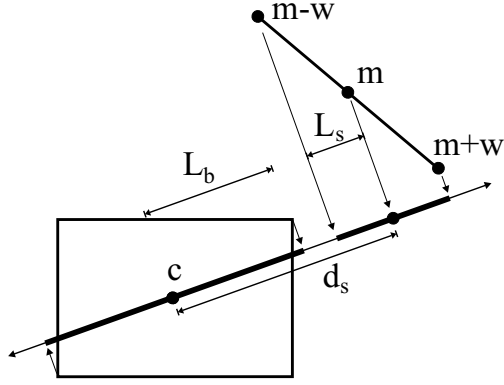
$$|d_s| > L_b + L_s$$

If we assume that the box is axis-aligned, then  $u^x = [1, 0, 0]^T$ ,  $u^y = [0, 1, 0]^T$ , and  $u^z = [0, 0, 1]^T$ , and the dot products with these vectors become simple component selections. This simplifies the box interval length computation to

$$L_b = |t^x v_x| + |t^y v_y| + |t^z v_z|$$

Now, recall that the candidate axis  $v$  is either a box face normal, or a cross product of a face normal with the line segment direction vector. Consider the former case, when  $v$  is a box face normal, for example  $[1, 0, 0]^T$ . In this case, the components  $v_y$  and  $v_z$  are zero, and the component  $v_x$  is one, and we are left with

$$L_b = t^x$$



**Figure 2.** Overlap test between a line segment and an OBB

The projection of the line segment onto the  $x$ -axis is also simple:

$$L_s = |w_x|$$

So, the test for the  $v = [1, 0, 0]^T$  axis is

$$|m_x| > t^x + |w_x|$$

The tests for the candidate axes  $v = [0, 1, 0]^T$  and  $v = [0, 0, 1]^T$  have similar structure.

The three cases where  $v$  is a cross product of  $w$  with one of the box faces are a little more complex. Recall that in general,

$$L_b = |t^x v \cdot u^x| + |t^y v \cdot u^y| + |t^z v \cdot u^z|$$

For the sake of concreteness, we will choose  $v = w \times u_y$ . Then this expression becomes

$$L_b = |t^x (w \times u^y) \cdot u^x| + |t^y (w \times u^y) \cdot u^y| + |t^z (w \times u^y) \cdot u^z|$$

Application of the triple product identity

$$(a \times b) \cdot c = (c \times a) \cdot b$$

yields

$$L_b = |t^x (u^y \times u^x) \cdot w| + |t^y (u^y \times u^y) \cdot w| + |t^z (u^y \times u^z) \cdot w|$$

All of these cross products simplify, because the  $u$  vectors are mutually orthogonal,  $u^x \times u^y = u^z$ ,  $u^y \times u^z = u^x$ , and  $u^z \times u^x = u^y$ , so

$$L_b = |t^x (-u^z) \cdot w| + |t^y (0) \cdot w| + |t^z (u^x) \cdot w|$$

And again, using the fact that  $u^x = [1, 0, 0]^T$ , and so forth,

$$L_b = t^x |w_z| + t^z |w_x|$$

The half-length of the segment interval is

$$L_s = |w \cdot (w \times u^y)| = |u^y \cdot (w \times w)| = |u^y \cdot 0| = 0$$

which is what we would expect, since we are projecting the segment onto a line orthogonal to it.

Finally, the projection of the segments midpoint falls at

$$d_s = (w \times u^y) \cdot m = (m \times w) \cdot u^y = m_z w_x - m_x w_z$$

which is just the  $y$ -component of  $m \times w$ . The final test is

$$|m_z w_x - m_x w_z| > t^x |w_z| + t^z |w_x|$$

Similar derivations are possible for the cases  $v = w \times u^x$  and  $v = w \times u^z$ .

Writing out the entire procedure, and precomputing a few common subexpressions, we have the following pseudo code:

```

let   $X = |w_x|$ 
let   $Y = |w_y|$ 
let   $Z = |w_z|$ 
if  $|m_x| > X + t_x$  return disjoint
if  $|m_y| > Y + t_y$  return disjoint
if  $|m_z| > Z + t_z$  return disjoint
if  $|m_y w_z - m_z w_y| > t_y Z + t_z Y$  return disjoint
if  $|m_x w_z - m_z w_x| > t_x Z + t_z X$  return disjoint
if  $|m_x w_y - m_y w_x| > t_x Y + t_y X$  return disjoint
otherwise return overlap

```

When a segment and an OBB are disjoint, the routine often encounters an early exit and only one (or two) out of the six expressions is executed. Total operation count for the worst case is: 9 absolute values, 6 comparisons, 9 add and subtracts, 12 multiplies. This does not include the cost of transforming, i.e. 36 operations, the problem into a coordinate system centered and aligned with the box.

## 4 Implementation Issues

H-COLLIDE has been successfully implemented in C++. We have interfaced H-COLLIDE with *GHOST*, a commercial software developer's toolkit for haptic rendering, and used it to find surface contact points between the probe of a PHANTOM arm and large geometric models (composed of tens of thousands of polygons). Here we describe some of the implementation issues.



## 4.1 Hashing Scheme

Clearly it is extremely inefficient to allocate storage for all these cells, since a polygonal surface is most likely to occupy a very small fraction of them. We use a hash table to alleviate the storage problem. From each cell location at  $(x, y, z)$  and a grid that has  $len$  cells in each dimension, we can compute a unique key using

$$key = x + y * len + z * len^2.$$

In order to avoid hashing too many cells with same pattern into the same table location we compute the actual location for a grid cell in the hash table with

$$TableLoc = random(key) \% TableLength.$$

Should the table have too many cells in one table location, we can simply grow the table. Hence, it is possible to determine which triangles we need to check in constant time and the amount of storage required is a constant factor (based on the grid grain) of the surface area of the object we want to “feel”.

Determining the optimal grid grain is a non-trivial problem. Please refer to [16] for a detailed retreatment and a possible analytical solution to this problem. We simply set the grain of the grids to be the average length of all edges. If the model has a very irregular triangulation it is very possible that there could be a large number of small triangles in a single grid cell.

Querying an OBBTree takes  $O(\log n)$  time, where  $n$  is the number of triangles in the tree. During the off-line computation, we can ensure that  $n$  is a small number compared to the total number of triangles in the model; thus the overall running time of our hybrid approach should be constant.

## 4.2 User Options

Since the hybrid approach used in H-COLLIDE has a higher storage requirement than either the individual technique alone, the system also allows the user to select a subset of the techniques, such as the algorithm purely based on OBBTrees, to opt for better performance on a machine with less memory.

## 5 System Performance

For comparison, we have implemented adaptive grids, our hybrid approach and an algorithm using only OBBTrees and the specialized overlap test described in Section 3.4. We have applied them to a wide range of models of varying sizes. (Due to the page limit, we invite the readers to view the Color Plates of these models at <http://www.cs.unc.edu/~geom/HCollide/model.pdf>.)

Their performance varies based on the models, the configuration of the probe relative to the model, machine configuration (e.g. cache and memory size) and the combination of techniques used by our system. Our hybrid approach results in a factor of 2-20 speed improvement as compared to a native *GHOST* method. For a number of models composed of 5,000 – 80,000 polygons, our system is able to compute all the contacts and response at rates higher than 1000 Hz on a 400MHz PC.

## 5.1 Obtaining Test Data

We first obtained the test data set by deriving a class from the triangle mesh primitive which comes with SensAble Technologies’ *GHOST* library, version 2.0 beta. This records the start and the endpoint of each segment used for collision detection during a real force-feedback session with a 3-DOF PHANTOM arm. We then implemented the three techniques mentioned above to interface with *GHOST* for comparison with a native *GHOST* method, and timed the collision detection routines for the different libraries using the data from the test set. The test set for each of these models contains 30,000 readings.

The distinction between a collision and an intersection shown in the tables is particular to *GHOST*’s haptic rendering. Each haptic update cycle contains a “collision” test to see if the line segment from the last position of the PHANTOM probe to its current position has intersected any of the geometry in the haptic scene. If there has been a collision, then the intersected primitive suggests a surface contact point for the PHANTOM probe to move towards. In this case it is now necessary to perform an “intersection” test to determine if the line segment from the last position of the PHANTOM probe to the suggested surface contact point intersects any of the geometry in the scene (including the primitive with which there was a “collision”).

The timings (in milliseconds) shown in Tables 1-5 were obtained by replaying the test data set on a 4 processor 400 MHz PC, with 1 GB of physical memory. Each timing was obtained using only one processor. For comparison, we ran the same suite of tests on a single processor 300 MHz Pentium Pro with 128 MB memory. The hybrid approach appeared to be the most favorable as well.

## 5.2 Comparison between Algorithms

Since the algorithms run on a real-time system, we are not only interested in the average performance, but also the worst case performance. Tables 1-5 show the

Method	Hash Grid	Hybrid	OBBTree	GHOST
Ave Col. Hit	0.0122	0.00883	0.0120	0.0917
Worst Col. Hit	0.157	0.171	0.0800	0.711
Ave Col. Miss	0.00964	0.00789	0.00856	0.0217
Worst Col. Miss	0.0753	0.0583	0.0683	0.663
Ave Int. Hit	0.0434	0.0467	0.0459	0.0668
Worst Int. Hit	0.108	0.102	0.0793	0.100
Ave Int. Miss	0.0330	0.0226	0.0261	0.0245
Worst Int. Miss	0.105	0.141	0.0890	0.364
<b>Ave. Query</b>	0.019	0.014	0.017	0.048

**Table 1.** Timings in msec for Man Symbol, 5K tris

Method	Hash Grid	Hybrid	OBBTree	GHOST
Ave Col. Hit	0.0115	0.0185	0.0109	0.131
Worst Col. Hit	0.142	0.213	0.138	0.622
Ave Col. Miss	0.0104	0.00846	0.0101	0.0176
Worst Col. Miss	0.0800	0.0603	0.0813	0.396
Ave Int. Hit	0.0583	0.0568	0.0652	0.0653
Worst Int. Hit	0.278	0.200	0.125	0.233
Ave Int. Miss	0.0446	0.0237	0.0349	0.0322
Worst Int. Miss	0.152	0.173	0.111	0.287
<b>Ave. Query</b>	0.030	0.025	0.028	0.070

**Table 2.** Timings in msec for Man with Hat, 7K tris

Method	Hash Grid	Hybrid	OBBTree	GHOST
Ave Col. Hit	0.0138	0.0101	0.0134	0.332
Worst Col. Hit	0.125	0.168	0.0663	0.724
Ave Col. Miss	0.00739	0.00508	0.00422	0.0109
Worst Col. Miss	0.0347	0.0377	0.0613	0.210
Ave Int. Hit	0.0428	0.0386	0.0447	0.0851
Worst Int. Hit	0.0877	0.102	0.0690	0.175
Ave Int. Miss	0.0268	0.0197	0.0213	0.0545
Worst Int. Miss	0.0757	0.0697	0.0587	0.284
<b>Ave. Query</b>	0.022	0.016	0.039	0.18

**Table 3.** Timings in msec for Nano Surface, 12K tris

Method	Hash Grid	Hybrid	OBBTree	GHOST
Ave Col. Hit	0.0113	0.00995	0.0125	0.104
Worst Col. Hit	0.136	0.132	0.177	0.495
Ave Col. Miss	0.0133	0.00731	0.0189	0.0280
Worst Col. Miss	0.128	0.0730	0.137	0.641
Ave Int. Hit	0.0566	0.0374	0.609	0.0671
Worst Int. Hit	0.145	0.105	0.170	0.293
Ave Int. Miss	0.0523	0.0225	0.0452	0.0423
Worst Int. Miss	0.132	0.133	0.167	0.556
<b>Ave. Query</b>	0.027	0.014	0.028	0.048

**Table 4.** Timings in msec for Bronco, 18K tris

Method	Hash Grid	Hybrid	OBBTree	GHOST
Ave Col. Hit	0.0232	0.0204	0.0163	1.33
Worst Col. Hit	0.545	0.198	0.100	5.37
Ave Col. Miss	0.00896	0.00405	0.00683	0.160
Worst Col. Miss	0.237	0.139	0.121	3.15
Ave Int. Hit	0.228	0.0659	0.0704	0.509
Worst Int. Hit	0.104	0.138	0.103	1.952
Ave Int. Miss	0.258	0.0279	0.0256	0.229
Worst Int. Miss	0.0544	0.131	0.0977	3.28
<b>Ave. Query</b>	0.030	0.016	0.016	0.320

**Table 5.** Timings in msec for Butterfly, 79K tris

timings in milliseconds obtained for both cases on each model and each contact configuration.

All our algorithms are able to perform collision queries at rates faster than the required 1000 Hz force update rate for *all* models in the worst case. Although the hybrid approach often outperforms the algorithm based on OBBTrees, it is sometimes slightly slower than the algorithm based on OBBTrees. We conjecture that this behavior is due to the cache size of the CPU (independent of the memory size) and memory paging algorithm of the operating system. Among techniques that use hierarchical representations, cache access patterns can often have a dramatic impact on run time performance.

The hybrid approach requires more memory and is likely to have a less cache-friendly memory access pattern than the algorithm purely based on OBBTrees, despite the fact that both were well within the realm of physical memory available to the machine. Furthermore, by partitioning polygons into groups using grids, the hybrid technique can enable real-time local surface modification.

The adaptive grids-hashing scheme, a commonly used technique in ray-tracing, did not perform equally well in all cases. Once again, our hypothesis is that its inferior worst case behavior is due to its cache access patterns, in addition to its storage requirements. We believe the native *GHOST* method uses an algorithm based BSP trees. While it is competitive for the smaller model sizes, its performance fails to scale up for larger models. Our hybrid approach and our algorithm purely based on OBBTrees and the specialized overlap test appear to be relatively unaffected by the model complexity.

## 6 Conclusion

We have presented a framework, H-COLLIDE, that consists of a suite of algorithms and a system implementation for fast and accurate collision detection for haptic interaction with polygonal models at rates higher than 1000Hz on a desk-top PC. This framework may be extended for supporting 6-DOF haptic devices to perform collision tests between a pair of 3D objects and flexible surfaces that may deform due to manipulation. In addition, it can be combined with the tracing algorithm [36] to handle complex sculptured models more efficiently, by using their control points.

**Acknowledgements:** We are grateful to National Science Foundation, National Institute of Health National Center for Research Resources and Intel Corporation for their support and the reviewers for their comments.

# References

- [1] J. Arvo and D. Kirk. A survey of ray tracing acceleration techniques. In *An Introduction to Ray Tracing*, pages 201–262, 1989.
- [2] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. *Proceedings of Visualization'96*, pages 197–204, 1996.
- [3] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *ACM Computer Graphics*, 24(4):19–28, 1990.
- [4] G. Barequet, B. Chazelle, L. Guibas, J. Mitchell, and A. Tal. Boxtree: A hierarchical representation of surfaces in 3d. In *Proc. of Eurographics'96*, 1996.
- [5] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The r\*-tree: An efficient and robust access method for points and rectangles. *Proc. SIGMOD Conf. on Management of Data*, pages 322–331, 1990.
- [6] Frederick P. Brooks, Jr., Ming Ouh-Young, James J. Batter, and P. Jerome Kilpatrick. Project GROPE — Haptic displays for scientific visualization. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 177–185, August 1990.
- [7] S. Cameron. Approximation hierarchies and s-bounds. In *Proceedings. Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 129–137, Austin, TX, 1991.
- [8] Stephen Cameron. A comparison of two fast algorithms for computing the distance between convex polyhedra. *IEEE Transactions on Robotics and Automation*, 13(6):915–920, December 1996.
- [9] J. Cohen, M. Lin, D. Manocha, and M. Ponamgi. I-collide: An interactive and exact collision detection system for large-scale environments. In *Proc. of ACM Interactive 3D Graphics Conference*, pages 189–196, 1995.
- [10] J. E. Colgate and J. M. Brown. Factors affecting the z-width of a haptic display. *IEEE Conference on Robotics and Automation*, pages 3205–3210, 1994.
- [11] J. E. Colgate and et al. Issues in the haptic display of tool use. *Proceedings of the ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 140–144, 1994.
- [12] M. Finch, M. Falvo, V. L. Chi, S. Washburn, R. M. Taylor, and R. Superfine. Surface modification tools in a virtual environment interface to a scanning probe microscope. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 13–18. ACM SIGGRAPH, April 1995.
- [13] S. Gibson. Beyond volume rendering: Visualization, haptic exploration, and physical modeling of element-based objects. In *Proc. Eurographics workshop on Visualization in Scientific Computing*, pages 10–24, 1995.
- [14] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE J. Robotics and Automation*, vol RA-4:193–203, 1988.
- [15] S. Gottschalk, M. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proc. of ACM Siggraph'96*, pages 171–180, 1996.
- [16] A. Gregory, M. Lin, S. Gottschalk, and R. Taylor. H-collide: A framework for fast and accurate collision detection for haptic interaction. Technical report, Department of Computer Science, University of North Carolina, 1998.
- [17] M. Held, J.T. Klosowski, and J.S.B. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. In *Canadian Conference on Computational Geometry*, 1995.
- [18] P. M. Hubbard. Interactive collision detection. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality*, October 1993.
- [19] D. Johnson and E. Cohen. A framework for efficient minimum distance computation. *IEEE Conference on Robotics and Automation*, pages 3678–3683, 1998.
- [20] J. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. In *Siggraph'96 Visual Proceedings*, page 151, 1996.
- [21] S. Krishnan, A. Pattekar, M. Lin, and D. Manocha. Spherical shell: A higher order bounding volume for fast proximity queries. In *Proc. of Third International Workshop on Algorithmic Foundations of Robotics*, pages 122–136, 1998.
- [22] M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, 1998.
- [23] M.C. Lin and John F. Canny. Efficient algorithms for incremental distance computation. In *IEEE Conference on Robotics and Automation*, pages 1008–1014, 1991.
- [24] William Mark, Scott Randolph, Mark Finch, James Van Verth, and Russell M. Taylor II. Adding force feedback to graphics systems: Issues and solutions. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 447–452, 1996.
- [25] T. M. Massie and J. K. Salisbury. The phantom haptic interface: A device for probing virtual objects. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 1:295–301, 1994.
- [26] A. Nahvi, D. Nelson, J. Hollerbach, and D. Johnson. Haptic manipulation of virtual mechanisms from mechanical cad designs. In *Proc. of 1998 Conference on Robotics and Automation*, pages 375–380, 1998.
- [27] B. Naylor, J. Amanatides, and W. Thibault. Merging bsp trees yield polyhedral modeling results. In *Proc. of ACM Siggraph*, pages 115–124, 1990.
- [28] M. Ouh-Young. *Force Display in Molecular Docking*. PhD thesis, University of North Carolina, Computer Science Department, 1990.
- [29] M. H. Overmars. Point location in fat subdivisions. *Inform. Proc. Lett.*, 44:261–265, 1992.
- [30] S. Quinlan. Efficient distance computation between non-convex objects. In *Proceedings of International Conference on Robotics and Automation*, pages 3324–3329, 1994.
- [31] D.C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH*, pages 345–352, 1997.
- [32] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles. Haptic rendering: Programming touch interaction with virtual objects. *Proc. of 1995 ACM Symposium on Interactive 3D Graphics*, pages 123–130, 1995.
- [33] H. Samet. *Spatial Data Structures: Quadtree, Octrees and Other Hierarchical Methods*. Addison Wesley, 1989.
- [34] Inc. SensAble Technologies. *ghost<sup>TM</sup>: Software developer's toolkit. Programmer's Guide*, 1997.
- [35] R. M. Taylor, W. Robinett, V.L. Chii, F. Brooks, and W. Wright. The nanomanipulator: A virtual-reality interface for a scanning tunneling microscope. In *Proc. of ACM Siggraph*, pages 127–134, 1993.
- [36] T.V. Thompson, D. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. *Proc. of ACM Interactive 3D Graphics*, pages 167–176, 1997.



# Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling

William A. McNeely

Kevin D. Puterbaugh

James J. Troy

The Boeing Company\*

## Abstract

A simple, fast, and approximate voxel-based approach to 6-DOF haptic rendering is presented. It can reliably sustain a 1000 Hz haptic refresh rate without resorting to asynchronous physics and haptic rendering loops. It enables the manipulation of a modestly complex rigid object within an arbitrarily complex environment of static rigid objects. It renders a short-range force field surrounding the static objects, which repels the manipulated object and strives to maintain a voxel-scale minimum separation distance that is known to preclude exact surface interpenetration. Force discontinuities arising from the use of a simple penalty force model are mitigated by a dynamic simulation based on virtual coupling. A generalization of octree improves voxel memory efficiency. In a preliminary implementation, a commercially available 6-DOF haptic prototype device is driven at a constant 1000 Hz haptic refresh rate from one dedicated haptic processor, with a separate processor for graphics. This system yields stable and convincing force feedback for a wide range of user controlled motion inside a large, complex virtual environment, with very few surface interpenetration events. This level of performance appears suited to applications such as certain maintenance and assembly task simulations that can tolerate voxel-scale minimum separation distances.

**CR Categories and Subject Descriptors:** H.5.2 [User Interfaces]: Haptic I/O, I.3.5 [Computational Geometry and Object Modeling]: Physically Based Modeling.

**Additional Keywords:** force feedback, voxel representations, virtual environments.

## 1. INTRODUCTION

The problem of simulating real-world engineering tasks — for example, objectives like design-for-assembly and design-for-maintenance — has been exacerbated by the modern transition from physical mockup to virtual mockup. Physical mockup provides natural surface constraints that prevent tools and parts from interpenetrating, whereas virtual mockup requires the user to satisfy such constraints by receiving collision cues and making appropriate body postural adjustments, which is usually tedious and may yield dubious results. In order to emulate the natural surface constraint satisfaction of physical mockup, one must introduce force

feedback into virtual mockup. Doing so shifts the burden of physical constraint satisfaction onto a haptic subsystem, and the user becomes free to concentrate on higher-level problems such as path planning and engineering rule satisfaction.

Tool and part manipulation inherently requires six degree-of-freedom (6-DOF) haptics, since extended objects are free to move in three translational and three rotational directions. Affordable high-bandwidth 6-DOF devices are becoming available, but 6-DOF haptic rendering remains an outstanding problem. It is considerably more difficult than 3-DOF point-contact haptic rendering. One can compare haptics with collision detection, since they share some technical similarity. Real-time collision detection is a challenging problem [13], but 6-DOF haptics adds stringent new requirements such as:

- Detect all surface contact (or proximity, for a force field), instead of stopping at the first evidence of it.
  - Calculate a reaction force and torque at every point or extended region of contact/proximity.
  - Reliably maintain a 1000 Hz refresh rate, independent of position and orientation of the manipulated object.
  - Control geometry driven haptic instabilities, such as forcing an object into a narrow wedge-shaped cavity.
- To address the needs of our targeted engineering applications, we adopt the following additional goals:
- Minimize the interpenetration of exact surface representations.
  - Handle complex static scenes, e.g., those containing several hundred thousand triangles, with reasonable memory efficiency.
  - The haptic rendering algorithm should parallelize easily.

Furthermore, we accept the limitation of voxel-scale accuracy. For example, a common engineering rule is to design at least 0.5 inch clearance into part removal paths, whenever possible, in order to accommodate tool access and human grasping and to serve as a cushion against assembly tolerance buildup.

We describe an approach that formally meets most of these requirements. It demonstrates the ability to drive a commercially available 6-DOF prototype device at a reliable 1000 Hz haptic refresh rate without the aid of asynchronous physics and haptic rendering loops. It supports the manipulation of a single rigid object within an arbitrarily rich environment of static rigid objects by rendering a half-voxel-deep force field that surrounds the static objects and serves to block potential interpenetration of the exact surface representations, as described in section 4.2. Given a predetermined spatial accuracy (i.e., voxel size), rendering performance depends linearly on the total exposed surface area of the manipulated object. There is also a relatively minor dependence on the instantaneous amount of contact/proximity, with a worst-case performance (e.g., maximum contact/proximity) of about half that of the best-case performance.

\* Bill McNeely, The Boeing Company, P.O. Box 3707, M/S 7L-43, Seattle, WA 98124, (bill.mcneely@boeing.com)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGGRAPH 99, Los Angeles, CA USA  
Copyright ACM 1999 0-201-48560-5/99/08...\$5.00

Our approach is distinguished primarily by its high haptic rendering speed, which is derived primarily from:

- A simple penalty force scheme called the tangent-plane force model, explained in section 3.
- A fixed-depth voxel tree, explained in section 4.3.
- A voxel map that collectively represents all static objects, explained in section 4.4.

Although the simplicity of our force model is critically important to performance, it is so simple that it generates force magnitude discontinuities (but not force direction discontinuities), especially under sliding motion. In 3-DOF point-contact haptics, force discontinuities can be devastating to force quality and stability, but under our 6-DOF approach there is a stochastic effect that lessens their impact. However, it proved necessary to introduce various measures to explicitly enhance force quality and stability, such as:

- A single-body dynamic model based on virtual coupling
- Pre-contact braking forces

All such measures are explained in section 5.

Data storage is often a secondary consideration in haptics work, because it is tempting to trade memory efficiency for higher performance. However, voxels are so relatively inefficient as geometric modeling elements that we improve their memory efficiency by generalizing the octree method, as explained in section 4.3.

## 2. PREVIOUS WORK

Although largely the result of unpublished work, there are numerous examples of 6-DOF haptic rendering for scenarios containing a very limited number of geometrically well behaved virtual objects, for example [6,7,24]. Our approach differs from this work primarily in its ability to render considerably more complex 6-DOF scenarios with no formal constraints on object shape, although at reduced accuracy.

Our approach includes a collision detection technique based on probing a voxelized environment with surface point samples. Voxel-based methods have been applied to non-haptic collision detection [12,15,16] and to 3-DOF haptics [3,18]. Sclaroff and Pentland [22] apply surface point sampling to implicit surfaces.

Intermediate representations for haptics were suggested by Adachi et al. [1], and have been subsequently elaborated [17]. This involves using a simple haptics proxy that approximates the exact scene and is simple enough to update the forces at the required high refresh rate, while a slower but more exact collision detection and/or dynamic simulation runs asynchronously and updates the proxy's parameters. Our work differs by tightly integrating collision detection, the force model, and the dynamic model into a single loop that updates forces directly at 1000 Hz.

There has been much work in multibody dynamic simulation for physically based modeling, for example [4,23]. Mirtich and Canny [19] track the contacts found from an iterative collision detection method and use this information to generate constant-size impulses. In general, such work is characterized by its emphasis on accuracy over rendering performance, and consequently it relies on methodology such as exact-surface collision detection and simultaneous surface constraint satisfaction, which currently fall far short of 6-DOF haptics performance requirements.

Our dynamic model adopts the practice of using an artificial coupling between the haptic display and virtual environment, as

originally proposed by Colgate et al. [10] and recently elaborated by Adams and Hannaford [2]. We also adopt a version of the “god object” concept suggested by Zilles and Salisbury [25] and others [21], generalized to 6-DOF and modified to use penalty forces that only approximately satisfy surface constraints. In addition, we use the concept of pre-contact braking force suggested by Clover [9].

Hierarchical techniques, such as employed by Gottschalk [13], can be used to alleviate convex-hull bounding box limitations for objects in very close proximity by recursively generating a tree of bounding volumes around finer features of the object. While this technique speeds collision detection, it also introduces indeterminacy in the cycle rate due to the varying cost of traversing the tree structure to an unknown depth to check each colliding polygon against object polygons. Cycle-rate should not only be fast but should also have a rate that is as constant as possible.

Temporal and spatial coherence can also be exploited [4,5,8] by assuming that objects move only slightly within each time step, thus allowing extrapolation from the previous state of the system. The number of polygon tests carried out at each time step is effectively reduced, increasing cycle-rate at the cost of introducing indeterminacy. With certain configurations or motions of objects, however, there are often noticeable drops in performance — a situation which is unacceptable in a real-time simulation.

## 3. TANGENT-PLANE FORCE MODEL

In our tangent-plane force model, dynamic objects are represented by a set of surface point samples, plus associated inward pointing surface normals, collectively called a point shell. During each haptic update the dynamic object's motion transformation is applied to every point of the point shell. The environment of static objects is collectively represented by a single spatial occupancy map called a voxmap, which is illustrated in Figure 1. Each haptically rendered frame involves sampling the voxmap at every point of the point shell.

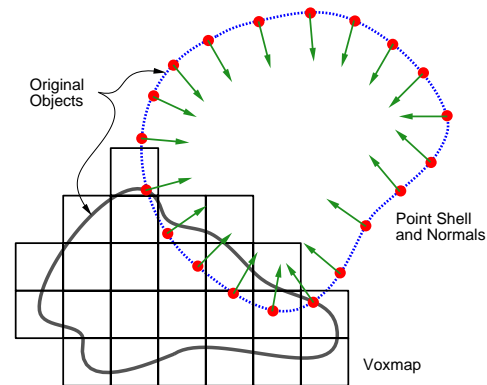
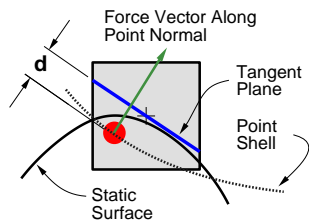


Figure 1. Voxmap colliding with point shell.

When a point interpenetrates a voxel (assumed for now to be a surface voxel) as shown in Figure 2, a depth of interpenetration is calculated as the distance  $d$  from the point to a plane within the voxel called the tangent plane.

The tangent plane is dynamically constructed to pass through the voxel's center point and to have the same normal as the point's associated normal. If the point has not penetrated below that plane (i.e., closer to the interior of the static object), then  $d$  is zero. Force is simply proportional to  $d$  by Hooke's law ( $F = K_{ff}d$ ). We call  $K_{ff}$  the “force field stiffness,” since the voxel represents a half-

voxel-deep force field. The net force and torque acting on the dynamic object is obtained as the sum of all force/torque contributions from such point-voxel intersections.



**Figure 2.** Tangent-plane force model.

The tangent-plane force model was inspired by the fact that the surfaces of contacting objects are tangent at an osculation point. It is important that the force takes its direction from a precomputed surface normal of the dynamic object. This proves to be considerably faster than the common practice of dynamically computing it from the static object's surface, or in the case of a force field, dynamically taking the gradient of a potential field.

One can see that this simple model has discontinuities in force magnitude when a point crosses a voxel boundary, for example, under sliding motion. Section 5 describes how discontinuities can be mitigated for haptic purposes.

#### 4. VOXEL DATA STRUCTURES

This section outlines the creation and usage of voxel-based data structures that are required under our approach. Exact (polygonal) surface penetration and memory usage will also be discussed.

##### 4.1 Voxmap and Point Shell

One begins by selecting a global voxel size,  $s$ , that meets the virtual scenario's requirements for accuracy and performance. The performance aspect is that the force model requires traversing a set of point samples, and  $s$  determines the number of such points. Consider a solid object such as the teapot in Figure 3(a). It partitions space into regions of free space, object surface, and object interior. Now tile this space into a volume occupancy map, or voxmap, as in Figure 3(b). The collection of center points of all surface voxels constitutes the point shell needed by the tangent-plane force model, as in Figure 3(c).



**Figure 3.** Teapot: (a) polygonal model, (b) voxel model, (c) point shell model.

This method for creating the point shell is not optimal, but it is convenient. Its accuracy may be improved by choosing points that lie on the exact geometrical representation.

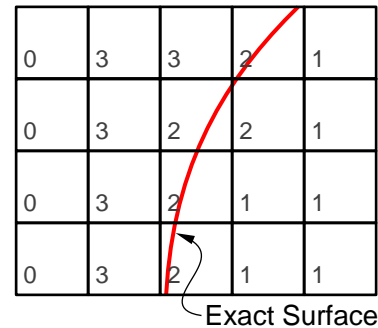
Each voxel is allocated two bits of memory that designate it as a free space, interior, surface, or proximity voxel. The 2-bit voxel types are defined in Table 1 and illustrated by an example in Figure 4.

A neighbor voxel is defined as sharing a vertex, edge, or face with the subject voxel. Each voxel has 26 neighbors. It is important that each static object be voxelized in its final position and ori-

entation in the world frame, because such transformations cause its voxelized representation to change shape slightly.

**Table 1.** Voxel types (2-bit)

Value	Voxel type	Description
0	Free space	Encloses only free-space volumes
1	Interior	Encloses only interior volumes
2	Surface	Encloses a mix of free-space, surface, and interior volumes
3	Proximity	Free-space neighbor of a surface voxel

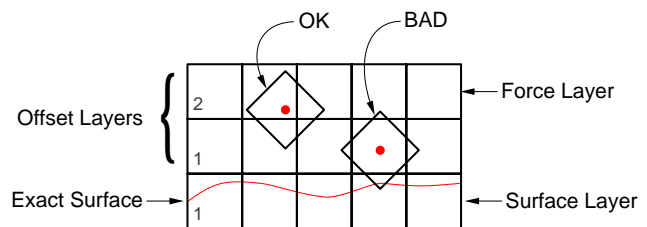


**Figure 4.** Assignment of 2-bit voxel values.

By the nature of 3D scan conversion, voxmaps are insensitive to surface imperfections such as gaps or cracks that are smaller than the voxel width. However, identifying the interior of a voxmap can be difficult. We adopt the practice of (1) scan-converting to create surface voxels, (2) identifying free-space voxels by propagating the voxelized walls of the object's bounding box inward until surface voxels are encountered, and (3) declaring all other voxels to be interior voxels. This ensures that objects with open surfaces will be voxelized instead of "leaking" and filling all voxels.

##### 4.2 Avoiding Exact Surface Interpenetration

In the tangent-plane force model shown in Figure 2, the exact surfaces of colliding objects are allowed to interpenetrate by voxel-scale distances during a point-voxel intersection. While this may be acceptable for some applications, we seek instead to preclude exact-surface interpenetration. We do this by offsetting the force field outward away from the surface by two voxel layers, as shown in Figure 5. (In this figure, the rotated boxes represent the surface voxels associated with the points of a pointshell, viewed as surface bounding volumes.) The offset force layer then serves to maintain a minimum object separation that provably precludes exact-surface interpenetration.



**Figure 5.** Criterion for exact-surface interpenetration.

The voxel legend described by Table 1 and Figure 4 is correspondingly redefined so that “surface” and “value 2” now refer to the offset force-layer voxels instead of geometric surface voxels, and similarly for the other voxel types. (Offset proximity voxels and free-space voxels are omitted from Figure 5, but they would occupy additional layers at the top of the figure.)

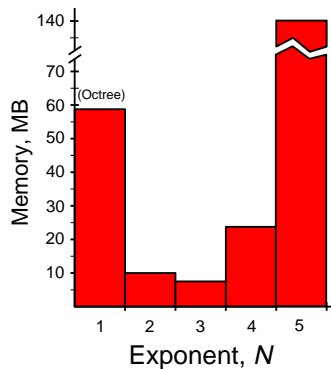
Force-layer offsetting is implemented as a final step of voxelization, in which the geometric surface voxel layer is grown outward by a process of promoting proximity voxels to surface values and demoting original surface voxels to interior values. This process is repeated to achieve the desired two-layer offset. (If voxels were allocated more than two bits, it would not be necessary to “recycle” voxel values in this manner, and there are other advantages to wider voxels that we are beginning to explore.)

Force-layer offsetting also serves to prevent any spike-like feature in the static object from generating a linear column of voxels that the point shell could completely fail to penetrate for certain orientations of the dynamic object. The force layer has no such features, because voxel values are propagated to 26 connected neighbors during the offsetting process.

### 4.3 Voxel Tree

A natural next step is to impose an octree organization on the voxels for the sake of memory efficiency and scalability. However, the need for a consistently fast haptic refresh rate is at odds with the variability in the tree traversal time. To address this, we have devised a hierarchy that represents a compromise between memory efficiency and haptic rendering performance. It is a generalization of octree with a tree depth that is limited to three levels, explained as follows.

At each level of the tree, the cubical volume of space is divided into  $2^{3N}$  sub-volumes, where  $N$  is a positive integer. ( $N$  is unity for an octree.) We have discovered that the most memory-efficient value for  $N$  may be at higher values, depending on the sparseness of the geometry. Figure 6 illustrates a study of the total memory consumed by a  $2^{3N}$ -tree as a function of  $N$  for geometry that is typical to our work. It has a minimum at  $N=3$ , which might be called a 512-tree.



**Figure 6.** Memory usage of  $2^{3N}$  tree as a function of  $N$ .

We further limit tree depth by fixing both the minimum and maximum dimensions of the bounding volumes in the tree. The minimum dimension is the size of voxels at the leaf level, and the maximum dimension is given implicitly by creating only three levels above the leaf level. The minimum-size requirement means that smaller features may not be adequately represented, but we fundamentally accept a global accuracy limitation, analogous to the

practice of accepting a fixed tessellation error in polygonal surface representations. The maximum-size requirement impacts memory efficiency and scalability, because one must cover all remaining space with the largest-size bounding volumes. However, these effects are mitigated by the use of  $2^{3N}$ -tree, since for a fixed number of levels, higher values of  $N$  increase the dynamic range of the bounding volume dimensions.

### 4.4 Merged Scene Voxmap

Our approach is limited to the case of a single dynamic rigid object interacting with an arbitrarily rich environment of static rigid objects. If it were necessary to separately calculate the interaction force for each of  $N$  static objects, then the computing burden would grow linearly with  $N$ . However, there is no inherent need to separately compute such interactions on a pairwise basis. For example, there is no need to identify the type of a contacted object in order to apply different material properties, since all static objects are treated as rigid. Furthermore, under our force-field approach, objects are never actually contacted in the sense of undergoing surface intersections. Therefore, we merge all static-object voxel representations together as if they were a single static object, applying straightforward precedence rules to merged voxel values and recalculating a voxel tree for the voxmap.

## 5. DYNAMIC MODEL

For the dynamic model, we use an impedance approach, in which user motion is sensed and a force/torque pair is produced. We further adopt what is called the “virtual coupler” scheme, which connects the user’s haptic motions with the motions of the dynamic object through a virtual spring and damper. This is a well known method for enhancing haptic stability [2].

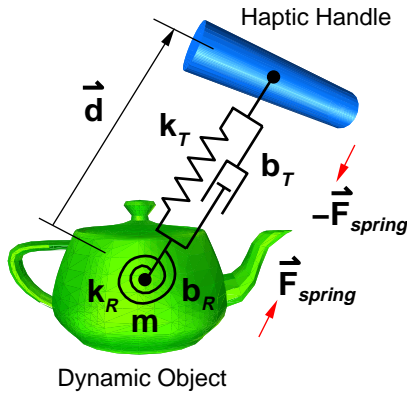
To solve for the motion of the dynamic object, we perform a numerical integration of the Newton-Euler equation, using a constant time step  $\Delta t$  corresponding to the time between force updates, e.g.,  $\Delta t = 1$  msec for 1000 Hz haptic refresh rate. We also must assign a mass  $m$  to the dynamic object equal to the apparent mass for the dynamic object that we want to feel at the haptic handle (in addition to the haptic device’s intrinsic friction and inertia, and assuming that its forces are not yet saturated). The net force and torque on the dynamic object is the sum of contributions from the spring-damper system, explained in section 5.1; stiffness considerations, explained in section 5.2; and the pre-contact braking force, explained in section 5.3.

### 5.1 A 6-DOF Spring-Damper System

Conceptually, a copy of the haptic handle is placed in the virtual scene and is coupled to the dynamic object through a spring-damper connection, as shown in Figure 7.

The real haptic handle controls the position and orientation of its virtual counterpart. This influences the spring’s displacement, which generates a virtual force/torque on the dynamic object and an opposite force/torque on the real haptic handle. Spring displacement also includes rotational motion, as shown in Figure 7 by the spiral at the center of the dynamic object (suggestive of a clock mainspring). Spring force is proportional to displacement, while spring torque is proportional to the angle of rotation from an equivalent-angle analysis and directed along an equivalent axis of rotation [11].





**Figure 7.** Dynamic model based on virtual coupling.

This 6-DOF spring makes the dynamic object tend to acquire the same position and orientation of the virtual haptic handle, assuming that the two objects are initially registered in some manner, e.g., with the center of the handle located at the dynamic object's center of mass and the handle's main axis aligned with one of the dynamic object's principal axes. The virtual object is assigned mass properties, which are reflected at the haptic interface as apparent mass that is added to the haptic device's intrinsic inertia. We operated at a small reflected mass of 12 g. The force and torque equations used here are:

$$\mathbf{F}_{spring} = k_T \mathbf{d} - b_T \mathbf{v}$$

$$\dot{\tau}_{spring} = k_R \dot{\theta} - b_R \dot{\omega}$$

where

$k_T, b_T$  = spring translational stiffness and viscosity

$k_R, b_R$  = spring rotational stiffness and viscosity

$\theta$  = equivalent-axis angle (including axis direction)

$\mathbf{v}, \dot{\omega}$  = dynamic object's relative linear and angular velocity.

Spring stiffness is set to a reasonably high value that is still comfortably consistent with stable numerical behavior at the known time sampling rate. Stiffness and viscosity are straightforwardly related to obtain critically damped behavior. A limitation of this simple formalism is that it is only valid for a dynamic object having equal moments of inertia in every direction, such as a sphere of uniform mass density. Since we were not interested in reflected moments of inertia, and indeed sought to minimize them, this was an acceptable limitation. It represents an implicit constraint on the virtual object's mass density distribution but not on its geometrical shape.

## 5.2 Virtual Stiffness Considerations

When the virtual object is in resting contact with the half-voxel-deep force field described by stiffness  $K_{ff}$ , we want to prevent the user from stretching the spring so far as to overcome the force field and drag the dynamic object through it. The spring force is clamped to its value at a displacement of  $s/2$ , where  $s$  is the voxel size. In the worst case, this contact force is entirely due to a single point-voxel interaction, which therefore determines an upper limit on the spring force. This can be viewed as a modification of the god-object concept [25], in which the god-object is allowed to penetrate a surface by up to a half voxel instead of being analytically constrained to that surface.

Whenever many point-voxel intersections occur simultaneously, the net stiffness may become so large as to provoke haptic instabilities associated with fixed-time-step numerical integration. To cope with this problem, we replace the vector sum of all point-voxel forces by their average, i.e., divide the total force by the current number of point-voxel intersections,  $N$ . This introduces force discontinuities as  $N$  varies with time, especially for small values of  $N$ , which degrades haptic stability. We mitigate this side effect by deferring the averaging process until  $N = 10$  is reached:

$$F_{Net} = F_{Total} \quad \text{if } N < 10$$

$$F_{Net} = \frac{F_{Total}}{N/10} \quad \text{if } N \geq 10$$

and similarly for torque.  $K_{ff}$  is adjusted to assure reasonably stable numerical integration for the fixed time step and at least 10 simultaneous point-voxel intersections. While this heuristic leads to relatively satisfactory results, we are investigating a hybrid of constraint-based and penalty-based approaches that formally address both the high-stiffness problem and its dual of low stiffness but high mechanical advantage. Forcing an object into a narrow wedge-shaped cavity is an example of the latter problem.

Dynamic simulation is subject to the well studied problem of non-passivity, which might be defined as the unintended generation of excessive virtual energy [2,10]. In a haptic system, non-passivity manifests itself as distracting forces and motions (notably, vibrations) with no apparent basis in the virtual scenario. Non-passivity is inherent in the use of time-sampled penalty forces and in the force discontinuity that is likely to occur whenever a point crosses a voxel boundary. Another potential source of non-passivity is insufficient physical damping in the haptic device [10]. Even a relatively passive dynamic simulation may become highly non-passive when placed in closed-loop interaction with a haptic device, depending on various details of the haptic device's design, its current kinematic posture, and even the user's motion behavior.

The most direct way to control non-passivity is to operate at the highest possible force-torque update rate supported by the haptic device, which for our work was the relatively high value of 1000 Hz. We also investigated the technique of computationally detecting and dissipating excessive virtual energy. While this had some success, it was eventually replaced by the simpler technique of empirically determining the largest value of  $K_{ff}$  consistent with stable operation over the entire workspace of the haptic device. As a further refinement, we discovered some residual instability in the dynamic object when it lies in free space. Whenever that occurs, therefore, we apply zero force and torque to the haptic device (overriding any non-zero spring values). A free-space configuration is trivially detected as every point of the dynamic object intersecting a free-space voxel of the environment.

## 5.3 Pre-Contact Braking Force

The treatment of spring-force clamping in section 5.2 ignored the fact that the dynamic object's momentum may induce deeper instantaneous point-voxel penetration than is possible under resting contact, thereby overcoming the force field. Currently, we do not attempt to avoid this outcome in every instance. Instead, we generate a force in the proximity voxel layer that acts to reduce the point's velocity, called the pre-contact braking force. In order to avoid a surface stickiness effect, the force must only act when the point is approaching contact, not receding from a prior contact. To determine whether the point is approaching or receding, consult its



associated inward-pointing surface normal,  $\hat{n}_i$ , and then calculate the force:

$$\mathbf{F}_i = -b\mathbf{v}_i(-\hat{n}_i \cdot \hat{\mathbf{v}}_i), \text{ if } \hat{n}_i \cdot \hat{\mathbf{v}}_i < 0$$

$$\mathbf{F}_i = 0, \text{ if } \hat{n}_i \cdot \hat{\mathbf{v}}_i \geq 0$$

where  $b$  is a “braking viscosity,”  $\mathbf{v}_i$  is the velocity of the  $i^{\text{th}}$  point in the point shell, and  $\hat{\mathbf{v}}_i$  is a unit vector along  $\mathbf{v}_i$ .

As a simple heuristic, therefore, adjust  $b$  so as to dissipate the object’s translational kinetic energy along the direction of approaching contact within one haptic cycle:

$$b = \frac{(\frac{1}{2}m\mathbf{v}^2)/\Delta t}{\mathbf{v} \cdot \sum_i \mathbf{v}_i(-\hat{n}_i \cdot \hat{\mathbf{v}}_i)}$$

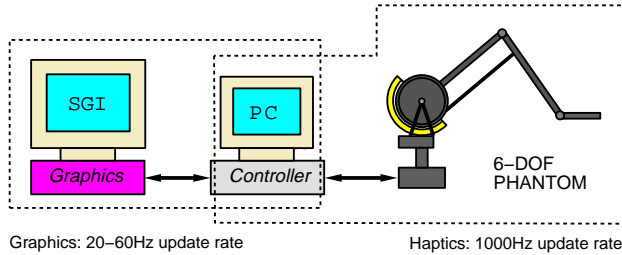
where  $m$  and  $\mathbf{v}$  are the dynamic object’s mass and velocity component along  $\sum \mathbf{F}_i$ , respectively, and the sum over  $i$  is understood to traverse only points for which  $\hat{n}_i \cdot \hat{\mathbf{v}}_i < 0$ .

We have not yet implemented a braking torque. Calculating this type of torque would be similar in form to the translational braking viscosity equation above.

A weakness of the braking technique is that an individual point’s velocity may become so large that the point skips over the proximity voxel in a single haptic cycle, or even worse, over all voxels of a thin object. We call this the “tunnelling problem.” This is particularly likely to happen for points of a long dynamic object that is rotated with sufficient angular velocity. One possible solution is to constrain the dynamic object’s translational and angular velocities such that no point’s velocity ever exceeds  $s/\Delta t$ .

## 6. RESULTS

The system configuration for our preliminary implementation is illustrated in Figure 8. Haptic rendering is performed on a dedicated haptics processor, which asserts updated force and torque information to the haptic device and reads position and orientation of the haptic handle in a closed loop running at 1000 Hz.



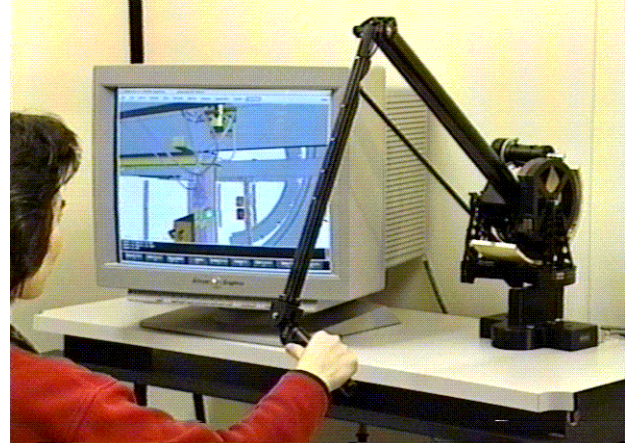
**Figure 8.** System Configuration.

In a separate asynchronous open loop, the haptics processor transmits UDP packets containing position and orientation information to a dedicated graphics processor, which renders the updated scene at about 20 Hz. This section provides more details on the system components and presents some preliminary results.

### 6.1 Haptics Device

We used a desk-mounted system called the PHANTOM™ Premium 6-DOF Prototype (shown in Figure 9), made by SensAble Technologies, Inc. This system includes the mechanism, its power electronics, a PCI interface card, and the GHOST® Software Developer’s Kit (SDK). Force feedback in three translational degrees-of-freedom is provided by a vertical 2-link planar structure, with a third orthogonal rotational axis at the base. Cable

transmission actuators drive linkages from the base. Its peak force is 22 N and the nominal positioning resolution is 0.025 mm at the end effector. The translational range of motion is about 42×59×82 cm, approximating the natural range of motion of the entire human arm. Torque feedback in three rotational degrees of freedom is provided by a powered gimbal mechanism that provides torques in yaw, pitch, and roll directions. Its peak torque is 0.67 Nm and the nominal resolution is 0.013° in each axis. The rotational range of motion is 330° in both yaw and roll, and 220° in pitch.



**Figure 9.** User with the 6-DOF haptic device.

Low-level interactions with the PCI interface card are handled by the PHANTOM device drivers provided with the system. The GHOST SDK transparently provides real-time motion control, including the use of a proprietary mechanism that guarantees a 1 kHz servo rate. A kinematic model that deals with conversions between joint space and Cartesian space, and dynamics algorithms that optimize the feel by compensating for device dynamics. Although the GHOST SDK supports numerous high-level interactions with the system, our usage is currently limited to (1) querying for global position and orientation of the end effector as a 4×4 homogeneous transformation matrix and (2) asserting the desired global force and torque.

### 6.2 Haptics and Graphics Processing

The dedicated haptics processor of our prototype system was a 350 MHz Pentium® II CPU with 128 MB of RAM running Windows NT®. The functions of voxelization, voxel-sampling, and force generation were provided by Boeing developed software known as Voxmap PointShell™, which implements the approach presented in this paper. Voxmap PointShell is interfaced with the GHOST SDK, which manages the 1 kHz servo loop. Within this loop, the haptic handle’s position and velocity information is received, a haptic frame is rendered, and updated force and torque information is sent to the device. GHOST monitors the time consumption of each loop and interrupts operation whenever a 1 kHz servo loop constraint is violated. Outside the servo loop, a separate, asynchronous loop samples the transformation matrices for the dynamic object and haptic handle, and sends them via UDP to a dedicated graphics processor.

Our dedicated graphics processor was an SGI Octane™ with one 250 MHz R10000 processor, 256 MB of RAM, and SI graphics. For visualization we use FlyThru®, a proprietary high-performance visualization system. This system was first used to virtually

“preassembly” the Boeing 777 and is now employed on commercial, military, and space programs throughout Boeing. FlyThru can maintain a frame rate of ~20 Hz, independent of the amount of static geometry. This is achieved by rendering the static geometry once to the color and Z-buffers, then reusing those images for subsequent frames [20]. This visualization scheme provided smooth motion with no noticeable lag.

One disadvantage of using two separate computers is that setup and usage tend to be cumbersome. In light of this, we have also implemented our approach on an Octane with two processors — one used strictly for haptics and the other for graphics.

### 6.3 Virtual Scenario

The static environment of our virtual scenario consisted of simulated aircraft geometry, with beams, tubes, wires, etc., voxelized at 5 mm resolution. Its polyhedral representation contains 593,409 polygons. Its FlyThru representation consumed 26 MB of memory, and its voxelized representation consumed 21 MB. Voxelization time on a 250 MHz SGI Octane was 70 sec. A closeup shot of a dynamic object (a teapot) maneuvering through a portion of this environment is shown in Figure 10.

The dynamic object for much of our testing was a small teapot (75 mm from spout to handle), logically representing a small tool or part, which when voxelized at 5 mm resolution yielded 380 points in its pointshell for the PC haptics processor. The dedicated haptics processor of the two-processor Octane system was able to achieve a maximum of 600 points for the same object.

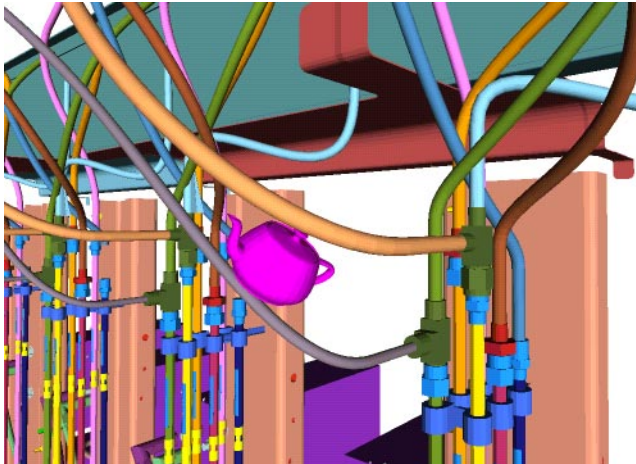


Figure 10. Dynamic object in the test environment.

### 6.4 Preliminary Test Results

We haptically rendered the motion of the teapot through the simulated aircraft geometry, paying particular attention to motion behavior and quality of force feedback. We evaluated the feeling of free space as well as resting and sliding contact (with the force field). In an attempt to explore the system’s limits, we sought to induce haptic instabilities and exact-surface interpenetrations by trapping the teapot in congested areas and by staging high-speed collisions.

Subjectively, the observed free-space behavior was indistinguishable from power-off operation, for translational as well as rotational motion. Sliding behavior on a flat or slowly curving surface was notably smooth. A relatively slight surface roughness was felt when sliding in contact with two surfaces. Torques were

clearly felt. We were able to move the teapot easily through congested areas where combinations of rotation and translation were required to find a path through the area, similar to path planning for maintenance access.

Throughout such investigation, a 1 kHz update requirement was maintained. We were unable to cause the teapot to pass completely through any of the environment surfaces, including relatively thin ones, even at maximum collision speed. There were remarkably few potential exact-surface interpenetration events. One natural metric is the ratio of penetration to collision events (PR) defined as the number of haptic frames registering one or more potential exact-surface penetrations divided by the number of haptic frames registering contact with the force-field layer (including penetration events).

We evaluated the benefit of the pre-contact braking force by selectively disabling it and re-measuring PR. The effect of this was fewer exact-surface penetrations, as shown in Table 2.

Table 2. Penetration ratio

Test	Braking	Penetrations	Contacts	PR
1	No	70	69,000	$1.0 \times 10^{-3}$
2	Yes	6	108,000	$6 \times 10^{-5}$

All such work was done with the haptic device limited to 15 N force and 0.1 Nm torque. At these limits we found the device to be stable for every possible type of motion.

## 7. CONCLUSIONS AND FUTURE WORK

The voxel-based approach to haptic rendering presented here enables 6-DOF manipulation of a modestly sized rigid object within an arbitrarily complex environment of static objects. The size of the moving object (i.e., the number of points in the point shell) is limited by the processor speed, while the size of the static environment is limited by memory. A force model was described in which the interaction of the moving object’s surface normals with the static voxmap was used to create haptic forces and torques. Results of testing an implementation of our approach on a 6-DOF haptic device showed that the performance appears to be acceptable for maintenance and assembly task simulations, provided that the task can tolerate voxel level accuracy.

It is apparent to us that we are just beginning to discover all the potential uses for the voxmap sampling method in haptics and other fields. Our primary focus will be to enhance the performance of the system for use in complex environments.

The voxel sampling method can be easily parallelized, using clones of the static environment and cyclic decomposition of the dynamic object’s pointshell. We intend to take advantage of this by investigating parallel computing environments, specifically low-latency cluster computing. This will allow haptic simulation of larger and more complex dynamic objects.

Another area of interest that we are pursuing involves using wider-bit-width voxel types (4-bit, 8-bit, etc.). This enhancement will allow for an extended force field range to model compliance when simulating varying material types.

We also intend to continue investigating solutions to problematic situations, like the wedge problem and tunnelling (moving through a thin object without detecting collision), as well as further reducing non-passivity.

## Acknowledgments

The authors express their thanks to colleagues Karel Zikan for the idea of voxel sampling, Jeff A. Heisserman for the idea of normal-aligned force direction, Robert A. Perry for creating simulated aircraft geometry, and Elaine Chen of SensAble Technologies, Inc. for literature research and technical information about the PHANTOM device and GHOST software support.

## References

- [1] Adachi, T., Kumano, T., Ogino, K., "Intermediate Representations for Stiff Virtual Objects," *Proc. IEEE Virtual Reality Annual Intl. Symposium*, pp. 203-210, 1995.
- [2] Adams, R.J. and Hannaford, B., "A Two-Port Framework for the Design of Unconditionally Stable Haptic Interfaces," *Proc. IROS*, Anaheim CA, 1998.
- [3] Avila, R.S. and Sobierajski, L.M., "A Haptic Interaction Method for Volume Visualization," *Proc. Visualization'96*, pp. 197-204, Oct. 1996.
- [4] Baraff, D., "Curved Surfaces and Coherence for Non-Penetrating Rigid Body Simulation," *Computer Graphics (proc. SIGGRAPH 90)*, vol 24, no. 4, pp. 19-28, Aug. 1990.
- [5] Baraff, D., "Fast Contact Force Computation for Nonpenetrating Rigid Bodies," *Computer Graphics (proc. SIGGRAPH 94)*, pp. 23-42, July 1994.
- [6] Berkelman, P.J. and Hollis, R.L., "Dynamic performance of a hemispherical magnetic levitation haptic interface device," in *SPIE Int. Symposium on Intelligent Systems and Intelligent Manufacturing*, (Proc. SPIE), Vol. 3602, Greensburg PA, Sept. 1997.
- [7] Brooks, F.P., Ouh-Young, M., Batter, J.J., Jerome, P., "Project GROPE — Haptic Displays for Scientific Visualization," *Computer Graphics (proc. SIGGRAPH 90)*, pp. 177-185, Aug. 1990.
- [8] Cohen, J.D., Lin, M.C., Manocha, D., and Ponamgi, M.K., "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments," *Computer Graphics (proc. SIGGRAPH 95)*, pp. 189-196, Aug. 1995.
- [9] Clover, C.L., *Control system design for robots used in simulating dynamic force and moment interaction in virtual reality applications*, Ph.D. thesis, Iowa State University, Ames, IA, Apr. 1996.
- [10] Colgate, J.E., Grafing, P.E., Stanley, M.C., and Schenkel, G., "Implementation of Stiff Virtual Walls in Force-Reflecting Interfaces," *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS)*, Seattle, WA, pp. 202-208, Sept., 1993.
- [11] Craig, J.J., *Introduction to Robotics: Mechanics and Control*. 2nd ed., Addison-Wesley, Reading MA, 1989.
- [12] Garcia-Alonso, A., Serrano, N., and Flaquer J., "Solving the Collision Detection Problem," *IEEE Computer Graphics and Applications*, vol. 14, no. 3, pp. 36-43, 1994.
- [13] Gottschalk, S., Lin, M.C., Manocha, D., "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *Computer Graphics (proc. SIGGRAPH 96)*, pp. 171-180, Aug. 1996.
- [14] Jackins, C., and Tanimoto, S.L., "Oct-Trees and Their Use in Representing Three-Dimensional Objects," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 249-270, 1980.
- [15] Kaufman, A., Cohen, D., Yagle, R., "Volume Graphics," *IEEE Computer*, 26(7), pp. 51-64, July, 1993.
- [16] Logan, I.P., Wills D.P.M., Avis N.J., Mohsen, A.M.M.A., and Sherman, K.P., "Virtual Environment Knee Arthroscopy Training System," *Society for Computer Simulation, Simulation Series*, vol. 28, no. 4, pp. 17-22, 1996.
- [17] Mark, W.R., Randolph, S.C., Finch, M., Van Verth, J.M., and Taylor II, R.M., "Adding Force Feedback to Graphics Systems: Issues and Solutions," *Computer Graphics (proc. SIGGRAPH 96)*, pp. 447-452, Aug. 1996.
- [18] Massie, T.H. and Salisbury, J.K., "The Phantom Haptic Interface: A Device for Probing Virtual Objects," *Proc. of the ASME International Mechanical Engineering Congress and Exhibition*, Chicago, pp. 295-302, 1994.
- [19] Mirtich, B. and Canny, J., "Impulse-based Dynamic Simulation," *Proceedings of Workshop on Algorithmic Foundations of Robotics*, Feb. 1994.
- [20] OpenGL Architecture Review Board, Woo, M., Neider, J., and Davis, T. *OpenGL Programming Guide*, 2nd, Addison-Wesley, Reading, MA, 1997.
- [21] Ruspini, D.C., Kolarov, K., and Khatib, O., "The Haptic Display of Complex Graphical Environments," *Computer Graphics (Proc. SIGGRAPH 97)*, pp. 345-352, Aug. 1997.
- [22] Sclaroff, S. and Pentland, A., "Generalized Implicit Functions for Computer Graphics," *Computer Graphics (Proc. SIGGRAPH 96)*, pp. 247-250, July, 1991.
- [23] Witkin A. and Welch, W., "Fast Animation and Control of Nonrigid Structures," *Computer Graphics (Proc. SIGGRAPH 90)*, pp. 243-252, Aug. 1990.
- [24] Yokokohji, Y., Hollis, R.L., and Kanade, T., "What you can see is what you can feel. Development of a visual/haptic interface to virtual environment," *Proc. IEEE Virtual Reality Annual Int. Symposium (VRAIS)*, pp. 46-53, Mar., 1996.
- [25] Zilles, C.B. and Salisbury, J.K., "A Constraint-based God-object Method for Haptics Display," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Pittsburgh, PA, pp. 146-151, 1995.

# Advances in Voxel-Based 6-DOF Haptic Rendering

William A. McNeely, Kevin D. Puterbaugh, James J. Troy

Boeing Phantom Works

April 13, 2005

## Abstract

*An approach is presented for realizing an order-of-magnitude improvement in spatial accuracy for voxel-based 6-DOF haptics. It trades constant-time performance for greater spatial accuracy. This helps to make 6-DOF haptics applicable to extraordinarily complex real-world task simulations, which often admit no other known solution short of physical mockup. A reduction of haptic fidelity is tactically incurred but simultaneously mitigated by augmenting standard voxel-sampling methodology with distance fields, temporal coherence, and culling of redundant polyhedral surface interactions. This is applied to large-scale haptic scenarios involving multiple moving objects and to collaborative virtual environments.*

**Keywords:** Haptics, physically based modeling, collision detection, voxel sampling, collaborative virtual environments

## 1. Introduction

The voxel sampling approach to 6-DOF haptics demonstrated the potential for simulating real-world tasks such as maintainability assessment of engineering design [13]. In its simplest form, voxel sampling also provides constant-time performance, which directly solves the problem of providing reliable 1000Hz haptic refresh rates without requiring decoupled simulation and haptic loops, which in turn avoids intermediate representations [14]. However, constant-time performance exacts a steep price in spatial accuracy. While some real-world tasks can be simulated adequately using, say, 10~15mm voxels for a scenario with the size and complexity of an automobile engine compartment, many more tasks would become accessible at smaller voxel size such as 1mm or even less. Since the number of surface point samples in the moving objects varies inversely as the square of voxel size, scenarios with realistically sized moving objects and/or small voxel size may easily exceed a million points. However, modern processors can perform only about 2000 point-voxel intersection tests per haptic frame, which falls short of satisfying constant-time performance by two orders of magnitude. Spatial accuracy is so important to real-world applicability that alternatives to constant-time performance should be sought.

We found that the spatial accuracy of voxel sampling may be improved by an order of magnitude, still without requiring decoupling the simulation and haptic loops, by sacrificing constant-time performance while introducing a suite of performance-enhancing measures. The most conspicuous cost of this

approach is a scenario-dependent viscous-like feeling at the haptic interface, although this is made more acceptable by an enhancement to haptic stability. The performance-enhancing measures presented here include:

- Distance fields (discussed in Section 3)
- "Geometrical awareness," which culls point-voxel samples by reducing surface-contact redundancy that is inherited from the underlying polyhedral representations (Section 4)
- Temporal coherence based on distance fields, dead reckoning, and the voxel tree that underlies the surface point samples of the moving object (Section 5)

The best measure of success of this approach is that it enables the haptic simulation of exceedingly complex tasks that cannot be simulated by any other known means short of physical mockup. This has been demonstrated for a series of real-world engineering tasks in aerospace and automotive applications. A part removal task from one of these environments will be discussed in this paper.

This complex tradeoff is proving acceptable in design-oriented applications such as assessing maintainability and producibility for complex mechanical assemblies, and indeed, it is enabling a new level of functional capability.

In addition to performance-enhancing measures, the paper also discusses some of our findings associated with implementing advanced voxel-based haptics for single and multi-user applications (Section 6) and results of performance testing for complex virtual environments (Section 7).

## **2. Related Work**

Modeling with polygons offers greater spatial accuracy than can be attained using voxels. However, polygon-based 6-DOF haptics is subject to severe performance barriers that limit its practical applicability, for example, by imposing a convexity requirement that constrains scenarios to 10~100 pairs of convex primitives [8]. A single concave object such as a dish antenna or a helical-spiral tube may decompose into hundreds or thousands of convex primitives, depending on modeling accuracy. Polygonal decimation may be used to reduce the number of convex primitives, but at the cost of compromising polygonal accuracy.

NURBS-based modeling offers superior spatial accuracy for 6-DOF haptics, but at the cost of even greater performance barriers and shape constraints. This approach has so far demonstrated the ability to simulate only relatively low-complexity scenarios that may not contain surface slope discontinuities such as sharp edges [16].

The voxel sampling approach of [13] imposes no formal shape constraints nor complexity constraints on the static objects. In this paper we adopt major elements of that approach, including 3-level 512-tree voxel hierarchy and tangent-plane force model. However, we abandon the goal of constant-time performance, because it limits scenarios to poor spatial accuracy and/or small moving objects, and we compensate by introducing a suite of performance-enhancing measures. The main cost of this change is to incur a tradeoff between the number of points in the moving object and haptic fidelity, but this is acceptable for our purposes, because it enables greater spatial accuracy and/or larger moving objects. The approach of [13] may be extended straightforwardly to support multiple moving objects [20] as well

as multi-user collaborative virtual environments, which will be discussed later in this paper. Quasi-static approximation has been introduced into voxel sampling [21], which avoids the need for a contact-stiffness-limiting heuristic, although it sacrifices dynamic realism. In other voxel-based haptics implementations, [19] presents a method for improving the pointshell accuracy and device stability, and [18] discusses a method to reduce the number of collision tests for multiple object pairs.

Geometrical awareness was exploited as core of the state-of-art feature-based collision detection algorithms between polyhedral objects [5][6][12][15]. Geometrical awareness is most often used to compute the distance between two disjoint polyhedra by considering only the two nearest features (vertex, edge, or face) between them. This lends itself to temporal coherence by tracking only the most recent nearest feature pairs. Most implementations impose a convexity requirement, but this is not strictly necessary, and so we avoid it. Another significant difference is our use of voxel-based distance fields to avoid expensive on-the-fly computation of feature-separation distance.

Like voxel sampling, the sensation preserving method of [17] achieves 6-DOF haptic update rates for complex objects by trading accuracy for speed while maintaining similar interaction forces. This method pre-processes the objects by breaking them down into convex pieces and multiple levels-of-detail (LOD). One of the compromises is that its filtered edge collapse technique allows for some object-to-object interpenetration. Pair-wise testing for multi-object collision is also used in this method.

In another 6-DOF method, [10] developed a technique to compute local minimum distances (LMDs) using spatialized normal cone hierarchies. Gradient search techniques have been used to refine the process to achieve haptic rates for moderately sized moving objects. The technique requires a pre-processing step to produce the spatial hierarchy.

Unlike our approach in which voxels replace polygons as representation primitives, voxels may be used to accelerate the search for intersecting polygons in a polygon-based approach [3]. However, any polygon-based approach incurs the performance barrier mentioned earlier, e.g., simulation update rates limited to low hundreds of Hz and moving objects limited to low tens of thousands of polygons. The tactics of loop decoupling and intermediate representations are helpful in this context [14], but they also degrade motion realism and incur the risk of force artifacts.

In the area of collaborative virtual environments, [4] developed a time-delayed collaborative haptics application where one user at a time had control of a single moving object. A shared virtual environment with simultaneous haptic interaction was demonstrated by [11] over long distances. Stable simulation was achieved with 150-200 ms of one way delay, but was limited to point contact interaction. A room-sized simulation trainer with 6-DOF force feedback was built by [9] for astronaut EVA that allowed two crew members to cooperatively manipulate the same large object. Simultaneous interaction with time delay was not taken into account.

## 2.1 Voxmap PointShell Review

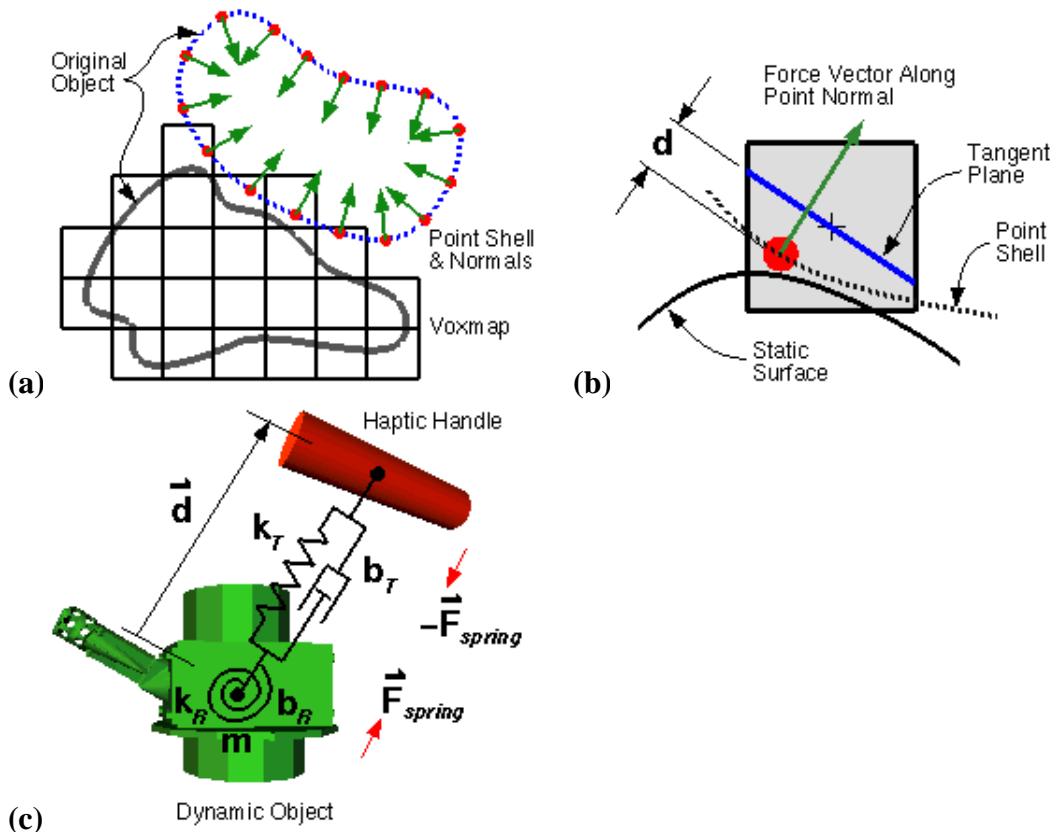
This section is intended to help clarify and review some of the concepts of voxel-based collision detection associated with our earlier Voxmap PointShell™ (VPS) paper [13].

Figure 1 shows the three basic steps associated with our voxel-based haptics process. The first step is determining which points of an object's pointshell representation have come into contact with the voxelized representation of another object (**a**). The magnitude and direction of the collision forces are



then determined. This process uses a model in which a force vector is calculated based on the depth of penetration along a normal perpendicular to a plane tangent to the object's surface at each point **(b)**. The sum of the collision forces are applied to the dynamic object and numerically integrated. The resulting motion is transferred through a virtual coupling **(c)** to generate the forces presented to the haptic device and the user. (A multi-user version of this technique will be discussed later in this paper.)

VPS works equally well for convex and concave objects, and it does not require decoupling the haptic and simulation loops. It uses a 3-level voxel hierarchy based on a 512-tree, where the leaf node is voxel, the middle level is called "chunk" and contains 512 voxels, and the highest level is called "hyperchunk" and contains 512 chunks.



**Figure 1.** (a) *Point-voxel collision detection*, (b) *Tangent plane force model*, (c) *Virtual coupling*

One of the aspects of this method that needs some clarification is the issue of multiple moving parts. The original VPS paper described the interaction between a pair of objects, where one object is represented as a collection of surface points and the other as a group of voxels. A common misconception is that one of the objects is not allowed to move. This is incorrect -- both objects in the collision pair are allowed to move.

All objects are voxelized in their initial positions in the global frame. Now consider the collision of a pair of objects that are both moving. In order to avoid the computationally expensive step of re-voxelizing objects on the fly, the positions and velocities of the points of the pointshell object (the smaller of the two, for performance reasons) are transformed into the rest frame of the voxmap object. The collision is analyzed in that frame, and the resulting force and torque are transformed back into the global frame and applied to the objects in their current positions.

For the simplest case, where only one of the objects is moving, the voxel object's coordinate system aligns with the world coordinate system and additional coordinate transformations are not required. For simplicity, this was the situation described in the earlier paper, but in the general case where both objects may be moving, the transformation to the voxel object's coordinate system will be needed.

Also, either object in the collision pair can be a combination of multiple parts that have been merged together to create a single logical object. Merged objects still maintain individual visual attributes, like color, but merged parts behave as a single entity from the collision detection method's point of view.

Environments with more than one pair of moving parts are also possible. The main issue is maintaining a list of collision pairs. For the worst case, the number of object pairs that need to be tested at each update is:

$$N_{pairs} = n(n-1)/2$$

where  $n$  is the number of moving objects.

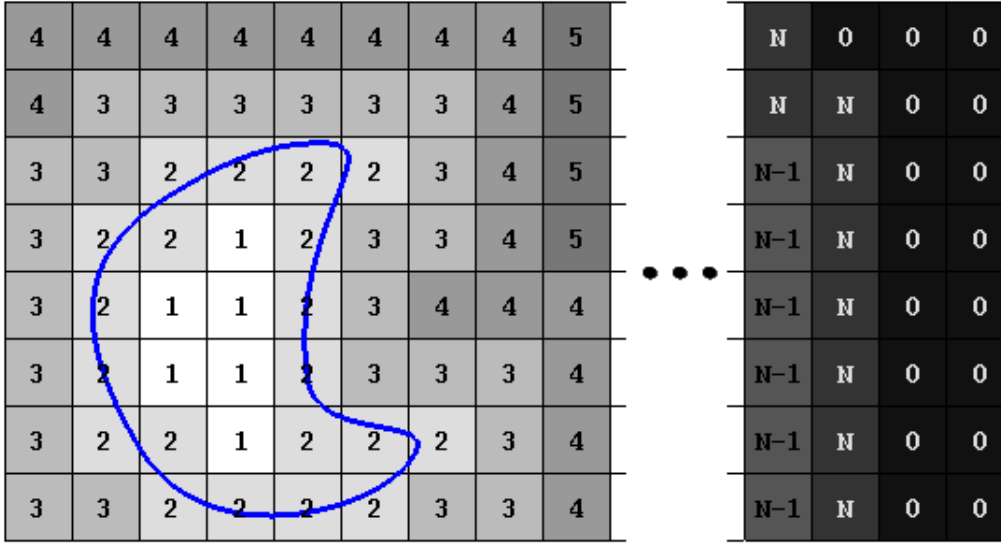
In the VPS method, each object is treated dynamically as a free object for the duration of a time step, subject to forces of collision with neighboring objects plus any external forces. This is only an approximation, of course, but it becomes a better approximation as the time step decreases, and it asymptotically approaches correct multi-body dynamic behavior in the limit of zero time step.

Other moving object issues associated with multi-user collaborative virtual environments will be discussed in Section 6.1

### 3. Distance Fields

It is useful to have advance warning of potential contact between pointshell and voxmap objects. For example, such warning is required by the temporal coherence technique described in Section 5. For that reason we extend the voxelization of an object beyond its surface into free space surrounding the object, marking such free-space voxels with integer values that represent a conservative estimate of distance-to-surface expressed in units of voxel size. This creates a voxel based distance field, as illustrated in the 2D example of Figure 2.





**Figure 2.** Voxel based distance field (in 2D)

We employ a simple "chess-board" distance-transformation algorithm [2] to calculate the distance field, which gives a conservative estimate of Euclidean distance along non-axis-aligned directions.

VPS supports 2, 4, or 8 bit voxels. The smallest positive value(s) are conventionally reserved for interior voxels, which in Figure 2 are marked 1. The distance field extends out to a user-specified maximum value, constrained only by the integer range.

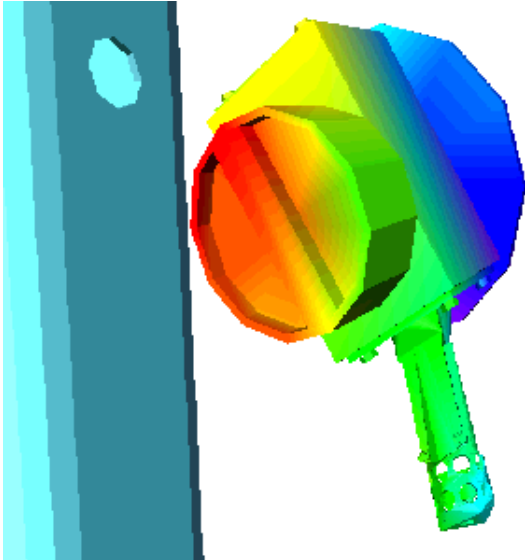
Unless noted otherwise, we assume the use of 4-bit voxels in this paper, since that is a practical choice for haptic applications in current computing environments. For 4-bit voxels the outermost positive voxels could be marked with values up to 15, representing a distance-to-surface of 13 voxels. However, the hierarchical extension of temporal coherence (Section 5.1) works optimally when the maximum distance-to-surface is matched to the power of the voxel hierarchy. Since we use a 512-tree [13], and 512 is the cube power of 8, the optimum maximum distance-to-surface is 8, corresponding to voxels marked 10 (since surface voxels are marked 2). Consequently, values 11 through 15 of the 4-bit range are unused.

The "geometrical awareness" technique described in Section 4 requires three different types of distance field, based on distance to selected geometrical features (vertex, edge, or face). Each field is independently pre-computed and packed into a word. For 4-bit voxels this implies 16-bit words, where the remaining 4 bits are unused. When discussing voxel bitwidth one must be careful to specify whether it refers to the bitwidth of an individual distance field or, less rigorously, to the size of the word required to store all three distance fields. Whenever the expression "16-bit voxels" is used in this paper, it refers to 16-bit words containing three distance fields of 4 bits each.

### 3.1 Other Proximity-Based Applications

One distance field applications that we have developed (and is now part of the VPS API) is a function that colors vertices of the dynamic object model based on its proximity to other objects. Figure 3 shows distance fields used for proximity-based coloring (warmer colors indicate closer proximity). The main

benefit from proximity coloring is that it aids haptic interaction by visually conveying distance to contact.



**Figure 3.** *Voxel-based proximity coloring*

Applications that use static environment voxel data are also possible. Highlighting surface voxels within a specific distance to the moving object produces a shadow-like effect that can also aid in distance-to-contact perception. Proximity-based distance measurement can be used to give a reasonable approximation for quickly determining minimum distances. The distance gradient information could also be useful for path planning, similar to potential field based path planning applications.

### 3.2 Collision Offsetting

Forces are generated using the tangent plane model, as reviewed in Section 2.1. One is free to select the voxel layer in which tangent-plane forces are generated, which we refer to here as the "force layer." If one selects surface voxels for that purpose, then as a potentially undesirable side effect, the exact surface of the pointshell object (e.g., its polygonal representation) may, in general, interpenetrate the exact surface of the voxmap object. In order to minimally avoid exact-surface interpenetration, one must adopt the second layer of free-space voxels as the force layer [13]. Since the distance field extends farther than two layers into free space, one may move the force layer to even more distant free-space layers and thereby create a collision offsetting effect. This is useful in task simulations where additional clearance is needed but is not formally modeled, e.g., to allow for human grasp in a part-manipulation task. In VPS one can dynamically vary the force layer and thereby dynamically vary the amount of clearance.

One might consider extending this scheme to the pointshell. The pointshell is normally derived from the centerpoints of surface voxels, but a free-space voxel layer might also be used for that purpose.

However, free-space layers contain more voxels than the surface layer, and VPS performance degrades as pointshell size increases. For that reason, VPS derives the pointshell from the surface layer, except in the situation that the user requests a range of collision offsetting that exceeds what is achievable by

dynamically varying the force layer inside the voxmap object. In that case, VPS derives the pointshell from the free-space layer that is both nearest the surface and minimally satisfies the user's requested range of collision offsetting.

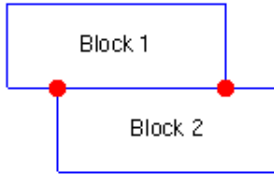
Despite the static nature of the pointshell as described above, it is possible to dynamically vary the locations of the points in the pointshell, by displacing them a short distance along the direction of the surface normal, either toward free space or toward the interior of the object. This provides the capability of fine-tuning the amount of collision offsetting. However, this has the problem that, depending on the direction of displacement and the local curvature of the surface, the displaced points may spread apart, creating a looser mesh of points that runs the risk of undetected penetration. One way to counteract this effect is to select a voxel size for the pointshell object that is smaller than that of the voxmap object, at the price of tactically degrading VPS performance.

An interesting application of pointshell displacement is mating-surface simulation, as illustrated in Results (Section 7) for a simple ball-and-socket scenario. In general, mating-surface simulation is problematic at haptic speeds, in the absence of kinematical constraints or similar special-case information, because manifold surface contact is computationally expensive. If mating parts are permanently constrained within a mechanism for the entire duration of a motion scenario, then kinematical constraints are certainly appropriate. However, it becomes problematic when kinematical constraints may engage or disengage during a simulation. For example, if a wrench can be used on a certain type of fastener, then the simulating system must know that association in advance. Any subsequent changes to tool or part geometry are liable to invalidate that association. Furthermore, the simulating system must somehow decide when to engage the constraint and when to disengage it, e.g., by detecting that the tool is sufficiently aligned with the part to engage the kinematical constraint. This leads to artifacts such as a mysterious attractive force that acts to seat the tool whenever it is sufficiently aligned with the part. Another artifact is a sticky feeling when trying to disengage the tool. VPS suggests an approach, albeit a computationally expensive one, to avoid such problems and artifacts by avoiding kinematic constraints altogether<sup>1</sup>.

## 4. Geometrical Awareness

Although the approach presented here is voxel-based, voxels may inherit properties of their parent polyhedral objects at discretization time, which has great value in culling point-voxel intersections at run time, as explained below.

To begin, consider the interaction of a pair of rigid non-penetrating polyhedral objects. Consider their surfaces as a pair of point manifolds that exhibit an arbitrary (even infinite) number of point intersections (surface contacts) for a given configuration. For physically-based modeling purposes, the only interesting contacts are those where one or both points belong to a C1 discontinuity in their respective parent surface. As a simple 2D example, the only interesting contacts between two blocks are their vertex-edge contacts, as illustrated in Figure 4.



**Figure 4.** *One 2D block rests upon another 2D block. (The red circles represent vertex-edge contacts.)*

In 3D, only vertex-surface and edge-edge contacts are interesting. ("Surface" is understood to include its edge boundaries and "edge" its vertex boundaries, hence edge-vertex and vertex-vertex contacts are both trivial subsets of edge-edge.) We refer to this powerful insight as geometrical awareness, to adopt the terminology of [5]. This result is entirely general for non-penetrating polyhedral objects, in particular, it does not require convexity. One may ignore all surface-surface and surface-edge contacts, which effectively reduces the problem's dimensionality and reduces computational load enormously.

Geometrical awareness can be applied to voxel sampling as follows. Point samples are taken as the center points of surface voxels. One labels each point as vertex, edge, or surface, according to whether its parent voxel inherited as a "priority feature" the vertex, edge, or surface attribute, respectively, from the underlying polyhedral geometry. By "priority feature" we mean the following priority ordering of feature inheritance. If a point's parent voxel intersects (i.e., contains) one or more vertices in the polyhedral geometry, then the point is labeled as vertex, even if its voxel also intersects edge or surface elements. Similarly, an edge point's voxel intersects one or more edges but no vertex, while a surface point's voxel intersects one or more surfaces but neither edge nor vertex.

To more efficiently apply geometrical awareness to point-voxel interactions such as in the tangent-plane force model, we precompute three voxel-based distance fields toward the nearest surface-, edge-, and vertex-voxel, respectively, as described below. Thus, one uses surface points to sample the vertex-distance field, vertex points to sample the surface-distance field, and edge points to sample the edge-distance field.

A known limitation of geometrical awareness is that it is not effective against manifold contact of 3D edges (e.g., a sword's edge perfectly aligned with another sword's edge). In that case, geometrical awareness prescribes testing a potentially large number of point-voxel contacts along the linear region of overlap. It is not clear how to generalize geometrical awareness so as to address both the common form of edge-edge contact (e.g., swords crossed at an angle) and the exotic case of edge-edge congruency. Fortunately, the latter almost never occurs in practical scenarios, not even within the accuracy of a voxel size.

## 4.1 Optimizing Voxel/Polygonal Accuracy

Feature-based distance fields are most effective when the accuracy of the underlying polyhedral geometry matches voxel accuracy, for the following reason. As one increases polyhedral accuracy (holding voxel size constant), one obtains more polygons of smaller dimensions, which increases the likelihood that a given voxel will contain a vertex and/or an edge. That increases the number of vertex-surface and edge-edge interactions at the expense of surface-surface interactions, which tends to defeat geometrical awareness and degrade performance. To compound matters, polyhedral accuracy is typically much better than voxel accuracy. Often it is decided casually, e.g. in the process of exporting it from a CAD system, oblivious to voxel size.

For best results, therefore, polyhedral accuracy must be reduced to voxel accuracy. We accomplish this through a process similar to decimation, at voxelization time, as follows. First, tessellate the polyhedral facets into triangles. Then, if any pair of adjacent triangles has the property that its non-shared vertices deviate from coplanarity by less than  $1/2$  voxel size, and also if their polyhedral angle is less than 90 degrees, then that pair of triangles is treated as a single quasi-planar quadrilateral for voxelization purposes. Otherwise, if those criteria are not met, then the pair of triangles remains dissociated. This process is repeated by considering triangles adjacent to a quasi-planar quadrilateral, which may lead to a quasi-planar pentagon, etc. After all triangles have been so processed, distance fields are constructed from the features of the resulting quasi-planar polygons. The 90-degree polyhedral-angle criterion prevents small curved objects (such as a sphere with diameter less than a voxel size) from being reduced to a single planar polygon.

## 5. Temporal Coherence

The voxel sampling method provides a natural opportunity for exploiting spatial and temporal coherence, or temporal coherence in short. This is done by tracking and predicting the status of points in the "pointshell" (set of surface point samples) of the dynamic object. A point that contacted a surface voxel in the previous frame is likely to remain in contact in the current frame.

Whenever a point samples its appropriate voxel-based distance field (see Section 3), it obtains a conservative estimate of its minimum distance from any contact. If we also know the point's maximum speed, then by dead reckoning we can predict how many frames will elapse before contact can possibly occur, which allows us to safely reduce the frequency of point sampling. Hence, the pointshell may contain more points than could possibly all be tested in a single haptic frame, and since the pointshell is derived from surface voxels, this enables the use of smaller voxels and greater spatial accuracy.

This requires knowing a point's maximum speed, but the latter is formally unlimited. A more serious problem is that the known speed may be so large that the available processing power cannot keep up with the burden of predicting contact for all free-space points. To solve these problems, we impose a speed limit that applies to all points. For this purpose we denote the maximum distance that any point may travel in a haptic frame as *MaxTravel*. In general, *MaxTravel* is adjusted on a frame-by-frame basis because it varies inversely with the number of points that require testing during that frame. As the amount of contact and near-contact increases more point tests become necessary. It is mandatory to test points that were in contact in the previous haptic frame. However, free-space points may be scheduled for testing at a reduced frequency.

*MaxTravel* has an absolute upper bound of  $1/2$  voxel size <sup>2</sup>, in order to prevent points from skipping over the penalty-force region of surface voxels and penetrating into the object's interior. Since the time duration of haptic frames is constant, *MaxTravel* is equivalent to a speed constraint. This expresses itself at the haptic interface as a viscous-like resistance whenever the virtual-world speed tries to exceed *MaxTravel* per haptic frame. For example, consider a scenario modeled using 2mm voxels and 1000Hz haptic refresh rate. A maximum speed of  $1/2$  voxel per millisecond is 1 meter/second. This corresponds to user motion of roughly one arm's length per second, which is unrealistically fast in the context of any application that involves manipulating objects with careful intent. In this simple example, therefore, the *MaxTravel* constraint has negligible impact at the haptic interface. However, in a more complete

analysis (1) the speed constraint applies to every point on the object's surface, which generates a more complicated constraint on the object's overall translational and rotational velocities, (2) any spatial scaling between the virtual world and the haptic interface must be considered, (3) *MaxTravel* may be smaller than its absolute upper bound of  $1/2$  voxel, as calculated below:

$$MaxTravel = (nCapacity - nMandatory) / \sum \frac{n_i}{0.5s \cdot i} \quad (1)$$

where *nCapacity* is the number of point tests that the processor can perform per haptic frame, *nMandatory*, is the number of "mandatory" tests (for points already in contact),  $n_i$  is the number of points in free space at  $i$  voxels ( $i > 0$ ) from contact, and  $s$  is voxel size. If Equation 1 yields  $MaxTravel < 0.5s$ , then we limit *MaxTravel* to its absolute upper bound of  $0.5s$ .

The worst case is that of more mandatory tests than can be performed, in which case *MaxTravel* in Equation 1 becomes zero or negative and further motion becomes impossible. Whenever this happens, VPS is unable to meet the user-requested time constraint, in which case it tests all mandatory points and abandons any attempt to maintain time criticality. However, in practice, geometrical awareness (Section 3) so sharply reduces the number of points in contact that we have rarely encountered this worst-case situation during a series of complex real-world task simulations.

We track and update point status using a mechanism called *distance-to-contact* queues. All points that currently have the same distance-to-contact value are considered to belong to the same value-specific queue. However, those points beyond the range of the distance fields belong to the same queue as those lying at a distance of exactly one voxel beyond that range. Therefore,  $n_i$  in Equation 1 is the number of points in queue  $i$ . (Since we use 4-bit distance fields, the number of queues is 16.) In general, there will not be enough processing power to test the entire contents of each queue during the current haptic frame, but it is only necessary to test the entire contents of the mandatory-point queues plus the following number of points  $m_i$  of each free-space queue  $i$ :

$$m_i = MaxTravel \cdot \frac{n_i}{0.5s \cdot i} \quad (2)$$

where  $m_i$  is rounded up to the nearest integer. We test  $m_i$  points per frame in round-robin fashion for each queue individually. This ensures that no point may travel into penetration undetected, i.e., before being re-tested. Whenever a point is re-tested, its distance-to-contact value may change, which then causes the point to migrate to a different queue. We make the assumption, borne out by observation, that *MaxTravel* varies so slowly with time that it may be considered constant while a point is waiting for re-testing. In fact, *MaxTravel* tends to be conservative, because its value typically decreases with time whenever objects are approaching contact.

The distance-to-contact queues are implemented as follows. Each queue is a bitmapped representation of the entire pointshell. Each point is represented as a one bit in just one of the queues, and for all other queues the bit at this same address is *zero*. During each haptic frame, a fraction of each queue's contents is traversed in order to satisfy the minimum sampling frequency. Whenever a *one* bit is encountered, its associated point is sampled.

Under this implementation, distance-to-contact queues become quite sparse. To accelerate their traversal, each queue is ordered into a 2-level hierarchy. The leaf level contains individual bits of the queue, while the upper level contains bits that are *one* whenever any of its 32 leaf-level children are *one*. This enables the skipping of entire 32-bit runs of *zero* bits. When  $n_i$  is zero, the entire queue is empty and may be skipped. While it may not be obvious that this implementation is preferable to more sophisticated point-scheduling schemes that can be imagined, in fact it yielded higher performance than several alternatives that we explored.

Temporal coherence conveys an important, if unexpected, benefit for haptic stability. Under virtual coupling (Figure 1c), the most likely source of instability is large transient movements of the dynamic object. However, *MaxTravel* inherently prevents large transient movements. Stability is a very complex topic, and there are many other possible sources of instability (e.g., limit-cycle oscillations, overly stiff virtual systems, unpredictable user-applied forces, device limitations, etc.). However, empirically, the stability benefit from *MaxTravel* has enabled perfectly stable haptic operation for all scenarios that we have ever tested.

## 5.1 Hierarchical Temporal Coherence

Since the pointshell is derived from the centroids of surface voxels, it inherits the spatial hierarchy of its parent voxel tree. All points that came from the same "chunk" of the voxel tree (the first level above leaf level) are assigned to contiguous bit addresses in the distance-to-contact queues. Then, whenever the entire chunk's worth of points is known to lie in free space, we may remove all such points from their queues and continue tracking only the chunk's distance-to-contact, e.g., by testing the chunk's centroid against the surface distance field. (Since the chunk's contents may be marked with a mixture of surface, edge, and vertex attributes, we must test against the most conservative distance field, which is the surface distance field.) This greatly reduces the point-testing burden, since in a 512-tree, a chunk contains about 100 points on average.

One may learn whether a chunk's entire point contents lie in free space as follows. Chunks are marked with a discretized distance-to-contact value in the same manner as voxels, thereby creating a chunk-level distance field. The pointshell-object's chunk centroid is then used to sample the static-object's chunk-level distance field, in precisely the same manner as point-voxel sampling. If such a test reveals that a chunk lies beyond the space spanned by voxel-level distance fields, then that chunk is considered to lie entirely in free space, and chunk-level temporal coherence is applied. On the other hand, if a previously free-space chunk enters the space spanned by voxel-level distance fields, then its contents are disgorged and re-inserted into the point queues. (The cost of such transitions may be greatly reduced by exploiting the fact that the points have contiguous bit addresses.)

Point sampling and chunk-centroid sampling behave identically in all respects except the following. "Contact" is re-defined to mean that the chunk enters the space spanned by voxel-level distance fields, as described above. Every chunk that lies in that space is considered to occupy a mandatory chunk queue. *MaxTravel* is modified straightforwardly in Equation 1 by augmenting  $nMandatory$  with a chunk-specific contribution and also extending the summation over queues to include the new chunk-level queues.

## 5.2 Point drifting

As a significant performance optimization, one may reduce the frequency of voxmap lookup during

point testing, as follows. Whenever voxmap lookup becomes necessary (as explained below), the point's current exact spatial position is stored, along with its current voxel-accurate distance-to-contact (as discovered through voxmap lookup and expressed implicitly by the point's distance-to-contact queue number). Subsequently, whenever that point falls due for testing under temporal coherence, one first computes its point drift, defined as the exact distance between its current position and its previously stored position. If so much drift has occurred that the point may be "too near contact" (as defined below), then voxmap lookup becomes necessary and drifting begins anew. Otherwise, if the amount of drift is not so great, then voxmap lookup is avoided, and the point is allowed to continue drifting. The criterion for being "too near contact" is that the point could possibly have drifted as much as two queues away from contact. In principle, one could more aggressively wait until it was only one queue from contact, but we elect to have a one-queue margin of safety.

When a point begins drifting, it stays in its initial distance-to-contact queue until the amount of point drift warrants re-queueing, e.g., due to drifting more than a voxel size. Whenever re-queueing becomes necessary, we conservatively assume that the point moved nearer to contact, i.e., to a lower-numbered queue. That incrementally increases the frequency of testing, but empirically, each test suddenly becomes about 7 times faster by avoiding voxmap lookup. This 7-fold advantage decreases as drifting proceeds, becoming minimal when the point drifts as near as two queues from contact, but when that happens the point is re-tested subject to voxmap lookup and properly re-queued, and drifting begins anew. The net performance benefit of point drifting depends in a complicated way on the motion scenario, but typically it is several-fold.

### 5.3 Dynamic Pre-Fetching of Voxel Data

It may easily happen that there is insufficient system memory to hold all voxel data for a given scenario, especially for large-scale scenarios and/or small voxel sizes. Under 32-bit operating systems the addressing limit is 4GB, which is often reduced further to 3GB or 2GB. While virtual memory is a good solution for non-time-critical applications, it is fundamentally incompatible with time-critical haptics. Just-in-time memory paging causes highly distracting force discontinuities or even haptic-controller timeouts. To avoid such adverse effects, one needs a predictive memory-paging scheme. For that reason, we implemented a dual-thread scheme that supports time-critical operation at haptic rates in one thread, coupled with dynamic pre-fetching of voxel data in the other thread.

A convenient way to implement dynamic pre-fetching is to integrate it with chunk-level temporal coherence as described in Section 5.1. The latter includes probing the space that lies beyond the space spanned by voxel-bearing chunks in the static distance fields. Consequently, one can readily detect when a given chunk of the dynamic object has reached a distance of one chunk size away from any voxel-bearing chunk(s) in the static distance fields. Whenever that happens (to recall Section 5.1), one immediately switches level-of-detail representations in the dynamic object, from using the chunk's centroid to using its constituent points. To extend that mechanism to dynamic pre-fetching, simply treat such representation-switching events as requests that voxel-bearing chunk(s) of the static distance fields should be fetched into real memory, if necessary. A separate thread can then perform such fetching in time to satisfy access by the haptic thread.

There is no way to guarantee that a pre-fetching thread can always act fast enough to satisfy the haptics thread, depending on the speed of the hard drives, scenario complexity, the backlog of pre-fetching requests, size of *MaxTravel* compared to chunk size, etc. To cover all such contingencies, we allow the haptics thread to be temporarily suspended as needed to allow the pre-fetching thread to catch up.



During a state of suspension, *MaxTravel* is set to zero, and no forces are sent to the haptic interface. We limit the duration of any suspension to 2 seconds, after which the simulation is terminated. Empirically, even with our largest scenarios, such suspensions occur so rarely and/or have such brief duration that they proved imperceptible. Furthermore, we have not yet encountered a scenario that was prematurely terminated by the 2-second timeout.

We did not extend this mechanism to hyperchunks, nor was temporal coherence extended to hyperchunks, on the grounds that the complexity of such an extension seemed to outweigh its potential benefits.

## 6. Implementation

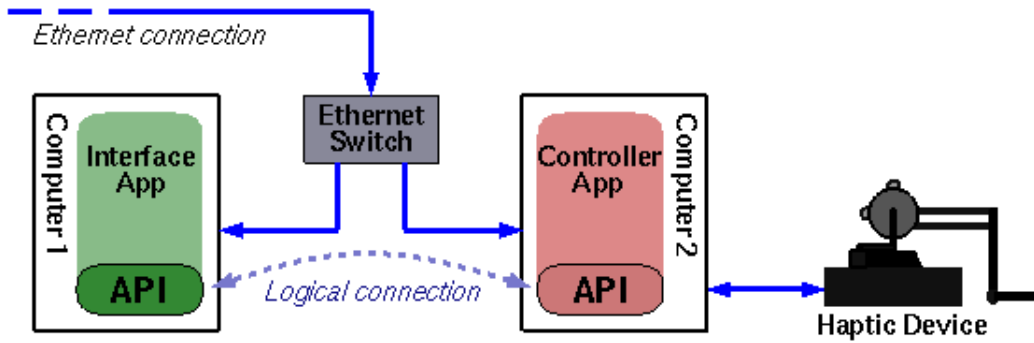
We have used a variety of architectures for experimentation and prototyping, using either one or two computers. For production use we implement the VPS collision detection and force generation algorithms in a separate computer we call the "Haptic Controller" using a client-server model. Our choice for this approach was driven by the mismatch between the computing requirements of physically based modeling and the available workstations used by typical engineering departments. Physically based modeling has these characteristics:

- Computationally intensive -- dual CPU's are best so one can be devoted to haptics and the other to secondary tasks
- Large amounts of memory (RAM) are required
- Large amounts of available high speed disk space are needed to save voxel data

Production workstation installations generally have these characteristics:

- A single CPU
- Modest amounts of memory
- Computation is already taxed by graphical rendering
- Memory fills with data representations optimized for graphical rendering
- Local disk space may be lower speed or inaccessible
- OS and application software installation tightly controlled by IT department

The mismatch between requirements and existing hardware is solved by putting the haptic process on a PC that is devoted to haptic processing. The haptic device (or other type of input device) is then connected to this PC as shown in Figure 5. The Haptic Controller PC is connected to the client workstation via ethernet and TCP/IP. If the PC is given an IP address on the same subnet as the workstation, connecting them via a switch minimizes bandwidth contention and allows them to communicate at 100Mbit/second, regardless of the network connection speed available to the workstation (often much slower). The Haptic Controller PC has no visual interaction with the user, and need not have an associated monitor. The Haptic Controller supports a variety of interaction devices including: various models of the PHANTOM haptic device, 6-DOF Spaceball (and similar) devices with no force feedback, and a 2-DOF mouse with no force feedback.



**Figure 5.** *Haptic Controller configuration*

Within the Haptic Controller, one thread is devoted to collision detection and force generation, and a second thread handles communication tasks with the client, and pre-processing. When the Spaceball is used, a third thread receives updates from it.

The Haptic Controller provides these services to the client: voxelization, transparently caches voxel data for later reuse, manages the haptic device, and supplies updated positions of the goal and moving objects on demand. An API is supplied for use by the host application. The API is designed to minimize the intrusion into the host application.

We have used the Haptic Controller with two applications: FlyThru®, a Boeing-proprietary visualization system used for design reviews, and a prototype application used for investigating collaborative haptics. The results reported here were obtained with FlyThru. FlyThru is designed to handle large amounts of geometry, and includes rendering optimization for the special case of a fixed eye point, and a small amount of moving geometry. This optimization is important because it allows the environment to be rendered in full detail during haptic interaction at responsive frame rates.

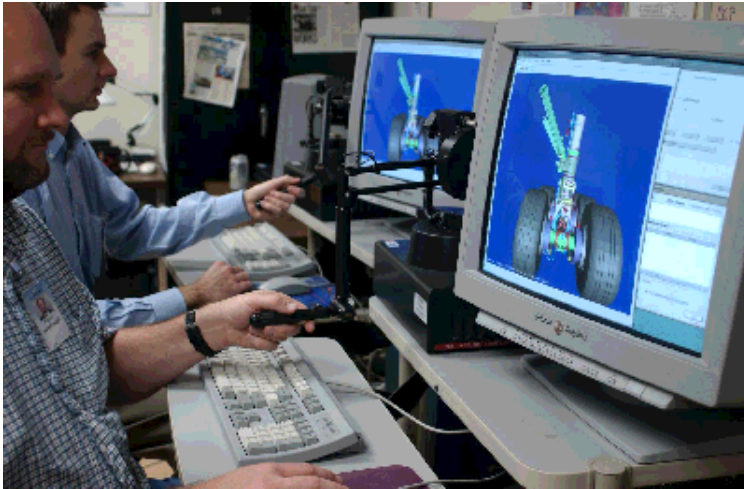
High-speed hard drives are desirable for the Haptic Controller for the sake of dynamically pre-fetching voxel data (Section 5.3). Empirically, hard drives with higher data transfer rates (like 10k-15k RPM SCSI drives) are more likely to meet pre-fetching demands for large-scale scenarios. If lower-speed hard drives are used, then haptic force quality acquires a rough and viscous feeling whenever two objects make contact for the first time, due to the fact that *MaxTravel* is set to zero while waiting for voxel data to appear in memory.

## 6.1 VPS-Based Collaborative Virtual Environments

In addition to building VPS-based applications with multiple constrained and unconstrained moving objects, we have recently implemented a multi-user environment for collaborative 6-DOF haptics that uses VPS for collision detection and response. The types of haptically enabled collaboration applications that we have been investigating include: design reviews, maintenance access, and training.

Implementing a collaborative virtual environment (CVE) with multiple simultaneous haptic users becomes more difficult when users are located at geographically separate sites. Haptic interaction is very sensitive to synchronization delays produced by communication over large distances. In order to maintain haptic stability, while minimizing the impact on interactive performance, the application needs to be designed with time delay compensation in mind. In our CVE implementation, we address the delay issue by using peer-to-peer communication and a multi-user virtual coupling configuration. Figure 6

shows our collaborative virtual environment application for maintenance access analysis.



**Figure 6.** *Haptic enabled collaborative virtual environment*

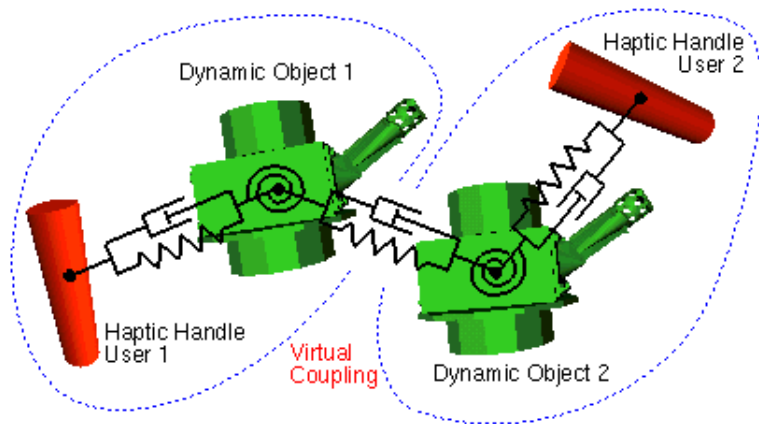
The peer-to-peer architecture synchronizes the CVE without a central server<sup>3</sup>. Each user is running a separate simulation of the environment in which models and motions are synchronized with the other users. The implementation uses TCP packets between the front-end graphical interface and UDP packets between haptic controllers. The system supports active users with haptics and non-haptic devices, as well as passive (visual only) users. A user can enter and leave the simulation at any time without impacting the other users.

The two main types of collaborative tasks that we have focused on are those involving: (1) each user controlling separate objects, and (2) multiple users controlling the same object. We will refer to these as type-1 and type-2, respectively. Both have the same type of infrastructure with respect to data and model synchronization, network connections, and device control. There are some significant differences as well.

The first type (control of different objects) has the same pair-wise collision checking requirements discussed in Section 2.1, but with the added requirement that users be aware that a voxel size mismatch between users will produce an asymmetric force response. A user with a smaller voxel size than other users will create an imbalance in contact forces between objects. This allows user A's pointshell object to contact user B's voxmap and generate repulsive forces before B's pointshell object makes contact with A's voxmap. This gives the user with the smaller voxels an enhanced ability to push/pull other users around without being affected equally by their interactions. Although the exact nature of this imbalance is probably unique to voxel-based haptics, this type of condition is a common problem in collaborative systems without centralized management -- for example, in a multi-player video game users can cheat by modifying the local front-end interface to give themselves special powers. In general, collaborative haptics applications will have asymmetric behavior if force calculation parameters are not the same for all users.

The second type of collaboration (users controlling the same object) requires a new type of coupling connection. In previous implementations [13], we have used virtual coupling elements [1][7] to connect the dynamic object to the haptic device. For the multi-user case, the virtual coupling model was extended to connect the instances of the object that all users control. Since each user is running an

independent simulation, there is an instance of the object independently calculated for each simulation. Coupling effects from the other instances of the object act as additional external forces on the local dynamic simulation of each object instance. Figure 7 shows this connection for a two user arrangement.



**Figure 7.** Multi-user connection model using virtual coupling elements

The multi-user virtual coupling effectively creates an environment for bilateral teleoperation of multiple haptic (or robotic) devices, with the addition of collision detection and response from objects and constraints in a virtual environment. One of the interaction drawbacks of this method is the potential for divergence of the multiple object instances. This can occur when another object (like a thin wall) gets trapped between the instances of the dynamic object.

Another interesting finding for both of these approaches to collaboration is that the haptic devices and dynamics simulations remain stable when force information from the other users is transmitted at rates below 1000Hz. The systems were functionally stable when external force updates from the other users were received at 100Hz. Note, we still maintained each users local simulation at 1000Hz to keep numerical integration and haptic loops stable.

A combined environment that simultaneously allows both types of interaction presents some interesting response possibilities. For example, what happens when two users are controlling one object (type-2) and then a third user joins the environment and controls another object (type-1)? In addition to feeling bilateral forces from each other, the first two users will see and feel contact interaction with the third as expected with type-1 contact. From the third user's point of view, he or she will see and interact with what appears to be a single instance of a moving object -- unless the users controlling that object enter into a divergent condition. One option for dealing with this situation is to allow user 3 to see and interact with both instances of that object. How well this works from a usability standpoint is still unknown, since a combined environment is not something we have tested yet.

In addition to multi-user interaction issues, time delay compensation is another major concern in collaborative virtual environments. Time delay is especially problematic when users are located at geographically separate sites. It is less critical for the type-1 collaboration, where the non-coupled users may not be aware of the delay -- at least not initially. They will still see the other users objects moving and instantly feel forces when they make contact with those objects. The delay will become apparent when objects have continuous contact. Although local contact forces are felt immediately, the reaction of the other user's object to the contact is delayed. A similar delayed reaction occurs when the contact is

removed. Fortunately, this delay does not appear to destabilize the simulations. But that is not the case for type-2 collaboration.

When multiple users simultaneously control the same object, time delay can cause the haptic devices to become unstable. For this situation, we have implemented a method for linking the current value of the time delay to the stiffness gains in the cross-user virtual coupling. A linear reduction of the stiffness for delays up to 1 second appears to keep both simulations stable. At this point, we are using gains that have been determined experimentally. We hope to develop a more theoretical basis for gain selection in the future.

## **6.2 Physically Based Modeling Without Force Feedback**

Although we have found that task performance of force feedback applications is superior to physically-based applications without force feedback, the cost of haptic devices appears to be a barrier to widespread adoption in the engineering community. Devices that have 6-DOF input, but no force feedback, like the Spaceball® provide a low cost alternative. However, just as with a haptic device, the VPS algorithms prevent part interpenetration and impose physically possible motion. The Spaceball is a force input device where the user pushes/pulls against internal spring-like elements. This motion is then converted into position and orientation data. Other 6-DOF, non-force feedback devices like the MicroScribe® have also been successfully tested with VPS-based virtual environments.

So far, our informal testing results indicate that these types of devices are adequate for tasks of low to moderate difficulty. We have found that very difficult tasks, like part extraction from congested environments, are only solvable with force feedback. In general, task performance is usually slower without force feedback, but is a reasonable alternative for some conditions. In our implementations, we usually attach the Spaceball to the Haptic Controller PC rather than the host workstation so that any existing use of another Spaceball by the host application (i.e., for view control) is unaffected. With such a system, an ambidextrous user can simultaneously change the viewpoint and manipulate objects.

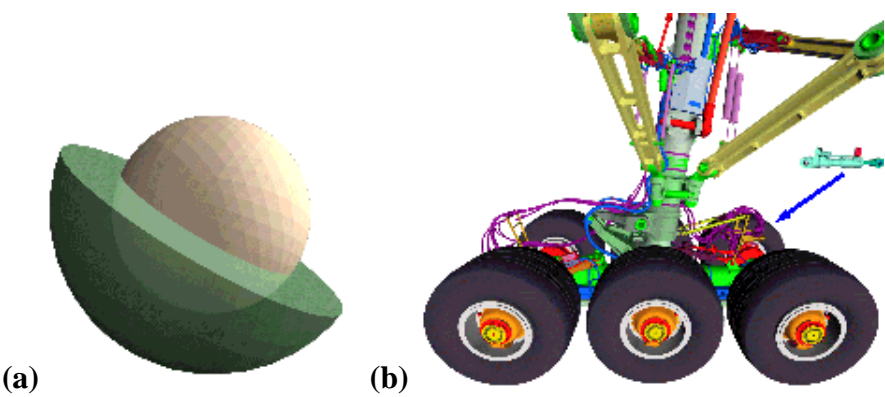
## **7. Experimental Results**

Our high-performance haptic rendering system has been implemented on Linux®, Microsoft Windows®, and SGI IRIX®. The performance results in the following discussion were obtained using a two processor 2.8 GHz Xeon PC with 2GB of RAM running Windows XP. Haptic rendering is performed on one processor to provide updated force and torque information to the haptic device and read position and orientation of the haptic handle in a closed-loop control system running at 1000Hz. Force feedback is provided by a PHANTOM® Premium 1.5/6-DOF haptic interface made by SensAble Technologies, Inc. The host graphics application for these experiments is FlyThru®, our internal high-performance visualization system, which uses a second computer to maintain a graphics frame rate of 10~20 Hz, depending on the complexity of the moving geometry, but independent of the amount of static geometry. This high performance is achieved by rendering the static geometry once to the color and Z-buffers, then reusing those images for subsequent frames [22]. (Other graphics display environments are also possible.)

VPS provides the capability to crudely simulate mating-surface scenarios without using kinematic constraints, as described in Section 3.2. This is illustrated here for the simple scenario of a ball that may

be rotated in a cradle-like socket (Figure 8a). This example illustrates a worse case scenario where a large amount of object-to-object contact occurs. In this case, the ball is the pointshell object, and its points are displaced by half a voxel toward the interior of the ball, in order to allow the ball to seat fully with the socket. For this scenario we measure VPS performance in terms of the time required for a full rotation of the ball. With a radius of 25 mm and a voxel size of 0.35 mm, this takes 1.28 seconds on a 2.8GHz processor. The speed of rotation is limited by *MaxTravel*, which is determined by voxel size and processor speed. In this scenario there are, on average, 250 points in contact at all times.

Figure 8b shows the 777 Main Landing Gear used here as an example of a large dataset for maintenance analysis tasks. The overall dimensions of the this dataset are approximately 4.1 x 1.9 x 4.8 m. The dynamic object chosen for testing is a large hydraulic actuator near the bottom of the scene that measures 0.9 x 0.2 x 0.2 m. For this test scenario, the user interacts with the environment by removing the dynamic object from its installed position. Simulation accuracy was adjusted over multiple tests by varying the voxel size.



**Figure 8.** Models used for testing: (a) Ball and socket model, (b) 777 Main landing gear (with dynamic object)

Table 1 collects the parameters of the dynamic object and the static environments in each of the above two scenarios, in which our approach was able to maintain a 1000Hz haptic refresh rate. Each scenario was evaluated twice, once with a relatively large voxel size and once with a small voxel size in relation to the overall dimensions of the scene. The Table includes the sampling resolution (voxel size), numbers of triangles, number of sampling points in each dynamic object, numbers of triangles, and number of voxels in each static environment.

**Table 1.** Virtual Scenario Measurements <sup>4</sup>

Scenarios	Voxel Size, mm	Voxelization Time, sec	Loading Time, sec	Dynamic Object		Static Environment	
				Triangles	Points	Triangles	Voxels
Ball and socket	0.35	5.8	1.7	2048	23960	2176	$5.91 \times 10^5$
Ball and socket	0.15	21.5	7.0	2048	130688	2176	$3.11 \times 10^6$
Landing gear	1.0 / 2.5	1472	313	40476	528653	$2.76 \times 10^6$	$4.59 \times 10^8$
Landing gear	0.7 / 1.25	5861	1355	40476	$1.14 \times 10^6$	$2.76 \times 10^6$	$1.78 \times 10^9$

One cannot straightforwardly assess the relative performance benefits of geometrical awareness and temporal coherence, since they depend sensitively on the motion scenario. However, one may directly compare the currently attainable accuracy (as represented by voxel size) against what was attainable before the advent of algorithmic enhancements such as geometrical awareness and temporal coherence. The maximum number of points that VPS could process in 1999 was reported as 600 [13]. Currently there is no formal limit, but up to 1M points is readily attainable and usable. We must also account for the fact that CPU speeds have increased about 8-fold since 1999. Consequently, 1M points was equivalent to 125,000 points in 1999, a 200-fold increase. Since the number of points varies inversely as the square of voxel size, a 200-fold increase in pointshell capacity corresponds to a 14-fold improvement in accuracy due to VPS algorithmic enhancements alone. Combining this with the CPU-speed increase, there has been a net 40-fold improvement in accuracy since 1999.

Throughout testing, we paid particular attention to motion behavior and quality of force and torque feedback. Artificial viscosity caused by *MaxTravel* (Section 5) was evident, especially at smaller voxel sizes, whenever objects were in contact or nearly so. However, both force and torque feedback are distinctly helpful to performing task simulations.

These results are from experiments performed in a single user environment, but the performance should be nearly identical in the multi-user environment described above, since each user will be running an identical simulation (with a small amount of communications related overhead).

## 8. Summary and Conclusions

In this paper we discussed geometric awareness, temporal coherence, and dynamic pre-fetching techniques for improving speed and accuracy of the Voxmap PointShell collision detection method for 6-DOF haptic rendering. The desired order-of-magnitude improvement in spatial accuracy was realized, at a cost of reduced haptic fidelity that is proving acceptable. Results from performance tests conducted on large datasets were presented.

Additional VPS capabilities for distance fields and surface offsetting were discussed that can be used to enhance collision detection and response, and also provide capabilities for path planning and other proximity related applications.

The use of VPS for multiple moving objects and shared objects in a collaborative virtual environment was discussed, as were collaboration related issues unique to voxel-based haptics.

We believe that haptic feedback will always remain a powerful adjunct to visual feedback, especially in busy environments with much occlusion.

## Footnotes:

1. Developers are still free to create additional constraints on top of the basic VPS collision detection implementation.
2. If the objects are sufficiently far apart, this upper bound may increase to  $\frac{1}{2}$  hyperchunk size, as a consequence of Hierarchical Temporal Coherence (Section 5.1)
3. The *collaborative architecture* is peer-to-peer, which should not be confused with the *Haptic Controller* architecture, which uses a client server model.
4. The total number of voxels shown in Table 1 include internal, surface, and distance field voxels (as described in Section 3). When multiple voxel sizes are listed for a scenario, the smaller number is the point spacing of the dynamic object and the larger number is the voxel size of the static environment.

## References

- [1] Adams, R. and Hannaford, B. "A Two-Port Framework for the Design of Unconditionally Stable Haptic Interfaces." Proc. IROS, Anaheim CA, 1998.
- [2] Borgefors, G., "Distance Transformations on Digital Images", Computer Vision Graphics Image Processing, v34, pp. 344-371, 1986.
- [3] Borro, D., Garcia-Alonso, A., and Matey, L., "Approximation of Optimal Voxel Size for Collision Detection in Maintainability Simulations within Massive Virtual Environments" Computer Graphics Forum, 23(1), p.13, Mar 2004.
- [4] Buttolo, P., Oboe, R., Hannaford, B., and McNeely W., "Force Feedback in Shared Virtual Simulations." Proc MICAD, Paris, 1996.
- [5] Choi, M. and Cremer, J., "Geometrically-Aware Interactive Object Manipulation", Computer Graphics Forum, 19(1), pp. 65-76, 2000.
- [6] Cohen, J., Lin, M., Manocha, D., and Ponamgi, M., "I-COLLIDE: An Interactive and Exact Collision Detection System for large-Scale Environments", 1995 Symposium on Interactive 3D Graphics, pp. 189-196, 1995.
- [7] Colgate, J.E., Grafing, P.E., Stanley, M.C., and Schenkel, G., "Implementation of Stiff Virtual Walls in Force-Reflecting Interfaces." PRoc. IEEE Virtual Reality Annual International Symposium (VRAIS), Seattle WA, pp. 202-208, Sept 1993.
- [8] Gregory, A., Mascarenhas, A., Ehmann, S., Lin, M., and Manocha, D., "Six Degree-of-Freedom



Haptic Display of Polygonal Models", Proc. IEEE Visualization, pp. 139-146, 2000.

[9] Homan, D. "Virtual Reality Applications Development", NASA JSC Annual Report for FY-1995 of the Automation, Robotics & Simulation Division,  
[http://tommy.jsc.nasa.gov/ARSD/report\\_FY95/homan\\_fy955.html](http://tommy.jsc.nasa.gov/ARSD/report_FY95/homan_fy955.html), including private correspondence.

[10] Johnson, D and Willemssen, P., "Accelerated Haptic Rendering of Polygonal Models through Local Decent", Int. Symposium on Haptic Interfaces for VR and Teleop Systems, pp. 18-23, Mar 2004.

[11] Kim, J., Kim, H., Tay, B.K., Muniyandi, M., Jordan, J., Mortensen, J., Oliveira, M., Slater, M., and Srinivasan, M.A., "Transatlantic Touch: A Study of Haptic Collaboration over Long Distance." Presence, 13(3), pp. 328-337, June 2004.

[12] Lin, M., "Efficient Collision Detection for Animation and Robotics", Ph.D. Thesis, University of California, Berkeley, 1993.

[13] McNeely, W., Puterbaugh, K., and Troy, J., "Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling", Proc. ACM SIGGRAPH, pp. 401-408, Aug 1999.

[14] Mark, W., Randolph, S., Finch, M., Jan Verth, J., and Taylor II, R., "Adding Force Feedback to Graphics Systems: Issues and Solutions", Proc. ACM SIGGRAPH, pp. 447-452, Aug 1996.

[15] Mirtich, B., "V-Clip: Fast and Robust Polyhedral Collision Detection", ACM Transactions on Graphics, 17(3), pp. 177-208, 1998.

[16] Nelson, D. and Cohen, E., "Optimization-Based Virtual Surface Contact Manipulation at Force Control Rates", Proc. IEEE Virtual Reality, pp. 37-44, 2000.

[17] Otaduy M. and Lin, M., "Sensation Preserving Simplification for Haptic Rendering", Proc. ACM SIGGRAPH, pp. 543-553, Aug 2003.

[18] Prior, A. and Haines, K., "The use of a Proximity Agent in a Collaborative Virtual Environment with 6 Degrees-of-Freedom Voxel-based Haptic Rendering." Proc. World Haptics Conf., Pisa, Italy, pp. 631-632, Mar 2005.

[19] Renz, M., Preusche, C., Pötke, M., Kriegel, H.-P., and Hirzinger, G., "Stable Haptic Interaction with Virtual Environments using an Adapted Voxmap-Pointshell Algorithm" Proc. Eurohaptics, Birmingham, UK, 2001.

[20] Troy, J., "Haptic Control of a Simplified Human Model with Multibody Dynamics" Phantom Users Group Conf., Aspen, CO, pp. 43-46, Oct 2000.

[21] Wan, M. and McNeely, W.A. "Quasi-Static Approximation for 6-DOF Haptic Rendering". Proc. IEEE Visualization conference, Seattle, WA, pp. 257-262, Oct 2003.

[22] Woo, M., Neider, J., Davis, T., and Shreiner, D., *OpenGL Programming Guide*, Version 1.2, 3rd, Addison Wesley, Reading, MA, 1999.

# Sensation Preserving Simplification for Haptic Rendering

Miguel A. Otaduy    Ming C. Lin  
Department of Computer Science  
University of North Carolina at Chapel Hill  
<http://gamma.cs.unc.edu/LODHaptics/>

## Abstract

We introduce a novel “sensation preserving” simplification algorithm for faster collision queries between two polyhedral objects in haptic rendering. Given a polyhedral model, we construct a multiresolution hierarchy using “filtered edge collapse”, subject to constraints imposed by collision detection. The resulting hierarchy is then used to compute fast contact response for haptic display. The computation model is inspired by human tactual perception of contact information. We have successfully applied and demonstrated the algorithm on a time-critical collision query framework for haptically displaying complex object-object interaction. Compared to existing exact contact query algorithms, we observe noticeable performance improvement in update rates with little degradation in the haptic perception of contacts.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and Geometric Transformations; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality

**Keywords:** Level-of-Detail Algorithms, Haptics, Collision Detection

## 1 Introduction

*Haptic rendering*, or force display, is emerging as an alternative form or an augmentation for information presentation, in addition to visual and auditory rendering. The sense of touch is one of the most important sensory channels, yet it is relatively poorly understood as a form of human-machine interface. Coupled with graphical rendering, force feedback can enhance the user’s ability to interact intuitively with complex synthetic environments and increase the sense of presence in exploring virtual worlds [Brooks, Jr. et al. 1990; Mark et al. 1996; Hollerbach et al. 1997; Salisbury 1999].

The first step in displaying force and torque between two 3D virtual objects is collision query and contact handling. Collision detection has been well studied, and many practical techniques and theoretical advances have been developed (see surveys by Lin and Gottschalk [1998] and Klosowski et al. [1998]). Yet, despite the huge body of literature in this area, the existing algorithms cannot run at the desired force update rates (at least hundreds of Hz but preferably several kHz) for haptic rendering of complex models.

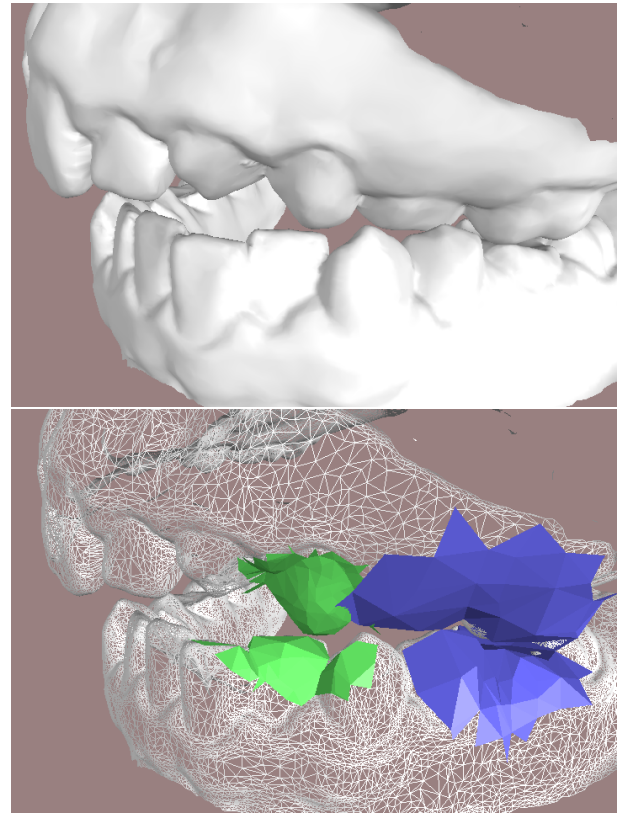


Figure 1: **Adaptive Resolution Selection.** *Top: Moving jaws in contact, rendered at their highest resolution; Bottom: The appropriate resolution (shown in blue and green) is selected adaptively for each contact location, while the finest resolution is displayed in wireframe.*

This is mainly due to the fact that the optimal running time of any collision detection algorithm intrinsically depends on both the input and output sizes of the problem. Those in turn depend on both the combinatorial complexity and the contact configuration of the objects involved in the queries. While we can render millions of polygons at interactive rates, we can barely create a force display of an environment consisting of just tens of thousands of polygons at the desired update rates.

Inspired by the large body of research in digital geometry processing and mesh simplification, we propose an algorithm based on multiresolution hierarchies of object geometry to perform time-critical collision queries for haptic rendering. In addition, our method is influenced by findings from tactual perception and spatial recognition to preserve pertinent contact information for haptic display.

**Main Contribution:** We introduce the notion of *sensation preserving simplification* to accelerate collision queries between two

complex 3D polyhedral models in haptic rendering. Given a polyhedral representation of an object, we generate a series of approximations for the object using *filtered edge collapse* operations, which smooth away high-frequency detail in low-resolution approximations while respecting the convexity constraints imposed by collision queries. Our goal is to generate a multiresolution hierarchy that can also be used as a bounding volume hierarchy for time-critical contact force computation in haptic rendering. Our computation model is based on a criterion that preserves perceivable contact details, effectively making the simplified model feel practically the same as the original. The resulting multiresolution hierarchy enables use of varying resolution approximations in contact queries at different locations across the surfaces of the objects in contact, depending on each contact configuration, as shown in Fig. 1. The key results in this paper include:

- A novel simplification algorithm, based on a formal definition of resolution, to generate representations for contact queries;
- A collision detection framework that dynamically selects adaptive levels of detail at each contact location;
- Application of this framework to real-time haptic rendering of complex object-object interaction.

This approach allows us to bound both the input and output sizes of the problem, thus achieving the desired contact query performance for force display. We have applied our approach to haptic rendering of complex models in contact configurations that are particularly challenging to collision detection algorithms. Compared to existing exact contact query algorithms, we are able to achieve up to two orders of magnitude performance improvement with little degradation in the haptic perception of contacts.

**Organization:** The rest of the paper is organized as follows. In Section 2, we give a brief survey of related work. Section 3 presents the haptic perception characteristics central to the design of our computational model and the overview of our approach. We describe the construction of the multiresolution hierarchy in Section 4 and sensation preserving contact queries using the hierarchy in Section 5. We address implementation issues and present results in Section 6. We conclude with a discussion and analysis of our approach and implementation, as well as future research directions.

## 2 Previous Work

Our research draws on a large collection of knowledge in mesh simplification, signal processing for digital geometry, collision detection, and haptic display. We briefly survey related work here.

### 2.1 Polygonal Simplification

Polygonal simplification has been an active research topic for the last decade. Numerous approaches have been proposed. We refer readers to an excellent new book on this subject [Luebke et al. 2002]. However, note that the growing interest in perception-based simplification for interactive rendering, e.g. [Luebke and Hallen 2001], has been based on human visual perceptual metrics. Our approach differs in many ways from existing work in this area, and our target application, haptic rendering, has a much higher performance requirement than visual display. Although we precompute the level of detail (LOD) hierarchy offline, the way we select the appropriate LOD on the fly is “contact-dependent” at each contact location across the object surfaces. Our approach bears a closer resemblance to view-dependent simplification [Luebke et al. 2002], which uses higher resolution representations on the silhouette of the object and much coarser approximations on the rest of the object that is not as noticeable to the viewpoint.

### 2.2 Signal Processing for Digital Geometry

Much of the work presented in this paper takes advantage of observations made in signal processing of meshes, since many concepts in multiresolution representations can be analyzed using frequency domain analysis. By generalizing discrete Fourier analysis to meshes, Taubin [1995] introduced a novel linear-time low-pass filtering algorithm for surface smoothing. This algorithm can be extended to accommodate different types of geometric constraints as well. Through a non-uniform relaxation procedure, whose weights depend on the geometry instead of connectivity, Guskov et al. [1999] generalized signal processing tools to irregular triangle meshes. Our work borrows some ideas from the relaxation techniques proposed in this paper.

### 2.3 Collision Detection

Hierarchical data structures have been widely used to design efficient algorithms for interference detection between geometric models (see surveys by Lin and Gottschalk [1998] and Klosowski et al. [1998]). Typical examples of bounding volumes include axis-aligned boxes and spheres, chosen for their simplicity in performing overlap tests between two such volumes. Other hierarchies include k-d trees and octrees, OBBTree, cone-trees, R-trees and their variants, trees based on S-bounds, etc. [Lin and Gottschalk 1998; Klosowski et al. 1998]. Additional spatial representations are based on BSP's and their extensions to multi-space partitions, space-time bounds or four-dimensional tests (see a brief survey by Redon et al. [2002]), and many more.

Hubbard [1994] first introduced the concept of time-critical collision detection using sphere-trees. Collision queries can be performed as far down the sphere-trees as time allows, without traversing the entire hierarchy. This concept can be applied to any type of bounding volume hierarchy (BVH). However, no tight error bounds have been provided using this approach. An error metric is often desirable for interactive applications to formally and rigorously quantify the amount of error introduced. Approaches that exploit motion coherence and hierarchical representations for fast distance computation between convex polytopes have been proposed [Guibas et al. 1999; Ehmann and Lin 2000]. However, these techniques are only applicable to convex polyhedra.

O'Sullivan and Dingliana [2001] studied LOD techniques for collision simulations and investigated different factors affecting collision perception, including eccentricity, separation, distractors, causality, and accuracy of simulation results. Based on a model of human visual perception validated by psychophysical experiments, the feasibility of using these factors for scheduling interruptible collision detection among large numbers of visually homogeneous objects is demonstrated. Instead of addressing the scheduling of multiple collision events among many objects, we focus on the problem of contact queries between two highly complex objects. Our approach, guided by a completely different tactual perception for haptic rendering, has distinct goals differing significantly from theirs.

Recently, GPU accelerated techniques have also been proposed for collision queries [Lombardo et al. 1999; Hoff et al. 2001]. Though fast, these approaches are not currently suitable for haptic rendering, since the readback from framebuffer and depth buffer cannot be done fast enough to perform queries at haptic update rates.

### 2.4 Haptics

Over the last few years, haptic rendering of geometric models has received much attention. Most previous work addresses issues related to rendering the interaction between a probe point and 3D objects [Ruspini et al. 1997]. This problem is characterized by high spatial coherence, and its computation can be localized. By

contrast, we attack the problem of force rendering for arbitrary 3D polyhedral object-object interaction, which involves a substantially higher computational complexity. Force rendering of object-object interaction also makes it much more challenging to correctly cache results from previous computations.

McNeely et al. [1999] proposed “point-voxel sampling”, a discretized approximation technique for contact queries that generates points on moving objects and voxels on static geometry. This approximation algorithm is the first to offer run-time performance independent of the environment’s input size by sampling the object geometry at a resolution that the given processor can handle. A recent approach proposed by Gregory et al. [2000] is limited to haptic display of object-object interaction for relatively simple models that can be easily represented as unions of convex pieces. Kim et al. [2002] attempt to increase the stability of the force feedback using contact clustering, but their algorithm for contact queries suffers from the same computational complexity.

The idea of using multiresolution representations for haptic rendering has been recently investigated by several researchers. Pai and Reissel [1997] investigated the use of multiresolution image curves for 2D haptic interaction. El-Sana and Varsheny [2000] proposed the construction of a multiresolution hierarchy of the model during preprocessing. At run-time, a high-detail representation is used for regions around the probe pointer and a coarser representation farther away. The proposed approach only applies to haptic rendering using a point probe exploring a 3D model. It does not extend naturally to force display of two interacting 3D objects, since multiple disjoint contacts can occur simultaneously at widely varying locations without much spatial coherence. The latter problem is the focus of our paper.

### 3 Overview

In this section, we first present important findings from studies on tactual perception that guide our computational model. Then, we describe the requirements for haptic rendering and our design goals.

#### 3.1 Haptic Perception of Surface Detail

From a perceptual perspective, both formal studies and experimental observations have been made regarding the impact of contact areas and relative size (or curvature) of features to the size of the contact probe (or finger) on identifying fine surface features.

Klatzky and Lederman [1995] conducted and documented studies on identification of objects using “haptic glance”, a brief haptic exposure that placed several temporal and spatial constraints on stimulus processing. They showed that a larger contact surface area helped in the identification of textures or patterns, though it was better to have a stimulus of the size comparable or just slightly smaller than that of the contact area when exploring geometric surface features.

Okamura and Cutkosky [1999] defined a fine (geometric) surface feature based on the ratio of its curvature to the radius of the fingertip acquiring the surface data. Their paper gives examples on how a larger fingertip, and thus a larger surface contact area, can miss some surface detail.

In this paper, we mainly focus on geometric surface features, not microscopic surface roughness or friction. We draw the following key observation from these studies relevant to our computational model:

*Human haptic perception of the existence of geometric surface features depends on the ratio between the contact area and the size of the feature, not the absolute size of the feature itself.*

Here we broadly define the size of a given feature in all three dimensions, namely width, length, and height. The width and length of a feature can be intuitively considered as the “inverse of resolution” (formally defined in Sec. 4) of the simplified model. That is, higher resolution around a local area implies that the width and length of the geometric surface features in that neighborhood are smaller, and vice versa. We extend the concept of “height” to include a perceivable amount of surface deviation introduced in the simplification process, according to haptic perception.

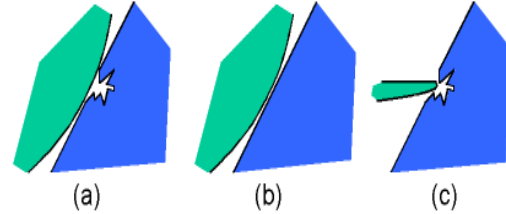


Figure 2: **Contact area and resolution:** (a) high resolution model with large contact area; (b) low resolution model with large contact area; (c) high resolution model with small contact area.

As illustrated in Fig. 2, the observation drawn by Okamura and Cutkosky [1999] for tactile feedback can extend to haptic rendering of contact forces between rigid bodies. The resolution at which the models are represented affects the number of contact points used to describe object interaction. However, increasing the resolution beyond a sufficiently large value does not affect the computed net force much, as shown in Fig. 2(a) and (b).

Our proposed model of acceptable error metrics differs notably from that of human visual perception in both the current mesh simplification literature and visual collision perception. In visual rendering, a combination of surface deviation (or Hausdorff distance) and the viewing distance from the object is used to determine if the representation of the objects requires higher resolution. In haptic rendering, on the other hand, this is governed by the relationship among the surface deviation, the resolution of the simplified model, and the contact surface area. We will later show how this relationship lays the foundation of our algorithmic design and contact query process for haptic rendering in Sec. 4 and Sec. 5.

#### 3.2 Requirements and Design Desiderata

We aim to create multiresolution representations where geometric surface detail is filtered when it cannot be perceived by the sense of touch. The resulting multiresolution hierarchies can be used to perform time-critical contact queries that stop when the reported result is accurate up to some tolerance value. This helps to automatically speed up the contact query computation for haptic rendering.

In our haptic rendering framework, we have chosen BVHs of convex hulls, because overlap tests between convex hulls can be executed rapidly in expected constant time with motion coherence [Guibas et al. 1999]. Furthermore, convex hulls provide at least equally good, if not superior, fitting to the underlying geometry as OBBs [Gottschalk et al. 1996] or k-dops [Klosowski et al. 1998].

We integrate BVHs of convex hulls with multiresolution representations so that the hierarchies, while being used for effective collision detection, can themselves be used to report contact points and normals with bounded errors at different levels of resolution. To summarize, our goal is to design **multiresolution hierarchies** that:

1. **Minimize perceptible surface deviation.** We achieve this goal by filtering the detail at appropriate resolutions and by using a novel sensation preserving refinement test for collision detection;

2. **Reduce the polygonal complexity of low resolution representations.** This objective is achieved by incorporating mesh decimation during the creation of the hierarchy;
3. **Are themselves BVHs of convex hulls.** We perform a surface convex decomposition on the given triangular mesh and maintain it across the hierarchy. The convex surface decomposition places both local and global convexity constraints on the mesh decimation process.

Our algorithm assumes that the input models can be represented as oriented 2-manifold triangular meshes with boundaries.

### 3.3 Notation and Terminology

We use bold-face letters to distinguish a vector (e.g. a point, normal, etc.) from a scalar value. In Table 1, we enumerate the notations we use throughout the paper.

Notation	Meaning
$r, r_i, r_j$	Different resolutions
$M^k$	An LOD of a mesh $M$ with a resolution $r_k$
$c_i$	A convex surface patch
$\hat{C}_i$	A convex piece constructed as the convex hull of a patch $c_i$
$e(\mathbf{v}_1, \mathbf{v}_2)$	An edge between two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$
$s, s_a, s_b$	Surface deviations
$D, D_a, D_b$	Contact areas
$q$	A distance query between two convex pieces
$\mathcal{Q}$	A contact query between two objects that consists of multiple distance queries $q$

Table 1: Notation Table

## 4 Multiresolution Hierarchy

In this section we describe the hierarchical representations of triangulated models used to perform sensation preserving contact queries for haptic rendering.

We create a hierarchy of static levels of detail (LOD), each level representing an approximation to the original triangular mesh at a different resolution (i.e. spatial scale), to be formally defined next. Because our goal is to provide an error bound arising from contact queries using simplified models, we must design a multiresolution hierarchy that computes error metrics between each LOD and the original model.

Conceptually, an LOD at resolution  $r_j$  of a mesh  $M$ ,  $M^j$ , can be obtained from an LOD at a lower resolution  $r_i$ ,  $M^i$ , by adding detail at resolutions in the range  $[r_i, r_j]$ . Our approach for generating LODs reverses this definition, so LODs at low resolution are obtained by removing detail at high resolution. While the detail is being removed, we quantify it and compute the surface deviation.

Following the LOD generation, we obtain a hierarchy where an LOD at resolution  $r_j$  preserves the lower resolution geometric information, while the higher resolution detail might have been culled away.

We generate each LOD by a sequence of **filtered edge collapse** operations (to be defined in Sec. 4.2) that perform filtering and mesh decimation,

subject to both **local and global convexity constraints** imposed by the collision detection module of the haptic rendering framework.

### 4.1 Definition and Computation of Resolution

Before we explain how we generate each LOD, we must first formally define what we consider as a resolution in our hierarchical representation. We follow the framework of signal processing for irregular meshes. Our definition of resolution for irregular meshes assumes that a triangular mesh  $M$  can be considered as a sampled version of a smooth surface  $S$ , which has been later reconstructed via linear interpolation. The vertices of the mesh are samples of the original surface, while edges and faces are the result of the reconstruction.

Our formal definition of sampling resolution for irregular meshes is based on the 1D setting. For a 1D function  $F(x)$ , the sampling resolution  $r$  is the inverse of the distance between two subsequent samples on the real line. This distance can also be interpreted as the projection of the segment between two samples  $v_1$  and  $v_2$  of the function on the average value. The average value is the low resolution representation of the function itself, and can be obtained by lowpass filtering.

Extrapolating this idea to irregular meshes, the sampling resolution of an edge  $(\mathbf{v}_1, \mathbf{v}_2)$  of the mesh  $M$  at resolution  $r_j$ ,  $M^j$ , can be estimated as the inverse of the projected length of the edge onto a low resolution representation of the mesh,  $M^i$ .

We locally compute the low resolution mesh  $M^i$  by filtering the mesh  $M^j$ , applying the filtered edge collapse operation to the edge  $(\mathbf{v}_1, \mathbf{v}_2)$ . Then we compute the normal  $\mathbf{N}$  of the resulting vertex  $\hat{\mathbf{v}}_3$  by averaging the normals of incident triangles. Finally, we project the edge on the tangent plane  $\Pi$  defined by  $\mathbf{N}$ . The resolution  $r$  is computed as the inverse of the length of the projected edge.

$$r = \frac{1}{\|(\mathbf{v}_1 - \mathbf{v}_2) - ((\mathbf{v}_1 - \mathbf{v}_2) \cdot \mathbf{N}) \cdot \mathbf{N}\|} \quad (1)$$

### 4.2 Filtered Edge Collapse

As stated, our multiresolution hierarchy is obtained through mesh simplification. We have selected edge collapse as the atomic decimation operation for two main reasons:

1. Under the required self-intersection tests, edge collapse can guarantee preservation of topology, a requirement for maintaining a surface convex decomposition of the object during the hierarchy construction.
2. Topologically, an edge collapse can be regarded as a local downsampling operation, where two samples (i.e. vertices) are merged into a single one.

In the construction of the hierarchy, we aim to:

1. Generate multiresolution representations with low polygonal complexity at low resolution for accelerating contact queries;
2. Filter detail as we compute low resolution LODs. This approach allows more aggressive simplification and enables faster merging of convex pieces to build the hierarchy.

These two goals are achieved by merging downsampling and filtering operations in one atomic operation, which we call *filtered edge collapse*.

In the filtered edge collapse operation, an edge  $(\mathbf{v}_1, \mathbf{v}_2)$  is first topologically collapsed to a vertex  $\hat{\mathbf{v}}_3$ . This step provides the downsampling. Then, given its connectivity,  $\hat{\mathbf{v}}_3$  is relaxed to a position  $\tilde{\mathbf{v}}_3$ , which provides the filtering. In our implementation, we used a relaxation operation based on the minimization of second order divided differences [Guskov et al. 1999]. Intuitively, this resembles the minimization of dihedral angles, without much affecting the shape of the triangles. We also tried other filtering techniques, such as those proposed by Taubin [1995], with very similar results. However, linear functions are invariant under the minimization of second order differences. This is consistent with the selection of

the tangent plane of the filtered mesh as the low resolution representation for the computation of resolutions.

In order to apply the relaxation to  $\hat{\mathbf{v}}_3$ , we need to compute a local parameterization. This local parameterization requires an initial position of  $\hat{\mathbf{v}}_3$ , which is computed using quadric error metrics, proposed by Garland and Heckbert [1997].

To sum up, the goal of our simplification and filtering process is to create multiresolution hierarchies for contact queries. As mentioned earlier, the collision detection module imposes convexity constraints on filtered edge collapse. Next, we will describe how the convexity constraints are satisfied.

### 4.3 Convexity Constraints

Due to the requirements of haptic rendering, we have chosen to perform collision detection using the Voronoi marching algorithm and surface convex decomposition as described by Ehmman and Lin [2001], for this approach provides us with both the distance and contact information needed for force display and its implementation is available to the public. A surface convex decomposition is first computed for the original mesh, and then a hierarchy of convex pieces is created.

The surface convex decomposition yields a set of convex surface patches  $\{c_1, c_2, \dots, c_n\}$  [Chazelle et al. 1997; Ehmman and Lin 2001]. For the correctness of the collision detection algorithm, the convex hulls of these patches are computed, resulting in convex pieces  $\{C_1, C_2, \dots, C_n\}$ .

The initial convex decomposition can be created using techniques presented by Chazelle et al. [1997]. However, our hierarchy is created in a novel way. Instead of creating convex hulls of pairs of convex pieces and joining them into a single convex piece, we merge neighboring sets of convex patches as long as they represent a single *valid* convex patch. The implications of this procedure for the collision detection algorithm are explained in Sec. 5.

Let  $c_1$  and  $c_2$  be two convex patches of LOD  $M^j$ . Let  $c = c_1 \cup c_2$  be a surface patch of  $M^j$ . After a filtered edge collapse operation is applied to  $M^j$ ,  $c_1$  and  $c_2$  will be merged if  $c$  constitutes a valid convex patch. The convex hull of  $c$ ,  $C$ , becomes the parent of  $C_1$  and  $C_2$  in the hierarchy of convex pieces for collision detection.

When a filtered edge collapse takes place, the convex patches in the neighborhood of the collapsed edge may be affected. Their boundary has to be updated accordingly.

A surface convex decomposition for the collision detection algorithm must meet several constraints:

1. All the interior edges of a convex patch must themselves be convex.
2. No vertex in a convex patch may be visible from any face in the patch, except the ones incident on it.
3. The virtual faces added to complete the convex hulls of the convex patches cannot intersect the mesh.

We consider the first constraint as a local constraint and the other two as global constraints. Before a filtered edge collapse operation is applied, we must check that the convexity constraints are preserved for all the convex pieces. Local and global convexity constraints are treated separately.

#### 4.3.1 Local Convexity Constraints

Let  $e \equiv (\mathbf{v}_1, \mathbf{v}_2)$  be a candidate edge that will be tested for a filtered edge collapse. Let  $\mathbf{v}_3$  represent the vertex resulting from the edge collapse, as well as its associated position. The edges in the 1-ring neighborhood of  $\mathbf{v}_3$  are susceptible to changing from convex to reflex and vice versa. Interior edges of convex patches are convex before the filtered edge collapse and must remain convex after

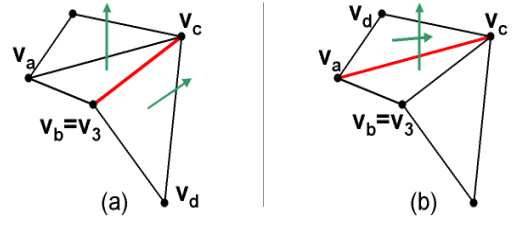


Figure 3: Local Convexity Constraints.

it. These constraints can be expressed as linear constraints in the position of  $\mathbf{v}_3$ .

Given  $e$ , the edge to be collapsed, two possible types of interior edges of convex patches exist: edges incident to  $\mathbf{v}_3$  and edges opposite to  $\mathbf{v}_3$ , as shown in Fig. 3. However, both cases can be treated equally. Assigning  $\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c$  and  $\mathbf{v}_d$  vertices as in Fig. 3, the convexity constraint of an edge can be expressed as a negative volume for the parallelepiped defined by the adjacent triangles:

$$((\mathbf{v}_b - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) \cdot (\mathbf{v}_d - \mathbf{v}_a) \leq 0 \quad (2)$$

To satisfy the convexity constraints, we have opted for formulating an optimization program, where  $\mathbf{v}_3$  is constrained to the segment between  $\hat{\mathbf{v}}_3$  and  $\tilde{\mathbf{v}}_3$ , and the objective function is the distance to  $\tilde{\mathbf{v}}_3$ . This optimization program is unidimensional. Because distance in one dimension is linear, it is a simple linear program in one dimension.

The position of the result of the constrained filtered edge collapse can be written as a linear interpolation between the initial position and the goal position:

$$\mathbf{v}_3 = u \cdot \hat{\mathbf{v}}_3 + (1 - u) \cdot \tilde{\mathbf{v}}_3 \quad (3)$$

The limit constraints can be expressed as  $u \geq 0$  and  $u \leq 1$ .

The convexity constraints in Eq. 2 can be rewritten as:

$$\begin{aligned} A \cdot u + B &\geq 0, \quad \text{where} \\ A &= ((\mathbf{v}_d - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) \cdot (\hat{\mathbf{v}}_3 - \tilde{\mathbf{v}}_3) \\ B &= ((\mathbf{v}_d - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)) \cdot (\tilde{\mathbf{v}}_3 - \mathbf{v}_a) \end{aligned} \quad (4)$$

$\mathbf{v}_3$  is computed for the minimum value of  $u$  that meets all the constraints. When  $\tilde{\mathbf{v}}_3$  is not a feasible solution but a solution exists, the constrained filtered edge collapse can be regarded as a partial filter.

#### 4.3.2 Global Convexity Constraints

The global convexity constraints are too complicated to be expressed explicitly, so they cannot be incorporated into the filtering process. Instead, they have to be verified after the filtering has been performed. We conduct this verification by computing the affected convex pieces after the edge collapse and performing the required intersection tests, using OBBs [Gottschalk et al. 1996] and spatial partitioning.

If a position  $\mathbf{v}_3$  that meets the local convexity constraints has been found, we check the global constraints. If they are met, the edge collapse is valid. If they are not met, then we check them at  $\hat{\mathbf{v}}_3$ . If they are not met at  $\hat{\mathbf{v}}_3$  either, the edge collapse is considered invalid and we disallow it. If  $\hat{\mathbf{v}}_3$  meets the global constraints, we perform a bisection search between  $\hat{\mathbf{v}}_3$  and  $\mathbf{v}_3$  of up to  $K$  iterations (in our current implementation  $K = 3$ ), searching for the position closest to  $\tilde{\mathbf{v}}_3$  that meets the global convexity constraints, as shown in Fig. 4.  $\mathbf{v}_3$  is reassigned to this position.



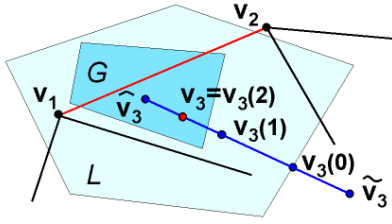


Figure 4: **Filtered Edge Collapse with Convexity Constraints.** The figure shows a filtered edge collapse where bisection search is required to find a position that meets the convexity constraints.  $G$  and  $L$  represent feasible regions of global and local constraints respectively.

#### 4.4 Multiresolution Hierarchy Generation

The hierarchy of LODs is created by applying successive filtered edge collapses on the given mesh, while performing a surface convex decomposition and merging convex pieces. First we compute the convex decomposition of the initial mesh. We then compute the value of resolution for all edges, and set them as valid for collapse. The edges are inserted in a priority queue, where edges with higher resolution have higher priority.

The main processing loop always tries to filter and collapse the edge with highest priority. If the filtered edge collapse is successful, the affected edges update their resolution and priority, and they are reset as valid for collapse. Moreover, the filtering and simplification may have relaxed some convexity constraints in the neighborhood of the collapsed edge, so we attempt to merge convex pieces in the process as well. If the filtered edge collapse fails, the edge is set as invalid for collapse. The process continues until no edges are valid for collapse.

This process must yield a hierarchy of static LODs. We have decided to generate a new LOD every time the number of convex pieces is halved. All the pieces in LOD  $M^j$  that are merged to a common piece  $C \in M^{j+1}$  during the processing will have  $C$  as their parent in the BVH.

Ideally, the process will end with one single convex piece, which serves as the root for the hierarchy to be used in the collision detection. However, this result is rarely achieved in practice, due to topological and geometric constraints that cannot be removed by a local operation such as filtered edge collapse. In such cases, the hierarchy is completed using a pairwise convex hull merging step. We call these remaining completing LODs “free” LODs.

During the process, we assign to each LOD  $M^j$  an associated resolution  $r_j$ . This resolution is the smallest resolution of an edge that has been collapsed before the LOD  $M^j$  is generated. Geometrically it means that the LOD  $M^j$  preserves all the detail of the original mesh at a resolution lower than  $r_j$ . In our sensation preserving simplification for haptic rendering, we wish to maximize the resolution at which LODs are generated. As will be explained in Sec. 5, the perceptual error for haptic rendering is measured by taking into account the resolution of the surface detail culled away. By maximizing the resolution at which LODs are generated, the contact queries can be completed faster. This is the basis for selecting edge resolution as the priority for filtered edge collapses. The pseudo code for the entire process of hierarchy construction is given in Appendix A on the conference proceedings CD.

Fig. 5 shows several of the LODs obtained when processing a model of a lower jaw (see Sec. 6 for statistics on this model). The LODs 3 and 6 shown in the figure are obtained from the original model by our simplification process. The convex pieces shown for the original model are successively merged to create the BVH during the process of simplification. Thus, the multiresolution hierarchy itself serves as BVH for collision detection. Unlike other types

of BVHs, with our simplification processing the different levels of the BVH only bound their associated LOD; they do not necessarily bound the original surface. This fact has some implications for the contact queries, described in Sec. 5.3. The free LODs 11 and 14 in the figure are obtained through pairwise merging of convex hulls. They serve to complete the BVH, but cannot be considered as LODs of a multiresolution hierarchy. Fig. 6 shows a more detailed view of the simplification and merging process. Notice how in the creation of the first LOD, most of the simplification and merging takes place at the gums. The gums are, indeed, the locations with detail at the highest resolution. When the processing reaches LOD 7, one tooth in particular is covered by a single convex patch, thus showing the success of the processing.

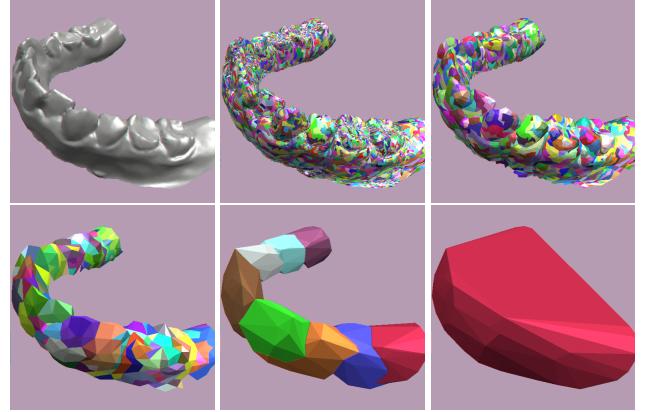


Figure 5: **Hierarchy of the Lower Jaw.** From left to right and top to bottom, original mesh, LOD<sub>0</sub>, and convex decompositions of LOD<sub>0</sub>, LOD<sub>3</sub>, LOD<sub>6</sub>, LOD<sub>11</sub> and LOD<sub>14</sub>.

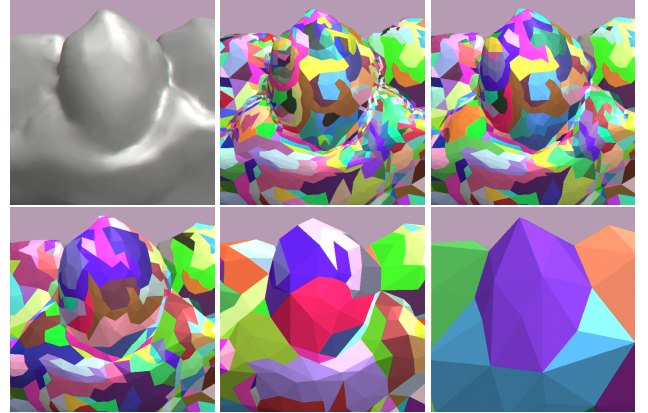


Figure 6: **Detail View of the Hierarchy.** From left to right and top to bottom, original mesh, LOD<sub>0</sub>, and convex decompositions of LOD<sub>0</sub>, LOD<sub>1</sub>, LOD<sub>2</sub>, LOD<sub>4</sub> and LOD<sub>7</sub>.

#### 4.5 Error Metrics

In this section, we present the parameters that must be computed after the hierarchy is created, in order to quantify the error for sensation preserving haptic rendering. The utilization of these parameters during the contact queries is explained in Sec. 5. To perform sensation preserving haptic rendering using a multiresolution hierarchy, we must measure the error that is introduced in the contact query and force computation and refine the query if the error is above a given tolerance. Once the hierarchies of LODs are created, with the resolution  $r$  computed for each LOD, we must compute several additional parameters for measuring the error:

1. The surface deviation,  $s$ , between every convex patch  $c$  and the original mesh. This is an upper bound on the size of the geometric surface detail lost during the simplification and filtering process.
2. A support area,  $D$ , for every vertex in the hierarchy. This value is used to calculate contact area at run-time. The support area  $D$  is computed for every vertex  $\mathbf{v}$  of the initial mesh  $M$  as the projected area onto the tangent plane of  $\mathbf{v}$  of the faces incident to  $\mathbf{v}$ , such that they are within a distance tolerance from  $\mathbf{v}$  along the direction of the normal  $\mathbf{N}$  of  $\mathbf{v}$ , and their normal lies inside the normal cone of  $\mathbf{N}$ . When an edge  $(\mathbf{v}_1, \mathbf{v}_2)$  is collapsed to a vertex  $\mathbf{v}_3$ , we assign to  $\mathbf{v}_3$  the minimum of the two support areas of  $\mathbf{v}_1$  and  $\mathbf{v}_2$ . We have typically used the same tolerance used in the contact queries (see Sec. 5) as the distance tolerance for this computation as well.
3. The maximum directed Hausdorff distance,  $h$ , computed for every convex piece  $C$ , from the descendant pieces of  $C$ .

The use of the surface deviation  $s$ , the support area  $D$ , and the resolution  $r$  of the LODs (whose computation is explained in Sec. 4.4) during the contact queries is described in Sec. 5.4. And the run-time use of the Hausdorff distance  $h$  is described in Sec. 5.3.

## 5 Contact Computation for Haptics

In this section, we describe how our collision detection algorithm uses the new multiresolution hierarchy described in Sec. 4 to compute contact response for haptic rendering. First, we describe the requirements of our collision detection system. Then, we present and analyze the data structures and algorithms. Finally, we show how to perform sensation preserving contact queries for force display.

### 5.1 Basic Haptic Rendering Framework

Our haptic rendering system uses a penalty-based force computation model, in which the amount of force displayed is proportional to the penetration depth or separation distance. Contact features within a given tolerance value are all considered as “contacts” for the purpose of force display. For more information about our haptic rendering framework, we refer readers to Appendix B on the conference proceedings CD.

We define the contact query between two objects  $A$  and  $B$  as  $Q(A, B, \delta)$ . From  $Q(A, B, \delta)$ , we obtain all local minima of the distance function between  $A$  and  $B$  that are closer than a distance tolerance  $\delta$ , as well as the associated contact information (i.e. distance, contact normal, etc.).  $Q(A, B, \delta)$  is performed by recursively traversing the bounding volume hierarchies (BVH) of  $A$  and  $B$  and performing “distance queries” for pairs of convex pieces. We define the distance query between two convex pieces  $a \in A$  and  $b \in B$ ,  $q(a, b, \delta)$ , as a boolean query that returns whether  $a$  and  $b$  are closer than  $\delta$ .

### 5.2 The Bounding Volume Test Tree

We use the concept of the *Bounding Volume Test Tree* (BVTT) [Larsen et al. 2000] to describe the algorithm and data structures used in our collision detection system. A node  $ab$  in the BVTT encapsulates a pair of pieces  $a \in A$  and  $b \in B$ , which might be tested with a query  $q(a, b, \delta)$ . Performing a contact query  $Q(A, B, \delta)$  can be understood as descending along the BVTT as long as the distance query  $q$  returns “true”. In the actual implementation, the BVTT is constructed dynamically while the contact query is performed. If the distance query result is “true” for a given pair, then the piece whose children have the lowest resolution is split. This

splitting policy yields a BVTT where the levels of the tree are sorted according to their resolution, as shown in Fig. 7. Nodes of the BVTT at coarser resolution are closer to the root. This is a key issue for optimizing our sensation preserving haptic rendering, because we obtain a BVTT where LODs with lower resolution and larger error are stored closer to the root. Descending the BVTT has the effect of selecting finer LODs.

As pointed out in Sec. 4.4, the top levels of the BVHs are “free” LODs, which are not obtained using our simplification algorithm, but pairwise convex hull merging. Therefore, the top levels of the BVTT have no associated metric of resolution. The boundary between “free” and regular LODs is indicated in Fig. 8 by the line  $\lambda$ .

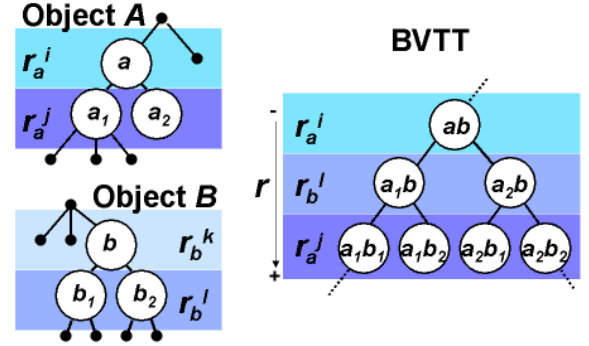


Figure 7: **Bounding Volume Test Tree.**

Instead of starting the contact query  $Q$  at the root of the BVTT every time, temporal coherence can be exploited using “generalized front tracking” [Ehmann and Lin 2001]. We store the “front” of the BVTT,  $\mathcal{F}$ , where the result of the distance query  $q$  switches from “true” to “false”, as shown in Fig. 8. The front is recorded at the end of a contact query  $Q_i$ , and the next query  $Q_{i+1}$  proceeds by starting recursive distance queries  $q$  at every node in the front  $\mathcal{F}$ .

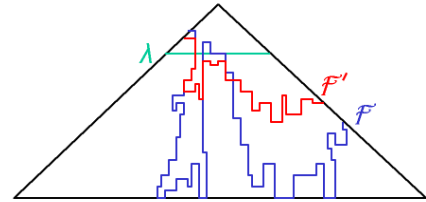


Figure 8: **Generalized Front Tracking of the BVTT.** The front of BVTT for the original model,  $\mathcal{F}$ , is raised up to the new front  $\mathcal{F}'$  using mesh simplification, since the contact queries can stop earlier using the sensation preserving selective refinement criterion.  $\lambda$  indicates the portion of the hierarchy constructed using the pairwise convex piece merging strategy, instead of mesh simplification, to form bounding volumes in the hierarchy.

### 5.3 Distance Query for Convex Pieces

In a contact query  $Q$  using BVHs, we need to ensure that if the distance query  $q$  is “true” for any node of the BVTT, then it must be “true” for all its ancestors. To guarantee this result with our multiresolution hierarchy, given a distance tolerance  $\delta$  for a contact query  $Q(A, B, \delta)$ , the distance tolerance  $\delta_{ab}$  for a distance query  $q(a, b, \delta_{ab})$  must be computed as:

$$\delta_{ab} = \delta + h(a^i, a) + h(b^j, b) \quad (5)$$



where  $h(a^i, a)$  and  $h(b^j, b)$  are maximum directed Hausdorff distances from the descendant pieces of  $a$  and  $b$  to  $a$  and  $b$  respectively. As explained in Sec. 4.5, these Hausdorff distances are pre-computed.

## 5.4 Sensation Preserving Selective Refinement

The time spent by a collision query  $Q$  depends directly on the number of nodes visited in the BVTT. Generalized front tracking considerably reduces the running time of  $Q$  when temporal coherence is high, which is the case in haptic rendering. Then, the time spent by  $Q$  is proportional to the size of the front  $\mathcal{F}$ . However, the cost is still  $O(nm)$  in the worst case, where  $n$  and  $m$  are the number of convex pieces of the objects.

In our system, we further take advantage of the multiresolution hierarchies to accelerate the performance of the query. The core idea of our sensation preserving selective refinement is that the nodes of the BVTT are only refined if the missing detail is perceptible. Note that the selective refinement does not apply to the “free” levels of the BVTT. Those levels must always be refined if the distance query  $q$  returns “true”.

As discussed in Sec. 3, the perceptibility of surface features depends on their size and the contact area. We have formalized this principle by devising a heuristic that assigns a functional  $\phi$  to surface features which is averaged over the contact area.

Given a node  $ab$  of the BVTT for which the distance query result  $q$  is “true”, we determine if the missing detail is perceptible by computing the functional of the missing detail and averaging it over the contact area of that node. For a convex piece  $a$  of the node  $ab$ , with resolution  $r_a$  and surface deviation from its descendent leaves  $s_a$ , we define the functional  $\phi$  as:

$$\phi_a = \frac{s_a}{r_a^2} \quad (6)$$

This definition of the functional can be regarded as a measure of the maximum volume of features that have been culled away in the convex piece  $a$ .

The online computation of the contact area for a pair of convex pieces is too expensive, given the time constraints of haptic rendering. Therefore, we have estimated the contact area by selecting the maximum support areas of the contact primitives (i.e. vertex, edge or triangle). As explained in Sec. 4.5, the support area  $D$  is stored for all vertices in the hierarchy. For edge or triangle contact primitives, we interpolate the support areas of the end vertices, using the barycentric coordinates of the contact point.

Given functional values of  $\phi_a$  and  $\phi_b$  for the convex pieces  $a$  and  $b$ , as well as support areas  $D_a$  and  $D_b$ , we compute a weighted surface deviation,  $s_{ab}^*$ , as:

$$s_{ab}^* = \frac{\max(\phi_a, \phi_b)}{\max(D_a, D_b)} \quad (7)$$

Note that  $s_{ab}^*$  can be considered as the surface deviation weighted by a constant that depends both on the resolution and the contact area. If  $s_{ab}^*$  is above a threshold  $s_0$ , the node  $ab$  has to be refined. Otherwise, the missing detail is considered to be imperceptible. The selection of the value of  $s_0$  is discussed in Sec. 6. As described in Sec. 4.4 and Sec. 4.5, the resolution  $r$ , the surface deviation  $s$ , and the support areas  $D$  are parameters computed as part of the preprocessing.

By using the described sensation preserving selective refinement of nodes of the BVTT, we achieve varying contact resolutions across the surfaces of the interacting objects, as shown in Fig. 1. In other words, every contact is treated independently, and its resolution is selected to cull away imperceptible local surface detail. As a consequence of the selective refinement, the active front of the

BVTT,  $\mathcal{F}'$ , is above the original front  $\mathcal{F}$  that separates nodes with “true” result for distance queries  $q$  from nodes with “false” result. The front does not need to reach the leaves of the BVTT as long as the missed detail is imperceptible, as depicted in Fig. 8. This approach results in a much faster processing of contact queries.

## 5.5 LOD Interpolation

A major issue in systems that use multiresolution hierarchies is the discontinuity that arises when the algorithm switches between different LODs. This problem is known as “popping” in multiresolution (visual) rendering. In haptic rendering its effects are discontinuities in the delivered force and torque, which are perceived by the user.

We have addressed the problems of discontinuities by interpolating contact information (e.g. contact normal and distance) from different LODs. When the sensation preserving selective refinement determines that no more refining is necessary, we perform a conservative refinement and compute contact information for the children of the current node of the BVTT. The contact information is interpolated between the two levels.

Naturally, LOD interpolation increases the number of nodes of the BVTT that are visited. However, for complex models and/or complex contact scenarios, the gain obtained from the selective refinement still makes sensation preserving simplification significantly outperform the exact technique, as presented in Sec. 6.

## 6 Implementation and Results

In this section we describe some of the models and experiments we have used to validate our sensation preserving simplification for haptic rendering.

### 6.1 System Demonstration

We have applied our sensation preserving simplification for haptic rendering on the models listed in Table 2. The complexity and surface detail of these models can be seen in Fig. 9.

Models	Lower Jaw	Upper Jaw	Ball Joint	Golf Club	Golf Ball
Orig. Tris	40180	47339	137060	104888	177876
Orig. Pcs	11323	14240	41913	27586	67704
Simp. Tris	386	1038	122	1468	826
Simp. Pcs	64	222	8	256	64
$r_1$	144.49	117.5	169.9	157.63	216.3
$r_\lambda$	12.23	19.21	6.75	8.31	7.16
Free LODs	6	8	3	8	6
LODs	15	15	17	16	18

Table 2: **Models and Associated Hierarchies.** *The number of triangles (Orig. Tris) and the number of convex pieces (Orig. Pcs) of the initial mesh of the models; the number of triangles (Simp. Tris) and the number of convex pieces (Simp. Pcs) of the coarsest LOD obtained through simplification; resolution ( $r_1$  and  $r_\lambda$ ) of the finest and coarsest LOD obtained through simplification; and “free” LODs and total number of LODs. The resolutions are computed for a radius of 1 for all the objects.*

As seen from the results in Table 2, we are able to simplify the models to LODs with only a couple hundred convex pieces or less. In other words, the sensation preserving selective refinement can be applied at earlier stages in the contact query, and this allows more aggressive culling of parts of the BVTT whenever the perceivable error is small.

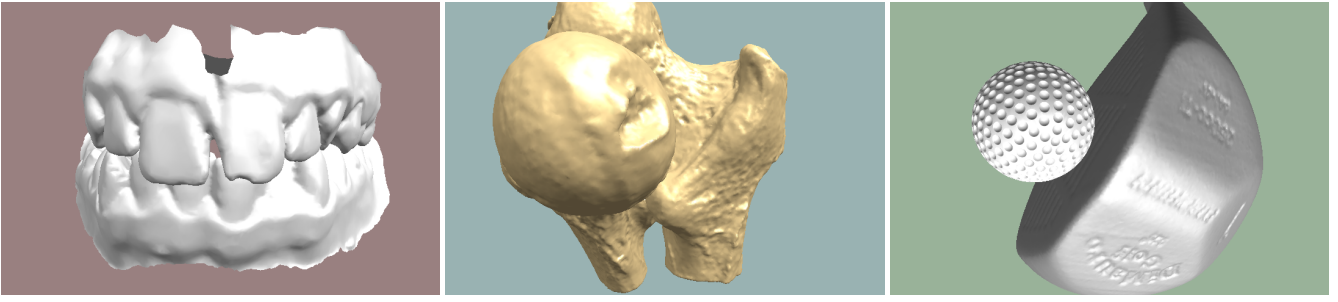


Figure 9: **Benchmark Models.** From left to right, moving upper and lower jaws, interlocking ball joints and interacting golf club and ball.

With the aforementioned models, we have performed the following proof-of-concept demonstrations:

- Moving upper and lower jaws.
- Golf club tapping a golf ball.
- Interlocking ball joints.

These demonstrations have been performed using our sensation preserving haptic rendering, a six-DOF *Phantom<sup>TM</sup>* haptic device, a dual Pentium-4 2.4GHz processor PC with 2.0 GB of memory and a NVidia GeForce-4 graphics card, and Windows2000 OS. Our implementation, both for preprocessing and for the haptic rendering, has been developed using C++. For the force computation of the haptic rendering we have used penalty methods based on the contact tolerance  $\delta$  [Kim et al. 2002]. We choose the value of  $\delta$  so that the maximum force of the haptic device is exerted for a 0 contact distance with the optimal value of stiffness.

## 6.2 Studies on Perceivable Contact Information

The performance of the sensation preserving haptic rendering is heavily determined by the selection of the threshold of weighted surface deviation  $s_0$ . If the chosen value is too high, the perceived contact information will deviate too much from the exact contact information. On the other hand, if the value is too low and the simplified models used are moderately complex consisting of more than a thousand convex pieces, the contact query will no longer be executable at the required rate. This severely affects the realism of haptic perception.

We have designed a scenario where we could test the fidelity of the sensation preserving selective refinement. In this scenario, users can touch the model of the golf ball with an ellipsoid. The ellipsoid has varying curvature, implying the existence of a wide range of contact scenarios, where the selective refinement will stop at varying LODs.

12 users experimented with this scenario and reported that the perception of contact information hardly varied for values of  $s_0$  in the range between 0.025 and 0.05 times the radius of the models. (For readers interested in the detail of experimental data, please refer to Appendix C on the conference proceedings CD).

## 6.3 Performance Demonstration

Based on the value of  $s_0$  obtained from the studies, we have successfully applied our algorithm to haptic rendering of object-object interaction on the benchmarks listed in Sec 6.1. We have also performed an analysis on contact forces and running time for the demonstrations previously mentioned. We have compared force profiles and statistics of the contact query of interactive haptic demonstrations with offline executions, using smaller error tolerances and an exact method. By exact, we mean that the distance computation for force display is accurate [Ehmann and Lin 2001]. In particular, Fig. 10 shows the contact profile, including the force

profile, the query time and the size of the front of the BVTT, for 200 frames of the moving jaws simulation. Fig. 11 shows the contact profile for 300 frames of the simulation on the golf scene. The contact profile of interlocking joints is quite similar to that of the interacting golf club and golf ball, thus omitted here.

For both scenarios, the simulation with  $s_0 < 5\%$  of the radii of the models has been performed in real time, using a haptic device to control the motion of the upper jaw and the golf club respectively, and to display the contact forces to the user. The trajectories of the upper jaw and the golf club are recorded and played back to perform the rest of the simulations offline, since the exact method was too slow to be used to keep up with the force update. As shown in the figure, we observed a gain of two orders of magnitude in the query time between the interactive haptic rendering using our approach and the exact offline simulation. Note that the spikes in the contact query time present in Fig. 10 and Fig. 11 result from lack of coherence in the traversal of the BVTT. As reflected in the graphs, the query time is more susceptible to lack of coherence when the error tolerance is lower.

Using our approach, the force profiles of simulations with varying error tolerances less than 5% of the radii of the models exhibit similar and sometimes nearly identical patterns as that of the original models. This resemblance validates our hypothesis on the haptic perception of contacts, inferred from human tactual perception.

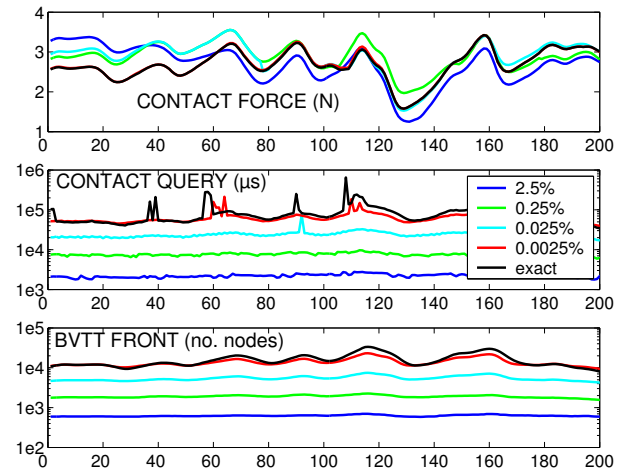


Figure 10: **Contact Profile for Moving Jaws.** Top: The profiles of the contact forces displayed using simplification, with varying error tolerances up to 2.5% of the radii of the jaws, all show very similar patterns. This similarity implies that the sensations of shape provided to the user are nearly identical. Middle: A log plot of contact query time using simplification with various error tolerances shows up to two orders of performance improvement. Bottom: The number of nodes in the front of the BVTT is also reduced by more than a factor of 10.

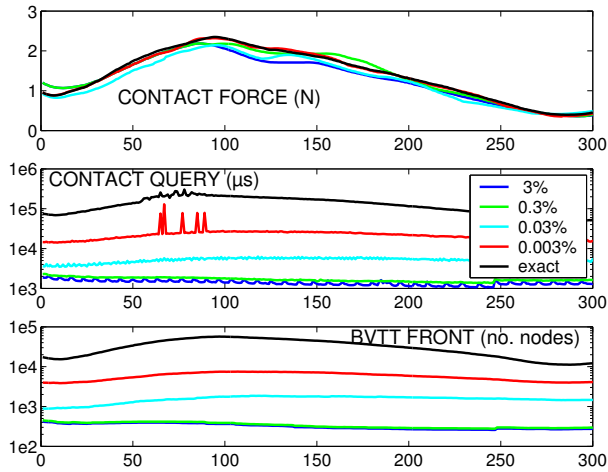


Figure 11: **Contact Profile for Golf Scene.** *Top: The profiles of the contact forces displayed using simplification, with varying error tolerances up to 3% of the radius of the ball, show nearly identical patterns. Middle: A log plot of contact query time using simplification with various error tolerances shows more than two orders of performance improvement. Bottom: The number of nodes in the front of the BVTT is reduced by nearly a factor of 100.*

## 7 Discussion and Analysis

In this section, we compare our approach with previous work in related areas and analyze the various algorithmic and performance issues.

### 7.1 Comparison with Related Work

**Mesh Decimation and Filtering:** The construction of our multiresolution hierarchy can be compared with both mesh decimation techniques and mesh filtering techniques. Representations obtained through these techniques might present better results than our hierarchies in certain aspects.

LODs created using our filtered edge collapse operation will have a larger surface deviation than LODs of traditional mesh decimation. This deviation inevitably results from combining decimation and filtering. In our framework, the detail at high resolution is filtered independently of its magnitude, while mesh decimation techniques will preserve detail to minimize the surface deviation. The elimination of the detail has beneficial consequences in the creation of the BVH and does not reflect on the output quality of the haptic rendering, since the filtered detail is quantified and taken into account in the sensation preserving refinement. Besides, multiresolution representations obtained through mesh decimation techniques are not valid by themselves to perform efficient contact queries.

Representations obtained through filtering appear smoother than our representations. The reduction in visual smoothness occurs because we use fewer samples (i.e. vertices) to represent meshes with the same frequency content. This approach is advantageous for our application, because it accelerates the contact queries. In addition, we have also presented a definition that allows comparing the resolution of the detail of the objects in contact.

**Contact Queries for Haptic Rendering:** As mentioned earlier, the running time of any contact query algorithm depends on both the input and output size of the problem. Given two polyhedra, characterized by their combinatorial complexity of  $n$  and  $m$  polygons, the contact query algorithm can have an output size and a run-time complexity as high as  $O(nm)$ .

The discretized approximation presented by McNeely et al. [1999] can avoid direct dependency on the input size of the problem by limiting the number of points sampled and the number of voxels generated. However, its performance on complex contact scenarios with many collisions is unknown. Both approaches by Gregory et al. [2000] and Kim et al. [2002] are limited to relatively simple models or modestly complex contact scenarios and do not scale well to highly complex object-object interaction.

In contrast, our approach, by reducing the combinatorial complexity of the input based on the contact configuration at each local neighborhood of (potential) collisions, automatically decreases the output size as well. In addition, its selection of LODs is contact-dependent to minimize the perceived difference in force display, while maximizing the amount of simplification and performance gain possible. This method is perhaps the first “contact-dependent simplification” algorithm for collision queries as well.

### 7.2 Generalization of the Algorithmic Framework

Due to the hard time constraints of haptic rendering, we have chosen a collision detection algorithm using BVHs of convex hulls and automatic convex surface decomposition. The choice of collision detection algorithm imposes convexity constraints on the simplification process and the hierarchy construction.

These constraints are rather specific. However, the algorithmic framework for generating the multiresolution hierarchy for sensation preserving contact queries that we have developed and presented in this paper is general and applicable to other collision detection algorithms.

Furthermore, although we focus on contact determination for haptic rendering in this paper, our approach for sensation preserving simplification can also be applied to other types of proximity queries, such as penetration depth estimation. Our approach can be generalized to multiresolution collision detection by automatically identifying superfluous proximity information and thus cleverly selecting the appropriate resolutions for performing the queries at different locations across the objects’ surfaces. This key concept can significantly accelerate the performance of any proximity query algorithm, as we have demonstrated in this paper.

A further analysis of the applicability of sensation preserving simplification to multiresolution collision detection for rigid body simulation has been conducted [Otaduy and Lin 2003]. More study is needed on the relationship between our sensation preserving error metrics and contact force models of rigid body simulations, but the preliminary results are promising.

### 7.3 Integration with Graphic Rendering

The LODs selected for haptic rendering are decoupled from the representation of the objects used for visual rendering. This difference in representations can potentially lead to some inconsistency between visual and haptic display, such as the existence of visual gaps when the displayed forces indicate that the objects are in contact. Future investigation is required for a better integration of multi-sensory cues in a multimedia environment.

### 7.4 Other Limitations

Our approach can handle triangular meshes with two-manifolds and boundaries. The current implementation is limited to polygonal models with connectivity information. As with all simplification algorithms that generate levels of detail offline, our approach has the similar memory requirement.

## 8 Summary and Future Work

We have presented a novel sensation preserving simplification to accelerate collision queries for force display of complex object-object interaction. The resulting multiresolution hierarchy constructed with a formal definition of resolution enables us to compute contact information at varying resolutions independently for different locations across the object surfaces. By selecting the most aggressive simplification possible on the fly based on the contact configuration, this approach considerably improves the run-time performance of contact queries. It makes haptic rendering of the interaction between highly complex models possible, while producing only relatively imperceptible changes in the force display. Our approach can also be easily extended to perform time-critical haptic rendering while optimizing the fidelity of the force display, using the technique described in [Otaduy and Lin 2003].

This new ability to perform force display of complex 3D object interaction enables many exciting applications, where haptic rendering of point-object interaction is insufficient. In addition to further optimizing and increasing the performance of sensation preserving haptic rendering, this research may be extended in several possible directions. These include haptic display of friction and textures exploiting our current framework, applications of 6-DOF haptic rendering to scientific visualization, engineering prototyping, and medical training, as well as formal user studies and task performance analysis.

## Acknowledgments

This research is supported in part by a fellowship of the Government of the Basque Country, National Science Foundation, Office of Naval Research, U.S. Army Research Office, and Intel Corporation. We would like to thank Stephen Ehmann and Young Kim for their help on integrating SWIFT++ and DEEP with our 6-DOF haptic rendering framework. We are also grateful to Dinesh Manocha, Russell Taylor, Mark Foskey, and the anonymous reviewers for their feedback on the earlier drafts of this paper.

## References

- BROOKS, JR., F. P., OUH-YOUNG, M., BATTER, J. J., AND KILPATRICK, P. J. 1990. Project GROPE — Haptic displays for scientific visualization. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, F. Baskett, Ed., vol. 24, 177–185.
- CHAZELLE, B., DOBKIN, D., SHOURABOURA, N., AND TAL, A. 1997. Strategies for polyhedral surface decomposition: An experimental study. *Computational Geometry: Theory and Applications* 7, 327–342.
- EHMANN, S., AND LIN, M. C. 2000. Accelerated proximity queries between convex polyhedra using multi-level voronoi marching. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- EHMANN, S., AND LIN, M. C. 2001. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum (Proc. of Eurographics'2001)* 20, 3.
- EL-SANA, J., AND VARSHNEY, A. 2000. Continuously-adaptive haptic rendering. *Virtual Environments 2000*, pp. 135–144.
- GARLAND, M., AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *Proc. of ACM SIGGRAPH*, 209–216.
- GOTTSCHALK, S., LIN, M., AND MANOCHA, D. 1996. OBB-Tree: A hierarchical structure for rapid interference detection. *Proc. of ACM SIGGRAPH*, pp. 171–180.
- GREGORY, A., MASCARENHAS, A., EHMANN, S., LIN, M. C., AND MANOCHA, D. 2000. 6-dof haptic display of polygonal models. *Proc. of IEEE Visualization Conference*.
- GUIBAS, L., HSU, D., AND ZHANG, L. 1999. *H-Walk: Hierarchical distance computation for moving convex bodies*. *Proc. of ACM Symposium on Computational Geometry*.
- GUSKOV, I., SWELDENS, W., AND SCHRODER, P. 1999. Multiresolution signal processing for meshes. *Proc. of ACM SIGGRAPH*, pp. 325–334.
- HOFF, K., ZAFERAKIS, A., LIN, M., AND MANOCHA, D. 2001. Fast and simple geometric proximity queries using graphics hardware. *Proc. of ACM Symposium on Interactive 3D Graphics*.
- HOLLERBACH, J., COHEN, E., THOMPSON, W., FREIER, R., JOHNSON, D., NAHVI, A., NELSON, D., AND II, T. T. 1997. Haptic interfacing for virtual prototyping of mechanical CAD designs. *CDROM Proc. of ASME Design for Manufacturing Symposium*.
- HUBBARD, P. 1994. *Collision Detection for Interactive Graphics Applications*. PhD thesis, Brown University.
- KIM, Y., OTADUY, M., LIN, M., AND MANOCHA, D. 2002. 6-dof haptic display using localized contact computations. *Proc. of Haptics Symposium*.
- KLATZKY, R., AND LEDERMAN, S. 1995. Identifying objects from a haptic glance. *Perception and Psychophysics* 57, pp. 1111–1123.
- KLOSOWSKI, J., HELD, M., MITCHELL, J., SOWIZRAL, H., AND ZIKAN, K. 1998. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Trans. on Visualization and Computer Graphics* 4, 1, 21–37.
- LARSEN, E., GOTTSCHALK, S., LIN, M., AND MANOCHA, D. 2000. Distance queries with rectangular swept sphere volumes. *Proc. of IEEE Int. Conference on Robotics and Automation*.
- LIN, M., AND GOTTSCHALK, S. 1998. Collision detection between geometric models: A survey. *Proc. of IMA Conference on Mathematics of Surfaces*.
- LOMBARDO, J. C., CANI, M.-P., AND NEYRET, F. 1999. Real-time collision detection for virtual surgery. *Proc. of Computer Animation*.
- LUEBKE, D., AND HALLEN, B. 2001. Perceptually driven simplification for interactive rendering. *Rendering Techniques; Springer-Verlag*.
- LUEBKE, D., REDDY, M., COHEN, J., VARSHNEY, A., WATSON, B., AND HUEBNER, R. 2002. *Level of Detail for 3D Graphics*. Morgan-Kaufmann.
- MARK, W., RANDOLPH, S., FINCH, M., VAN VERTH, J., AND TAYLOR II, R. M. 1996. Adding force feedback to graphics systems: Issues and solutions. *Proc. of ACM SIGGRAPH*, 447–452.
- MCNEELY, W., PUTERBAUGH, K., AND TROY, J. 1999. Six degree-of-freedom haptic rendering using voxel sampling. *Proc. of ACM SIGGRAPH*, 401–408.
- OKAMURA, A., AND CUTKOSKY, M. 1999. Haptic exploration of fine surface features. *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2930–2936.
- O'SULLIVAN, C., AND DINGLIANA, C. 2001. Collisions and perception. *ACM Trans. on Graphics* 20, 3, pp. 151–168.
- OTADUY, M. A., AND LIN, M. C. 2003. CLODs: Dual hierarchies for multiresolution collision detection. *UNC Technical Report TR03-013*.
- PAI, D. K., AND REISSEL, L. M. 1997. Haptic interaction with multiresolution image curves. *Computer and Graphics* 21, 405–411.
- REDON, S., KHEDDAR, A., AND COQUILLART, S. 2002. Fast continuous collision detection between rigid bodies. *Proc. of Eurographics (Computer Graphics Forum)*.
- RUPINI, D., KOLAROV, K., AND KHATIB, O. 1997. The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH*, 345–352.
- SALISBURY, J. K. 1999. Making graphics physically tangible. *Communications of the ACM* 42, 8.
- TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proc. of ACM SIGGRAPH*, 351–358.

## APPENDIX A: Pseudo Code for Multiresolution Hierarchy Generation

```

Compute surface convex decomposition
Dump initial LOD
 $n$  = number of convex pieces
Compute resolution of edges
Initialize edges as valid
Create priority queue
while Valid(Top(queue)),
  if FilteredEdgeCollapse(Top(queue)) then
    PopTop(queue)
    Recompute resolution of affected edges
    Reset affected edges as valid
    Update priority of affected edges
    Attempt merge of convex pieces
  else
    Set Top(queue) as invalid
    Update priority of Top(queue)
  endif
  if Number of pieces  $\leq n/2$  then
    Dump new LOD
     $n$  = number of convex pieces
  endif
endwhile
while Number of pieces  $> 1$ ,
  Binary merge of pieces
endwhile

```

ALGORITHM 0.1: Creation of the Hierarchy

## APPENDIX B: Overview of the Haptic Rendering Framework

In this appendix, we give an overview of our six-degree-of-freedom haptic rendering framework for displaying force and torque between two objects in contact. We compute the displayed force based on the following steps:

1. Perform a contact query between two objects  $A$  and  $B$ , collecting the set  $S$  of nodes of the front of the BVTT that are inside a distance tolerance  $\delta$ , at the appropriate resolution.
2. For all nodes  $ab \in S$ , compute the contact information:
  - (a) If the pair  $ab$  is disjoint, compute the distance, the contact normal and the closest points and features (i.e. vertex, edge or face).
  - (b) If the pair  $ab$  is intersecting, compute the penetration depth, the penetration direction, and penetration features<sup>1</sup>.
3. Cluster all contacts based on their proximity.
4. Compute a representative contact for each cluster, averaging contact information weighted by the contact distance.
5. Compute a penalty-based restoring force at the representative contact of each cluster.
6. Apply the net force and torque to the haptic probe.

<sup>1</sup>By the penetration features, we mean a pair of features on both objects whose supporting planes realize the penetration depth.

## APPENDIX C: Studies on Perceivable Contact Details

In this appendix we briefly describe an informal user study that has been conducted to test the fidelity of the sensation preserving simplification for haptic rendering, and also to help us identify what are the error tolerances for which the missing surface detail is not perceptible to the users.

The scenario of our experiment consists of a golf ball (please refer to the paper for statistics of the model) that is explored with an ellipsoid as shown in Fig. 1. The ellipsoid consists of 2000 triangles and one single convex piece. For simplicity, we have only applied the simplification to the golf ball and left the ellipsoid invariant. Thus, the fidelity of the sensation preserving simplification relies on the adequacy of the resolution of the golf ball that is selected at each contact.

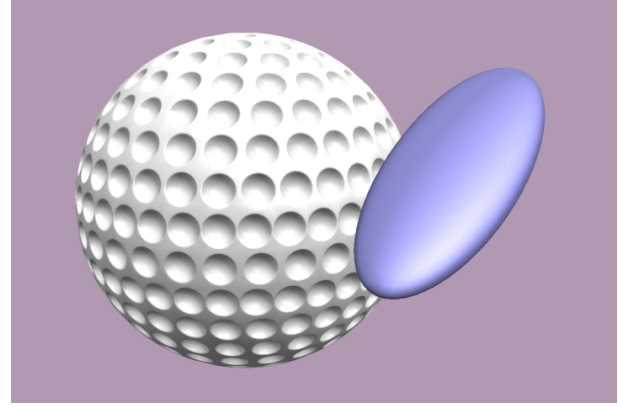


Figure 1: Scenario of the User Study.

12 users have experimented with this scenario. They were asked to identify at what value of the threshold  $s_0$  of the sensation preserving selective refinement they started perceiving a deviation in the perception of the surface detail of the golf ball. The values of  $s_0$  were in the range from 0.05% to 20% of the radius of the ball. In Table 1 we indicate how many users picked each threshold value. The users also reported that the main characteristic that they explored was the perceptibility of the dimples of the golf ball.

$s_0$	$\geq 10\%$	5%	2.5%	1%	$\leq 0.5\%$
no. users	0	4	7	1	0

Table 1: Results of the User Study.

Based on the results from this informal user study, we selected values of  $s_0$  in the range of 2.5% to 5% of the radii of the models for the demonstrations presented in the paper.



## DETC2004/DAC-57507

### A HAPTIC SYSTEM FOR VIRTUAL PROTOTYPING OF POLYGONAL MODELS

**David E. Johnson**  
School of Computing  
University of Utah  
Salt Lake City, USA  
dejohnso@cs.utah.edu

**Peter Willemsen**  
School of Computing  
University of Utah  
Salt Lake City, USA  
willemsn@cs.utah.edu

**Elaine Cohen**  
School of Computing  
University of Utah  
Salt Lake City, USA  
cohen@cs.utah.edu

#### ABSTRACT

Virtual prototyping attempts to replace physical models with virtual models for the purpose of design evaluation. One task a virtual prototyping environment can address is the accessibility of the components of a mechanical system. In this paper, we demonstrate a haptics-based virtual prototyping system for finding collision-free paths for moving models in complex polygonal environments. The system can handle models and environments with hundreds of thousands of triangles, and augments innate human talents at searching for collision-free paths.

#### INTRODUCTION

While mechanical model design increasingly relies upon computer-aided design (CAD) and sophisticated simulation programs, physical prototypes still play an important role in design evaluation. Since physical prototypes are expensive to build, and may take significant time to manufacture, virtual prototyping environments attempt to replace as much functionality of the physical prototypes as possible with a virtual prototype.

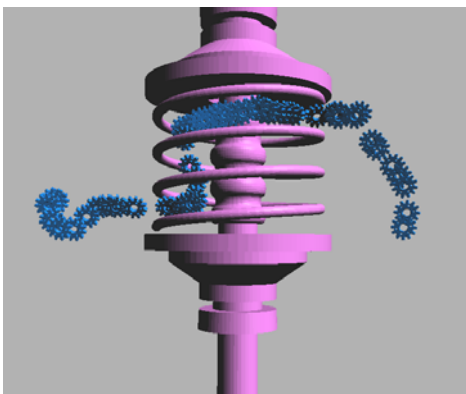


Figure 1: GUIDING A GEAR PAST A SPRING PART IN A SAMPLE VIRTUAL PROTOTYPING SESSION

Accessibility is a design evaluation task that is difficult to simulate on a computer. Two main reasons preclude easy automatic simulation:

- Computation of a collision-free path for complex models is difficult and time-consuming,
- Modeling human manipulation capabilities is difficult.

We propose a haptic system for virtual prototyping that allows human guidance and intuition in developing a collision-free path between virtual models (Figure 1). This type of system provides the intuitive usability of a physical prototype, yet retains the cost and time advantages of a computer model.

Our system has several advantages over existing haptic systems:

- Exact distances between models are computed, so the simulation is accurate to the resolution of the models.
- Haptic feedback is provided before models collide, so no invalid interpenetration of the models needs to occur.
- Models in the environment are freely movable, providing real-time adjustment to the scene as desired.

The virtual prototyping environment is demonstrated on an example mechanical system with hundreds of thousands of triangles.

#### BACKGROUND

Virtual prototyping of accessibility tasks is closely related to the area of path planning. The main distinction is that in virtual prototyping, there is some assumption of human involvement, whereas path planning is usually more of an automatic technique.

Path planning methods fall into two main categories – global methods and local methods. Global methods try to sample the configuration space of the model and the environment, and then connect together collision-free instances into a collision-free path[1][2][3]. Local methods use local

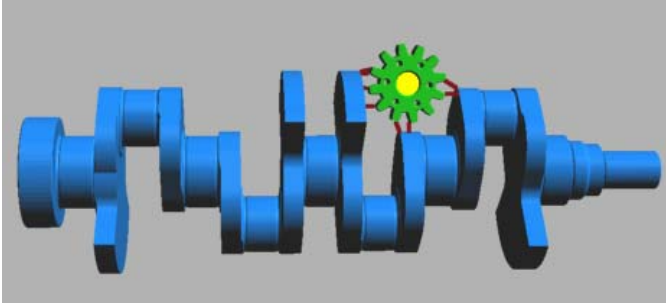


Figure 2: THE SET OF LMDs PRODUCES REPULSIVE FORCES THAT KEEP MODELS FROM COLLIDING.

repulsion techniques to avoid collisions, while being drawn towards a distant goal[4]. However, these local methods can get stuck in local minima and never reach the goal. Our haptics system is similar to the local path planning approach, but uses human guidance to push models past local minima.

Haptics[5] has been proposed as a virtual prototyping interface in prior work. Hollerbach et al. [6] computed fast penetration depths between a point and a spline model to create a sensation of contact with the model, as did researchers at Ford Motor company in [7]. Nelson[8] developed a fast technique for haptic rendering of two spline models in contact and also adapted the method to moving linkages[9]. Ruspini created a haptic rendering system for point interactions with complex, moving polygonal models[10].

McNeely used a six degree-of-freedom (DOF) device to manipulate a point-sampled model with a large-scale voxel environment[11]. The environment in that system is static; however, that approach guarantees a worst case computation time, important for reliable haptic rendering. They report that they were able to use haptics to find collision-free paths in complex environments for which global path planning algorithms failed.

## APPROACH

Our virtual prototyping system is based on a 6-DOF haptic rendering method for complex polygonal models[12][13]. That haptic rendering method uses techniques for finding local minimum distances (LMDs) between polygonal models and computing local updates to the LMDs while the models move. This paper uses those techniques in conjunction with methods for maintaining a collision-free path and visualizing the result to produce a virtual prototyping environment.

The basic haptic rendering approach is to find LMDs between the moving polygonal model and the environment. Each LMD acts like a spring guiding the moving model away from collision with the environment (Figure 2), while human interaction guides the model towards its goal. In order to maintain haptic rates, the set of LMDs are updated using a local gradient search.

These two steps, a search for LMDs and a local update, are summarized in the next two sections, please refer to the original papers for a more complete description.

### Spatialized Normal Cone Pruning

The haptic rendering searches for LMDs using spatialized normal cones[14], which hierarchically encapsulate the position and spread of surface normals over a model. The algorithm uses surface normal information to find portions of each model that

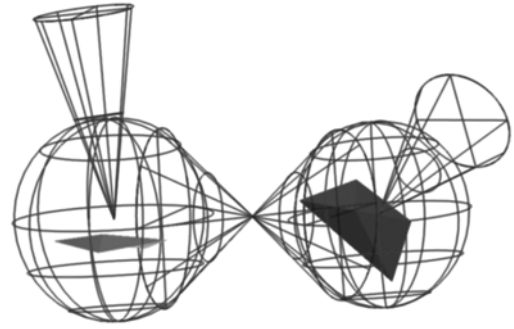


Figure 3: A SPHERE BOUNDS A PORTION OF POLYGONAL GEOMETRY AT A NODE OF THE HIERARCHY. THE NORMAL SPAN OF THE NODE GEOMETRY IS ENCAPSULATED IN A NORMAL CONE. THE RANGE OF POSSIBLE LINES BETWEEN CLOSEST POINTS BETWEEN NODES IS BOUNDED BY A DOUBLE CONE BETWEEN BOUNDING SPHERES. AT EACH PAIR OF NODES, THE ALGORITHM CHECKS TO SEE IF THE NORMAL SPANS OVERLAP AND ARE ALONG A VECTOR CONTAINED WITHIN THE SPAN OF POSSIBLE MINIMUM DISTANCE LINES.

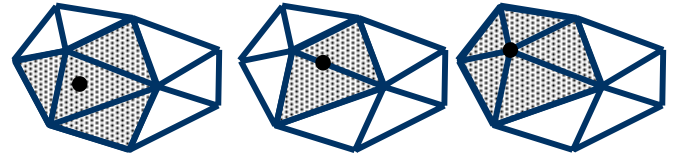


Figure 4: THE LOCAL GRADIENT SEARCH CHECKS NEIGHBORING FEATURES FOR NEW CLOSEST POINTS. WHEN THE LAST POINT IS ON A FACE, NEIGHBORING FACES ARE CHECKED. WHEN THE LAST POINT IS ON AN EDGE, ONLY TWO NEIGHBORING FACES ARE USED. THE LAST CASE IS FOR A VERTEX, WHERE ALL THE TOUCHING FACES ARE CHECKED.

point towards each other and are collinear with the line between potential closest points on the models, a condition which represents a local minimum distance (Figure 3). Nodes in the hierarchy that cannot form a local minimum distance solution are pruned, while remaining nodes are subdivided and the algorithm recursively applied. At leaf triangles, exact tests compute whether a local minimum distance solution exists.

### Local Search with Gradient Descent

Each LMD pair from the normal cone computation represents locally closest points on the two models. The LMD pairs are updated at haptic rates by a local gradient search algorithm[13]. The gradient search checks neighboring features on the polygonal model for a new pair of points that are closer than the current pair (Figure 4). This search repeats for each LMD until the LMD converges to a new distance.

### Combined Search and Update

As fast as the global normal cone search can compute LMDs, it introduces new LMDs and deletes those that are no longer needed. In our examples, this update happened at 10-100Hz. The local gradient search then updated the LMDs at over 1000Hz for stable haptic rendering.

### Force and Torque

Each LMD pair within a cutoff distance contributes to the total force and torque being applied to the model under the control of the force-feedback device. We approximate the center of mass and moments of the moving model using an oriented bounding box approximation. These forces and torques are reflected back to the user by the 6-DOF force-feedback device.



Figure5: THE 6-DOF PHANTOM.

## SYSTEM DESIGN

Our virtual prototyping system is based on a Sensable six DOF PHANTOM haptic interface (Figure 5). The computations run on a dual processor Pentium 4 2.4 GHz Linux computer with a gigabyte of RAM and a GeForce 4 Ti 4400 graphics card.

The software architecture uses a multi-threaded design to handle the different rates of computation needed for graphics, global normal cone search, and the local updates with force computations. The application uses three threads: a global search thread, a local update thread, and a graphics thread. This architecture allows us to restrict the computational load of the graphics and global threads, and let the local update run as fast as possible. On a two-processor system, this translates into the local update getting one processor to itself and the other threads sharing the second processor.

## VIRTUAL PROTOTYPING SYSTEM

Since we compute LMDs while the moving model is still some distance from the environment models, haptic forces are used to guide the moving model away from collision with the environment (Figure 6). The onset distance for forces is adjustable, so the user can decide how much clearance between models is desired during testing. In general, the LMDs tend to approximate the local distance field between the models, and the forces tend to push the moving model towards the medial axis between the models. Since the medial axis is the surface of maximum clearance between models, these forces tend to guide the moving model towards the safest path.

### Collision-Free Path

While the test object is being moved by the haptic interface, its position and orientation are stored in a buffer. This buffer allows the motion of the test object to be played back for review, analysis, or further modification.

If the moving model is forced to penetrate an environment model by the user, the simulation is no longer valid. A collision state is detected and the simulation is rolled back, using the stored positions and orientations in the buffer, until the model state is valid. The simulation can then resume, and the user can try new approaches for finding a collision-free path. This means that the path stored by our virtual prototyping program

is always valid, and if the moving model can reach its goal, the problem has been solved.

## Detecting Collisions

Collisions are detected when the smallest LMD falls below an adjustable parameter. This parameter can represent error in the fit of the polygonal model to an original CAD model, or it can represent a desired minimum clearance between models. Detecting collisions in this fashion, instead of with actual model intersection, provides more control over simulation accuracy.

## Path Visualization

Since we store model positions and orientations during the simulation, a sampling of the path of the model can be visualized. Drawing a copy of the moving model at each sampled location (or some subset), allows the user to check the validity of the collision-free path, and to examine any unusual maneuvering needed to safely guide the model. One drawback is that if many positions are visualized simultaneously, the frame rate of the display can slow.

## Interface

The main interface is the 6-DOF haptic interface. After loading models into the environment, the position of the currently selected model is controlled by the user moving the haptic interface. The selected model is changed with keyboard commands, so any model in the environment is freely movable by the haptic interface.

Keyboard commands also control the recording of the collision-free path, stopping of recording, and visualization of the path in playback mode.

The current set of LMDs are displayed as red lines between the two models. They help provide feedback cues to the relative positions of the two models in the absence of stereo viewing.

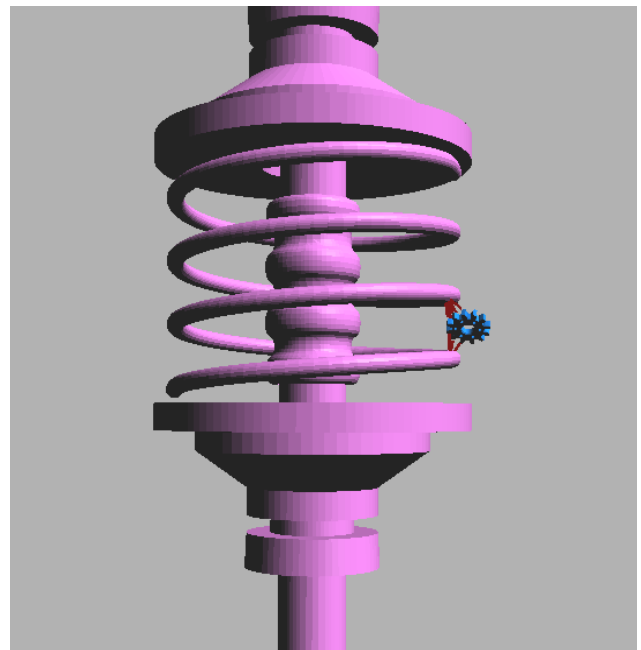


Figure 6: THE LMDs PROVIDE GUIDANCE IN REGIONS OF LIMITED CLEARANCE.



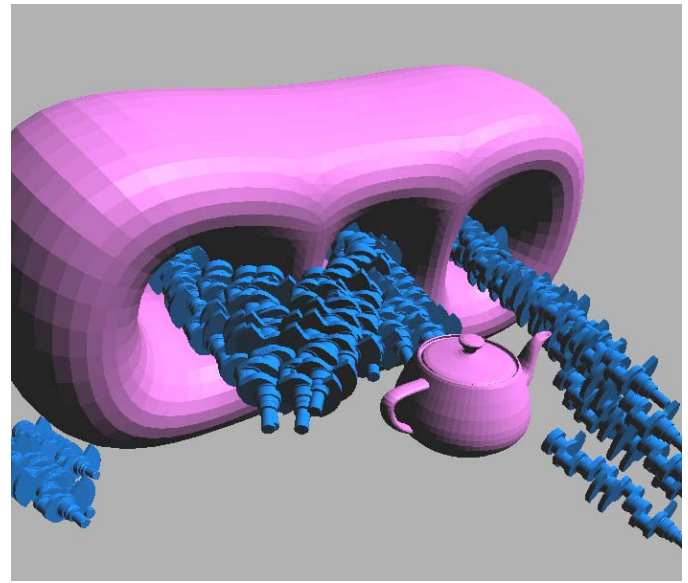
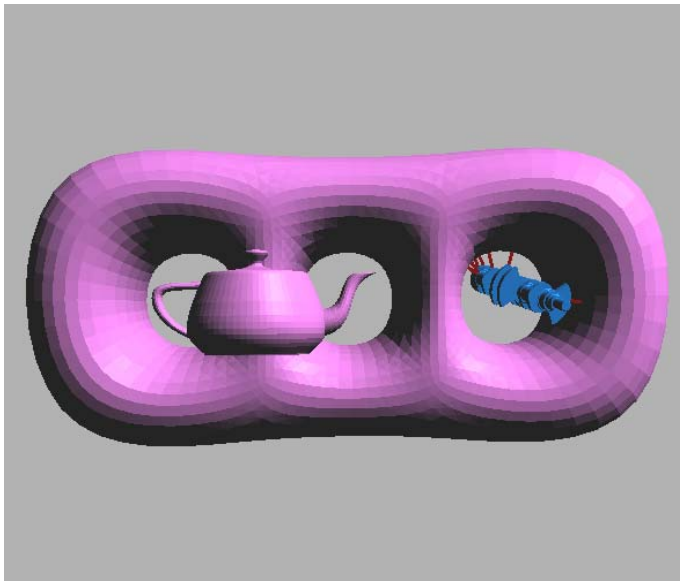


Figure 7: HAPTICS GUIDES THE CRANK MODEL THROUGH THE HOLES WHILE AVOIDING THE TEAPOT MODEL. THE FINAL PATH IS VISUALIZED AS A SAMPLING OF MODEL POSITIONS DURING THE SIMULATION.

## RESULTS

We tested our virtual prototyping system with a variety of models. In the tests, we threaded one moving model around and inside the environment model. The tests were such that the moving models needed to be oriented properly to fit through the gaps in the environment model, so that success without haptic feedback would be difficult. The model and environment sizes ranged from around 6,000 triangles to 113,000 triangles. In all the cases, we were able to intuitively find a collision-free path to accomplish the goal.

### Gear-Spring Part

In the first test, we used a gear model with 6,300 triangles and a spring part model with 23,500 triangles. The goal was to have the gear enter the spring, traverse down the body, and then exit the spring. There was limited clearance between the gear and the spring and spring body, as well as between the coils of the spring. The gear model had to be turned almost flat to fit through the coils. However, with haptics, the forces naturally guided the model (Figure 6) along a safe path (Figure 1).

### Crank-Holes-Teapot

In this example, we used a crank model with 45,000 triangles, a three-holed block with almost 12,000 triangles, and a teapot model with 5,600 triangles. We used the haptic interface to position the block and the teapot in such a way that there was not a clear path from one hole to the next. The goal in this test was to thread the crank model through all three holes while avoiding the teapot.

The haptic interface provided enough cues to the user to find a path out of the middle hole and to tilt around the teapot, even though that portion of the path was occluded by the teapot during the test (Figure 7 shows the view during the test and a tilted view after the test for the path visualization).

### Helicopter-Rocker

The final test used a helicopter model with 113,000 triangles and a rocker arm model with 40,000 triangles. For this test, we wanted to pass the rocker through the open window of the helicopter, around the interior, and back out, avoiding other structures such as the helicopter blade and tail (Figure 8).

The haptic rendering system was able to provide useful feedback during this test, creating a collision-free path under user guidance.

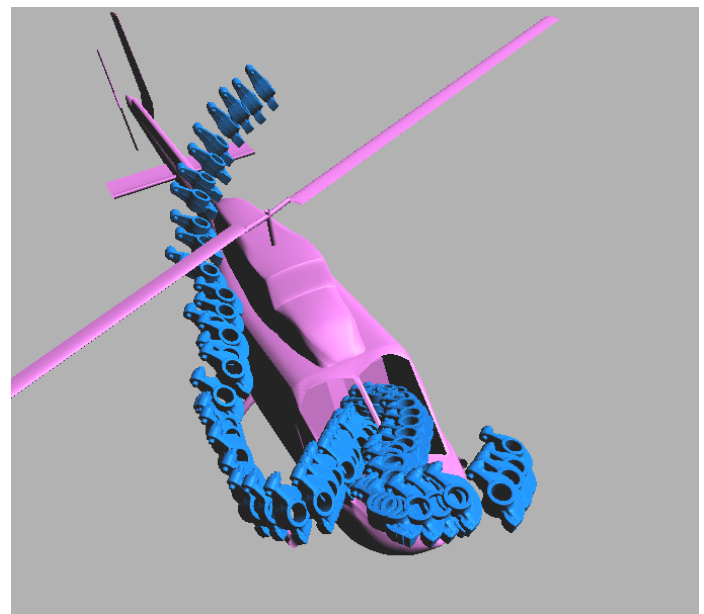


Figure 8: THIS TEST INVOLVED MODELS WITH OVER 150,000 COMBINED TRIANGLES.

## DISCUSSION AND CONCLUSION

The presented system advances the state-of-the-art in haptically-enhanced virtual prototyping systems by allowing virtual prototyping on general, freely positioned, polygonal models. In addition, our mechanism for rolling back model collision states to a collision-free position and orientation simplifies motion planning by always storing a safe path. The use of adjustable distances for both force onset and collision distance also enhances the capabilities of the system by simulating different clearance constraints during the task.

Comparison with other systems is difficult. Most other haptic rendering methods for polygonal models depend on penetration to generate forces, which would invalidate the simulation results. The highest-performing system[15] to compare with can render similarly sized models as our system but uses potentially low resolution levels of details in computing forces. In contrast, we are able to compute exact distances between high-resolution models and generate forces before penetration.

The tests demonstrated the system on a variety of model types and sizes. While the chosen models were not solving an actual real-world problem, the model shapes, resolutions, and task types are representative of the kinds of problems the system can solve.

The locality of the LMD computation means that these environments can scale to very large number of triangles, since most of the triangles will not come into consideration at any one time.

We hope to improve the power of the virtual prototyping system by continuing to improve the speed of the haptic rendering sub-system, by including other measures besides distance as a way of generating forces (such as clearance), and giving feedback to the user indicating the quality of the path as it is being generated.

## ACKNOWLEDGMENTS

The authors would like to acknowledge support in part from the following grants: NSF DMI9978603, NSF CDA-96-23614, and ARO DAAD 19-01-1-0013.

## REFERENCES

- [1] J.F. Canny. The Complexity of Robot Motion Planning. ACM Doctoral Dissertation Award. MIT Press, 1988.
- [2] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, pages 12(4):566–580, 1996.
- [3] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A Voronoi-Based Hybrid Motion Planner", Proc. IEEE/RSJ International Conf. on Intelligent Robots and Systems, 2001.
- [4] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots," in *IJRR*, 5(1):90–98, 1986.
- [5] K. Salisbury et al., "Haptic rendering: programming touch interaction with virtual objects", in *Symp. on Interactive 3D Graphics*, 1995. pp. 123-130.
- [6] J. Hollerbach et al., "Haptic interfacing for virtual prototyping of mechanical CAD designs," ASME Design for Manufacturing Symposium, (Sacramento, CA), Sept. 14-17, 1997.
- [7] P. Stewart et al., "CAD Data Representations For Haptic Virtual Prototyping", Proceedings of DETC'97, 1997 ASME Design Engineering Technical Conferences, Sept. 14-17, 1997, Sacramento, California.
- [8] D. Nelson, D. Johnson, and E. Cohen, "Haptic Rendering of Surface-to-Surface Sculpted Model Interaction," in Proc. 8th Annual Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems, (Nashville, TN), ASME, November 1999.
- [9] D. Nelson and E. Cohen., "Optimization-Based Virtual Surface Contact Manipulation at Force Control Rates," in IEEE Virtual Reality 2000, (New Brunswick, NJ), IEEE, March 2000.
- [10] D. Ruspini, K. Kolarov, and O. Khatib, "The Haptic Display of Complex Graphical Environments," in *Computer Graphics Proceedings, SIGGRAPH 1997*. Aug. 3-8. pp. 345-352.
- [11] W. McNeely, K. Puterbaugh, and J. Troy. "Six degree-of-freedom haptic rendering using voxel sampling". *Proc. of ACM SIGGRAPH*, pages 401–408, 1999.
- [12] D. Johnson and P. Willemsen, "Six Degree-of-Freedom Haptic Rendering of Complex Polygonal Models," in Proc. 2003 Haptics Symposium, 2003.
- [13] D. Johnson and P. Willemsen, "Accelerated Haptic Rendering of Polygonal Models through Local Descent," in Haptics Symposium 2004, IEEE, 2004.
- [14] D. Johnson and E. Cohen, "Spatialized Normal Cone Hierarchies," in Proc. 2001 ACM Symposium on Interactive 3D Graphics, Research Triangle Park, NC, March 19-21, 2001. pp. 129-134.
- [15] M. Otaduy and M. Lin, "Sensation Preserving Simplification for Haptic Rendering", in Proceedings of ACM SIGGRAPH 2003/ACM Transactions on Graphics, Vol. 22, pp. 543-553, San Diego, CA. 2003.

## DIRECT HAPTIC RENDERING OF COMPLEX TRIMMED NURBS MODELS

Thomas V Thompson II      Elaine Cohen  
Department of Computer Science  
University of Utah

### ABSTRACT

*The most accurate haptic rendition of a virtual model is produced when the haptic algorithm acts directly on the actual model and not an intermediate representation. In the modeling and design communities the de facto model representation standard is NURBS. Further, more powerful systems provide trimming and adjacency information within the models representation. This additional information permits more complex models to be expressed succinctly but also increases the complexity of representation. In this paper we present an algorithm that supports direct haptic rendering of models constructed from trimmed NURBS surfaces. Our distributed system links an advanced modeling system to a force-reflecting device. In addition, we present extensions to the algorithm which support model manipulation, dimensioned probes, and multi-probe contact.*

### 1 INTRODUCTION

The passive graphical display of complex models can convey only limited visual information. Even with the array of presentation options supplied to a user by modeling packages, such as isoline drawings, shaded images, and animations, the designer is often left without information that could easily be gathered if the model could be interrogated by touch (Gibson, 1966). Haptic rendering supplies this channel of feedback to the user by simulating the forces generated by contact with, and surface tracing of, a virtual model (Fig. 1). This increased level of interaction facilitates a greater understanding of complex models and adds to the sense of realism in virtual environments (Hollerbach et al., 1996; Stewart et al., 1997).

While the graphical display of a model is almost exclusively accomplished by first converting it to a collection of polygons, the model itself often starts with a different geometric representation. In fact, a master CAD model is almost always described by NURBS throughout its life (Piegl and Tiller, 1995). Being a parametric representation, NURBS surfaces have the advantage of compactness, higher order continuity, and exact computation of surface tangents and normals. All of these properties are useful in complex, realistic virtual environments (Snyder, 1995).

The ability to trim away arbitrary portions of a NURBS surface and define adjacencies under boolean set operations is a powerful extension to any modeling package. These construc-

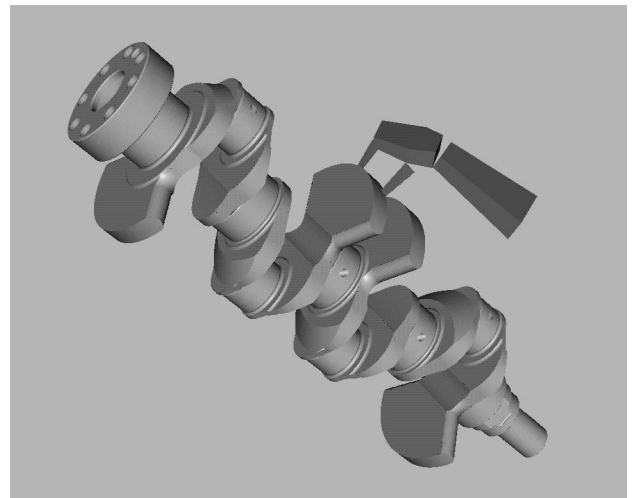


Figure 1. Multiple endpoint haptic rendering of a four cylinder crank shaft constructed from trimmed NURBS surfaces.

tive solid geometry results form a large class of models which can be much more readily and succinctly expressed using trimmed NURBS than non-trimmed NURBS. Furthermore, the use of trimmed NURBS is fairly widespread, making it important for a haptic rendering algorithm to handle them. This increase in expressiveness does not come without a cost. Since the trimming curve (or curves) representing the boundary of a surface is frequently expressed as a piecewise linear curve with a high number of segments, determining when haptic contact is to transition from one surface to another can be a complex task. However, once such a transition is detected, topological adjacency information can allow for efficient computation of the exact transition point.

A major contribution of this paper is that we use a model-oriented approach which enables an environment populated by a broad class of models to be haptically rendered (Fig. 1). Enabling the user to trace directly on the actual CAD model instead of an intermediate or alternate representation achieves the most accurate haptic rendering results. To this end, we introduce direct haptic rendering of complex models constructed from trimmed NURBS surfaces. In order to realize this approach,

we have developed and tested algorithms for model proximity testing, fast update of global and local closest point approximations, computationally efficient surface evaluation and normal evaluation techniques, and smooth efficient transitioning across trimmed surface boundaries. Further, we present extensions including model manipulation, dimensioned probes, and multiple probe contact. These algorithmic results are tested within a complete system that integrates a research modeling package, Alpha-1 (Riesenfeld, 1989; Riesenfeld, 1993) with both a Sarcos Dextrous Arm Master (Jacobsen et al., 1990) and a Phantom haptic feedback device (Massie and Salisbury, 1994).

## 2 BACKGROUND

The goal of a haptic rendering system is to generate forces that can be applied to a user's hand or arm to accurately produce a sense of contact with a virtual model. These forces, called restoring forces, resist penetration into the virtual model and are calculated using a wall model. These response models often have a restoring force proportional to the penetration depth (Colgate and Brown, 1994) and in the direction of the surface normal at the contact point, typically a local closest point. An accurate force computation depends on a good tracking and tracing algorithm since the magnitude and direction of the force depends on the accuracy of the calculated closest point and normal. Importantly, the force servo loop, and hence the closest point calculation, must run at several hundred Hz in order to maintain stiff virtual surfaces (Minsky et al., 1990). This high update rate limits the complexity of the algorithms for finding the closest point and thereby has dictated the types of models that could be rendered.

For instance, Zilles and Salisbury (1995) advocate using a constraint-based system to trace polygonal models. These systems are often limited to simple models since the high polygon count of complex models requires too much processing time. Ruspini et al. (1997) extend this work to handle larger polygon counts and to permit more general graphics primitives, such as points and lines, to be traced by a dimensioned probe.

Adachi (1993) employs distribution functions and Salisbury and Tarr (1997) use implicit surfaces to permit quality tracing of sculptured models. However, the design and graphics communities by and large model using NURBS. As such, to use these methods requires a conversion from the NURBS model into one of these other representations. This conversion is not only a complicated task but can be numerically unstable resulting in inaccurate models defined by high order functions.

Others (Adachi et al., 1995; Mark et al., 1996) propose using intermediate representations to simplify the tracing of sculptured models. These systems are bound by network limitations, and therefore updates are slow. While this approach permits the tracing of complex models, it suffers from poor temporal performance and cannot accurately reproduce surfaces with high curvature.

Thompson et al. (1997a) as well as Johnson and Cohen (1998) have demonstrated direct haptic rendering of sculptured models constructed from untrimmed NURBS surfaces using direct parametric tracing. While sculptured models can be traced

with quality results using these methods, they are predominantly surface oriented approaches since the models contain no topological adjacency information. The method requires tracking a closest point on each proximal surface to allow tracing across surface boundaries. This in turn limits both the complexity and number of models populating the environment.

Nelson et al. (1999) provide a closed form solution in the velocity domain for tracing sculptured NURBS models using a locally convex end-effector model. While this method allows the end-effector to be sculptured, the contact model is a single point and the approach is surface oriented with no transitioning algorithm.

## 3 SYSTEM OVERVIEW

A haptic virtual environment must meet the update rate constraints of the visual and haptic displays. While it may be acceptable for the visual display to update at twenty frames per second, the haptic display must maintain rates of several hundred Hz for the virtual surfaces to feel solid. Both the visual and haptic display individually tax a system, therefore, we distribute the *simulation process* and *haptic process* onto different special purpose computers (Thompson et al., 1997a; Thompson et al., 1997b).

The modeling environment runs within the simulation process on a graphics workstation which handles the visual display and performs computations typically global in scope. When the simulation process detects an appropriate global event, such as the proximity of the probe to a model, it signals the haptic process, which continues the computation with low-latency, local methods. The haptic process runs on real-time microcomputer boards and performs computations which are typically local in scope.

Several Ethernet channels connect the two processes to facilitate the various forms of communication in our system. Both processes have a model manager whose sole function is to maintain data consistency across the system. Data caching on both sides of the system, in conjunction with the distinct division of labor between the two processes, keeps the communication overhead at a minimum. The combination of independent processes maximizing computational capacity, low communication overhead, and synchronized visual and haptic display creates a realistic haptic virtual environment.

## 4 TRIMMED NURBS MODELS

Non-Uniform Rational B-Spline (NURBS) surfaces are highly compact and yet very expressive as a representation for modeling. A NURBS surface is a bivariate vector-valued piecewise rational function of the form

$$S(u, v) = \frac{\sum_{i=0}^m \sum_{j=0}^n P_{i,j} w_{i,j} B_{j,k_v}(v) B_{i,k_u}(u)}{\sum_{i=0}^m \sum_{j=0}^n w_{i,j} B_{j,k_v}(v) B_{i,k_u}(u)}, \quad (1)$$

where the  $\{P_{i,j}\}$  form the control mesh, the  $\{w_{i,j}\}$  are the weights, and the  $\{B_{i,k_u}\}$  and  $\{B_{j,k_v}\}$  are the basis functions

defined on the knot vectors  $\{u\}$  and  $\{v\}$  for a surface of order  $k_u$  in the  $u$  direction and  $k_v$  in the  $v$  direction.

The various properties of a NURBS surface, including a local convex hull property, and the ability to evaluate surface points, normals and tangents, along with its intuitive control characteristics make it a good representation for modeling and design. These properties have led to NURBS becoming the *de facto* industry standard for the representation and data exchange of geometric models (Piegl and Tiller, 1995).

Trimmed NURBS models are constructed by cutting away portions of a NURBS surface using trimming curves in parametric space. In our system trimming information is represented as directed closed polygons called *trimming loops*. Each individual linear portion of the loop is called a *segment*. A collection of connected segments that represents shared boundary between two surfaces is referred to as an *edge*. Portions of the surface domain to the left of a loop are considered cut-away while pieces to the right are deemed part of the model. Note that each surface that is part of a model contains at least one trimming loop. If there is no portion of the surface being cut away then this loop simply surrounds the domain of the surface.

Consider the model in Fig. 2a of a simple disc with a hole cut through it. The top surface will have two trimming loops within its parametric domain as shown in Fig. 2b. Notice that the direction of the two loops indicate that the dark regions are to be cut away. The outer clockwise loop cuts away the outermost region while the inner counterclockwise loop cuts away the center region representing the hole. The edges in Fig. 2b are illustrated in alternating brightness to indicate the patch in Fig. 2a that is adjacent to each edge.

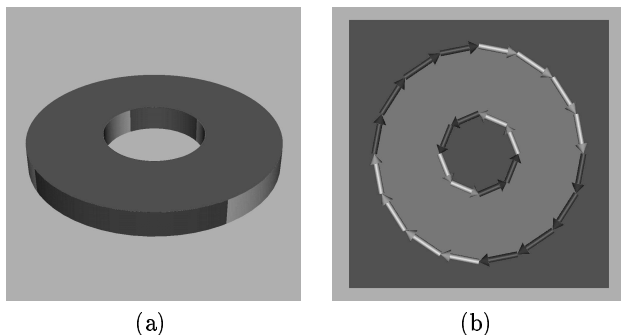


Figure 2. A trimmed NURBS model (a) and its parametric domain containing the trimming loop information (b).

## 5 DIRECT HAPTIC RENDERING

The goal of direct haptic rendering is to enable the user to feel the actual designed model instead of some secondary representation. In addition to an enhanced tracing experience, using the actual model also allows the designer to modify the model without having to wait for the haptic system to convert the model in a time-consuming preprocessing step. The difficult part of this approach lies in the evaluation of the model

to acquire the requisite closest point and surface normal needed by the force response model. These calculations must complete in a short enough time to permit the entire haptic process to maintain its high update rate. Also, the method must produce what we refer to as a *local closest point* since tracing using a global closest point results in several complications and, in fact, potentially erroneous results (Thompson et al., 1997a).

The act of haptically tracing a virtual model can be broken down into several phases. The system checks each model for proximity to the probe and activates those models deemed proximal. The tracking algorithm then tracks a closest point on the model until contact is made, at which point the tracing algorithm takes over. While in contact, the tracing algorithm maps the movement of the probe to movement along the surface of the model. Transitioning is indicated if the probe's movement causes the local closest point to cross a trimming loop boundary.

### 5.1 Proximity Detection

Our algorithm needs to trace only a single point per model per end-effector. However, it is still advantageous to keep the number of models active to a minimum. Doing so saves processor time and maintains high update rates. To this end, our system checks the proximity of the probe as it moves throughout the environment to each model. We use a first order approximation to the closest point found by a method referred to as nodal mapping (Thompson et al., 1997a). The simulation process performs this computation since it is a global computation that requires the processing power of the workstation. When the probe moves close enough to potentially contact a model, an activation packet is sent to the haptic process indicating that a closest point should be tracked. This packet contains the ID for the model being activated along with the surface ID and parametric location of the closest point. The parametric value then seeds the tracking algorithm.

### 5.2 Tracking

Any movement near an active model, while not in contact with that model, is referred to as tracking. Contrary to tracing, the closest point for tracking must be the global closest point. This ensures proper contact detection by permitting the closest point to jump across concavities and to climb convex regions. For example, Fig. 3 demonstrates the different closest point requirements. During tracing, the probe moves from position *A* to position *B* resulting in the closest point becoming bound to the intersection of two surfaces (Fig. 3a). This is the correct response: the probe is trying to move inside a second surface and should be restricted from movement in that direction. If the same algorithm was used for tracking, a similar scenario could occur (Fig. 3b). Again the probe moves from position *A* to position *B*, which results in the tracked point being bound to an edge. In this case, however, the closest point should not be bound. In fact, if the probe were to continue along the current path and intersect the model, the contact would not be detected since the penetration would not occur at the location of the bound tracked point.

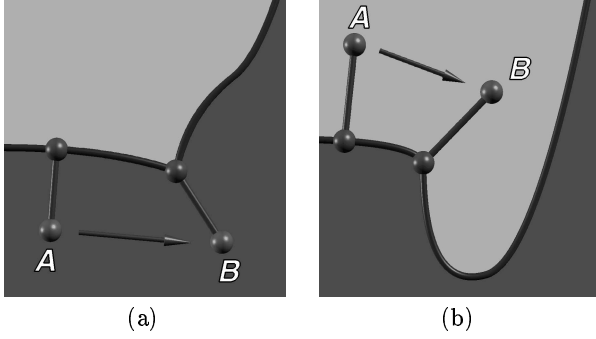


Figure 3. A bound closest point is desired when in contact and tracing (a) but when tracking while not in contact (b) it is not.

The tracing algorithm cannot be used exclusively for tracking since it assumes the probe is in contact and therefore the next desirable closest point should correlate to a movement bound to the surface of the model. In the tracing algorithm, it is appropriate for the closest point to hold its position since its purpose is to restrict the probe's movement in certain directions. During tracking, however, there should be no restrictions on the probe's movement or the closest point which shadows it.

Currently, it is not feasible to determine the global closest point to a trimmed NURBS model at haptic rates. Therefore, we use a hybrid approach where the tracing algorithm is augmented by periodic re-seeding with the global closest point. The simulation process calculates an approximate global closest point using an algorithm based on a time-critical method that spends less time on objects outside the region of interest (Johnson and Cohen, 1997). The haptic process accepts this point and uses it to re-seed the tracing algorithm. This periodic re-seeding allows the tracing algorithm to form a good approximation to the global closest point between updates. We have found this to be very effective since the probe typically does not move very far between updates. Even though contact detection may be delayed for several cycles, the cycle rate is so high that the delay is, in practice, imperceptible to the user.

### 5.3 Contact and Tracing

At the heart of every haptic system is its ability to detect the contact of the probe with a virtual model and to simulate the act of tracing along the surface of the model. Contact occurs when the probe first intersects the virtual model, resulting in a positive penetration depth. Once contact has been established, any lateral movement along the surface indicates tracing. As the probe moves, a local closest point on the surface of the model is found that shadows the probe's movement. The accuracy of this closest point and its associated normal is essential since the restoring force calculation depends directly upon this result.

We relate movement of the probe to movement along the surface using direct parametric tracing (Thompson et al., 1997a). The algorithm has been shown to run at interactive rates and produce accurate results making it suitable for direct haptic rendering. The algorithm is seeded with the surface evaluation point

$S(u^*, v^*)$ , calculated using refinement (Cohen et al., 1980) where  $(u^*, v^*)$  are the parametric coordinates for the point of contact.

Movement along a surface in Euclidean space relates to movement in parametric space by the partial derivatives of the surface

$$\frac{\partial S}{\partial u} \approx \frac{\Delta S_u}{\Delta u}, \quad \frac{\partial S}{\partial v} \approx \frac{\Delta S_v}{\Delta v},$$

where  $\Delta S_u$  and  $\Delta S_v$  are the change on the surface along the  $u$  and  $v$  isocurves respectively. A good approximation for  $\Delta S_u$  and  $\Delta S_v$  is the projection of the probe onto the surface tangent plane. The coordinates of this projection within the tangent plane are used to derive  $\Delta u$  and  $\Delta v$ .

A key element of the algorithm is the efficient computation of the surface tangents  $\frac{\partial S}{\partial u}$  and  $\frac{\partial S}{\partial v}$ . The calculation of  $S(u^*, v^*)$  by refinement results in the new knot vectors  $\{\hat{u}\}$  and  $\{\hat{v}\}$ , the new weights  $\{\hat{w}_i\}$  and a new control mesh  $\{\hat{P}_{i,j}\}$  where  $\hat{P}_{i^*,j^*} = S(u^*, v^*)$ . Furthermore, the  $i^*$  column and the  $j^*$  row in the control mesh form control polygons for isocurves that pass through the point  $\hat{P}_{i^*,j^*}$ . This allows the calculation of the two tangent vectors to be performed using the more simple curve equation. The isocurve defined at  $v^*$  in the  $j^*$  row of the control mesh is given by

$$\gamma_{j^*}(u) = \frac{\sum_{i=0}^m \hat{P}_i \hat{w}_i B_{i,k_u}(u)}{\sum_{j=0}^m \hat{w}_j B_{j,k_u}(u)}, \quad (2)$$

where the  $\{\hat{P}_i\}$  form the extracted control polygon, the  $\{\hat{w}_i\}$  are the associated weights, and the  $\{B_{i,k_u}\}$  are the basis functions defined over the knot vector  $\{\hat{u}\}$  for a curve of order  $k_u$ . The  $\gamma_{j^*}(v)$  curve can be formed in similar fashion.

We then calculate the tangent vector by evaluating the derivative of the isocurve at  $u^*$ . Using the quotient rule to form the velocity curve for Eq. (2) yields,

$$\gamma'_{j^*}(u) = \left( \frac{\sum_{i=0}^m \sum_{j=0}^m \hat{P}_i \hat{w}_i \hat{w}_j B_{j,k_u}(u) B'_{i,k_u}(u) - \left( \sum_{i=0}^m \hat{P}_i \hat{w}_i \hat{w}_j B_{i,k_u}(u) B'_{j,k_u}(u) \right)}{\left( \sum_{j=0}^m \hat{w}_j B_{j,k_u}(u) \right)^2} \right).$$

However, since the curve was refined to form an evaluation point, each basis function takes on a value of either zero or one when the velocity curve is evaluated at  $u^*$ . The two basis functions that remain active,  $B_{i^*,k_u}(u^*)$  and  $B_{i^*+1,k_u-1}(u^*)$ , result in the simplified expression,

$$\gamma'_{j^*}(u^*) = \frac{(k-1)}{\hat{u}_{i^*+k} - \hat{u}_{i^*+1}} \frac{\hat{w}_{i^*+1}}{\hat{w}_{i^*}} (\hat{P}_{i^*+1} - \hat{P}_{i^*}). \quad (3)$$

Notice that the tangent vector is computed efficiently using only the control polygon, associated weights, and the knot vector. With this and the resulting calculation of  $\Delta u$  and  $\Delta v$ , the new approximation to the local closest point can be evaluated. Since this new point,  $S(u^* + \Delta u, v^* + \Delta v)$ , is calculated using refinement, the algorithm can continue directly into the next time step.

## 5.4 Transitioning

Three basic transitions can occur during a trace. Two forms result when the probe's movement causes the local closest point to hit a trim boundary. In the first form of transitioning, the local closest point should transition across the trim and onto an adjacent surface (Fig. 4a). This occurs most often in areas of a model constructed by pasting surfaces together along an adjacent edge. The second possibility is that the closest point should remain on the trim edge. This is the case in concave areas where the probe can trace along the intersection of two surfaces (Fig. 4b). The final form is the special case of transitioning off the model. When the penetration depth becomes negative, tracing ends and tracking resumes, which is effectively the inverse of the contact problem.

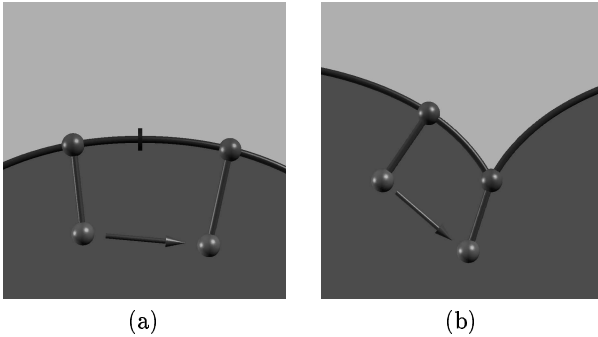


Figure 4. (a) Transitioning across a trimming edge and onto another surface. (b) Transitioning onto the intersection of two surfaces.

There are four core modules to the tracing algorithm that permit the proper detection and handling of transitions. The `check_cross` module detects when a movement along a surface intersects a trimming loop. Once such an intersection is found the `find_adjacent` module determines the exact corresponding point on the neighboring surface. Tracing along a trim boundary is handled by the `slide3d` module with the `release` module being used to determine when the edge tracing should terminate and normal surface tracing should resume. The path the tracing algorithm takes through these modules is illustrated in Fig. 5.

**5.4.1 Trim Intersection.** Discrete movement along the surface correlates to a directed line segment in parametric space. This segment is constructed using the current contact point's parametric coordinates and the next location calculated using direct parametric tracing. If this segment, or *movement vector*, intersects any of the surface's trimming segments then a boundary has been crossed. The location of the intersection is determined by selecting the intersection point closest to the current contact point.

Since the number of trimming segments per surface can be very large, it is not possible to check every segment for intersection. Our solution to this problem is to overlay each surface

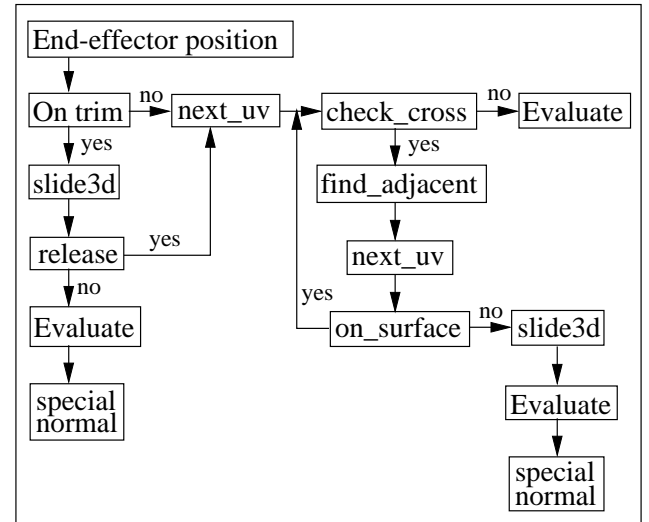


Figure 5. The tracing algorithm follows a different path depending upon when and if the trace results in crossing a trim boundary.

with a grid. Each cell in the grid contains the trim segments that lie within or intersect it. Ideally, each cell would contain one segment and each segment would be contained in exactly one cell. In practice this is not necessary and would result in heavy preprocessing overhead. We locate the grid so that its boundary coincides with the bounding box of all trim segments. Further, we construct the grid to have four times as many cells as segments with the number of rows and columns being equal. In practice, this heuristic has proven to be effective.

Each call to `check_cross` results in only checking those segments lying within the cells the movement vector intersects. In addition, we use a grid walking algorithm in order to check these cells in the order the movement vector traverses through them. The intersection checks conclude at the first valid intersection, further cutting down on the number of intersection checks performed.

**5.4.2 Adjacency.** In order to smoothly transition from one surface to another it is necessary to calculate an accurate transition point on the neighboring surface. Our system maintains an *edge adjacency table* for each surface. This table allows efficient determination of the adjacent surface as well as the appropriate trimming loop and edge onto which the transition should occur. The segment is found by indexing into the edge to the segment that corresponds to the one intersected. Since adjacent trimming edges run in opposite directions, and the number of segments in these edges is the same for both surfaces, the index of the proper segment can be found directly as  $N - i - 1$  where  $N$  is the number of segments in the edge and  $i$  is the index of the segment intersected. The exact point of transition along this segment,  $s$ , is then given by  $s(1 - p)$  where  $p$  is the percent along the intersected segment that the intersection occurred.

**5.4.3 Edge Tracing and Release.** Tracing along a trim edge is closely related to tracing along the surface. The edge tracing algorithm must slide along the edge in Euclidean space to a point locally close to the probe's position. Our `slide3d` module does precisely that. The algorithm projects the probe onto the current segment. If this projection remains on the current segment then this projection is our result. If, however, the projection is beyond either endpoint of the segment then we continue to project onto segments along the loop in that direction until a local minimum is found.

Once the local closest point is found the algorithm checks to see if the tracked point should release from the trim. Our algorithm first evaluates one of the surfaces and then uses direct parametric tracing to determine the next location on the surface. If the calculated parametric point is on the surface (i.e. on the correct side of the trimming loop – the right side of the closest segment) then the trace releases from the trim. If the trace does not release onto the first surface then the second is checked. When the trace does not release onto either surface, then a special normal is computed directed from the probe's location to the local closest point. This normal ensures a smooth trace along the edge and also deters further movement into the model.

## 6 EXTENSIONS

Using the direct haptic rendering algorithm as a black box entity, several worthwhile extensions are possible. We have implemented model manipulation, dimensioned probes, and multi-probe contact. While these are but a few of the potential improvements and extensions, they demonstrate the power and flexibility of the algorithm.

### 6.1 Models in Motion

Whether through manipulation, animation, or dynamic properties, mobile models are a fundamental property of virtual environments. The direct haptic rendering algorithm presented in this paper is designed for probe movement with static models, but can be extended so that both probe and models can move. We accomplish this by tracing the original (non-transformed) model with a probe position that has been transformed into model space (Thompson et al., 1997b).

For each model being traced or tracked, we transform the probe through the inverse of the model's transformation matrix. This process transforms the movement of the model into a component of the probe's movement. The resulting closest point and normal are then transformed back from model space to world space. This embedding of the tracing algorithm requires minimal overhead and does not affect the update rate of the haptic process.

### 6.2 Dimensioned Probe

One of the drawbacks to point probe methods is the dimensionless nature of the probe. If two models are placed directly adjacent to one another, a probe without finite size could still move between the two models without making contact. To elim-

inate this possibility we compute a model that is projected outward by the radius of the desired probe. A model of this type is often referred to as an offset model (Ho, 1997). Our system uses the offset model in the haptic rendering process while using the original model for the visual display.

Figure 6 illustrates the construction and use of an offset model. The original model in Fig. 6a is offset by the radius of the probe in the direction of the surface normal resulting in the model in Fig. 6b. Isolated regions are trimmed away, producing the offset model in Fig. 6c. Contact with the surface of this offset model represents contact with the original model with a dimensioned probe (Fig. 6c). Notice that any part of the offset model that is trimmed away represents a portion of the original model that could not be contacted with the dimensioned probe. Tracing with a point probe along an edge created by trimming away a region corresponds to tracing multiple contact points of the original model with a dimensioned probe (Fig. 6c).

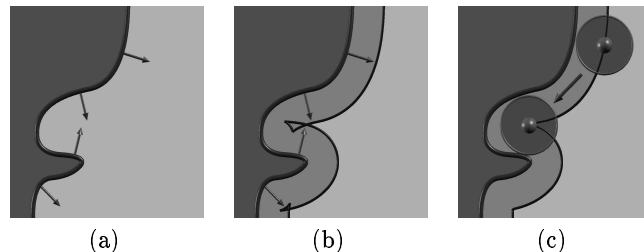


Figure 6. (a) Actual model. (b) Initial offset model. (c) Final offset model with possible trace positions.

It is important to note that this process depends on trimming and adjacency information. Further, while this approach uses an auxiliary representation it is not a simplifying “intermediate” representation, since the offset model exactly represents the parts of the original model that can be contacted by the dimensioned probe. Producing the offset model adds significant preprocessing, but it does not affect performance of the tracing algorithm as long as the model geometry does not change during the trace.

### 6.3 Multi-Probe Contact

Our current implementation of direct haptic rendering of trimmed models runs at over 1000 Hz. However, when using the Sarcos master we notice no improvement when running at any rate over 500 Hz. By running at this lower rate, there is extra time within each cycle of the haptic process. Since the Sarcos device can reflect forces to multiple end-points, we make two calls to the trace algorithm using the location of the finger and thumb as the probe locations. This adds to the overall tracing experience with only a slight data overhead and minimal impact on the tracing algorithms performance. We are currently working on using this additional information to produce stable



grasps as well as allow better user control of the models and the environment (Maekawa and Hollerbach, 1998).

## 7 RESULTS

Our trace algorithm is model-based with a near constant time grid-based transitioning algorithm. This results in equal performance and accuracy for each model, regardless of complexity (Fig. 7).

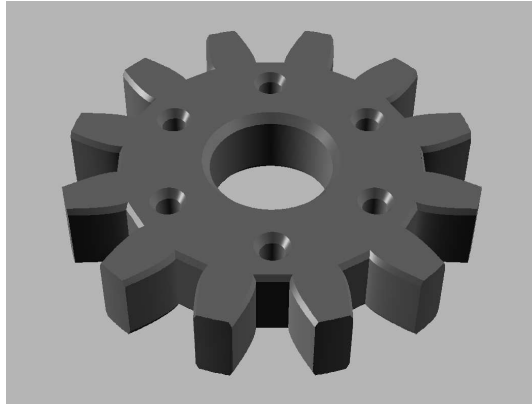


Figure 7. Highly trimmed model of a mechanical gear.

Trimmed NURBS models vary greatly in the number of surfaces and trim segments required to represent them. Table 1 lists a sampling of models against which we tested the system. The *Srfs* column indicates the total number of surfaces for the model. *Segs* indicates the average number of trim segments per surface. Grid statistics are represented in the final three columns. The column labeled *Empty* gives a percentage for the number of cells in a surfaces grid that contain no trim segments. Empty cells translate into essentially zero work for the transitioning algorithm. *Max* gives the maximum number of segments in any one cell. This number represents the worst case for the transitioning algorithm for the given model. Finally, the *Mean* column shows the average number of segments in cells that actually contain segments. This number indicates the amount of work the transitioning algorithm can be expected to perform when near a surface boundary. Note that both the *Max* and the *Mean* columns contain very small numbers in comparison to the *Segs* column. These numbers indicate the drastic reduction in work the transitioning algorithm performs when compared to an algorithm that would check every segment.

The two haptic devices used are very different yet both produced accurate results. The Sarcos Dextrous master is a high inertia device with 10 degrees-of-freedom and a complex dynamics structure. For this device, the haptic process ran on a hybrid PowerPC 604 and Motorola 68040 VME system with a surface stiffness of  $6000N/m$ . The Phantom is a low inertia device with 3 degrees of freedom and a rather simple dynamics structure. The haptic process in this instance ran on an SGI Indigo R10000 under IRIX 6.5 with a surface stiffness of  $1200N/m$ . Both used a

Model	Srfs	Segs	Empty	Max	Mean
Goblet	3	254.00	89.92	13	3.38
Brake	28	168.14	72.52	6	1.47
Gear	22	1256.27	92.11	15	4.05
Crank	73	412.00	89.56	36	3.30

Table 1. Statistics on models used in system testing.

nonlinear response model to provide a physically-accurate model of probe-model collision (Marhefka and Orin, 1996). In both cases, the simulation process ran on an Octane with dual R10000 processors and RealityEngine<sup>2</sup> graphics.

Since the quality of the trace is directly related to the calculated closest point and surface normal, we ran simulations to determine the errors for these values. The simulation consisted of a surface trace at a constant penetration depth of  $5mm$ . The algorithm was able to resolve the closest point to within  $0.2mm$  and the error for the surface normal error was under  $0.02degrees$ . In practice, the penetration depth averages  $3mm$  for the Sarcos master and  $1mm$  for the Phantom with low variance. This illustrates that the penetration is not only consistently small, but also consistently near the mean. This combination produces a smooth tracing experience.

## 8 FUTURE WORK

The goal of our research is to produce a virtual environment that allows intuitive interaction with complex models. To achieve this goal, algorithms to support important capabilities remain under investigation and include:

- An arbitrary probe model would allow the trace to more closely represent the device being used. More complex haptic devices permit tracing with the full hand.
- Collision contact and response for model impacts is a difficult problem, especially when using trimmed NURBS, but a low latency solution needs to be developed for more realistic force response.
- Surface properties such as texture can add to the realism of the tracing experience. Soft and deformable models are a challenge since changes in the geometry can occur during contact.

We are also investigating alternative uses for the tracing algorithm. Among these are methods for modifying arbitrary curves that lie on a surface, sketching on a surface interactively, and finding the silhouette curves of NURBS models.

## 9 CONCLUSION

We have presented a powerful algorithm that supports the direct haptic rendering of models constructed from trimmed NURBS. This model-based approach permits more complex scenes as well as models to be haptically rendered. Additionally, we have demonstrated the ability of the algorithm to be

extended in order to permit model manipulation, tracing by a dimensioned probe, and multi-probe contact. Finally, the distributed system design allows high update rates on both sides of the system, resulting in an interactive visual display coupled with an accurate haptic rendition produced from the actual model.

## ACKNOWLEDGMENTS

The authors would like to thank the Biorobotics group for helping operate and setup the robotic devices, their various control systems, and the networking software. Thanks also go to the students and staff of the GDC project, within which this work was developed. Support for this research was provided by NSF Grant MIP-9420352, by DARPA grant F33615-96-C-5621, and by the NSF and DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219).

## REFERENCES

- Adachi, Y., 1993, "Touch And Trace On The Free-Form Surface Of Virtual Object," in *Proc. Virtual Reality Annual Intl. Symp.*, Seattle, WA, pp. 162-168.
- Adachi, Y., Kumano, T., and Ogino, K., 1995, "Intermediate Representation For Stiff Virtual Objects," in *Proc. Virtual Reality Annual Intl. Symp.*, Research Triangle Park, NC, pp. 203-210.
- Cohen, E., Lyche, T., and Riesenfeld, R., 1980, "Discrete B-Splines And Subdivision Techniques In Computer Aided Geometric Design And Computer Graphics," *Computer Graphics and Image Processing*, Vol. 14, Number 2.
- Colgate, J.E., and Brown, J.M., 1994, "Factors Affecting The Z-Width Of A Haptic Display," in *Proc. IEEE 1994 International Conference on Robotics & Automation*, San Diego, CA, pp. 3205-10.
- Nelson, D., Johnson, D., Cohen, E., 1999, "Haptic Rendering of Surface-to-Surface Sculpted Model Interaction," in *8th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Nashville, TN.
- Gibson, J.J., 1966, *The Senses Considered as a Perceptual System*, Boston, MA, Houghton Mifflin Co.
- Ho, C., 1997, *Feature-Based Process Planning and Automatic Numerical Control Part Programming*, Ph.D. Thesis, University of Utah, Computer Science Department.
- Hollerbach, J.M., Cohen, E.C., Thompson, W.B., and Jacobsen, S.C., 1996, "Rapid Virtual Prototyping Of Mechanical Assemblies," *NSF Design and Manufacturing Grantees Conference*, Albuquerque, NM.
- Jacobsen, S.C., Smith, F.M., Iversen, E.K., and Backman, D.K., 1990, "High Performance, High Dexterity, Force Reflective Teleoperator," in *Proc. 38th Conf. Remote Systems Technology*, Washington, D.C., pp. 180-185.
- Johnson, D.E., and Cohen, E., 1997, "Minimum Distance Queries For Polygonal and Parametric Models," Technical Report UUCS-97-003, University of Utah, Department of Computer Science.
- Johnson, D.E., and Cohen, E., 1998, "An improved method for haptic tracing of sculptured surfaces," in *7th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Anaheim, CA.
- Maekawa, H. and Hollerbach, J.M., 1998, "Haptic Display for Object Grasping and Manipulating in Virtual Environment," in *Proc. IEEE Intl. Conf. Robotics & Automation*, Leuven, Belgium, pp. 2566-2573.
- Marhefka, D.W., and Orin, D.E., 1996, "Simulation Of Contact Using A Nonlinear Damping Model," in *Proc. International Conference on Robotics and Animation*, Minneapolis, Minnesota, pp. 1662-1668.
- Mark, W.R., Randolph, S.C., Finch, M., Van Verth, J.M., and Taylor III, R.M., 1996, "Adding Force Feedback To Graphics Systems: Issues And Solutions," in *Proc. SIGGRAPH 96*, New Orleans, LA, pp. 447-452.
- Massie, T.M. and Salisbury, J.K., 1994, "The PHANTOM Haptic Interface: A Device for Probing Virtual Objects," in *3rd Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Chicago, IL, DSC-Vol 1, pp. 295-301.
- Minsky, M., Ouh-Young, M., Steele, M., Brooks, F.P. Jr., Behensky, M., 1990, "Feeling And Seeing: Issues In Force Display," in *Proc. Symposium on Interactive 3D Graphics*, Snowbird, UT, pp. 235-243.
- Piegl, L. and Tiller, W., 1995, *The NURBS Book*, Berlin, Springer.
- Riesenfeld, R., 1989, "Design Tools For Shaping Spline Models," in *Mathematical Methods in Computer Aided Geometric Design*, (Edited by T. Lyche and L. Schumaker), Academic Press.
- Riesenfeld, R., 1993, "Modeling With Nurbs Curves And Surfaces," in *Fundamental Developments of Computer Aided Geometric Design*, L. Piegl (ed.), Academic Press.
- Ruspini, D.C., Koloarov, K., and Khatib, O., 1997, "The Haptic Display of Complex Graphical Environments," in *Proc. SIGGRAPH 97*, Los Angeles, CA, pp. 345-351.
- Salisbury, K. and Tarr, C., 1997, "Haptic Rendering of Surfaces Defined by Implicit Functions," in *Proc. 6th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Dallas, TX, DSC-Vol. 61, pp. 61-67.
- Snyder, J., 1995, "An Interactive Tool For Placing Curved Surfaces Without Interpenetration," in *Proc. SIGGRAPH 95*, Los Angeles, CA, pp. 209-218.
- Stewart, P., Buttolo, P., and Chen, Y., 1997, "CAD Data Representations for Haptic Virtual Prototyping," in *Proc. ASME Design Engineering Technical Conference*, Sacramento, CA.
- Thompson II, T.V., Johnson, D.E., Cohen, E., 1997, "Direct Haptic Rendering Of Sculptured Models," in *Proc. Symposium on Interactive 3D Graphics*, Providence, RI, pp. 167-176.
- Thompson II, T.V., Nelson, D.D., Cohen, E., and Hollerbach, J.M., 1997, "Maneuverable NURBS Models within a Haptic Virtual Environment," in *Proc. 6th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Dallas, TX, DSC-Vol. 61, pp. 37-44.
- Zilles, C.B., and Salisbury, J.K., 1995, "A Constraint-Based God-Object Method For Haptic Display," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots*, Vol 3, pp. 146-151.

# HAPTIC RENDERING OF SURFACE-TO-SURFACE SCULPTED MODEL INTERACTION

Donald D. Nelson     David E. Johnson     Elaine Cohen  
 University of Utah, Computer Science Dept.  
 50 S Central Campus Dr Rm 3190  
 Salt Lake City, UT 84112-9205  
 Email: {dnelson,dejohnso,cohen}@cs.utah.edu

## ABSTRACT

*Previous work in haptics surface tracing for virtual prototyping and surface design applications has used a point model for virtual finger-surface interaction. We extend this tracing method for surface-to-surface interactions. A straightforward extension of the point-surface formulation to surface-surface can yield extraneous, undesirable solutions, although we rework the formulation to yield more satisfactory solutions. Additionally, we derive an alternative novel velocity formulation for use in a surface-surface tracing paradigm that exhibits additional stability beyond the Newton methods. Both methods require evaluating the surface point and first and second surface partial derivatives for both surfaces, an efficient kilohertz rate computation. These methods are integrated into a three step tracking process that uses a global minimum distance method, the local Newton formulation, and the new velocity formulation.*

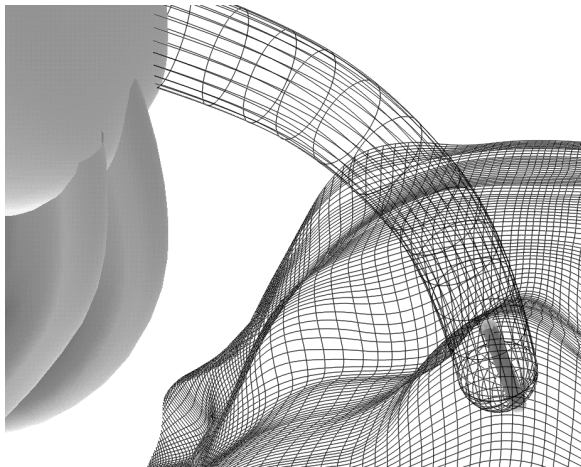


Figure 1: Well-behaved finger penetration into a surface shown by the “penetration cylinder”. The velocity method and modified Newton method return the maximum distance between the two surfaces upon penetration.



Figure 2: Virtual proxies are important in the penetrating case. The maximal distance is required by the haptics tracing algorithm. Global solution discontinuities such as “chopping though” an object are not desirable. Because the velocity formulation (shown) is the “most local,” it is the method of choice for the penetrating case.

## 1 INTRODUCTION

User interactions with surfaces with force feedback are an important design and visualization tool. Current work has mostly utilized a point position hand model for interaction, but a desirable extension is to allow more realistic geometry for the hand model. However, this extension creates some severe geometric computation challenges, as well much more complicated contact and response scenarios.

In this paper, we address one portion of the surface-surface haptic rendering problem, namely, computation of proper penetration depth between two surfaces. Once the penetration vector has been obtained, rendering force feedback will be done with established

haptics techniques [Thompson, 1997].

This penetration depth computation will be placed within a framework for reliably finding and tracking multiple contact points between models. This framework breaks the haptic rendering problem into several phases — distant tracking using global minimum distance methods, nearby tracking using local Newton methods, and tracking during contact using a reformulated Newton’s method or a novel velocity formulation.

## 1.1 Distance Extrema

Following [Baraff, 1990], the extremal distance may be defined as the minimum distance between the two models when they are disjoint, zero during tangential contact, and the locally maximum penetration depth when they are inter-penetrated. This measure is related to the minimum translation distance defined by Cameron [Cameron, 1997].

When a user touches a virtual surface with his virtual finger model, a curve  $\gamma(t)$  embedded on the finger surface  $\mathbf{f}$  and a curve  $\zeta(t)$  on the embedded on a model surface  $\mathbf{g}$  define the path of distance extrema required by the haptics tracing algorithm.

$$\text{distance extrema} = \|\mathbf{f}(\gamma(t)) - \mathbf{g}(\zeta(t))\| \quad (1)$$

Our goal is to find the piecewise continuous curves  $\gamma(t)$  and  $\zeta(t)$  for penetration depth computations in a real-time haptics tracing environment.

When the finger model is penetrated into the CAD model, the maximal distance is required for force computations as shown in Fig.1. When the finger is not penetrating the surface the curves  $\gamma$  and  $\zeta$  may be discontinuous since the distance between surface models is non-differentiable in general. Both surfaces may be moving. A minimal distance measure between surfaces is useful in this case for a global monitoring and restarting mechanism.

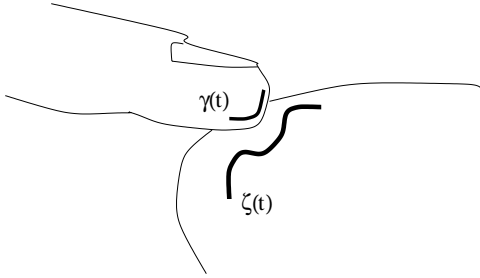


Figure 3: Curves of distance extrema embedded into the finger surface model and the CAD model.

## 2 BACKGROUND

The robotics community has considerable literature in the area of finding the minimal distance between a point and model or between model and model [Quinlan, 1994, Gilbert, 1988, Lin, 1994]. However, the minimum distance between models is zero during penetration; we desire the penetration depth for haptic force computations. The minimum distance between parametric surfaces  $\mathbf{f}(u, v)$  and  $\mathbf{g}(s, t)$  may be described by the following system of equations

$$(\mathbf{f}(u, v) - \mathbf{g}(s, t)) \cdot \mathbf{f}_u = 0 \quad (2)$$

$$(\mathbf{f}(u, v) - \mathbf{g}(s, t)) \cdot \mathbf{f}_v = 0 \quad (3)$$

$$(\mathbf{f}(u, v) - \mathbf{g}(s, t)) \cdot \mathbf{g}_s = 0 \quad (4)$$

$$(\mathbf{f}(u, v) - \mathbf{g}(s, t)) \cdot \mathbf{g}_t = 0 \quad (5)$$

which correctly describes that the line between the closest points is normal to each surface. This system has been solved with a search over the four-dimensional parameter space [Snyder, 1995], with global, high-dimensional resultant methods [Lin, 1994], and Euclidean space bounding methods [Johnson, 1998a].

Note that in haptic applications, we are often interested in the local solution to distance. Using a local solution constrains the force computation to a continuous solution, similar to a “God-object” [Zilles, 1995] or virtual proxy [Ruspini, 1997] as used in polygonal methods (see Fig.2).

The assumption that the situation upon penetration is local for haptics tracing is fortunate because the haptics controller often requires the greatest update rates precisely at the impact or penetrating event. Greater stability is achieved with the high update rates.

Local solution methods have been applied to point - surface analogs of Equations 2-5. A first-order solution to the minimum distance is described in [Thompson, 1997] and extended to a Newton formulation in [Johnson, 1998b, Stewart, 1997]. For a parametric surface  $\mathbf{f}(u, v)$  and point  $\mathbf{P}$ , we can describe the minimum distance constraint equations as

$$(\mathbf{f}(u, v) - \mathbf{P}) \cdot \mathbf{f}_u = 0 \quad (6)$$

$$(\mathbf{f}(u, v) - \mathbf{P}) \cdot \mathbf{f}_v = 0. \quad (7)$$

Offset surfaces have recently been used to extend the point-surface finger model with a sphere-surface model [Ruspini, 1997, Thompson, 1999]. A surface that is offset from an original surface by a constant amount can be traced with the point-surface model. When the original surface is displayed, the point model is effectively performing sphere-surface tracing. A sphere is still a very simple finger model; rotating the finger has no effect. We develop a method in this work that allows more complicated finger models.

## 3 APPROACH

We have broken the haptic rendering of surface-surface interactions into three phases. In the first phase, the far phase, a global monitoring mechanism returns all portions of a surface within some distance of some part of the other surface. In the second phase, the near phase, local Newton methods determine the closest points between all portions of the surfaces returned from the far phase. The third phase, the penetration phase, uses these closest points to initiate a velocity formulation that maintains the proper penetration depth vector between surfaces.

### 3.1 Global Minimum Distance

The global minimum distance mechanism depends on the subdivision properties of the surface. For NURBS surfaces, the surface is made up of a patchwork of polynomial pieces. Each polynomial piece is contained with the convex hull of its defining control points. These pieces may be refined, such that each piece splits into several pieces that maintain the original surface shape, yet have additional control points. These additional control points converge quadratically to the surface as the refinement is increased.

We can exploit these properties to prune away portions of surfaces that are further away than some threshold, refine the remaining portions, and repeat [Johnson, 1998a]. Remaining areas may be used to find approximate minimum distances to initiate faster, local, Newton methods.

### 3.2 Local Minimum Distance

The local minimum distance phase uses Newton methods to quickly update the minimum distance between portions of the model. We

use a reformulated extremal distance approach described in the following section for extra stability relative to the standard minimum distance formulation from the minimum distance formulation of Equations 2-5.

### 3.3 Local Penetration

We have developed two methods for maintaining the proper penetration depth vector for surface-surface interactions. The first is a reformulation of the Newton minimum distance method. The second is a velocity formulation. The Newton method has the advantage of being able to find the extremal distance given nearby starting locations on the surfaces. The velocity formulation is able to maintain the proper extremal distance and shows stability beyond that of the reformulated Newton method.

### 3.4 Extremal Distance Representation

The minimum distance equations 2-5 are not adequate to properly describe the extremal distance needed for surface-surface haptics. Along with a root at the extremal distance, a set of roots occurs along the curve of intersection between the two surfaces, where  $(\mathbf{f}(u, v) - \mathbf{g}(s, t))$  goes to zero. The local methods may very easily “slide” into these solutions. Our reworked formulations avoid the zero distance roots.

The following methods are applicable to any parametric surface representation, including NURBS and subdivision surfaces, the most commonly used representations. Let  $\mathbf{u} = [u \ v \ s \ t]^T$  designate the closest parametric contact coordinates between any two parametric surfaces  $\mathbf{f}(u, v)$  and  $\mathbf{g}(s, t)$ .  $\mathbf{f}$  and  $\mathbf{g}$  denote surface evaluations, or mappings from parametric space to Cartesian space.

### 3.5 Newton Extremal Distance Formulation

The extremal distance between parametric surfaces  $\mathbf{f}(u, v)$  and  $\mathbf{g}(s, t)$  may be described by the following equation:

$$\mathbf{E}(u, v, s, t) = (\mathbf{f}(u, v) - \mathbf{g}(s, t)) \cdot \mathbf{N} \quad (8)$$

where  $\mathbf{N}$  is the surface normal of  $\mathbf{f}$  at  $(u, v)$ . We wish to find the extrema of  $\mathbf{E}$ , which may be found at simultaneous zeros of its partials. The partials are

$$\mathbf{f}_u \cdot \mathbf{N} + (\mathbf{f}(u, v) - \mathbf{g}(s, t)) \cdot \mathbf{N}_u = 0 \quad (9)$$

$$\mathbf{f}_v \cdot \mathbf{N} + (\mathbf{f}(u, v) - \mathbf{g}(s, t)) \cdot \mathbf{N}_v = 0 \quad (10)$$

$$-\mathbf{g}_s \cdot \mathbf{N} = 0 \quad (11)$$

$$-\mathbf{g}_t \cdot \mathbf{N} = 0. \quad (12)$$

Noting that the normal  $\mathbf{N}$  is orthogonal to the tangent plane formed by the partials  $\mathbf{f}_u$  and  $\mathbf{f}_v$ , we may remove the  $\mathbf{f}_u \cdot \mathbf{N}$  and  $\mathbf{f}_v \cdot \mathbf{N}$  terms. Additionally, the partials of  $\mathbf{N}$  lie in the tangent plane of  $\mathbf{f}$ . The equivalent constraint may be formulated by replacing these partials with the partials of  $\mathbf{f}$ . These substitutions form a simplified set of equations.

$$\mathbf{N} \cdot \mathbf{g}_s = 0 \quad (13)$$

$$\mathbf{N} \cdot \mathbf{g}_t = 0 \quad (14)$$

$$(\mathbf{f} - \mathbf{g}) \cdot \mathbf{f}_u = 0 \quad (15)$$

$$(\mathbf{f} - \mathbf{g}) \cdot \mathbf{f}_v = 0. \quad (16)$$

The first two equations constrain the solution to collinear normals and the second two maintain collinearity of the closest points with the surface normals. This set of equations is analogous to those

used in [Baraff, 1990] and [Snyder, 1995], however, we have expressed them in a form more suitable to the demands of haptic rate computation.

This system of equations may be locally solved through incremental updates of  $\mathbf{u}$  using multi-dimensional Newton’s method,

$$\Delta \mathbf{u} = \mathbf{J}^{-1}(-\mathbf{F}) \quad (17)$$

where  $\mathbf{F}$  is the constraint violation defined by Eqs. 13-16, and  $\mathbf{J}$  is the Jacobian of  $\mathbf{F}$ .

This formulation may result in extraneous zeros, since there may be multiple locations where the surfaces’ tangent planes are parallel and are at a local distance extrema. However, these undesired roots are typically at polar opposites of the model and are less common than for the minimum distance formulation.

### 3.6 Velocity Extremal Distance Formulation

A different approach in the penetrating case is to take surface velocity into account. The relation of parametric contact differentials with the relative linear and angular velocity between the two surfaces can be used to provide incremental tracing updates. In the Appendix, the authors have extended the results of [Cremer, 1996, Montana, 1986] to arbitrary surface parameterizations, i.e. the contact velocity relations have been extended to surfaces whose partials are not everywhere perpendicular, making the relations useful for common models. The parametric contact coordinates  $\mathbf{u}$  may be integrated through time using the following relation, derived in the Appendix,

$$\dot{\mathbf{u}} = \mathbf{A} \begin{bmatrix} \mathbf{v}_{x,y} \\ \omega_{x,y} \end{bmatrix} \quad (18)$$

where  $[\mathbf{v}^T, \omega^T]^T$  are the relative linear and angular velocity between the surfaces relative to the frame  $\mathbf{x}_g = \mathbf{g}_u / \|\mathbf{g}_u\|$ ,  $\mathbf{z}_g = \mathbf{g}_u \times \mathbf{g}_v / \|\mathbf{g}_u \times \mathbf{g}_v\|$ , and  $\mathbf{y}_g = \mathbf{z}_g \times \mathbf{x}_g$  located at the contact point  $\mathbf{g}(\mathbf{u})$  (Fig.6), and

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_\theta (\mathbf{E}^f - \beta \mathbf{F}^f_{-y,x}) & -\mathbf{E}^g \\ -(\mathbf{R}_\theta \mathbf{F}^f) & -\mathbf{F}^g \end{bmatrix}^{-1} \quad (19)$$

$$\mathbf{E}^f = \begin{bmatrix} \mathbf{x}_f & \mathbf{y}_f \end{bmatrix}^T \mathbf{f}_u \quad (20)$$

$$\mathbf{F}^f = \begin{bmatrix} \mathbf{x}_f^T \mathbf{z}_u \\ \mathbf{y}_f^T \mathbf{z}_u \end{bmatrix} \quad (21)$$

$$\mathbf{R}_\theta = \begin{bmatrix} \cos \theta & -\sin \theta \\ -\sin \theta & -\cos \theta \end{bmatrix} \quad (22)$$

where  $\theta$  represents the angle between parametric axes  $\mathbf{g}_s$  and  $\mathbf{f}_u$ ,  $\beta$  is the distance between contacts,  $\mathbf{x}_f = \mathbf{f}_u / \|\mathbf{f}_u\|$ ,  $\mathbf{z}_f = \mathbf{f}_u \times \mathbf{f}_v / \|\mathbf{f}_u \times \mathbf{f}_v\|$ , and  $\mathbf{y}_f = \mathbf{z}_f \times \mathbf{x}_f$ .  $\mathbf{E}^g, \mathbf{F}^g$  are defined in a similar manner.

The authors have also developed the relation in terms of world frame quaternion velocity body coordinates in the Appendix, which may be more convenient for use in haptics.

### 3.7 Comparing the Methods

The advantage to using Newton’s iterative method is that it converges to a solution given a close initial guess. We are required to use the Newton method during the non-penetrating case so that we obtain an exact starting point for the velocity method.

The advantage to the velocity space method is that it is an exact relation at that instant in time; it is not an iterative numerical method. The integration of  $\dot{\mathbf{u}}$  provides a highly accurate, strictly

continuous tracing update. It is very well conditioned and does not suffer from the optimization problems of Newton's method. However, it does not converge to the true minimal distance given only an approximate starting point. It is a good algorithm for generating virtual proxy information because it is a strictly local distance update; the curves  $\gamma(t)$  and  $\zeta(t)$  are continuous, where they may be discontinuous in small intervals with the Newton method.

The authors have found that haptics surface tracing should be as minimally confusing as possible. When given a choice of switching to a non-local point, which would cause discontinuous force feedback, it has been our experience that the more local choice is desirable. This choice is enforced by the continuous updates provided by the velocity method.

## 4 ALGORITHM & IMPLEMENTATION

In sum, we have the following algorithm for the non-penetrating and penetrating case:

- Far : Global distance refinement, obtain approximate  $\mathbf{u}$
- Near : Newton iteration, obtain exact  $\mathbf{u}$
- Very Near and Penetrating : Velocity formulation, using exact  $\mathbf{u}$

During tracking (the outside, non-penetration case), we use a global "monitoring" mechanism. Newton iteration and the velocity method are run concurrently during this case. kilohertz rate updates are not critical since the haptics control is returning no force during non-penetration.

Once the Newton or velocity methods detect a penetration, the monitoring and Newton tracking are turned off. It is assumed that non-local jumps are not possible because a haptics device can hold a user to within .3 mm of the surface. Global jumps are also not desirable because of the need for virtual proxies for the finger model (see Fig.2).

Due to the use of extremal points in tracking and tracing parametric surfaces, the existing NURBS trimming implementation [Thompson, 1999] may still be used. This model is an approximation for effects such as falling off or transitioning between edges, because the "middle" of the finger is considered to be completely off as soon as the extremal point is off. A full model for tracing edges is considerably more costly in terms of computation and is a subject of future work. However, the important perception of falling off edges that is remarkably well rendered with haptic devices is not significantly diminished with this approximate model.

## 5 RESULTS

Our approach is efficient because two surface evaluations for the contact points and surface partial derivatives at the points are required for the velocity and Newton formulations. Timing results [Johnson, 1998b] have shown that the point and partial evaluations are only slightly slower than evaluating only the point. Running times on an SGI R10000 Onyx 2 for two surface evaluations inside Alpha\_1 are about .07 milliseconds. Other operations including the inverse of the  $4 \times 4$  matrix and other restacking required in the velocity and Newton formulation are not insignificant, but run in .01 milliseconds (that is, the cost of the methods excluding the surface evaluations). Thus, a single processor system can easily perform control and surface-surface analysis at several kilohertz update rates.

Figure 4 shows the concurrent global monitoring and local Newton and velocity parametric tracing to enable surface-surface contact analysis at kilohertz rates for haptics control.

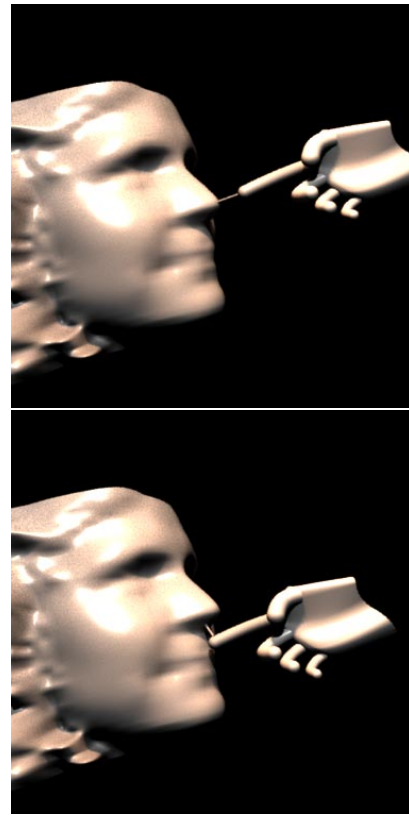


Figure 4: Combined global monitoring and local parametric tracing enable surface-surface contact analysis at kilohertz rates for haptics control.

We have avoided the introduction of unstable artifacts from a purely iterative numerical method at the time of impact that would be felt by the user. The generally accepted noticeable level of vibration are on the order of less than a micron at various frequencies from 1 Hz to 1kHz [NRC, 1995]. The artifacts caused by the numerical methods may be many orders of magnitude greater than the just noticeable error that a user may perceive.

Numerical integration of the results from the velocity formulation may be done with the simple Euler's method for short intervals of time in a haptics environment due to the very small step sizes between servo loop cycles. For other applications that use larger timesteps, as occurs in our simulation debugging code, we have used standard fourth order numerical integration techniques. A very long tracing sequence, on the order of  $10^8$  seconds for typical user motions, can be performed in practice with this integration technique without accumulating noticeable errors. Periodic "restarts" due to user transition to the non-penetrating condition and subsequent tracking by Newton's method occur quite often. Even higher order integration methods can be employed if some unusual circumstance or application requires it without excessive cost due to the efficiency of our tracking techniques.

The reliability of these distance methods depends partially on the underlying stability of the numerical methods. Singular regions in the case of point-to-surface computations were derived in [Johnson, 1998b]; we expect similar conditions for the surface-to-surface extremal distance computations. During nearly singular concave cases, as in Fig.5, the condition numbers during the first iteration of Newton's method is roughly 3 or 4, where it has usually been between 1 and 2 in other configurations. The velocity method maintains a condition number of roughly 1 or 1.2 for the



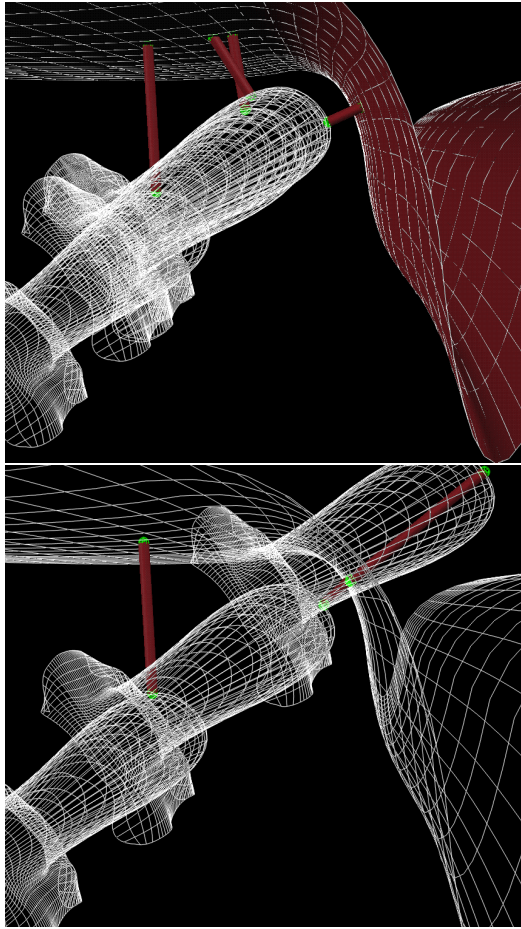


Figure 5: The reformulated Newton local distance method (top) may have problems with concave cases that the velocity (bottom) does not have.

cases tried so far, for nearly singular and other cases. Because condition numbers less than 100 are considered to be small, the loss of precision upon matrix inverse operations is shown to be small with both methods. Upon absolute singular configurations, both methods have singular matrices. However, Newton’s method has additional convergence requirements [McCalla, 1967] that can produce additional instability in concave regions, even when the condition number is very small. The velocity method does not exhibit these instabilities. We are investigating more sophisticated numerical methods to improve the reliability of the Newton method approach.

## 6 CONCLUSION

The tracing update formulations presented here provide improved surface tracing interactions. A fast, compact method for surface-surface updates for the nearly penetrating and penetrating case have been developed and analyzed. The velocity formulation has been introduced for use in haptics penetration.

## ACKNOWLEDGMENT

Thanks go to the students and staff of the *Alpha-I* project, within which this work was developed. Support for this research was pro-

vided by NSF Grant MIP-9420352, by DARPA grant F33615-96-C-5621, and by the NSF and DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219).

## 7 APPENDIX: Surface Contact Velocity Formulation

A number of different derivations of the kinematics of contact have been developed in the last 15 years. These formulations relate the rate of change of the parametric contact coordinates to the Cartesian velocity and angular velocity of the bodies in contact. Previous works have been limited to surfaces parameterized have orthogonal surface partial derivatives. Some are also limited the in-contact case [Montana, 1986, Montana, 1988, Murray, 1990, Cai, 1987]. While any surface may be reparameterized to be orthogonal, it may be expensive and impractical to find such a parameterization. The finger may bend in our interactive application. Finding the reparameterization with full numerical precision is also a problem [Maekawa, 1996]. We develop a new derivation for the not-in-contact, non-orthogonal surface parameterizations.

When both surfaces have partials that are everywhere orthogonal, i.e.  $\mathbf{f}_u \cdot \mathbf{f}_v = 0$  and  $\mathbf{g}_s \cdot \mathbf{g}_t = 0$ ,  $\forall \mathbf{u}$  in the parametric domain, an original result from [Montana, 1986], extended by [Cremer, 1996] for the not in contact case, derives the surface kinematics as

$$\begin{aligned}\dot{\mathbf{u}}_f &= \mathbf{I}_f^{-1} \mathbf{R}_\theta (\mathbf{I}_g + \tilde{\mathbf{I}}_f + \beta \mathbf{I}_g \tilde{\mathbf{I}}_f)^{-1} \begin{bmatrix} -w_y \\ w_x \end{bmatrix} + \mathbf{I}_g \begin{bmatrix} v_x \\ v_y \end{bmatrix} \\ \dot{\mathbf{u}}_g &= \mathbf{I}_g^{-1} (\mathbf{I}_g + \tilde{\mathbf{I}}_f + \beta \tilde{\mathbf{I}}_f \mathbf{I}_g)^{-1} ((1 + \beta \tilde{\mathbf{I}}_f) \begin{bmatrix} -w_y \\ w_x \end{bmatrix} - \tilde{\mathbf{I}}_f \begin{bmatrix} v_x \\ v_y \end{bmatrix})\end{aligned}$$

where  $\beta$  is the (signed) distance between contacts,  $\mathbf{1}$  is the  $2 \times 2$  identity matrix, the relative linear and angular surface velocities of surface  $\mathbf{g}$  relative to surface  $\mathbf{f}$  be denoted by  $\mathbf{v}, \omega$ , the surface contact velocities be  $\mathbf{v}^f, \omega^f$  and  $\mathbf{v}^g, \omega^g$ ,  $\mathbf{I}$  is the first fundamental form and  $\mathbf{II}$  is the surface curvature or second fundamental form, with subscripts for surfaces  $\mathbf{f}$  and  $\mathbf{g}$ .  $\theta$  represents the angle between parametric axes  $\mathbf{g}_s$  and  $\mathbf{f}_u$ , let  $\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ -\sin\theta & -\cos\theta \end{bmatrix}$  and  $\tilde{\mathbf{I}} = \mathbf{R}_\theta \mathbf{I} \mathbf{R}_\theta^T$ . The relation may be written in the following matrix form,

$$\dot{\mathbf{u}} = \mathbf{A} \begin{bmatrix} \mathbf{v}_{x,y} \\ \omega_{x,y} \end{bmatrix}. \quad (23)$$

### 7.1 Non-Orthogonal Parameterizations

We provide a new derivation for  $\mathbf{A}$  for regular parametric surfaces whose partials are not orthogonal so the result in Eqn. 23 can be used for typical models constructed by CAD systems.

The parametric contact frames in Fig. 6 for the extremal distance context are defined with an orthonormal set of vectors.  $\mathbf{R}_f = [\mathbf{x}_f \ \mathbf{y}_f \ \mathbf{z}_f]$  is the rotation matrix from the local contact frame to the world frame, where  $\mathbf{x}_f = \mathbf{f}_u / \|\mathbf{f}_u\|$ ,  $\mathbf{z}_f = \mathbf{f}_u \times \mathbf{f}_v / \|\mathbf{f}_u \times \mathbf{f}_v\|$ ,  $\mathbf{y}_f = \mathbf{z}_f \times \mathbf{x}_f$ .  $\mathbf{R}_g$  is similarly defined.  $\mathbf{z}_f$  and  $\mathbf{z}_g$  are parallel free vectors (Fig. 6).

Comparison of the relative surface velocities<sup>1</sup> can be used to re-

<sup>1</sup>A subscript with x or y such as  $\mathbf{a}_x$  will denote the first component of the vector. Several subscripts, such as  $\mathbf{a}_{-x,y}$  will represent a two-vector containing the negative of the first component and the second component of  $\mathbf{a}$ . A superscript  $T$  as in  $\mathbf{a}^T$  will denote a vector or matrix transpose. Partitions of matrices may be selected with (row,column) indexing, with “a:b” for a range or “:” for all rows or columns, as in the *Matlab*<sup>TM</sup> notation. The operator *extract\_skew\_symmetric* retrieves 3 independent components from 9 elements of a skew symmetric matrix.

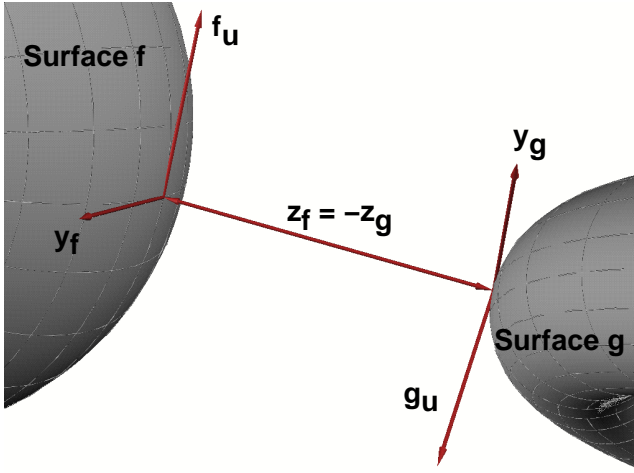


Figure 6: Closest point surface contact frames. Velocity relations allow the contact coordinate velocities to be found.

late the parametric contact coordinates  $\mathbf{u}_f$  and  $\mathbf{u}_g$  with the linear and angular surface velocities. Let the surface velocities  $\mathbf{v}^f$  and  $\mathbf{v}^g$  and relative surface velocity  $\mathbf{v}$  be in the frame  $\begin{bmatrix} \mathbf{x}_g & \mathbf{y}_g & \mathbf{z}_g \end{bmatrix}$ . In the extremal distance context (see Fig. 6), we have

$$\mathbf{v}_{x,y}^g + \mathbf{v}_{x,y} = \mathbf{R}_\theta \mathbf{v}_{x,y}^f + \beta \mathbf{R}_\theta \omega_{y,-x}^f, \quad (24)$$

$$\omega_{y,-x}^g + \omega_{y,-x} = -\mathbf{R}_\theta \omega_{y,-x}^f. \quad (25)$$

The terms  $\mathbf{v}^f$  and  $\mathbf{v}^g$  will contain  $\dot{\mathbf{u}}_f$  and  $\dot{\mathbf{u}}_g$  in the following equations through due to the chain rule of differentiation. We derive matrices  $\mathbf{E}_{x,y}^f$  and  $\mathbf{F}_{x,y}^f$  for surface  $f$  (and analogously for surface  $g$ ) from the relations of linear and angular velocity to separate  $\dot{\mathbf{u}}_f$  and  $\dot{\mathbf{u}}_g$  from other terms,

$$\mathbf{v}_{x,y}^f = \mathbf{R}_{f,x,y}^T \dot{\mathbf{f}}_{x,y} = \begin{bmatrix} \mathbf{x}_f & \mathbf{y}_f \end{bmatrix}^T \mathbf{f}_u \quad \dot{\mathbf{u}}_f = \mathbf{E}_{x,y}^f \dot{\mathbf{u}}_f, \quad (26)$$

$$\begin{aligned} \omega_{y,-x}^f &= \text{extract\_skew\_symmetric}(\mathbf{R}_f^T \dot{\mathbf{R}}_f)_{y,-x} \\ &= \begin{bmatrix} \mathbf{x}_f^T \mathbf{z}_u \\ \mathbf{y}_f^T \mathbf{z}_u \end{bmatrix}_{2 \times 2} \dot{\mathbf{u}}_f = \mathbf{F}_{y,-x}^f \dot{\mathbf{u}}_f. \end{aligned} \quad (27)$$

Using Eqns. 24,25,26,27, the general non-orthogonal case is reduced to the  $4 \times 4$  system,

$$\begin{bmatrix} \mathbf{R}_\theta(\mathbf{E}_{x,y}^f - \beta \mathbf{F}_{y,-x}^f) & -\mathbf{E}_{x,y}^g \\ (-\mathbf{R}_\theta \mathbf{F}_{y,-x}^f)_{x,y} & -\mathbf{F}_{x,y}^g \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_f \\ \dot{\mathbf{u}}_g \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{x,y} \\ \omega_{x,y} \end{bmatrix}. \quad (28)$$

This system can be solved quickly for  $\dot{\mathbf{u}}$  using the following pseudocode fragment for arbitrary surface parameterizations. We rewrite the inverse of the  $4 \times 4$  coefficient matrix in Eq. 28,

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_\theta(\mathbf{E}_{x,y}^f - \beta \mathbf{F}_{y,-x}^f) & -\mathbf{E}_{x,y}^g \\ -(\mathbf{R}_\theta \mathbf{F}_{y,-x}^f)_{x,y} & -\mathbf{F}_{x,y}^g \end{bmatrix}^{-1} \quad (29)$$

to be solved even more efficiently as a series of  $2 \times 2$  matrix inverses and multiplications,

```
iEg_xy=inv(Eg_xy);
RphiFf_xy=Rphi*Ff_xy;
dRphiFf_xy=d*RphiFf_xy;
Fg_xyiEg_xy=Fg_xy*iEg_xy;
RphiEf_xy=Rphi*Ef_xy;

H=Fg_xyiEg_xy*(RphiEf_xy+dRphiFf_xy)-
RphiFf_xy;
iH=inv(H);
J=iH*[Fg_xyiEg_xy -eye(2)];

A=[J; iEg_xy*(RphiEf_xy*J+
dRphiFf_xy*J-[eye(2) zeros(2)])];
```

**Proof:** From Eqs.24,26, we may write

$$\dot{\mathbf{u}}_g = \mathbf{E}_{x,y}^g{}^{-1}[\mathbf{R}_\theta \mathbf{E}_{x,y}^f \dot{\mathbf{u}}_f + \beta \mathbf{F}_{y,-x}^f \dot{\mathbf{u}}_f - \mathbf{v}]. \quad (30)$$

Substituting into Eq.25, we have

$$\mathbf{F}^g \mathbf{E}_{x,y}^g{}^{-1}[\mathbf{R}_\theta \mathbf{E}_{x,y}^f \dot{\mathbf{u}}_f + \beta \mathbf{F}_{y,-x}^f \dot{\mathbf{u}}_f - \mathbf{v}] + \omega = -\mathbf{R}_\theta \mathbf{F}^f \dot{\mathbf{u}}_f. \quad (31)$$

Gathering  $\dot{\mathbf{u}}_f$ ,

$$\mathbf{F}^g \mathbf{E}_{x,y}^g{}^{-1}[\mathbf{R}_\theta \mathbf{E}_{x,y}^f + \beta \mathbf{F}_{y,-x}^f + \mathbf{R}_\theta \mathbf{F}^f] \dot{\mathbf{u}}_f = \mathbf{F}^g \mathbf{E}_{x,y}^g{}^{-1} \mathbf{v} - \omega. \quad (32)$$

Expressing this equation as a linear system, we have

$$\mathbf{H} \dot{\mathbf{u}}_f = \mathbf{F}^g \mathbf{E}_{x,y}^g{}^{-1} \mathbf{v} - \omega, \quad (33)$$

for which we can solve for the contact coordinates for surface  $f$ ,

$$\dot{\mathbf{u}}_f = \mathbf{H}^{-1}[\mathbf{F}^g \mathbf{E}_{x,y}^g{}^{-1} \quad -\mathbf{1}_{2 \times 2}] \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix}, \quad (34)$$

For convenience, let us represent

$$\mathbf{J}_{2 \times 4} = \mathbf{H}^{-1}[\mathbf{F}^g \mathbf{E}_{x,y}^g{}^{-1} \quad -\mathbf{1}_{2 \times 2}] \text{ so that}$$

$$\dot{\mathbf{u}}_f = \mathbf{J} \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix}, \quad (35)$$

Now substituting this solution back into Eq.30,

$$\dot{\mathbf{u}}_g = \mathbf{E}_{x,y}^g{}^{-1}[\mathbf{R}_\theta \mathbf{E}_{x,y}^f \mathbf{J} \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} + \beta \mathbf{F}_{y,-x}^f \mathbf{J} \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} - \mathbf{v}]. \quad (36)$$

$$= \mathbf{E}_{x,y}^g{}^{-1}[\mathbf{R}_\theta \mathbf{E}_{x,y}^f \mathbf{J} + \beta \mathbf{F}_{y,-x}^f \mathbf{J} - [\mathbf{1}_{2 \times 2} \quad \mathbf{0}_{2 \times 2}]] \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} \quad (37)$$

Finally, letting

$$\mathbf{J}_b = \mathbf{E}_{x,y}^g{}^{-1}[\mathbf{R}_\theta \mathbf{E}_{x,y}^f \mathbf{J} + \beta \mathbf{F}_{y,-x}^f \mathbf{J} - [\mathbf{1}_{2 \times 2} \quad \mathbf{0}_{2 \times 2}]], \text{ we may write}$$

$$\dot{\mathbf{u}} = \begin{bmatrix} \mathbf{J} \\ \mathbf{J}_b \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix}. \quad (38)$$

The matrix  $\mathbf{A}$  from Eq.29 is  $\begin{bmatrix} \mathbf{J} \\ \mathbf{J}_b \end{bmatrix}$ , completing the proof. The solution is roughly as efficient as previous methods since the inverse of two  $2 \times 2$  matrices and nine matrix multiplications, rather than four  $2 \times 2$  matrix inversions and six matrix multiplications, is required (see pseudocode fragment). The optimized matrix inverse and multiplication implementation is a constant cost and is a small fraction of the cost associated with a surface evaluation.



## 7.2 Cartesian and Quaternion Generalized Velocities

Now we express the relative surface velocities in terms of world frame body coordinate velocities so that integration of orientation is possible (integration of angular velocity is meaningless). Define the local contact frame through the rotation matrix

$$\mathbf{R}_{loc} = \begin{bmatrix} \mathbf{x}_g & \mathbf{y}_g & \mathbf{z}_g \end{bmatrix}^T. \quad (39)$$

Let  $\mathbf{G}_f = \mathbf{G}(\mathbf{q}_{f,rot})$ ,  $\mathbf{G}_g = \mathbf{G}(\mathbf{q}_{g,rot})$ , where  $\mathbf{G}(\mathbf{q}_{rot})$  is the matrix operator mapping quaternion velocities to angular velocities [Haug, 1992, Shabana, 1998], given by

$$\mathbf{G}(\mathbf{q}_{rot}) = 2 \begin{bmatrix} -q_{rot2} & q_{rot1} & q_{rot4} & -q_{rot3} \\ -q_{rot3} & -q_{rot4} & q_{rot1} & q_{rot2} \\ -q_{rot4} & q_{rot3} & -q_{rot2} & q_{rot1} \end{bmatrix}. \quad (40)$$

The velocity  $[\mathbf{v}^T \omega^T]^T$  is the motion of surface  $g$  relative to surface  $f$ . We write our world space velocities in terms of the local frame. We relate relative angular velocity and world frame quaternion velocity by

$$\omega = \mathbf{R}_{loc} \begin{bmatrix} -\mathbf{T}_f \mathbf{G}_f & \mathbf{T}_g \mathbf{G}_g \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{f,rot} \\ \dot{\mathbf{q}}_{g,rot} \end{bmatrix}, \quad (41)$$

and relative linear velocity to Cartesian velocity through

$$\mathbf{v} = \mathbf{R}_{loc}(\dot{\mathbf{q}}_{g,tr} - \dot{\mathbf{q}}_{f,tr} + (\omega_g^{gl} - \omega_f^{gl}) \times (\mathbf{g}(\mathbf{u}) - \mathbf{q}_{g,tr})). \quad (42)$$

$\times$  denotes the vector cross product.  $\mathbf{T}_f$  and  $\mathbf{T}_g$  are rotations from the local frame to the world frame defined by quaternions  $\mathbf{q}_{f,rot}$  and  $\mathbf{q}_{g,rot}$ . We write the relative surface velocities  $[\mathbf{v}^T, \omega^T]^T$  as

$$\begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{loc} & -\mathbf{R}_{loc}(\mathbf{g}(\mathbf{u}) - \mathbf{q}_{g,tr}) \times \\ \mathbf{0} & \mathbf{R}_{loc} \end{bmatrix} * \begin{bmatrix} -\mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 4} \\ \mathbf{0}_{3 \times 3} & -2\mathbf{T}_f \mathbf{G}_f & \mathbf{0}_{3 \times 3} & 2\mathbf{T}_g \mathbf{G}_g \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_f \\ \dot{\mathbf{q}}_g \end{bmatrix} \quad (43)$$

where  $\times$  in Eq. 43 denotes the  $3 \times 3$  skew symmetric matrix that performs the operation of a cross product (obtained from the three components of a vector, i.e.  $\mathbf{a} \times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$ ). From Eq. 38, the truncated part  $[\mathbf{v}_{x,y}^T, \omega_{x,y}^T]^T$  is all that is required. Let  $\mathbf{B}$  contain the first two rows and rows four and five of Eq. 43. Substituting  $[\mathbf{v}_{x,y}^T, \omega_{x,y}^T]^T$  into Eq. 38, yields

$$\dot{\mathbf{u}} = \mathbf{AB} \begin{bmatrix} \dot{\mathbf{q}}_f \\ \dot{\mathbf{q}}_g \end{bmatrix}. \quad (44)$$

## 7.3 Non-Orthogonal Surface-Curve Velocity Formulation

Similarly, it may be shown that the surface-curve extremal distance equations may be extended to for arbitrary surface parameterizations.

The time derivative of the parametric contact coordinates for surface  $f$  and curve  $g$  may be written as a function of linear and angular velocity multiplying a matrix operator,

$$\dot{\mathbf{u}}_{3 \times 1} = \begin{bmatrix} \mathbf{R}_\theta(\mathbf{E}_u^f + d\mathbf{F}_{x,y}^f) & -\mathbf{g}_{u,x,y} \\ -\mathbf{R}_\theta \mathbf{F}_{x,y}^f & -k_g \sin(\phi) \mathbf{g}_{u,x,y} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{v}_{x,y} \\ \omega_y \end{bmatrix}_{3 \times 1}, \quad (45)$$

where

$$k_g = \frac{\|\mathbf{g}_u \times \mathbf{g}_{uu}\|}{\|\mathbf{g}_u\|^3} \quad (46)$$

and  $\phi$  is the angle between the curve normal (which is  $-\mathbf{z}^f$  for the extremal distance case) and the curve binormal  $\mathbf{b}$ , about the curve  $x$  axis  $\mathbf{g}_u$ . The binormal is

$$\mathbf{b} = \frac{\mathbf{g}_u \times \mathbf{g}_{uu}}{\|\mathbf{g}_u \times \mathbf{g}_{uu}\|}. \quad (47)$$

Eq. 43 is again used to establish this relation in terms of quaternion and Cartesian velocities, using rows 1,2, and 5.

## References

- [Bajaj, 1988] Bajaj, C., Hoffmann, C., Lynch, R., Hopcroft, J., "Tracing surface intersections," in *Computer Aided Geometric Design*, Vol 5, 1988, pp. 285-307.
- [Baraff, 1990] Baraff, D., "Curved surfaces and coherence for non-penetrating rigid body simulation", in *Computer Graphics Proceedings, SIGGRAPH*, 24(4): 19-28, 1990.
- [Cai, 1987] Cai, C., Roth, B., "On the spatial motion of rigid bodies with point contact," in *International Conference on Robotics and Automation*, pp. 686-695, March 1987.
- [Cameron, 1997] Cameron, S., "Enhancing GFK: Computing Minimum and Penetration Distances between Convex Polyhedra," in *International Conference on Robotics and Automation*, April 1997.
- [Cremer, 1996] Anitescu, M., Cremer, J., and Potra, F., "Formulating 3D Contact Dynamics Problems," *Mech. Struct. & Mach.*, vol. 24, no. 4, pp. 405-437, Nov. 1996.
- [Cohen, 1980] Cohen, E., Lyche, T., and Riesenfeld, R., "Discrete B-Splines And Subdivision Techniques In Computer Aided Geometric Design And Computer Graphics," *Computer Graphics and Image Processing*, Vol 14, Number 2, October 1980.
- [De, 1998] De, S., Srinivasan, M., "Rapid Rendering of "Tool-Tissue" Interactions in Surgical Simulations: Thin Walled Membrane Models", PHANTOM User's Group Proceedings, PUG 1998, [http://www.sensable.com/community/PUG98\\_papers.htm](http://www.sensable.com/community/PUG98_papers.htm).
- [Gregory, 1998] Gregory, A., Lin, M., Gottschalk, S., Taylor, R., "H-Collide: A Framework for Fast and Accurate Collision Detection for Haptic Interaction," UNC Report, currently unpublished, 1998.
- [Gilbert, 1988] Gilbert, E., Johnson, D., Keerthi, S. "A Fast Procedure for Computing the Distance between Complex Objects in Three-Dimensional Space," *IEEE Journal of Robotics and Automation*, pp. 193-203, April 1988.
- [Haug, 1992] Haug, E., *Intermediate Dynamics*, Prentice Hall, 1992.
- [Johnson, 1998a] Johnson, D. E., and Cohen, E., "A framework for efficient minimum distance computations," *Proc. IEEE Intl. Conf. Robotics & Automation*, Leuven, Belgium, May 16-21, 1998, pp. 3678-3684.

- [Johnson, 1998b] Johnson, D.E., and Cohen, E., "An improved method for haptic tracing of sculptured surfaces," Symp. on Haptic Interfaces, ASME International Mechanical Engineering Congress and Exposition, Anaheim, CA, Nov. 15-20, 1998, in press.
- [Kriezis, 1992] Kriezis, G., Patrikalakis, N., Wolder, F., "Topological and differential-equation methods for surface intersections," in *Computer-Aided Design*, vol. 24, no. 1, January 1992, pp. 41-51.
- [Lin, 1994] M.C. Lin, D. Manocha, and J. Canny. Fast contact determination in dynamic environments. in *IEEE Conference on Robotics and Automation*, pp. 602-609, 1994.
- [Maekawa, 1996] Maekawa, T., Wolter, F., Patrikalakis, N., "Umbilics and lines of curvature for shape interrogation," in *Computer Aided Geometric Design*, vol. 13, no. 2, March 1996.
- [McCalla, 1967] McCalla, T., Introduction to Numerical Methods and FORTRAN Programming, John Wiley & Sons, Inc., 1967.
- [Montana, 1986] Montana, D., J., 1986, Tactile sensing and the kinematics of contact, Ph.D. Thesis, Division of Applied Sciences, Harvard University.
- [Montana, 1988] Montana, D., "The Kinematics of Contact and Grasp", in *Int. J. of Rob. Res.*, vol. 7, no. 3, June 1988.
- [Murray, 1990] Murray, R., Robotic Control and Nonholonomic Motion Planning, Ph.D. Dissertation, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, 1990.
- [NRC, 1995] National Research Council, Virtual Reality: Scientific and Technological Challenges, Committee on Virtual Reality Research and Development, National Academy Press, 1995, p.69.
- [Quinlan, 1994] Quinlan, S. "Efficient Distance Computation between Non-Convex Objects," IEEE Int. Conference on Robotics and Automation, pp. 3324-3329, 1994.
- [Ruspini, 1997] Ruspini, D., Kolarov, K., Khatib, O., "The Haptic Display of Complex Graphical Environments," in *Computer Graphics Proceedings, SIGGRAPH*, August, 1997.
- [Ruspini, 1998] Ruspini, D., Khatib, O., "Dynamic Models for Haptic Rendering Systems," in *Advances in Robot Kinematics: ARK'98*, June 1998, Strobl/Salzburg, Austria, pp 523-532.
- [Sato, 1996] Sato, Y., Hirata M., Maruyama T., Arita Y., "Efficient Collision Detection using Fast Distance Calculation: Algorithms for Convex and Non-Convex Objects," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, April, 1996, pp. 771-778.
- [Shabana, 1998] Shabana, A., Dynamics of Multibody Systems, Cambridge University Press, 1998.
- [Stewart, 1997] Stewart P., Chen Y., Buttolo P., "CAD Data Representations for Haptics Virtual Prototyping," in *Proceedings of DETC '97*, 1997 ASME Design Engineering Technical Conferences, Sept. 14-17, 1997, Sacramento, CA.
- [Snyder, 1995] Snyder, J., "An Interactive Tool for Placing Curved Surfaces without Interpenetration," in *Computer Graphics Proceedings, SIGGRAPH*, Los Angeles, August 6-11, 1995, pp. 209-216.
- [Thompson, 1997] Thompson II, T.V., Nelson, D.D., Cohen, E., and Hollerbach, J.M., "Maneuverable NURBS models within a haptic virtual environment," *6th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, DSC-Vol. 61, (Dallas, TX), pp. 37-44, Nov. 15-21, 1997.
- [Thompson, 1999] Thompson II, T.V., Cohen, E., "DIRECT HAPTIC RENDERING OF COMPLEX TRIMMED NURBS MODELS," in *8th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, (Nashville, TN), 1999.
- [Zilles, 1995] Zilles, C.B., and Salisbury, J.K., "A Constraint-based God-object Method For Haptic Display," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots*, Vol 3, pp. 146-151, 1995.

## 18

# Tactual Displays for Sensory Substitution and Wearable Computers

Hong Z. Tan

*Purdue University*

Alex Pentland

*Massachusetts Institute of Technology*

### 1. INTRODUCTION

A major challenge in building practical wearable computer systems is the development of output devices to display or transmit information to the human user. Much effort has been devoted to visual displays that are lightweight and have high resolution. Such efforts are warranted since visual displays are still the dominant output devices used by most computing systems. Auditory displays are now becoming the norm of multimedia systems in addition to visual displays. Whereas vision is best suited for perceiving text and graphics, and audition for speech and music, the sense of touch is intimately involved in nonverbal communication. Whether it is a tap on the shoulder to get someone's attention or a firm handshake to convey trust, touch enables us to exchange information directly with people and the environment through physical contact. The skin is the largest organ of our body, yet only a small portion of it (i.e., the hands) is engaged in most human-computer interactions.

For a long time, the sense of touch has been regarded as the inferior sense as compared to vision or audition. However, the potential to receive

information tactually is well illustrated by some natural (i.e., nondevice related) methods of tactual speech communication. Particularly noteworthy is the so-called Tadoma method that is employed by some individuals who are both deaf and blind. In Tadoma, one places a hand on the face and neck of a talker and monitors a variety of actions associated with speech production. Previous research has documented the remarkable abilities of experienced Tadoma users (e.g., Reed, Rabinowitz, Durlach, Braida, Conway-Fithian, & Schultz, 1985). Not only can these individuals converse with both familiar and unfamiliar talkers at high performance levels, but they pick up additional features such as the speaker's accent. The Tadoma method is a living proof that high information transmission is possible through the somatosensory system.

Earlier work on wearable tactual displays has concentrated on assisting the blind to see and the deaf to hear through their sense of touch (i.e., sensory substitution). For example, the Optacon (Linville & Bliss, 1966) was initially developed in the 1960s for the daughter of one of its inventors as a reading aid for the blind. It converted images of printed materials to vibrational patterns on the index finger. The Optacon was probably one of the first commercially successful wearable tactual displays ever developed. With sufficient training, typical reading rates of 30–50 wpm can be achieved (Craig & Sherrick, 1982). Some exceptional subjects have demonstrated rates as high as 70–100 wpm (Craig, 1977). Recently, force-feedback systems have been developed for applications in teleoperation and virtual/augmented reality systems (Burdea, 1996). These systems can simulate forces that would have been generated by remote or virtual objects during manual manipulation, thereby enhancing an operator's sense of presence and task performance. Our current work is aimed at developing wearable tactual displays for general-purpose human-computer interactions. We are exploring new ways of utilizing the wearable tactual display technology developed for sensory substitution to convey information that is as intuitive as a sense of resistive force. Wearable computing also provides a unique environment for developing paradigms to distribute human-computer interfaces across the entire body and its various sensory channels.

The organization of this chapter is as follows. Section 2 defines terms that are used throughout this chapter. Section 3 presents a brief historical review of tactual displays that have been developed for sensory substitution. Section 4 discusses the requirements for wearable tactual displays. Section 5 describes a general-purpose tactual directional display that has been developed in our laboratory for wearable computing. Section 6 provides a summary.

## 2. DEFINITION OF TERMS

The term *haptics* refers to sensing and manipulation through the tactual sense. The human *tactual* sensory system is generally regarded as made up of two subsystems: the tactile and kinesthetic senses (see reviews by Loomis & Lederman, 1986; Clark & Horch, 1986). The *tactile* (or *cutaneous*) sense refers to the awareness of stimulation to the body surfaces mediated by sensors close to skin surfaces such as the mechanoreceptors. For example, when you touch a loudspeaker, your hands receive tactile stimulation. The *kinesthetic* sense (or *proprioception*) refers to the awareness of limb positions, movements, and muscle tensions mediated by sensors in the muscles, skin, and joints as well as a knowledge of motor commands sent to muscles (i.e., efference copy). For example, when you touch your nose with closed eyes, you rely on the kinesthetic sense to know where your fingertip is relative to your nose. When you hold an object in your hand, you use the kinesthetic sense to estimate the weight of the object. In both the nose touching and weight estimation cases, the tactile sense is also activated since the fingertip touches the nose, and the hand is in contact with the object. However, the information that is crucial to the successful execution of these tasks is derived primarily from the kinesthetic sense. In other words, the above distinction between tactile and kinesthetic senses is functional and task dependent.

The term *haptics* is often used in the literature. In this chapter, it refers to manipulation as well as perception through the tactual sense. An example of haptic interfaces is force-feedback joysticks for video games.

The term *display* refers to a human-machine interface that mainly transmits information from a machine to a human. The term *controller* refers to a human-machine interface that is mainly used by a human to control certain processes of a machine (Figure 18.1). The term *interface* is used in this chapter either to refer to a device that is both a display and a controller, or when such distinction is not important.

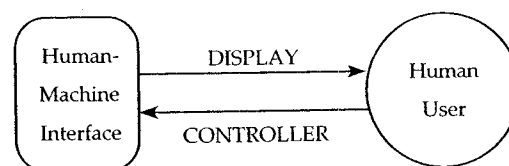


FIG. 18.1. Display vs. controller for human-machine interfaces.

### 3. TACTUAL DISPLAYS FOR SENSORY SUBSTITUTION

Most tactual communication systems for sensory substitution have been developed based upon two major principles: pictorial or frequency-to-place transformation. Devices for the blind tend to adopt the pictorial approach (i.e., direct translation of spatial-temporal visual information to the skin). Devices for the deaf are usually based on the cochlea model of speech (i.e., positional encoding of frequency information). Examples of pictorial tactual communication systems are the Optacon (OPTical-to-TACTile-CONverter), the Optohapt (OPTical-TO-HAPTics), the TVSS (Tactile Vision Substitution System), and the Kinotact (KINesthetic, OPTical and TACTile display).

The Optacon (Telesensory Corp, Mountain View, CA) is a direct translation reading aid for the blind that quantizes an area roughly the size of a letter into 144 black and white image points (24 rows and 6 columns) via photocells and displays these image points on a corresponding 24-by-6 array of vibrating pins (Linvill & Bliss, 1966). This portable system consisted of a small hand-held camera and a tactile display measuring 1.1 cm by 2.7 cm that fits under the fingertip of an index finger (Figure 18.2). Whenever a photocell detected a "black" spot, the corresponding pin vibrated. Reading speed with the Optacon varied from 10 to 100 wpm depending on the individual, the amount of training, and experience, with typical rates of 30–50 wpm (Craig & Sherrick, 1982).

The Optohapt consisted of a linear array of nine photocells that scanned vertically the output of an electric typewriter on paper and nine vibrators distributed along the body surface (Figure 18.3). The stimulation sites

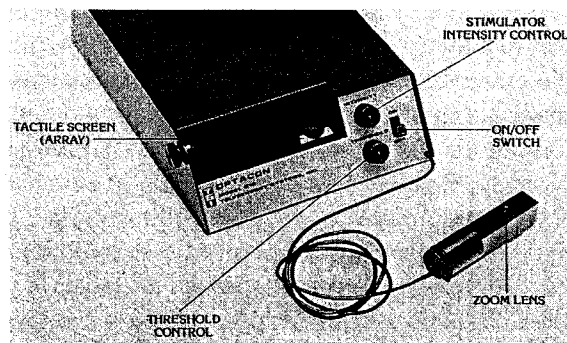


FIG. 18.2. The Optacon. Photograph courtesy of Telesensory Corp.

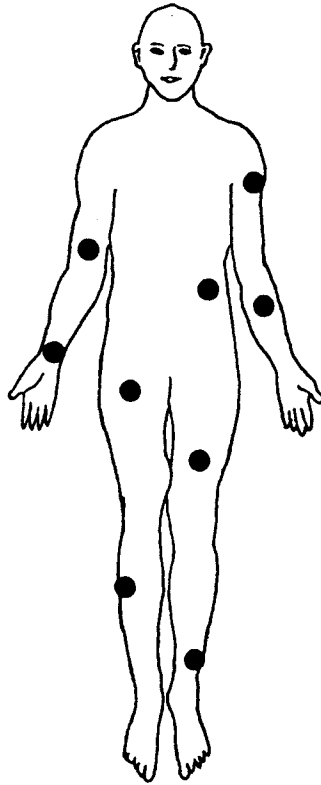


FIG. 18.3. The locations of the nine vibrators used in the Optohapt system. From Geldard (1966), reprinted by permission of Psychonomic Society, Inc.

were selected to avoid corresponding bodily points and were distributed as widely as possible over the skin surface. It was found that "raw" letters of the alphabet were not the most efficacious symbols because they lacked discriminability. It was suggested that the twenty-six most readily discriminated signals (such as the period, the colon, a filled square, etc.) be selected to encode letters of English (Geldard, 1966).

Whereas the Optacon and the Optohapt were aimed toward the transmission of text material, the TVSS was designed to transmit general visual images. It consisted of a television camera controlled by the user, and a 20-by-20 matrix of solenoid vibrators (spaced 12 mm apart) mounted in the back of a dental chair (Figure 18.4). Each solenoid would vibrate when the corresponding region in the camera's viewfinder was illuminated. Initial results showed that both sighted and blind subjects learned to recognize

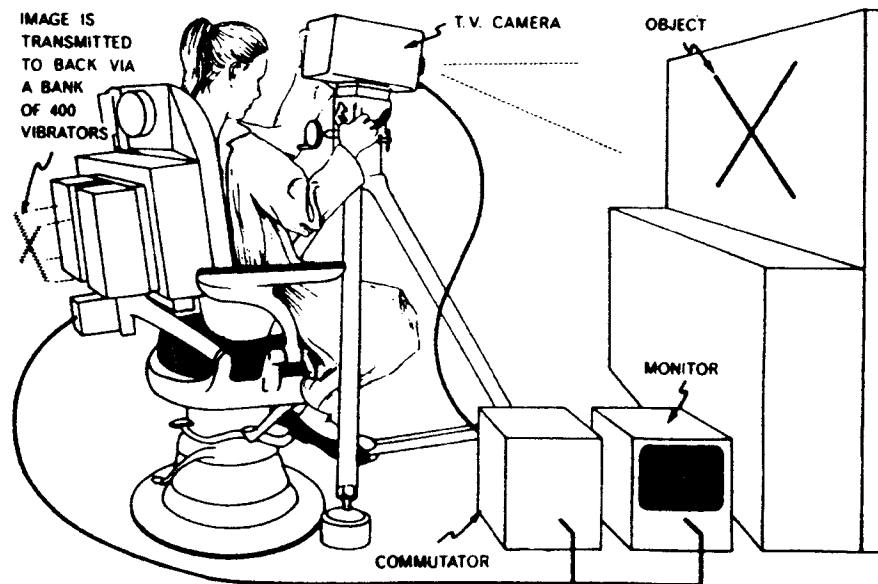


FIG. 18.4. The TVSS. From White et al. (1970), reprinted by permission of Psychonomic Society, Inc.

common objects (e.g., telephone, cup, etc.) and their arrangements in three-dimensional space. When given the control of the camera, the subjects also learned to externalize the objects presented tactually on their backs as being in front of them (Bach-y-Rita, 1972; White, Saunders, Scadden, Bach-Y-Rita, & Collins, 1970). Further investigation showed, however, that subjects had considerable difficulties in identifying internal details of a pattern, thus casting doubts on the skin's ability to identify complex visual patterns (White, 1973).

The Kinotact was very similar to TVSS in construction. It consisted of a 10-by-10 array of photocells, 100 switching circuits, and a corresponding 10-by-10 array of vibrators mounted on the back of a chair. Cutting off light to a particular photocell would switch on a corresponding vibrator. Block letters of the alphabet were used as stimuli in experiments during which subjects were required to identify letters based on the vibration patterns presented on their back. It was found that subjects could perform almost equally well whether or not the correspondence between columns of photocells and those of vibrators were in the same order (i.e., column 1 of photocells was connected to column 1 of vibrators, column 2 of photocells was connected to column 2 of vibrators, and so forth) or randomized (e.g.,



column 1 of photocells was connected to column 7 of vibrators, column 2 of photocells was connected to column 6 of vibrators, etc.), as long as there was a one-to-one correspondence between columns of photocells and vibrators. The amount of additional training for the subjects to learn the random mapping was minimal (Craig, 1973).

The results of studies with the Optohapt and Kinotact systems suggest that whereas a direct spatial-temporal mapping of visual images to vibrational patterns seems to be the most natural approach to take, encoding visual information in a way that results in highly discriminable vibrational patterns warrants higher performance levels at very little additional cost on training (Geldard, 1966; Craig, 1973).

In a typical tactual hearing aid using the frequency-to-place transformation model, the acoustic signal of speech is sent through an array of bandpass filters with increasing center frequencies. The outputs of these filters are rectified and used to modulate the amplitudes of a corresponding array of vibrators (Keidel, 1973). Examples of such systems range from the "Felix" system developed by Dr. Nobert Wiener in the early 1950s at the Research Laboratory of Electronics at MIT (Figure 18.5) to the modern day wearable version of Tactaid VII. The Tactaid VII system (Audiological Engineering Corp., Somerville, MA) consists of a small processing unit with an embedded microphone, which can be clipped to a belt or fit into a shirt pocket, and a harness with seven resonant vibrators (Figure 18.6). The harness can be worn on the forearm, the chest, the abdomen or around the neck. When used alone, it can convey useful information such as environmental sounds, but understanding speech is difficult. When used in conjunction with speechreading (i.e., lipreading), Tactaid VII provides a limited improvement to sentence reception accuracy with a typical increase of around 10% (the so-called ceiling effect) (Reed & Delhorne, 1995).

In general, performance levels with artificial tactual speech communication devices do not reach anywhere near that demonstrated by Tadoma users (Reed, Durlach, Delhorne, Rabinowitz, & Grant, 1989). Previous research has documented that these individuals can understand everyday speech at very high levels, allowing rich two-way conversation with both familiar and unfamiliar talkers (Figure 18.7). The information transfer rate for the Tadoma method has been estimated to be roughly 12 bits/sec, which is about half the rate of daily conversation conducted by hearing individuals (Reed, Durlach, & Delhorne, 1992). In contrast, the typical reading rates with the Optacon is about 30–50 wpm, or about 4–6.7 bits/sec in information transfer rate. The conversion from word rate to information transfer rate

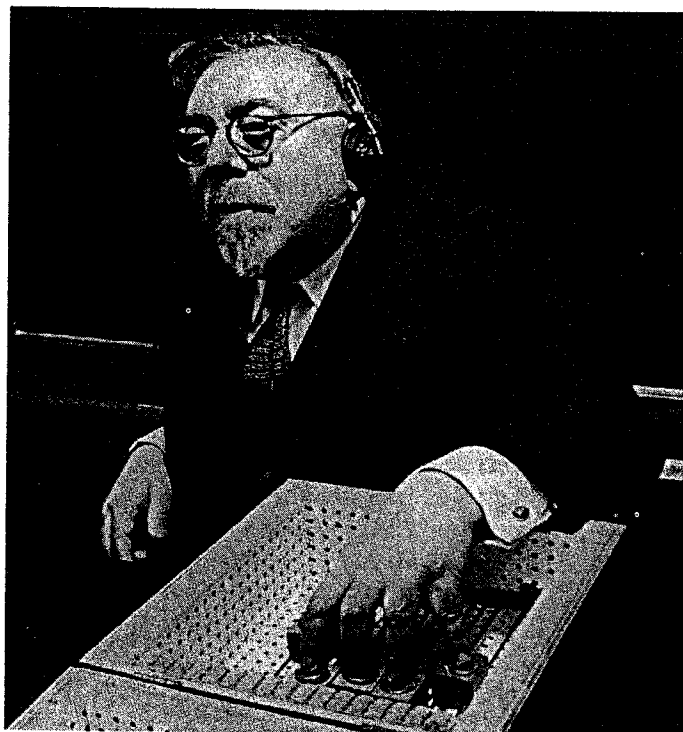


FIG. 18.5. Dr. Norbert Wiener with the "Felix" system. Photograph by Alfred Eisenstaedt, 1950.

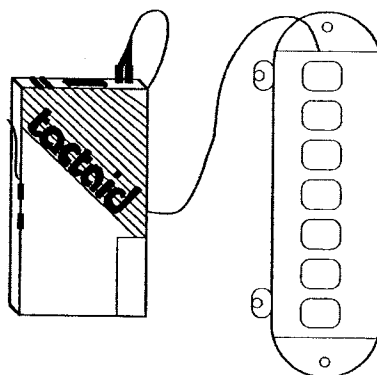


FIG. 18.6. The Tactaid VII with its processor and the seven-vibrator array. Reprinted by permission from Weisenberger & Percy (1994). © 1994 Alexander Graham Bell Association for the Deaf, Inc.



FIG. 18.7. Two experienced Tadoma method users who are deaf and blind (Leonard Dowdy on the left and Raymond Boduch on the right) communicate with each other and Senior Research Scientist Nathaniel I. Durlach (center). Photograph by Hansi Durlach, 1980.

is based on two assumptions. First, according to Shannon (1951, Fig. 4), the uncertainty for strings of eight letters (including the 26 letters of the English alphabet and space) or more has an upper bound of 2 bits/sec. For simplicity, it is assumed that the test material is longer than eight letters. Second, it is assumed that the average word length is 4 letters. It follows that the information content in words is  $2 \text{ bits/letter} \times 4 \text{ letter/word}$ , or equivalently, 8 bits/word. The information rate for 30 wpm is, therefore,  $8 \text{ bits/word} \times 30 \text{ words/minute}$ , or equivalently, 4 bits/sec.

One problem with tactile aids is that they are composed of multiple stimulators that deliver "homogeneous" high-frequency vibrations to the tactual sensory system. In contrast, a talking face for Tadoma is perceptually rich, displaying various stimulation qualities (e.g., mouth opening, airflow, muscle tension around the cheeks, laryngeal vibration, etc.) that engage both the kinesthetic and tactile sensory systems. Recognition of the need

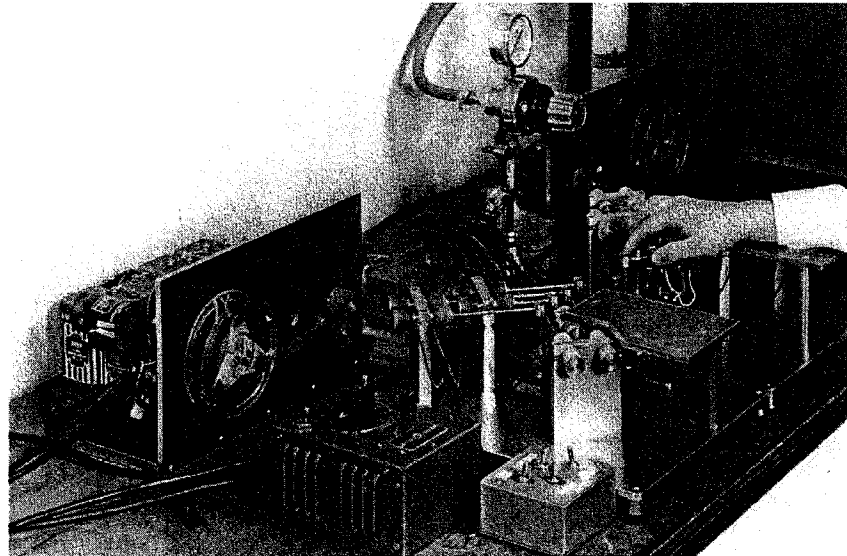


FIG. 18.8. The "reverse-typewriter" system. Photograph by James Bliss.

to develop devices that engage both the tactile and the kinesthetic senses is now prevalent. Examples of such displays are the "reverse-typewriter" system, the "OMAR" system, the MIT Morse code display, and the Tactuator, all of which stimulate the fingers.

The "reverse-typewriter" system was developed by Bliss (1961) prior to his invention of the Optacon. It was a pneumatic display that consisted of eight finger rests arranged in two groups on which the user could place the fingers of both hands in a manner similar to that of the "home" position of a typewriter (Figure 18.8). Each stimulator was capable of generating motions corresponding to the active movements of a typist's fingers in reaching the upper and lower rows on a keyboard. One experienced typist was trained to receive sequences of 30 symbols (the alphabet, comma, period, space, and upper case) and reached a maximum information transfer rate of 4.5 bits/sec (Bliss, 1961).

The "OMAR" system was a two-degree-of-freedom (up-down and front-back) finger stimulator that used motion, vibration, and stiffness cues to encode speech information (Figure 18.9). Initial experiments demonstrated that subjects were able to judge onset asynchronies of vibration and movement with this system (Eberhardt, Bernstein, Barac-Cikoja, Coulter, & Jordan, 1994).

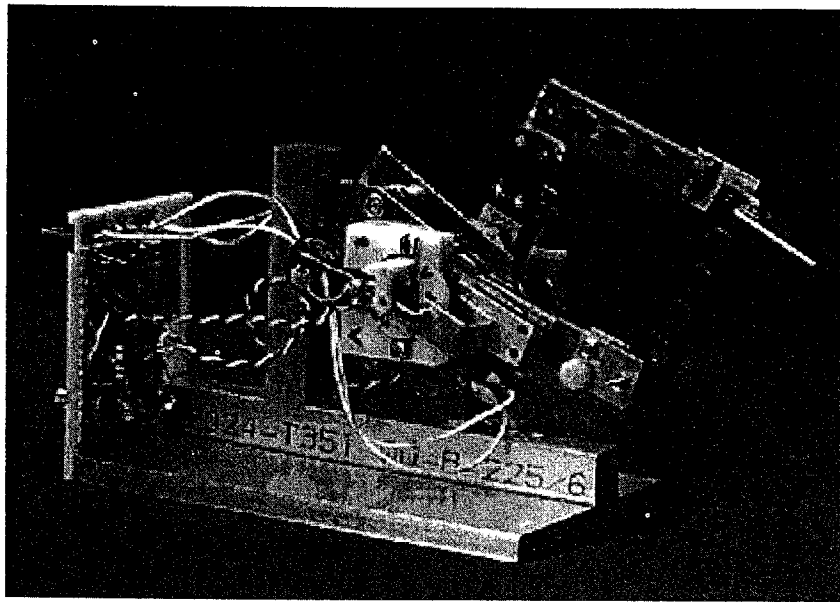


FIG. 18.9. The "OMAR" system. Photograph courtesy of Dr. Lynne E. Bernstein. © 1998 House Ear Institute.

The MIT Morse code display was designed to move the fingertip of one finger up and down in a way that was similar to the motions generated by ham radio operators using the straight keys for sending the code (Figure 18.10). The ability of two experienced Morse code operators to receive the code of everyday English sentences through this device was estimated to be around 20 wpm, or 2.7 bits/sec (Tan, Durlach, Rabinowitz, Reed, & Santos, 1997).

The Tactuator consisted of three independent, point-contact, one-degree-of-freedom actuators interfaced individually with the fingerpads of the thumb, the index finger, and the middle finger (Figure 18.11). Each movement channel is capable of delivering stimuli from absolute detection threshold (i.e., the smallest displacement that can be detected by a human observer) to approximately 50 dB SL (Sensation Level, defined relative to detection threshold) throughout the frequency range from near DC to above 300 Hz, thereby encompassing the perceptual range from gross motion to vibration (Tan & Rabinowitz, 1996). Information transfer rate with the Tactuator was estimated to be about 12 bits/sec, which is roughly the same as that achieved by Tadoma users in tactual speech communication (Tan, Durlach, Reed, & Rabinowitz, 1998). This promising result was mainly

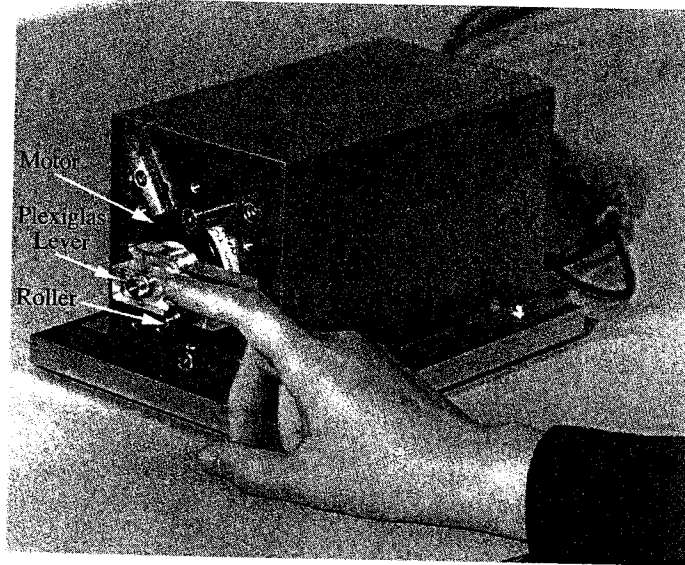


FIG. 18.10. The one-degree-of-freedom (up and down) stimulator used in the MIT Morse code study. From Tan et al. (1997), reprinted by permission of Psychonomic Society, Inc.

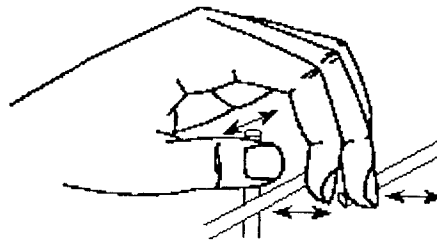


FIG. 18.11. A diagram of the Tactuator. Reprinted by permission of the ASME from Tan & Rabinowitz (1996).

attributed to the relative richness of the Tactuator as a tactual display (i.e., it used features such as finger location, motional and vibrational stimulation, etc. to convey tactual information).

Despite the recent promising results in the research laboratories, however, much work still remains before these experimental apparatus become practical for the daily use by individuals with sensory impairments.

The above review, although brief and incomplete [e.g., we did not discuss electrocutaneous stimulation at all because of its tendency to induce

pain and discomfort (Rollman, 1973)], brings several conclusions that can be drawn from work in the area of sensory substitution. First of all, among the above-reviewed tactual communication systems developed for sensory substitution, the Optacon and the Tactaid VII (including its earlier versions) are the only commercially available, portable aids for the blind and the deaf, respectively. Laboratory apparatus are usually developed with the aim of precise stimulus control, not necessarily the device portability. Second, both the Optacon and the Tactaid VII require intensive and extensive training for their users. Only those individuals with severe sensory impairments are motivated enough to go through the training. Third, performance with tactile aids for the blind and/or the deaf do not match that achieved by people with normal sensory capabilities as far as pictorial and speech communication are concerned. Fourth, there remains much work in developing displays that deliver rich stimulation to the tactual sensory systems, and in devising coding schemes that best match the information content of stimulating signals with the capability of the somatosensory system.

#### 4. CONSIDERATIONS FOR WEARABLE TACTUAL DISPLAYS

The development of tactual interfaces in general requires advancement in two areas. On the one hand, an understanding of the human somatosensory system enables us to associate physical stimulation parameters with well-defined percepts. On the other hand, advances in technologies make it possible for us to design apparatus that can deliver desired stimulation patterns. The development of wearable tactual displays presents additional challenges and opportunities.

The first consideration is the wearability of wearable tactual displays. Desktop-based displays such as the Tactuator and the PHANTOM™ (Massie & Salisbury, 1994) are unlikely candidates for a wearable computer. Portable haptic displays, such as the exoskeleton system developed by EXOS Inc. for NASA astronauts, requires the user to carry the weight of the structure and absorb the excessive force at body sites strapped to the device. One human factor study recommends that such devices be worn for no more than an hour or two due to user fatigue (Tan, Srinivasan, Eberman, & Cheng, 1994). Given the state of current technology, vibrotactile displays are good candidates for wearable tactual displays for their light weight and low power consumption.

The second consideration is the body site to be stimulated by wearable tactual displays. The desire for high spatial resolution should be balanced with the accessibility and size of contact area. For example, the hands (especially the fingertips) are the best candidates for tactual displays in terms of sensory resolution. However, the hands are already engaged in many daily tasks, especially those involving human-computer interactions. The back has poorer spatial resolution, yet it is usually not engaged by any human-computer interfaces and can be easily accessed. Its relatively poor spatial resolution can be compensated for by its relatively large contact area.

The third consideration is the intended users of wearable tactual displays. We believe that with proper design, tactual displays should be useful for all users of wearable computers, whether they are sensory impaired or not. A good example of a well-designed universal adaptive structure is the street curb. Although originally conceived to provide access for individuals on wheelchairs, curbs are used by mothers with baby strollers, students on roller blades, and couriers with dollies. Tactual displays should be developed to enhance the functionality of wearable computers for all users.

The fourth consideration is the amount of training associated with the use of wearable tactual displays. A good wearable tactual display should minimize training by displaying information that is salient, intuitive, and easy to interpret through the sense of touch. For example, a buzzing on one's shoulder can attract immediate attention from the user. Increased pressure on one's back can signal something approaching from behind.

The fifth issue concerns the long-term wearability of tactual displays. For example, it is well known that all sensory systems adapt to the stimulation environment. For example, we cease to notice our clothes after we have put them on for a while although they are in constant contact with the skin surfaces. Static stimulation (such as a constant vibration) tends to "numb" the skin and "fades" after a short while. Dynamic stimuli (such as pulsing) are more likely to evoke the same perceptual intensity time after time.

Finally, a new paradigm is needed to seamlessly integrate tactual interfaces with visual and auditory displays. Given the existing well-developed visual and auditory interfaces, tactual interfaces would be most useful when used to supplement visual and auditory information, or to clarify it when vision or audition is overloaded. An example of such a system would be a tactile vest that redirects a driver's visual attention when a collision-warning alarm goes off.

In summary, a good wearable tactual display should be not only portable but wearable, should not interfere with the user's daily activities, should



be useful for people with all degrees of sensory capabilities, should require minimum amount of training, should be resistant to sensory adaptation, and should be well integrated with existing visual and auditory interfaces.

### 5. A TACTILE DIRECTIONAL DISPLAY

Recently, a wearable tactile directional display has been developed at our research laboratory. It consists of an array of micromotors embedded in the back of a vest that delivers vibrational patterns to the back of the wearer. The display elicits salient and vivid movement sensations that are resistant to sensory adaptation. Informal tests with first-time users indicate that it requires no training and the interpretation of directional signals is highly consistent. Because the directional information is presented relative to the user's body coordinates, it eliminates the need to transform coordinates as is often the case with maps and building layouts. What's more, the perceived spatial resolution of the display can be manipulated by the signals delivered to the individual actuators and is not limited by the physical layout of actuator arrays.

All this is accomplished by taking advantage of a perceptual illusion called sensory saltation (also known affectionately as the "rabbit"), discovered by Dr. Frank Geldard and his colleagues at the Princeton Cutaneous Communication Laboratory in the early 1970s (Geldard, 1975). In a typical setup for eliciting the "rabbit," three mechanical stimulators are placed with equal distance on the forearm (Figure 18.12). Three brief pulses are delivered to the first stimulator closest to the wrist, followed by three more at the middle stimulator, followed by another three at the stimulator near

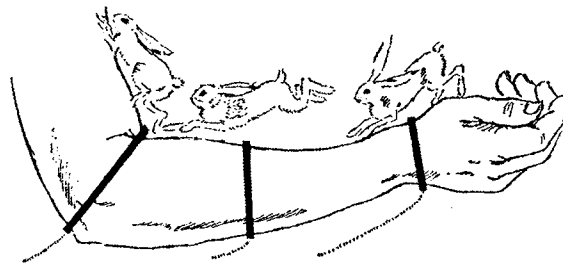


FIG. 18.12. A Norwegian cartoonist's illustration of sensory saltation. From Geldard (1975), reprinted by permission of Lawrence Erlbaum Associates, Inc. Publishers.

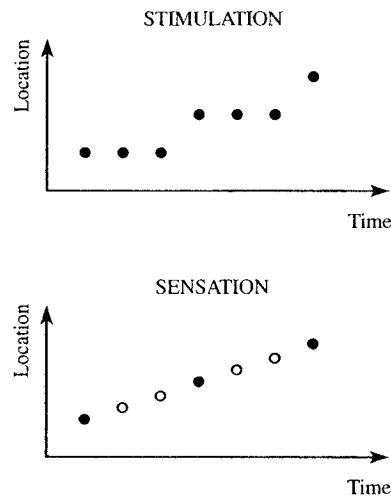


FIG. 18.13. An illustration of sensation vs. stimulation pattern for sensory saltation. Open circles indicate perceived pulses at phantom locations between stimulators. Reprinted by permission from Tan & Pentland (1997). © 1997 IEEE.

the elbow. The pulses are evenly spaced in time. Instead of feeling the successive taps localized at the three stimulator sites, the observer is under the impression that the pulses are distributed with more or less uniform spacing from the wrist to the elbow (as illustrated in Figure 18.13). The saltation effect can be elicited with mechanical, electrocutaneous, or thermal stimulation. The sensation is discontinuous and discrete as if a tiny rabbit was hopping up the arm from the wrist to the elbow, leading to the nickname "cutaneous rabbit." The same vivid hopping sensation can also be elicited in vision and audition (Geldard, 1975).

An important feature of the cutaneous rabbit is its ability to simulate higher spatial resolution than the actual spacing of stimulators, yet mimic the sensation produced by a veridical set of stimulators with the same higher-density spacing (Cholewiak & Collins, 1995; Collins, 1996; Cholewiak, Sherrick, & Collins, 1996). The perceived spacing of adjacent taps is inversely proportional to the number of pulses sent to each stimulator. In theory, only two stimulators are needed in order to produce the sensory saltation effect. Additional stimulators add redundancy and robustness to the overall setup. Thus a sparse stimulator array can be effectively used to produce a dense perception. Another important feature of the sensory saltation phenomenon is that the sensation remains vivid and

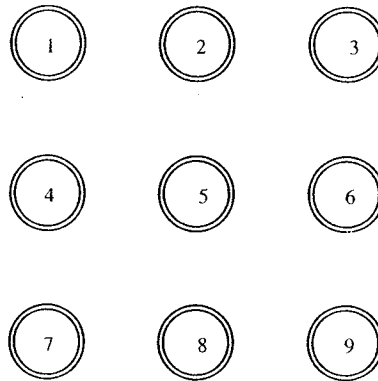


FIG. 18.14. Layout of the 3-by-3 two-dimensional "rabbit" display.  
Reprinted by permission from Tan & Pentland (1997). ©1997 IEEE.

does not fade away after repeated stimulation due to its discontinuity and discreteness.

Our wearable rabbit displays are implemented on the back of vests. An initial protocol consisted of 9 actuators arranged in a 3-by-3 array that measured 18 cm by 18 cm (Figure 18.14). The final version consisted of 16 actuators arranged in a 4-by-4 array covering an area of 18 cm by 18 cm. The main reasons for using a 4-by-4 array are to avoid direct stimulation to the spine area and to provide some additional redundancy. Each actuator is based on a micromotor with a biased load (MicroMo, Clearwater, Florida) that weighs 3.2 grams. The micromotor is mounted inside a square plastic tubing. The tubing is then attached to the garment with elastic bands. The state of the actuator (on or off) and its timing can be controlled by the parallel port of a personal computer through additional circuitry.

To test the effectiveness of our rabbit display, observers who are not aware of the sensory saltation phenomenon have been asked to wear the vests and report on the sensations associated with several stimulation patterns. One typical pattern consisted of three pulses sent to stimulator #8 (see Figure 18.14), followed by three pulses sent to #5, and followed by another three sent to #2. Most observers commented on a sensation of something "hopping" or "crawling" up the spine. When asked how many pulses were felt, most gave an answer between 6 and 8, thus indicating a perception of pulses in between stimulator locations. This is quite consistent with the classical definition of the sensory saltation phenomenon. In a two-dimensional pattern, three pulses of each were sent sequentially to stimulators in the order of #1, #2, #3, #6, #9, #8, #7, and #4. Instead of

feeling a square, most observers reported that they felt a circular pattern. This was an interesting finding and its interpretation awaits further investigation of the sensory saltation phenomenon, especially in the case of a two-dimensional stimulator array.

In general, novice observers find it intuitive to perceive the directional information indicated by these patterns, and their interpretations of the signals are highly consistent. The signals can elicit vivid movement sensations (up, down, left, right, along a line of  $45^\circ$  or  $-45^\circ$  incline). Because the directional cues are relative to the user's own body coordinate, no additional coordinate transformations are necessary. The fact that a circular pattern can be perceived suggests that other patterns might be "drawn" as well.

In an implementation of a navigational guidance system, the rabbit vest has been integrated with a driving simulation software system to provide directional cues to a driver. The overall system consists of an SGI, a PC, the vest, and input hardware that simulated the steering wheel, acceleration pedal, and brake pedal. The SGI is dedicated to the traffic simulation software SIRCA (Nissan Cambridge Basic Research, Cambridge, Massachusetts). The PC is used to interface with the hardware. The driver controls the steering wheel and acceleration and brake pedals. The positions of these input devices, read by the game port of the PC and transmitted to the SGI through a serial link, are used to control a virtual car in SIRCA simulation. Depending on the location of the virtual car, surrounding traffic condition, road configuration, and the driver's intended destination, SIRCA generates one of the following directional commands: turn left, turn right, or go straight at the next intersection. This command, sent to the PC through a serial connection, triggers a preprogrammed signal pattern at the parallel port of the PC that controls the micromotors in the rabbit vest. The driver wearing the vest feels a left, right, or upward arrow on the back. Such a system demonstrates the feasibility and effectiveness of wearable tactile displays for navigation guidance.

## 6. SUMMARY

This chapter has presented a general review of tactual displays for sensory substitution and a description of a tactile directional display designed for wearable computing. The two-dimensional "rabbit" display has many features that make it an attractive candidate for displaying directional information in applications such as navigation guidance. One implementation has already demonstrated the usefulness of simple directional signals for

providing directional cues to a driver. The possibilities of displaying more complicated patterns and using them to encode useful information have yet to be explored.

As wearable computing becomes more ubiquitous and distributed, a new generation of interfaces for wearable computers is emerging. In the future, smart clothing and furniture will become part of human-computer interfaces through contact sensing and display. By presenting our current efforts toward that goal, we hope to stimulate similar work on new and novel haptic displays that can work in concert with wearable visual and auditory displays.

**Acknowledgments** This work has been supported in part by British Telecom and in part by the Things That Think Consortium at the MIT Media Lab. Part of this work has been presented at the First International Symposium on Wearable Computers.

## REFERENCES

- Bach-y-Rita, P. (1972). *Brain Mechanisms in Sensory Substitution*. New York: Academic Press.
- Bliss, J. C. (1961). *Communication via the kinesthetic and tactile senses*. Ph.D. Dissertation, Dept. of Electrical Engineering, M.I.T.
- Burdea, G. C. (1996). *Force and Touch Feedback for Virtual Reality*. New York: John Wiley & Sons, Inc.
- Cholewiak, R. W., & Collins, A. A. (1995). Exploring the conditions that generate a good vibrotactile line. Presented at the Psychonomic Society Meeting, Los Angeles, CA.
- Cholewiak, R. W., Sherrick, C. E., & Collins, A. A. (1996). Studies of saltation. *Princeton Cutaneous Research Project (No. 62)*. Princeton University, Department of Psychology.
- Clark, F. J., & Horch, K. W. (1986). Kinesthesia. In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook of Perception and Human Performance: Sensory Processes and Perception* (pp. 13/1-13/62). New York: Wiley.
- Collins, A. A. (1996). Presentation at the Tactile Research Group Meeting of the Psychonomic Society Meetings.
- Craig, J. C. (1973). Pictorial and abstract cutaneous displays. In F. A. Geldard (Ed.), *Cutaneous Communication Systems and Devices* (pp. 78-83). The Psychonomic Society, Inc.
- Craig, J. C. (1977). Vibrotactile pattern perception: Extraordinary observers. *Science*, **196**, 450-452.
- Craig, J. C., & Sherrick, C. E. (1982). Dynamic tactile displays. In W. Schiff & E. Foulke (Eds.), *Tactual Perception: A Sourcebook*. Cambridge University Press.
- Eberhardt, S. P., Bernstein, L. E., Barac-Cikoja, D., Coulter, D. C., & Jordan, J. (1994). Inducing dynamic haptic perception by the hand: system description and some results. In *Proceedings of the American Society of Mechanical Engineers, DSC-1*, 345-351.
- Geldard, F. A. (1966). Cutaneous coding of optical signals: The optohapt. *Perception & Psychophysics*, **1**, 377-381.
- Geldard, F. A. (1975). *Sensory Saltation: Metastability in the Perceptual World*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

- Keidel, W. D. (1973). The cochlear model in skin stimulation. In F. A. Geldard (Ed.), *Cutaneous Communication Systems and Devices* (pp. 27-32). The Psychonomic Society, Inc.
- Linville, J. G., & Bliss, J. C. (1966). A direct translation reading aid for the blind. *Proceedings of the Institute of Electrical and Electronics Engineers*, **54**, 40-51.
- Loomis, J. M., & Lederman, S. J. (1986). Tactual perception. In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook of Perception and Human Performance: Cognitive Processes and Performance* (pp. 31/1-31/41). New York: Wiley.
- Massie, T. H., & Salisbury, J. K. (1994). The PHANTOM haptic interface: A device for probing virtual objects. In *Proceedings of the American Society of Mechanical Engineers*, **DSC-55(1)**, 295-299.
- Reed, C. M., & Delhorne, L. A. (1995). Current results of field study of adult users of tactile aids. *Seminars in Hearing*, **16(4)**, 305-315.
- Reed, C. M., Durlach, N. I., & Delhorne, L. A. (1992). The tactual reception of speech, fingerspelling, and sign language by the deaf-blind. *Digest of Technical Papers of the Society for Information Display International Symposium*, **XXIII**, 102-105.
- Reed, C. M., Durlach, N. I., Delhorne, L. A., Rabinowitz, W. M., & Grant, K. W. (1989). Research on tactual communication of speech: Ideas, issues, and findings. *The Volta Review*, **91**, 65-78.
- Reed, C. M., Rabinowitz, W. M., Durlach, N. I., Braida, L. D., Conway-Fithian, S., & Schultz, M. C. (1985). Research on the Tadoma method of speech communication. *Journal of the Acoustical Society of America*, **77(1)**, 247-257.
- Rollman, G. B. (1973). Electrocutaneous stimulation. In F. A. Geldard (Ed.), *Cutaneous Communication Systems and Devices* (pp. 38-51). The Psychonomic Society, Inc.
- Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal*, **30**, 50-64.
- Tan, H. Z., Durlach, N. I., Rabinowitz, W. M., Reed, C. M., & Santos, J. R. (1997). Reception of Morse code through motional, vibrotactile, and auditory stimulation. *Perception & Psychophysics*, **59(7)**, 1004-1017.
- Tan, H. Z., Durlach, N. I., Reed, C. M., & Rabinowitz, W. M. (1999). Information transmission with a multi-finger tactual display. *Perception & Psychophysics*, **61(6)**, 993-1008.
- Tan, H. Z., & Pentland, A. (1997). Tactual displays for wearable computing. *Digest of the First International Symposium on Wearable Computers*, 84-89. IEEE Computer Society.
- Tan, H. Z., & Rabinowitz, W. M. (1996). A new multi-finger tactual display. In *Proceedings of the American Society of Mechanical Engineers*, **DSC-58**, 515-522.
- Tan, H. Z., Srinivasan, M. A., Eberman, B., & Cheng, B. (1994). Human factors for the design of force-reflecting haptic interfaces. In *Proceedings of the American Society of Mechanical Engineers*, **DSC-55(1)**, 353-359.
- Weisenberger, J. M., & Percy, M. E. (1994). Use of the Tactaid II and Tactaid VII with children. *The Volta Review*, **96(5)**, 41-57.
- White, B. W. (1973). What other senses can tell us about cutaneous communication. In F. A. Geldard (Ed.), *Cutaneous Communication Systems and Devices* (pp. 15-19). The Psychonomic Society, Inc.
- White, B. W., Saunders, F. A., Scadden, L., Bach-Y-Rita, P., & Collins, C. C. (1970). Seeing with the skin. *Perception & Psychophysics*, **7(1)**, 23-27.

# Toward Realistic Haptic Rendering of Surface Textures

Seungmoon Choi and Hong Z. Tan  
Purdue University

**T**he emerging science of haptic rendering consists of delivering properties of physical objects through the sense of touch. Owing to the recent development of sophisticated haptic-rendering algorithms, users can now experience virtual objects through touch in many exciting applications, including surgical simulations, virtual prototyping, and data perceptualization. Haptics holds great promise to enrich the sensory attributes of virtual objects that these systems can produce.

**New sophisticated haptic-rendering algorithms let users experience virtual objects through touch. The authors systematically investigate the unrealistic behavior of virtual haptic textures.**

One area that has received increasing attention in the haptics community is haptic texture rendering, the goal of which is to introduce micro-geometry-scale features on object surfaces. Haptic objects rendered without textures usually feel smooth, and sometimes slippery. Appropriate haptic textures superimposed on haptic objects enhance an object's realism. For example, we can make the same cubic structure feel like a brick with rough surface textures or a cardboard carton with finer textures. Clearly, haptic texture rendering is an exciting research field that can take haptic rendering to the next level.

Although much effort has been devoted to haptic texture rendering—mostly in modeling and rendering techniques<sup>1,2</sup>—the research community must overcome many challenges before haptic texture rendering can be widely used in real applications. One common problem in haptically rendered textures is that they are sometimes perceived to behave unrealistically, for example, by buzzing or by the apparent aliveness of a textured surface. Due to the complex nature of the haptic-rendering pipeline and the human somatosensory system, it remains a difficult problem to expose all factors contributing to such perceptual artifacts.

At the Haptic Interface Research Laboratory at Purdue University, we are among the first to have systematically investigated the unrealistic behavior of virtual

haptic textures. This article presents a summary of our recent work in this area. We hope this article will stimulate further discussion among haptics researchers and applications developers who are interested in haptic texture rendering. Interested readers may refer to our previous publications for more details.<sup>3-7</sup>

## Perceived instability

We use the term *perceived instability* to refer to all unrealistic sensations—such as buzzing—that cannot be attributed to the physical properties of a textured surface being rendered with a force-feedback device. To develop haptic texture-rendering models and methods that can deliver realistic textures to human users, you must understand the conditions under which textured virtual objects are free of perceptual artifacts, and you must also recognize the sources of perceived instabilities. We developed the notion of perceived instability to include the effects of all factors in haptic interaction that can result in unrealistic sensations. As shown in Figure 1, haptic interaction occurs at a haptic interaction tool that mechanically connects two symmetric dynamic systems. In principle, each block in the diagram can contribute to the perception of instability by the human user.

A crucial difference between real and virtual haptic interactions with an object is that a virtual environment imparts no haptic sensation to the user unless the interaction tool penetrates the object surface. The first phase of haptic texture rendering is the computation of the penetration depth and the resulting force command using a haptic texture renderer stored in the computer. For this purpose, most haptic systems repeat several procedures at a high update rate, usually 1 KHz or higher. First, the system measures the position of the haptic interaction tool using position sensors embedded in the haptic interface. The system then compares the measured position of the interaction tool with the location of objects in the virtual environment. If the interaction tool penetrates the surface of any virtual object, a response force is computed and sent to the haptic interface to create the intended haptic effect. Finally, if the state of any virtual object has changed due to the interaction, the system updates the

database of virtual objects.

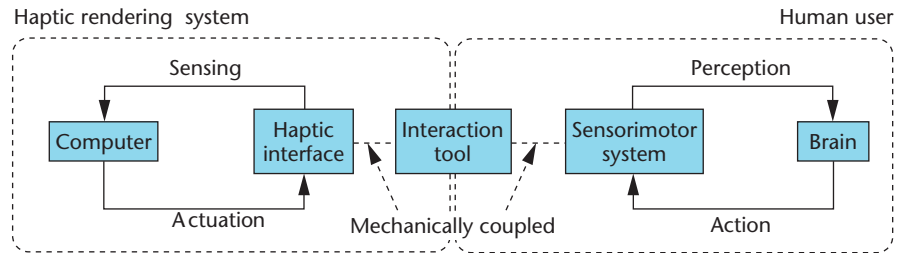
Two of these four steps—collision detection and response force computation—can have a significant effect on perceived instability. These two steps determine the so-called environment dynamics, the reaction dynamics of the haptic renderer to a user input. In most cases, the environment dynamics is an approximation of the corresponding real-world contact dynamics because simulating the actual physics is usually too complex to accomplish in real time. The simplified environment dynamics must preserve the essence of the real contact dynamics to produce sensations consistent with a user's experience and expectation. Otherwise, the user perceives unrealistic behaviors of haptically rendered objects, and perceived instability occurs. This issue has received little attention from the research community because a majority of studies on haptic texture rendering have focused on the development of time-efficient rendering algorithms.

The next phase of haptic texture rendering is the delivery of force to the human user. During this process, the force-feedback device must remain stable to avoid perceived instability. Device instability, such as mechanical resonance, can result in force variations in addition to the force command received from the haptic texture renderer. In our experiences, a user usually judges the haptically rendered textured surface as unstable when an extraneous signal—such as high-frequency buzzing—occurs from the haptic interface. This issue has received much attention in the context of control engineering, although most studies assume a much simpler virtual environment such as a flat wall without any textures.<sup>8</sup> Much work is needed to extend the techniques for solving the hard-wall stability problem.

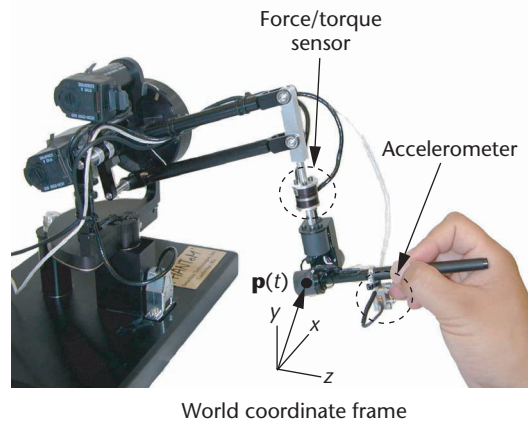
The last phase of haptic texture rendering is the perception of force by a human user. The human user senses the mechanical stimuli from the haptic interface, extracts information from force variations, forms a percept of the virtual object being rendered, and determines whether the virtual object is realistic. To determine whether the user perceives instability from a textured surface rendered by the haptic interface, we must resort to psychophysical studies. Psychophysics is a branch of psychology with well-developed methodology for studying the relation between geometrical and physical properties of objects and the percept. At this point, little knowledge is available in the literature on the perceived instability of haptically rendered objects, because this is a new research topic that has only become relevant with the recent capability to render haptic virtual environments.

## Goals and approaches

Our long-term goal is to develop haptic texture-rendering systems that deliver realistic sensations of virtual haptic textures. To do so requires the appropriate design of a texture renderer, the stable control of a haptic interface, and a better understanding of the somatosensory system. As a first step, our research has



**1** Illustration of the key components of the interaction between a haptic rendering system and a human user.



**2** Phantom 1.0A used in our studies. This is a modified Phantom instrumented with additional force-torque and acceleration sensors.

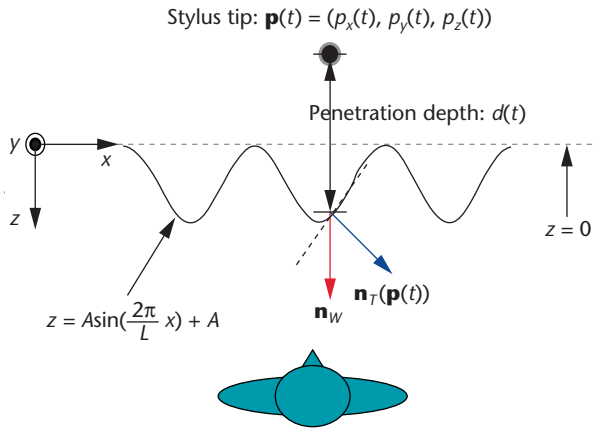
focused on understanding the nature of perceived instability. Specifically, we

- investigated the conditions under which perceived instability of virtual textures occurs,
- discovered the types of perceived instability frequently reported by human users,
- identified the proximal stimuli that contributed to the perception of instability, and
- unveiled the sources that produced the stimuli.

We conducted psychophysical experiments to quantify the conditions under which users perceived instability from virtual textures and to understand the associated percept. We also measured the physical stimuli delivered by the haptic interaction tool to the user's hand under various conditions where the textures were perceived to be stable and unstable. By analyzing the measured data, we located signal components responsible for the perception of instability. We achieved the last goal by investigating which component in the haptic texture-rendering system generated the signals that led to perceived instability.

Because of the plethora of texture-rendering models and methods, we chose a benchmark consisting of the most essential features common to many texture-rendering systems for studying perceived instability. In addition, we had to consider the effect of user exploration patterns because the user is mechanically coupled to the haptic interface. For the apparatus, we used a Phantom force-reflecting device (from SensAble Technologies) shown in Figure 2. This is the most widely used device for haptics research and applications.





**3 Illustration of the texture rendering models and methods used in our studies.**

For the texture model, we used a 1D sinusoidal grating superimposed on a flat surface (see Figure 3). The grating is represented by

$$z = A \sin\left(\frac{2\pi}{L}x\right) + A$$

in the Phantom world coordinate frame where  $A$  and  $L$  are the amplitude and wavelength of the grating. Sinusoidal gratings have been widely used as basic building blocks for textured surfaces in studies on haptic texture perception and as a basis function set for modeling real haptic textures.

For collision detection, we used two methods of computing penetration depth  $d(t)$ :

$$d_1(t) = \begin{cases} 0 & \text{if } p_z(t) > 0 \\ A \sin\left(\frac{2\pi}{L}p_x(t)\right) + A - p_z(t) & \text{if } p_z(t) \leq 0 \end{cases}$$

and

$$d_2(t) = \begin{cases} 0 & \text{if } p_z(t) > h(p_x(t)) \\ A \sin\left(\frac{2\pi}{L}p_x(t)\right) + A - p_z(t) & \text{if } p_z(t) \leq h(p_x(t)) \end{cases}$$

where  $\mathbf{p}(t) = (p_x(t), p_y(t), p_z(t))$  was the position of the Phantom stylus tip and  $h(p_x(t)) = A \sin(2\pi/L p_x(t)) + A$  was the height of the textured surface at  $p_x(t)$ . The first method,  $d_1(t)$ , assumed that collision detection was based on the plane underlying the textured surface ( $z = 0$ ). The advantage of  $d_1(t)$  was that we can easily generalize it to textured objects with a large number of underlying polygons because the plane could represent a face on a polygon. The disadvantage was that it introduced discontinuity in computed penetration depth (and subsequently in response force) when the Phantom stylus entered and left the textured surfaces.

The second method  $d_2(t)$  declared a collision as soon as the stylus entered the texture boundary. The advantage of this method was that it ensured a continuous

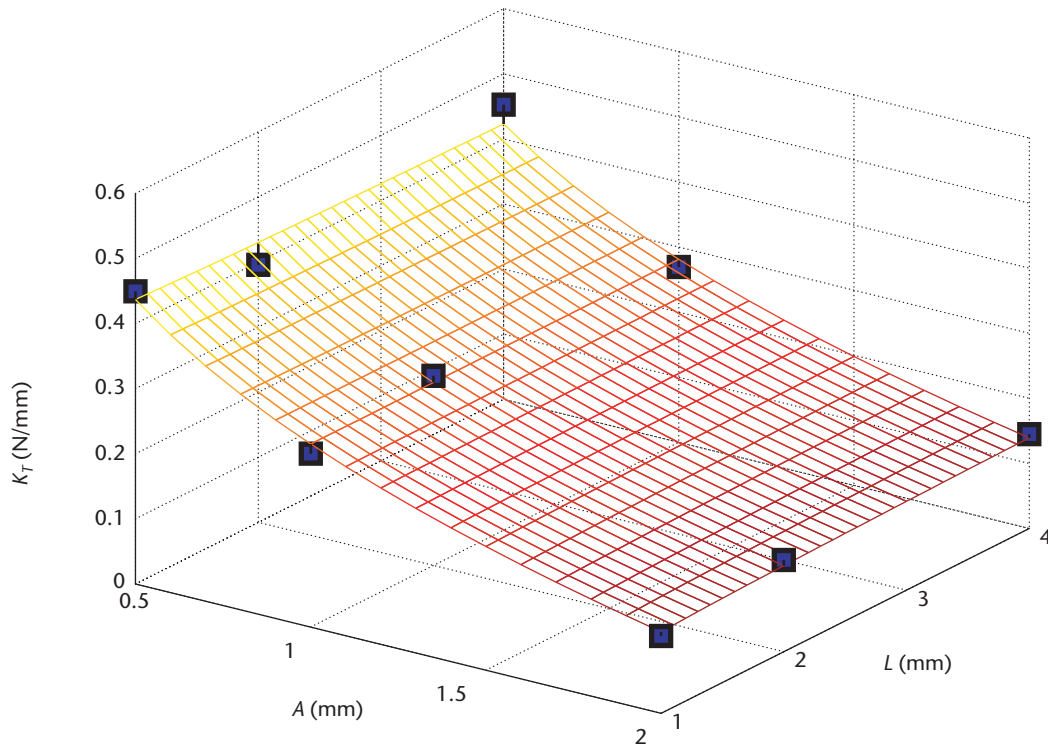
change in computed penetration depth and response force. The disadvantage was that it's much more difficult to apply this algorithm to textured polygonal objects for two reasons. One is the nonlinearity associated with the representation of textured object surfaces that usually requires iterative numerical algorithms for collision detection. The other is the lack of a global representation of the boundaries of the textured virtual objects. In a typical implementation, the polygons and the texture model are stored separately, and the texture model is locally mapped onto a point on the polygon whenever necessary. It's often infeasible to do a global collision detection using only the local information. To the best of our knowledge, the application of the collision detection method based on  $d_2(t)$  to a general class of textured objects is still an open research issue.

We employed two basic texture-rendering methods. Both used a spring model to calculate the magnitude of rendered force, but they differed in the way they rendered force directions. Force magnitudes were calculated as  $K \cdot d(t)$ , where  $K$  was the stiffness of the textured surface and  $d(t)$  was the penetration depth of the stylus at time  $t$  (see Figure 3). In terms of force directions, the first method rendered a force  $\mathbf{F}_{mag}(t)$  with a constant direction normal to the flat wall underlying the textured surface. The second method rendered a force  $\mathbf{F}_{vec}(t)$  with varying direction such that it remained normal to the local microgeometry of the sinusoidal texture model. Mathematically,  $\mathbf{F}_{mag}(t) = Kd(t)\mathbf{n}_w$ , and  $\mathbf{F}_{vec}(t) = Kd(t)\mathbf{n}_T(\mathbf{p}(t))$ , where  $\mathbf{n}_w$  was the normal vector of the underlying flat wall, and  $\mathbf{n}_T(\mathbf{p}(t))$  was the normal vector of the textured surface at  $\mathbf{p}(t)$ . Both methods kept the force vectors in horizontal planes, thereby minimizing the effect of gravity on rendered forces.

The two texture-rendering methods are natural extensions of virtual wall rendering techniques. Perceptually, they are very different: Textures rendered by  $\mathbf{F}_{vec}(t)$  feel rougher than those rendered by  $\mathbf{F}_{mag}(t)$  for the same texture model. Textures rendered by  $\mathbf{F}_{vec}(t)$  also feel sticky sometimes.

Our exploration mode refers to a stereotypical pattern of the motions that a user employs to perceive a certain attribute of objects through haptic interaction. We tested two exploration modes—free exploration and stroking. In the free exploration mode, users could determine and use the interaction pattern that was most effective at discovering the instability of the rendered textures. We selected this mode as the most challenging interaction pattern for a haptic texture rendering system in terms of perceived stability. With the stroking mode, users should move the stylus laterally across the textured surfaces. We chose this mode to be representative of the typical and preferred exploration pattern for accurate texture perception.<sup>9</sup>

We conducted psychophysical experiments to quantify the parameter space within which textures were perceived as stable and to categorize the types of perceived instability discovered by users. We employed the method of limits, a well-established classical psychophysical method, in all our experiments.<sup>10</sup> And we employed a diverse range of experimental conditions, with factors including texture model parameter (amplitude and wave-



4 Example of psychophysical results.

length of the 1D sinusoidal gratings), texture rendering method, exploration mode, and collision-detection method. In each experimental condition, a subject's task was to explore a virtual textured plane rendered with Phantom and to decide whether the textured plane exhibited any perceived instability. The dependent variable measured in the experiments was the maximum stiffness  $K_T$  under which the rendered textured plane did not contain any perceived instability. A more detailed description of experiment design can be found elsewhere.<sup>3,4,6</sup>

We measured physical stimuli to isolate the signals responsible for the perception of instability and to identify their sources. For this purpose, we added two more sensors—6D force-torque sensor and 3D accelerometer—to the Phantom, as shown in Figure 2, and measured the nine associated physical variables—3D position, 3D force, and 3D acceleration—that were delivered to a user's hand. We collected data for many experimental conditions on the basis of the parameter space obtained from the psychophysical experiments. By comparing the measured data of both perceptually stable and unstable cases in the time and frequency domains, we isolated the physical stimuli that induced the perception of instability. We also investigated the sources for these signals using additional hypothesis-driven experiments.

## Parameter spaces

Figure 4 shows an example of a parameter space for perceptually stable haptic texture rendering based on data obtained from the psychophysical experiments. We measured the data when the subject stroked virtual textures rendered with  $d_1(t)$  (collision detection based on the plane underlying the textured surface) and  $F_{vec}(t)$  (variable force directions). In the figure,  $A$  and  $L$  represent the amplitude and wavelength of the sinusoidal texture model, respectively, and  $K_T$  denotes the maximum stiffness value under which a virtual texture felt stable. The blue rectangles in the figure represent the stiffness thresholds averaged over three subjects for the corresponding texture model parameters. Also shown is a best-fit surface to the measured data found by regression analysis. The region under the mesh surface represents the parameter space of  $(A, L, K)$  for perceptually stable haptic texture rendering, and the region above the mesh surface contains parameters that result in virtual textures that were perceived as unstable.

The most significant result of the psychophysical experiments was that the parameter spaces for perceptually stable texture rendering were limited. See Table 1 for a summary. Under most experimental conditions, the virtual textures that could be rendered without any

Table 1. Average stiffness thresholds for perceptually stable texture rendering.

Experiments	$d_1(t)$		$d_2(t)$	
	Range (N/mm)	Mean (N/mm)	Range (N/mm)	Mean (N/mm)
$F_{mag}(t)$ , free exploration	0.0586 – 0.1023	0.0799	0.1813 – 0.5383	0.3486
$F_{mag}(t)$ , stroking	0.4488 – 0.1664	0.3116	0.2490 – 0.6410	0.3603
$F_{vec}(t)$ , free exploration	0.0097 – 0.0367	0.0209	0.0181 – 0.0260	0.0235
$F_{vec}(t)$ , stroking	0.0718 – 0.3292	0.1848	0.3254 – 0.4638	0.3808

perceived instability felt soft—similar to the feel of cor-duroy. Textures rendered with higher stiffness values usually contained unrealistic sensations, such as buzzing and aliveness. For the haptic texture rendering system used in our experiments to be useful for generating a large range of textures, we need to enlarge the parameter spaces for perceptually stable rendering.

We examined the effects of experiment factors—texture model parameters, collision detection method, texture rendering method, and exploration mode—by applying statistical analysis on the psychophysical results. In general, stiffness thresholds tended to increase when the amplitude of the sinusoidal texture model decreased or when the wavelength increased. Collision detection method  $d_2(t)$  resulted in larger stiffness thresholds than  $d_1(t)$ , except for experiments using  $\mathbf{F}_{vec}(t)$  and free exploration where the thresholds were too small to exhibit any trends (see Table 1). In most cases, textures rendered with  $\mathbf{F}_{mag}(t)$  (constant force direction) showed larger stiffness thresholds than those with  $\mathbf{F}_{vec}(t)$  (variable force direction). On average, textures explored by stroking resulted in larger stiffness thresholds than those by free exploration.

To gain insight into the effects of texture model parameters on perceived instability, we consider the derivative of force magnitude. Let  $g(t) = |\mathbf{F}_{mag}(t)| = |\mathbf{F}_{vec}(t)|$  denote force magnitude, and assume that the stylus is in contact with the textured surface. From there we have

$$g(t) = K \left[ A \sin \left( \frac{2\pi}{L} p_x(t) \right) + A - p_z(t) \right]$$

Differentiating  $g(t)$  with respect to the time variable  $t$  results in

$$\dot{g}(t) = 2\pi \frac{KA}{L} \cos \left( \frac{2\pi}{L} p_x(t) \right) \dot{p}_x(t) - K\dot{p}_z(t) \quad (1)$$

There are two terms in Equation 1 that determine the rate of change of force magnitude. The term on the right,  $K\dot{p}_z(t)$ , responds to stylus velocity in the normal direction to the underlying plane  $\dot{p}_z(t)$  with a gain of  $K$ . The term on the left is due to the virtual textures. Here, the lateral velocity of the stylus  $\dot{p}_x(t)$  is amplified with three constant gains ( $K$ ,  $A$ , and  $1/L$ ) and one variable gain that depends on the lateral position of the stylus  $p_x(t)$ . Increasing  $A$  or decreasing  $L$  results in a faster change in force magnitude which can cause a textured surface to be perceived as less stable, or equivalently, result in a smaller stiffness threshold  $K_T$ .

We expected that  $d_2(t)$  would generate perceptually more stable textures than  $d_1(t)$  because it removed discontinuities in force commands at the texture entry points. This expectation was confirmed, except for the condition where the subjects freely explored the virtual haptic textures rendered with  $\mathbf{F}_{vec}(t)$ . The stiffness thresholds measured using  $\mathbf{F}_{vec}(t)$  and free exploration were practically zero for both  $d_1(t)$  and  $d_2(t)$ , and hence did not exhibit any significant trend. The reason that the textures felt unstable was the presence of strong buzzing noises whenever we positioned the Phantom stylus deep

inside the textured surfaces.

Our finding that textures rendered with  $\mathbf{F}_{mag}(t)$  resulted in larger stiffness thresholds than those rendered with  $\mathbf{F}_{vec}(t)$  was also consistent with the nature of these two rendering methods. While  $\mathbf{F}_{mag}(t)$  imposed perturbations in the force magnitude only,  $\mathbf{F}_{vec}(t)$  resulted in perturbations in the force direction as well as force magnitude. The sometimes abrupt changes in force direction could cause virtual textures rendered with  $\mathbf{F}_{vec}(t)$  to be perceived as less stable than those rendered with  $\mathbf{F}_{mag}(t)$ . Perceptually,  $\mathbf{F}_{vec}(t)$  is a useful rendering method because it can produce textures that feel much rougher than those rendered with  $\mathbf{F}_{mag}(t)$  using the same texture model.

We expected the experimentally confirmed fact that stroking would result in a larger stiffness threshold than free exploration for the same rendering parameters. Our subjects rarely used stroking in the free exploration mode although it was allowed. Instead, they chose to position the stylus at various locations on or inside the virtual textured surface to focus on the detection of perceived instability. Therefore, in the free exploration mode, the subjects concentrated on the detection of unrealistic vibrations in the absence of any other signals. In the stroking mode, the subjects always felt the vibrations due to the stylus stroking the virtual textured surface. They had to detect additional noise to declare the textured surface as unstable. Due to possible masking of the different vibrations coming from the textured surface, it's conceivable that subjects could not detect instability with stroking as easily as they would with static positioning of the stylus. Indeed, our subjects reported that the experiments with stroking were more difficult to perform.

### Frequently observed perceived instabilities

We found three types of frequently reported perceived instability in the psychophysical experiments: buzzing, aliveness, and ridge instability. The first two relate to the perception of force magnitude, while the other relates to the perception of force direction. Buzzing refers to high-frequency vibrations that subjects felt at the Phantom stylus when it touched virtual textured surfaces. We observed this type of perceived instability in most experimental conditions, particularly when the stiffness values were much higher than the thresholds measured in the psychophysical experiments. The subjects reported that buzzing appears to be of higher frequencies than the vibrations induced by the stroking of virtual textures.

Measurement data supported the anecdotal report. Whenever the subjects felt buzzing, spectral components in a high-frequency region (roughly 150 to 250 Hz) appeared in the power spectral densities of the vibrations transmitted through the stylus. An example is shown in Figure 5. In this figure, the horizontal axis represents frequency from 10 to 500 Hz, while the vertical axis shows the power spectrum density of  $p_z(t)$  (the measured stylus position along the normal direction of the textured plane) in dB relative to 1-micrometer peak sinusoidal motion. We can observe a spectral peak at around 71 Hz. Additional prominent spectral peaks appear in the high

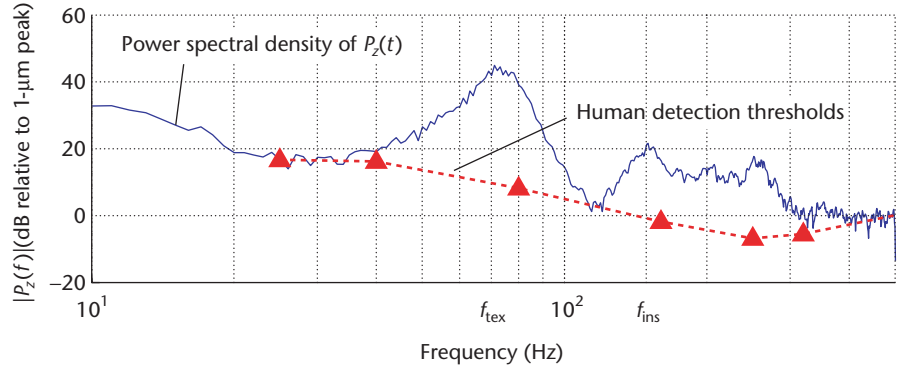
frequency region starting at around 150 Hz. The intensities of the high-frequency vibrations are up to 25 dB above the human detection thresholds at the corresponding frequency (the red dotted line). These high-frequency spectral peaks caused the buzzing.

We suspected that the rapidly changing force commands for texture rendering might have excited the high-frequency dynamics of the Phantom, thereby causing high-frequency vibration to be transmitted through the stylus. We therefore measured the frequency response of the Phantom near the origin of its world coordinate frame and found that the Phantom indeed exhibited a mechanical resonance at 218 Hz. This resonance was likely the source of the high-frequency spectral peaks that invoked the perception of buzzing.

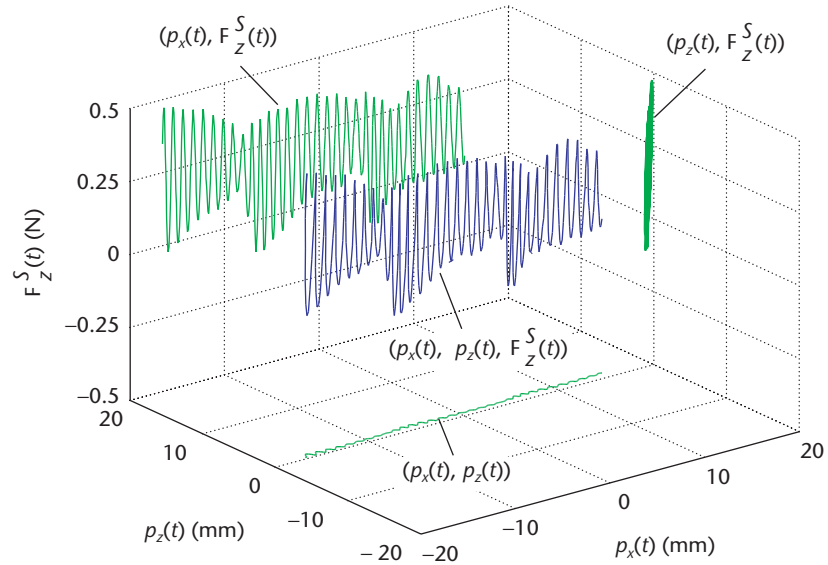
The second type of instability that the subjects frequently observed was aliveness. It occurred when the Phantom stylus was apparently held still yet the subject felt pulsating force changes emanating from the textured surface. The sensation appeared to be at a lower frequency than that of buzzing. Aliveness was reported for textures rendered with  $\mathbf{F}_{mag}(t)$  (fixed force-direction) using  $d_z(t)$  as penetration depth (continuously varying force commands). The measured physical characteristics of perceived aliveness were different from those of buzzing.

Analyses in the frequency domain shed little insight on the signals responsible for the percept of aliveness. However, examination of data in the time domain revealed many instances where perceptible changes in force occurred while the stylus was perceived to be stationary in space along the direction of the force changes.

In Figure 6, the two horizontal axes indicate position normal to the textured surface  $p_z(t)$  and along the lateral stroking direction  $p_x(t)$ . The vertical axis shows forces felt by the subject's hand. The duration of the data set is 400 ms. The large change in  $p_x(t)$  was the result of the subject stroking the textured surface. In contrast, there was little change in  $p_z(t)$ . The change in force was on the order of 0.5 Newtons. As a result, the subject felt a noticeable change in normal force although the stylus was perceived to be barely moving into the textured surface. Therefore, the force variation was interpreted as coming from an alive textured surface. Indeed, subjects sometimes referred to the virtual object as a pulsating textured surface. These observations suggest that aliveness was caused by larger-than-expected force variations in spite of position changes that were barely perceivable.



5 Frequency domain analysis of the signals responsible for buzzing.

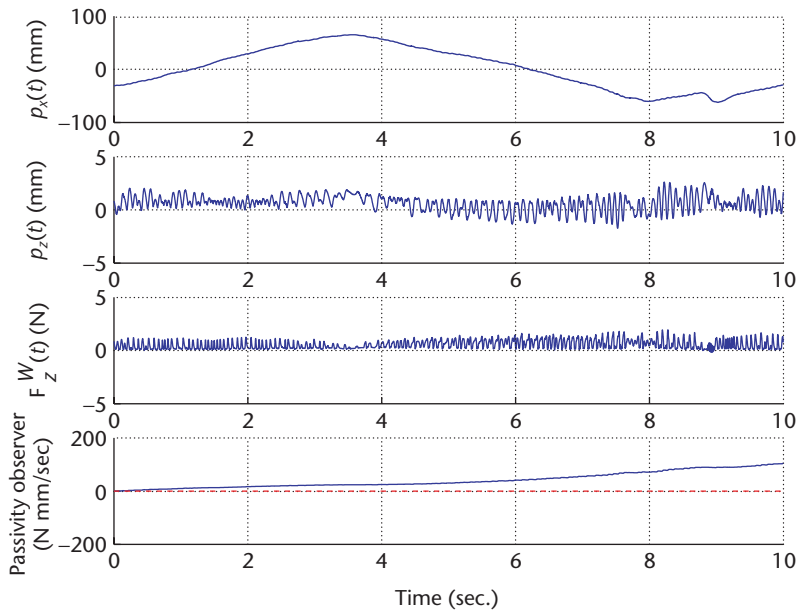


6 Analysis of aliveness perception in the time domain.  $F_z^S(t)$  denotes the force measured along the long axis of the styles.

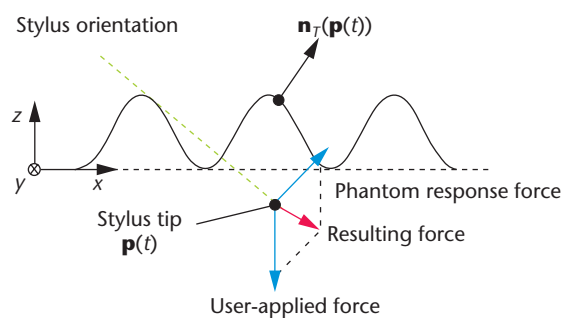
We suspected that, unlike buzzing, which was caused by unstable control of haptic interface, aliveness was probably caused by inaccurate environment dynamics. To investigate this hypothesis, we examined whether it was possible for a user to perceive aliveness while the texture-rendering system including the force-feedback device was stable in the control sense. We applied a passivity-based control theory to data measured from a user interacting with virtual textured surfaces. You can regard a dynamic system as passive if it preserves or dissipates its initial energy despite its interaction with an external system. Because passivity is a sufficient condition for control stability,<sup>8</sup> our hypothesis could be confirmed if we found cases in which a user perceived aliveness from a passive texture rendering system.

Using a passivity observer—an online observer for monitoring the energy flow of a dynamic system—we confirmed that aliveness perception could indeed occur when the haptic texture-rendering system was passive and stable. An example is shown in Figure 7. In this figure, the top panel shows the position data along the lateral stroking direction  $p_x(t)$ , the second panel shows the





**7** Example of using a passivity observer to analyze data corresponding to aliveness perception.  $F_z^W$  denotes the force measured along the direction normal to the textured surface.



**8** Illustration of the force components involved in ridge instability.

position variable in the normal direction  $p_z(t)$ , the third panel shows the force along the normal direction  $F_z^W(t)$ , and the bottom panel shows the values of the passivity observer. We can see that despite the abrupt force changes that resulted in the perception of aliveness, the passivity observer remained positive. These results provide unequivocal evidence that perceived instability can occur even when a haptic texture-rendering system is passive and stable. We have therefore shown indirectly that environment modeling and human perception can also play important roles in perceived quality of a haptic texture-rendering system.

Consider the difference between touching a real and a virtual surface. When a stylus touches a real surface, it's either on or off the surface, but not inside the surface. When a stylus touches a virtual surface, however, the stylus must penetrate the virtual surface for the user to form a perception of that surface through the resultant force variations. With a real surface, a stylus resting on the surface can remain stationary. With a virtual surface, however, the stylus's position can fluctuate inside the surface and this fluctuation is amplified to result in perceivable

force variations by a texture renderer, thereby contributing to the perception of aliveness. It is well known that humans tend to rely more on vision for position-movement information, and that we can easily integrate visual position information with haptic force information. Our relatively poor kinesthetic resolution of unsupported hand movements in free space—combined with our relatively high sensitivity to force changes—is also responsible for the perception of aliveness.

The last type of perceived instability, called ridge instability, is different from the first two types in the sense that it is related to the perception of force directions. We use the term ridge instability to refer to the phenomenon that the Phantom stylus was actively pushed to the valleys of the virtual textures rendered with  $\mathbf{F}_{\text{vec}}(t)$  when the stylus was placed on the ridges of the textures. When a

real stylus rests on the ridge of a real surface with sinusoidal gratings, the reaction force and friction of the surface combine to counterbalance the force exerted by the user's hand holding the stylus, thereby creating an equilibrium. The force rendered by  $\mathbf{F}_{\text{vec}}(t)$ , however, was solely based on the local texture geometry and did not take into account the direction of user-applied force, as illustrated in Figure 8. In this figure, we assume that the force applied by the user was normal to the plane underneath the texture. According to the environment model  $\mathbf{F}_{\text{vec}}(t)$ , the force applied by the Phantom was always in the direction of the surface normal  $\mathbf{n}_T(\mathbf{p}(t))$ . As a result, the net force exerted on the tip of the stylus—the sum of the forces applied by the user and the Phantom—was directed toward the valley of the sinusoidal grating. Therefore, the subject who tried to rest the stylus on the ridge could feel the stylus being actively pushed into the valley.

## Conclusions

In this article, we have shown that current haptic texture-rendering systems might suffer from several types of perceived instability. We also have demonstrated that perceived instability can come from many sources, including the traditional control instability of haptic interfaces as well as inaccurate modeling of environment dynamics and the difference in sensitivity to force and position changes of the human somatosensory system. Our work underscores the importance of developing texture-rendering algorithms that guarantee the perceptual realism of virtual haptic textures. It is our hope that this article will encourage more researchers to contribute to the study of perceived instability of virtual haptic textures. ■

## Acknowledgments

This work was supported in part by a National Science Foundation (NSF) Faculty Early Career Develop-

ment (CAREER) Award under grant 9984991-IIS and in part by an NSF award under grant 0098443-IIS. We thank Blake Hannaford and Jee-Hwan Ryu for discussions on the passivity observer. We also thank Vincent Hayward for discussions on velocity estimation. The thoughtful comments from the anonymous reviewers are greatly appreciated.

## References

1. T.H. Massie, *Initial Haptic Explorations with the Phantom: Virtual Touch through Point Interaction*, master's thesis, Dept. of Mechanical Eng., Massachusetts Inst. of Technology, 1996.
2. C. Ho, C. Basdogan, and M.A. Srinivasan, "Efficient Point-Based Rendering Techniques for Haptic Display of Virtual Objects," *Presence*, vol. 8, no. 5, 1999, pp. 477-491.
3. S. Choi and H.Z. Tan, "Perceived Instability of Virtual Haptic Texture. I. Experimental Studies," *Presence*, 2003, to be published.
4. S. Choi and H.Z. Tan, "An Analysis of Perceptual Instability During Haptic Texture Rendering," *Proc. 10th Int'l Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems (IEEE VR 02)*, IEEE CS Press, 2002, pp. 129-36.
5. S. Choi and H.Z. Tan, "A Study on the Sources of Perceptual Instability During Haptic Texture Rendering," *Proc. IEEE Int'l Conf. Robotics and Automation*, IEEE CS Press, 2002, pp. 1261-1268.
6. S. Choi and H.Z. Tan, "An Experimental Study of Perceived Instability During Haptic Texture Rendering: Effects of Collision Detection Algorithm," *Proc. 11th Int'l Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems (IEEE VR 03)*, IEEE CS Press, 2003, pp. 197-204.
7. S. Choi and H.Z. Tan, "Aliveness: Perceived Instability from a Passive Haptic Texture Rendering System," *Proc.*

*IEEE/RSJ Int'l Conf. Intelligent Robots and Systems*, IEEE CS Press, 2003, pp. 2678-2683.

8. B. Hannaford and J.-H. Ryu, "Time-Domain Passivity Control of Haptic Interfaces," *IEEE Trans. Robotics and Automation*, IEEE CS Press, vol. 18, no. 1, 2002, pp. 1-10.
9. S.J. Lederman and R.L. Klatzky, "Hand Movement: A Window into Haptic Object Recognition," *Cognitive Psychology*, vol. 19, 1987, pp. 342-368.
10. G.A. Gescheider, *Psychophysics: Method, Theory, and Application*, 2nd ed., Lawrence Erlbaum Assoc., 1985.



**Seungmoon Choi** is a postdoctoral research associate at the Haptic Interface Research Laboratory at Purdue University. His research interests include haptic rendering, psychophysics, and robotics. Choi received a PhD in electrical and computer engineering from Purdue University.



**Hong Z. Tan** is an associate professor in the school of electrical and computer engineering and the school of mechanical engineering (courtesy) at Purdue University. Her research interests include haptic interfaces, psychophysics, haptic rendering, and distributed contact sensing. Tan received a PhD in electrical engineering and computer science from the Massachusetts Institute of Technology.

Readers may contact Hong Tan at Purdue Univ., Electrical Eng. Building, 465 Northwestern Ave., West Lafayette, IN 47907; hongtan@purdue.edu.

## IEEE Transactions on Mobile Computing

**A** revolutionary new quarterly journal that seeks out and delivers the very best peer-reviewed research results on mobility of users, systems, data, computing information organization and access, services, management, and applications. *IEEE Transactions on Mobile Computing* gives you remarkable breadth and depth of coverage ...



**Architectures**  
**Support Services**  
**Algorithm/Protocol Design and Analysis**  
**Mobile Environment**  
**Mobile Communication Systems**  
**Applications**  
**Emerging Technologies**



To subscribe:  
<http://computer.org/tmc>  
 or call  
 USA and CANADA:  
**+1 800 678 4333**  
 WORLDWIDE:  
**+1 732 981 0060**

# Haptic Display of Interaction between Textured Models

Miguel A. Otaduy

Nitin Jain

Avneesh Sud

Ming C. Lin

Department of Computer Science  
University of North Carolina at Chapel Hill

<http://gamma.cs.unc.edu/HTextures>

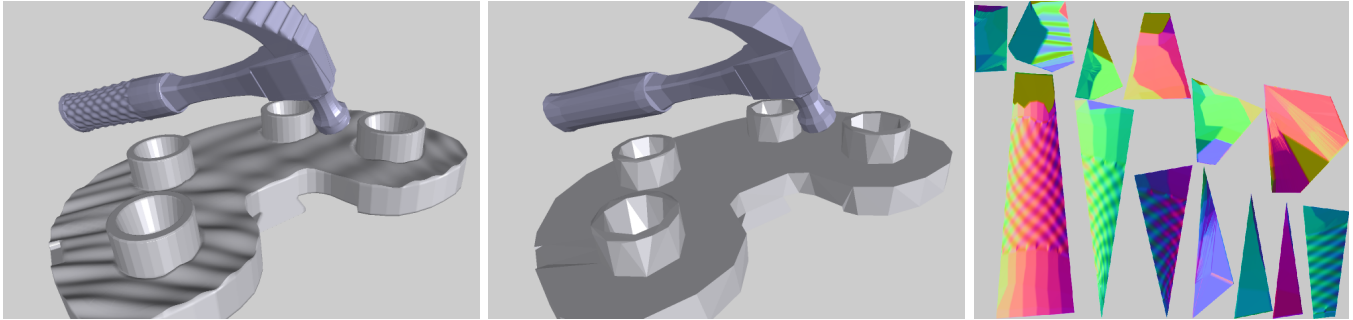


Figure 1: **Haptic Display of Interaction between Textured Models.** From left to right: (a) high-resolution textured hammer (433K polygons) and CAD part (658K polygons), (b) low-resolution models (518 & 720 polygons), (c) hammer texture with fine geometric detail.

## ABSTRACT

Surface texture is among the most salient haptic characteristics of objects; it can induce vibratory contact forces that lead to perception of roughness. In this paper, we present a new algorithm to display haptic texture information resulting from the interaction between two textured objects. We compute contact forces and torques using low-resolution geometric representations along with texture images that encode surface details. We also introduce a novel force model based on directional penetration depth and describe an efficient implementation on programmable graphics hardware that enables interactive haptic texture rendering of complex models. Our force model takes into account important factors identified by psychophysics studies and is able to haptically display interaction due to fine surface textures that previous algorithms do not capture.

**Keywords:** haptics, textures, graphics hardware

## 1 INTRODUCTION

Haptic rendering provides a unique, two-way communication between humans and interactive systems, enabling bi-directional interaction via tactile sensory cues. By harnessing the sense of touch, haptic display can further enhance a user's experience in a multi-modal synthetic environment, providing a more natural and intuitive interface with the virtual world. A key area in haptics that has received increasing attention is the rendering of *surface texture*, i.e. fine geometric features on an object's surface. The intrinsic surface property of texture is among the most salient haptic characteristics of objects. It can be a compelling cue to object identity, and it can strongly influence forces during manipulation [16]. In medical applications with limited visual feedback, such as minimally-invasive or endoscopic surgery [24], and virtual prototyping applications of

mechanical assembly and maintainability assessment [27], accurate haptic feedback of surface detail is a key factor for successful meticulous operations.

Most of the existing haptic rendering algorithms have focused primarily on force rendering of rigid or deformable flat polygonal models. This paper addresses the simulation of forces and torques due to interaction *between two textured objects*. Effective physically-based force models have been proposed to render the interaction between the tip (a point) of a haptic probe and a textured object [18, 10]. However, no technique is known to display both interaction forces and torques between two textured models. In fact, computation of texture-induced forces using full-resolution geometric representations of the objects and handling contacts at micro-geometric scale is computationally prohibitive.

Similar to graphical texture rendering [2], objects with high combinatorial complexity (i.e. with a high polygon count) can be described by coarse representations with their fine geometric detail stored in texture images, which we will refer to as *haptic textures* in this paper. Given this representation and a new force model that captures the effect of geometric surface details, we are able to haptically display intricate interaction between highly complex models using haptic textures instead of actual surface geometry.

**Main Contributions:** In this paper, we introduce a physically-based algorithm for incorporating texture effects to haptic display of interaction between two polygonal models. This algorithm enables, for the first time, interactive haptic display of forces and torques due to fine surface details. The main results of our paper are:

- A novel force model for haptic texture rendering, based on the gradient of directional penetration depth, that accounts for important factors identified by psychophysics studies;
- A fast algorithm for approximating directional penetration depth between textured objects;
- An efficient implementation on programmable graphics hardware that enables interactive haptic display of forces and torques between complex textured models;

- A new approach to haptically render complex interaction due to fine surface details using simplified representations of the original models and the corresponding haptic textures.

Our algorithm can be integrated in state-of-the-art haptic rendering algorithms to enhance the range of displayed stimuli. We have successfully tested and demonstrated our algorithm and implementation on several complex textured models. Some examples are shown in Fig. 1. Subjects were able to perceive roughness of various surface textures.

**Organization:** The rest of the paper is organized as follows. In Sec. 2 we discuss related work. Sec. 3 defines key terminology and describes several important concepts central to our force model. Sec. 4 presents the force computation model. Sec. 5 introduces a simple yet effective algorithm for approximating directional penetration depth and its parallel implementation on graphics processors. We then describe our results in Sec. 6. Finally, we discuss and analyze our approach in Sec. 7 and conclude with possible future research directions in Sec. 8.

## 2 PREVIOUS WORK

In this section we briefly discuss related work on haptic rendering and penetration depth computations.

### 2.1 Six Degree-of-Freedom Haptics

Haptic display of forces and torques between two interacting objects is commonly known as 6 degree-of-freedom (DoF) haptics. In all approaches to 6-DoF haptics, collision detection is a dominant computational cost. The performance of collision detection algorithms depends on the size of the input models, which in turn depends on the sampling density of the models, both for polygonal representations [23, 15, 11] and for voxel-based representations [17, 27].

To be correctly represented, surfaces with high-frequency geometric texture detail require higher sampling densities, thereby increasing the cost of collision detection. As a result, haptic rendering of forces between textured objects becomes computationally infeasible to achieve, and new representations must be considered.

Otaduy and Lin [20] recently suggested multiresolution representations to minimize the computational impact of collision detection and to adaptively select the appropriate resolution at each contact location. However, their approach filters out high resolution geometric features, thus ignoring all texture effects.

### 2.2 Haptic Texture Rendering

Rendering and perception of textures has been one of the most active areas in haptics research. Please refer to [16] for a survey on psychophysics of tactile texture perception. Klatzky and Lederman made important distinctions between perception of textures with bare skin vs. perception through a rigid object. When perceived through a rigid probe, roughness of a textured surface is encoded as vibration.

Several researchers have successfully developed haptic texture rendering techniques for interaction between a probe point and an object, using coarse geometric approximations and geometric texture images. These techniques use the idea of computing geometry-dependent high frequency forces, which transmit vibratory information to the user, and are perceived as virtual roughness. Minsky [18] showed that texture information can be conveyed by displaying forces on the tangent plane defined by the contact normal. Minsky computed a texture-induced force proportional to the gradient of a 2D height field stored in a texture map. Ho et al. [10] have proposed techniques that alter the magnitude and direction of 3D normal force based on height field gradient. Siira and Pai [26] followed

a stochastic approach, where texture forces are computed according to a Gaussian distribution.

All these techniques exploit the fact that, for point-object contact, a pair of texture coordinates can be well defined, and this is used to query height fields stored in texture maps. Note that only geometric effects of one object are captured. We are interested in rendering forces occurring during the interaction of two surfaces. In this case, the geometric interaction is not limited to and cannot be described by a pair of contact points. Moreover, the local kinematics of the contact between two surfaces include rotational degrees of freedom, not captured by point-based haptic rendering methods.

Choi and Tan [3] have studied the influence of collision detection and penetration depth computation on point-based haptic rendering, and their findings appear to be applicable to 6-DoF haptics as well.

### 2.3 Penetration Depth Computation

Several algorithms [12, 6, 5, 13] have been proposed for computing a measure of penetration depth using various definitions. However, each of them assumes that at least one of the input models is a convex polytope. It is commonly known that if two polytopes intersect, then the difference of their reference vectors with respect to the origin of the world coordinate system lies in their convolution or Minkowski sum [8]. The problem of penetration depth computation reduces to calculating the minimum distance from the origin to the boundary of the Minkowski sum of two polyhedra. The worst case complexity for two general, non-convex polyhedra can be as high as  $O(m^3n^3)$ , where  $m, n$  are the number of polygons in each model. Kim et al. [14] presented an algorithm for estimating penetration depth between two polyhedral models using rasterization hardware and hierarchical refinement. Although it offers better performance than previous techniques, this approach may take up to minutes to compute the penetration depth, making it inadequate for haptic simulation.

In this paper we present a new algorithm to estimate directional penetration depth between models described by low-resolution representations and haptic textures. Unlike the algorithm by Kim et al. [14], it does not compute the global penetration depth between two models, but its performance makes it suitable for haptic display.

## 3 PRELIMINARIES

In this section we first introduce notation used in the paper. Then, we present definitions related to penetration depth, which is an essential element of our force model. Finally, we describe the computational pipeline for haptic rendering of interaction between textured models.

### 3.1 Notations

A *height field*  $H$  is defined as a set  $H = \{(x, y, z) \mid z = h(x, y), (x, y, z) \in \mathbb{R}^3\}$ . We call  $h : \mathbb{R}^2 \rightarrow \mathbb{R}$  a *height function*. Let  $\mathbf{q}$  denote a point in  $\mathbb{R}^3$ , let  $\mathbf{q}_{xyz} = (q_x \ q_y \ q_z)^T$  denote the coordinates of  $\mathbf{q}$  in a global reference system, and  $\mathbf{q}_{u,v,n} = (q_u \ q_v \ q_n)^T$  its coordinates in a rotated reference system  $\{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$ . A surface patch  $S \subset \mathbb{R}^3$  can be represented as a height field along a direction  $\mathbf{n}$  if  $q_n = h(q_u, q_v), \forall \mathbf{q} \in S$ . Then, we can define a mapping  $g : D \rightarrow S, D \subset \mathbb{R}^2$ , as  $g(q_u, q_v) = \mathbf{q}_{xyz}$ , where:

$$h(q_u, q_v) = q_n = \mathbf{n} \cdot \mathbf{q}_{xyz} = \mathbf{n} \cdot g(q_u, q_v) \quad (1)$$

The inverse of the mapping  $g$  is the orthographic projection of  $S$  onto the plane  $(\mathbf{u}, \mathbf{v})$  along the direction  $\mathbf{n}$ .



### 3.2 Definitions of Penetration Depth

Penetration depth  $\delta$  between two intersecting polytopes is typically defined as the minimum translational distance required to separate them (see Fig. 2-b). As mentioned in Sec. 2.3, this distance is equivalent to the distance from the origin to the Minkowski sum of the polyhedra. *Directional penetration depth*  $\delta_{\mathbf{n}}$  along the direction  $\mathbf{n}$  is defined as the minimum translation along  $\mathbf{n}$  to separate the polyhedra (see Fig. 2-c). The penetration depth between two intersecting surface patches will be referred to as *local penetration depth*.

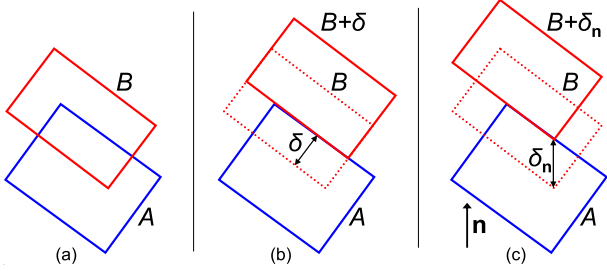


Figure 2: **Definitions of Penetration Depth.** (a) Intersecting objects  $A$  and  $B$ , (b) global penetration depth  $\delta$ , and (c) directional penetration depth  $\delta_{\mathbf{n}}$  along  $\mathbf{n}$ .

Let us assume that two intersecting surface patches  $S_A$  and  $S_B$  can be represented as height fields along a direction  $\mathbf{n}$ . Consequently,  $S_A$  and  $S_B$  can be parameterized by orthographic projection along  $\mathbf{n}$ , as expressed in Sec. 3.1. As a result of the parameterization, we obtain mappings  $g_A : D_A \rightarrow S_A$  and  $g_B : D_B \rightarrow S_B$ , as well as height functions  $h_A : D_A \rightarrow \mathbb{R}$  and  $h_B : D_B \rightarrow \mathbb{R}$ . The directional penetration depth  $\delta_{\mathbf{n}}$  of the surface patches  $S_A$  and  $S_B$  is the maximum height difference along the direction  $\mathbf{n}$ , as illustrated in Fig. 3 by a 2D example. Therefore, we can define the directional penetration depth  $\delta_{\mathbf{n}}$  as:

$$\delta_{\mathbf{n}} = \max_{(u,v) \in (D_A \cap D_B)} (h_A(u,v) - h_B(u,v)) \quad (2)$$

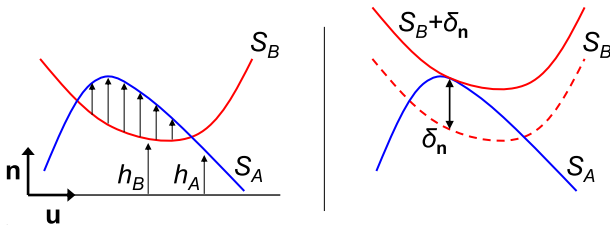


Figure 3: **Penetration Depth of Height Fields.** Directional penetration depth of surface patches expressed as height difference.

### 3.3 Haptic Display Pipeline

We assume that the interacting objects can be described as parameterized low-resolution triangle meshes with texture maps that store fine geometric detail. In a haptic simulation of object-object interaction, the object whose motion is controlled by the user is called the *probe object*. Contacts between the probe object and the rest of the objects in the environment generate forces that are displayed to the user.

Following a common approach in 6-DoF haptics, we simulate the dynamics of the probe object as a result of contact forces and a virtual coupling force that ensures stable interaction with the user [1]. We propose a novel algorithm for computing contact forces, taking into account texture effects. We follow the steps below to compute contact forces:

1. Each haptic simulation frame starts by performing collision detection between the low-resolution meshes. We then identify intersecting surface patches as contacts. We characterize each contact by a pair of contact points on the patches and a penetration direction  $\mathbf{n}$ .
2. For each contact, we compute force and torque using our novel force model for texture rendering, based on the penetration depth and its gradient. The penetration depth is approximated taking into account fine geometric detail stored in haptic textures.
3. The forces and torques of all contacts are combined to compute the net force and torque on the probe object.

Other effects, such as friction [9], can easily be incorporated into this display pipeline using the contact information computed between the low-resolution meshes.

## 4 A FORCE MODEL FOR TEXTURE RENDERING

In this section we describe our force model for haptic display of interaction between textured surfaces. We first show how factors highlighted by psychophysics studies are taken into account. Then, we introduce a penalty-based force model for texture rendering. Finally, we present the formulation of the gradient of penetration depth used in our force model.

### 4.1 Foundation of the Proposed Force Model

Roughness of surface textures perceived through a rigid probe is mainly encoded as vibration and strongly influences the forces that must be applied to manipulate the objects [16]. In point-based haptic texture rendering, vibrating forces are commonly computed using a height field gradient [18, 10]. Our force model generalizes the point-based approach by computing forces based on the *gradient of penetration depth* between two objects.

Based on psychophysics studies, Klatzky and Lederman [16] highlight factors influencing perception of roughness through a rigid spherical probe. These factors are:

**Probe Radius:** For spherical probes, the texture frequency at which perception of roughness is maximum depends on probe radius. At low frequencies, roughness increases with texture frequency, but after reaching a peak, roughness decreases as texture frequency increases. Our conjecture is that roughness perception is tightly coupled to the trajectory traced by the probe, which can be regarded as an offset surface of the perceived geometry. Okamura and Cutkosky [19] also modeled interaction between robotic fingers and textured surfaces by tracing offset surfaces. They defined an offset surface as the boundary of the Minkowski sum of a given surface and a sphere. Therefore, the height of the offset surface at a particular point is the distance to the boundary of the Minkowski sum for a particular position of the probe, also known to be the penetration depth<sup>1</sup>. In other words, the height of the offset surface reflects the distance that the probe must move in order to avoid interpenetration with the surface. Since, for spherical probes, perception of roughness seems to be tightly coupled with the oscillation of offset surfaces, in our force model for general surfaces we have taken into account the *variation of penetration depth*, i.e. its gradient.

**Normal Force:** Perception of roughness grows monotonically with normal force. This relation is also captured by our force model in

<sup>1</sup> Actually, the height of the offset surface is the distance to the surface along a particular direction, so the distance to the boundary of the Minkowski sum must also be measured along a particular direction. This is known to be the *directional penetration depth*.

a qualitative way, in making tangential forces and torques proportional to the normal force.

**Exploratory Speed:** The exploratory speed, or velocity of the probe in the plane of contact with the surface, affects the perception of roughness. Our force model is intrinsically geometry-based, but in a haptic simulation dynamic effects are introduced by the haptic device and the user. We have analyzed the dynamic behavior of our force model, and we have observed that vibratory motion produced by simulated forces behaves in a way similar to physical roughness perception. The results of our experiments are described in detail in [21].

The influence of probe geometry, normal force and exploratory speed is taken into consideration in the design of our force model, which will be presented next.

## 4.2 Penalty-Based Texture Force

For two objects  $A$  and  $B$  in contact, we define a penalty-based force proportional to the penetration depth  $\delta$  between them. Penalty-based forces are conservative, and they define an elastic potential field. In our force model we have extended this principle to compute texture-induced forces between two objects.

We define an elastic penetration energy  $U$  with stiffness  $k$  as:

$$U = \frac{1}{2} k \delta^2 \quad (3)$$

Based on this energy, we define force  $\mathbf{F}$  and torque  $\mathbf{T}$  as:

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{T} \end{pmatrix} = -\nabla U = -k\delta(\nabla\delta) \quad (4)$$

where  $\nabla = \left( \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, \frac{\partial}{\partial \theta_x}, \frac{\partial}{\partial \theta_y}, \frac{\partial}{\partial \theta_z} \right)$  is the gradient in 6-DoF configuration space.

As described in Sec. 3.3, each contact between objects  $A$  and  $B$  can be described by a pair of contact points  $\mathbf{p}_A$  and  $\mathbf{p}_B$ , and by a penetration direction  $\mathbf{n}$ . We assume that, locally, the penetration depth between objects  $A$  and  $B$  can be approximated by the directional penetration depth  $\delta_{\mathbf{n}}$  along  $\mathbf{n}$ . We rewrite Eq. 4 for  $\delta_{\mathbf{n}}$  in a reference system  $\{\mathbf{u}, \mathbf{v}, \mathbf{n}\}^2$ . In this case, Eq. 4 reduces to:

$$\begin{pmatrix} F_u & F_v & F_n & T_u & T_v & T_n \end{pmatrix}^T = -k\delta_{\mathbf{n}} \begin{pmatrix} \frac{\partial \delta_{\mathbf{n}}}{\partial u} & \frac{\partial \delta_{\mathbf{n}}}{\partial v} & 1 & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_u} & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_v} & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_n} \end{pmatrix}^T \quad (5)$$

where  $\theta_u$ ,  $\theta_v$  and  $\theta_n$  are the rotation angles around the axes  $\mathbf{u}$ ,  $\mathbf{v}$  and  $\mathbf{n}$  respectively.

The force and torque on object  $A$  (and similarly on object  $B$ ) for each contact can be expressed in the global reference system as:

$$\begin{aligned} \mathbf{F}_A &= (\mathbf{u} \ \mathbf{v} \ \mathbf{n}) (F_u \ F_v \ F_n)^T \\ \mathbf{T}_A &= (\mathbf{u} \ \mathbf{v} \ \mathbf{n}) (T_u \ T_v \ T_n)^T \end{aligned} \quad (6)$$

As explained in Sec. 3.3, forces and torques of all contacts are summed up to compute the net force and torque.

Generalizing Minsky's approach [18], we define tangential forces  $F_u$  and  $F_v$  proportional to the gradient of penetration depth. However, we also define a penalty-based normal force and gradient-dependent torque that describe full 3D object-object interaction. In addition, in our model the tangential force and the torque are proportional to the normal force, which is consistent with psychophysics studies showing that perceived roughness increases with the magnitude of the normal force [16].

<sup>2</sup> $\mathbf{u}$  and  $\mathbf{v}$  may be selected arbitrarily as long as they form an orthonormal basis with  $\mathbf{n}$ .

## 4.3 Penetration Depth and Gradient

In our formulation,  $\delta$  and  $\delta_{\mathbf{n}}$  are functions defined on a 6-DoF configuration space. We have opted for central differencing over one-sided differencing to approximate  $\nabla\delta_{\mathbf{n}}$ , because it offers better interpolation properties and higher order approximation. The partial derivatives are computed as:

$$\frac{\partial \delta_{\mathbf{n}}}{\partial u} = \frac{\delta_{\mathbf{n}}(u + \Delta u, v, n, \theta_u, \theta_v, \theta_n) - \delta_{\mathbf{n}}(u - \Delta u, v, n, \theta_u, \theta_v, \theta_n)}{2\Delta u} \quad (7)$$

and similarly for  $\frac{\partial \delta_{\mathbf{n}}}{\partial v}$ ,  $\frac{\partial \delta_{\mathbf{n}}}{\partial \theta_u}$ ,  $\frac{\partial \delta_{\mathbf{n}}}{\partial \theta_v}$  and  $\frac{\partial \delta_{\mathbf{n}}}{\partial \theta_n}$ .

$\delta_{\mathbf{n}}(u + \Delta u, \dots)$  can be obtained by translating object  $A$  a distance  $\Delta u$  along the  $\mathbf{u}$  axis and computing the directional penetration depth. A similar procedure is followed for other penetration depth values.

## 5 DIRECTIONAL PENETRATION DEPTH

In this section we present an algorithm for approximating local directional penetration depth for textured models and describe a parallel implementation on graphics hardware.

### 5.1 Approximate Directional Penetration Depth between Textured Models

A contact between objects  $A$  and  $B$  is defined by two intersecting surface patches  $S_A$  and  $S_B$ . The surface patch  $S_A$  is approximated by a low-resolution surface patch  $\hat{S}_A$  (and similarly for  $S_B$ ). We define  $f_A : \hat{S}_A \rightarrow S_A$ , a mapping function from the low-resolution surface patch  $\hat{S}_A$  to the surface patch  $S_A$ .

Collision detection between two low-resolution surfaces patches  $\hat{S}_A$  and  $\hat{S}_B$  returns a penetration direction  $\mathbf{n}$ . Let us assume that both  $S_A$  and  $\hat{S}_A$  (and similarly for  $S_B$  and  $\hat{S}_B$ ) can be represented as height fields along  $\mathbf{n}$ , following the definition in Sec. 3.1. Given a rotated reference system  $\{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$ , we can project  $S_A$  and  $\hat{S}_A$  orthographically along  $\mathbf{n}$  onto the plane  $(\mathbf{u}, \mathbf{v})$ . As the result of this projection, we obtain mappings  $g_A : D_A \rightarrow S_A$  and  $\hat{g}_A : \hat{D}_A \rightarrow \hat{S}_A$ . We define  $\bar{D}_A = D_A \cap \hat{D}_A$ .

The mapping function  $g_A$  can be approximated by a composite mapping function  $f_A \circ \hat{g}_A : \bar{D}_A \rightarrow S_A$  (See Fig. 4). From Eq. 1, we define an approximate height function  $\hat{h} : \bar{D}_A \rightarrow \mathbb{R}$  as:

$$\hat{h}(u, v) = \mathbf{n} \cdot (f_A \circ \hat{g}_A(u, v)) \quad (8)$$

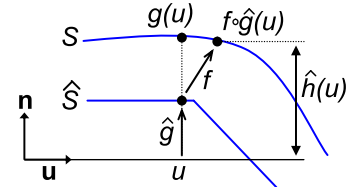


Figure 4: **Approximate Height Function.** Height function of a surface patch approximated by a composite mapping function.

Given approximate height functions  $\hat{h}_A$  and  $\hat{h}_B$ , a domain  $D = \bar{D}_A \cap \bar{D}_B$ , and Eq. 2, we can approximate the directional penetration depth  $\delta_{\mathbf{n}}$  of  $S_A$  and  $S_B$  by:

$$\delta_{\mathbf{n}} = \max_{(u,v) \in D} (\hat{h}_A(u, v) - \hat{h}_B(u, v)) \quad (9)$$

Although this algorithm can be realized on CPUs, it is best suited for implementation on graphics processors (GPUs), as we will present next.

## 5.2 Computation on Graphics Hardware

As shown in Eq. 5, computation of 3D texture-induced force and torque according to our model requires the computation of directional penetration depth  $\delta_n$  and its gradient at every contact. From Eq. 7, this reduces to computing  $\delta_n$  all together at 11 configurations of object  $A^3$ . As pointed out in section 2.3, computation of penetration depth using exact object-space or configuration-space algorithms is too expensive for haptic rendering applications. Instead, the approximation  $\hat{\delta}_n$  according to Eqs. 8 and 9 leads to a natural and efficient image-based implementation on programmable graphics hardware. The mappings  $\hat{g}$  and  $f$  correspond, respectively, to orthographic projection and texture mapping operations, which are best suited for the parallel and grid-based nature of GPUs.

For every contact, we first compute  $\hat{h}_B$ , and then perform two operations for each of the 11 object configurations: compute  $\hat{h}_A$  for the transformed object  $A$ , and then find the penetration depth  $\hat{\delta}_n = \max(\Delta\hat{h}) = \max(\hat{h}_A - \hat{h}_B)^4$ .

### Height Computation

In our GPU-based implementation, the mapping  $f: \hat{S} \rightarrow S$  is implemented as a texture map that stores geometric detail of the high-resolution surface patch  $S$ . We refer to  $f$  as a “haptic texture”. The mapping  $\hat{g}$  is implemented by rendering  $\hat{S}$  using an orthographic projection along  $\mathbf{n}$ . We compute the height function  $\hat{h}$  in a fragment program. We obtain a point in  $S$  by looking up the haptic texture  $f$  and then we project it onto  $\mathbf{n}$ . The result is stored in a floating point texture  $t$ .

We choose geometric texture mapping over other methods for approximating  $h$  (e.g. rendering  $S$  directly or performing displacement mapping) in order to maximize performance. We store the input haptic texture  $f$  as a floating point texture, thus alleviating precision problems.

### Max Search

The  $\max$  function in Eq. 9 can be implemented as a combination of frame buffer read-back and CPU-based search. However, we avoid expensive read-backs by posing the  $\max$  function as a binary search on the GPU [7]. Given two height functions  $\hat{h}_A$  and  $\hat{h}_B$  stored in textures  $t_1$  and  $t_2$ , we compute their difference and store it in the depth buffer. We scale and offset the height difference to fit in the depth range. Height subtraction and copy to depth buffer are performed in a fragment program, by rendering a quad that covers the entire buffer. For a depth buffer with  $N$  bits of precision, the search domain is the integer interval  $[0, 2^N)$ . The binary search starts by querying if there is any value larger than  $2^{N-1}$ . We render a quad at depth  $2^{N-1}$  and perform an occlusion query<sup>5</sup>, which will report if any pixel passed the depth test, i.e. the stored depth was larger than  $2^{N-1}$ . Based on the result, we set the depth of a new quad and continue the binary search.

### Gradient Computation

The height functions  $\hat{h}_A(\pm\Delta u)$ ,  $\hat{h}_A(\pm\Delta v)$  and  $\hat{h}_A(\pm\Delta\theta_n)$  may be obtained by simply translating or rotating  $\hat{h}_A(0)$ . As a result, only 6 height functions  $\hat{h}_A(0)$ ,  $\hat{h}_B(0)$ ,  $\hat{h}_A(\pm\Delta u)$  and  $\hat{h}_A(\pm\Delta\theta_n)$  need to be computed for each pair of contact patches. These 6 height functions are tiled in one single texture  $t$  to minimize context switches and increase performance (See Fig. 5). Moreover, the domain of each height function is split into 4 quarters, each of which is mapped to

<sup>3</sup>Note that, since we use central differencing to compute partial derivatives of  $\delta_n$ , we need to transform object  $A$  to two different configurations and recompute  $\delta_n$ . All together we compute  $\delta_n$  itself and 5 partial derivatives, hence 11 configurations

<sup>4</sup>We denote the height difference at the actual object configuration by  $\Delta\hat{h}(0)$ , and the height differences at the transformed configurations by  $\Delta\hat{h}(\pm\Delta u)$ ,  $\Delta\hat{h}(\pm\Delta v)$ ,  $\Delta\hat{h}(\pm\Delta\theta_u)$ ,  $\Delta\hat{h}(\pm\Delta\theta_v)$  and  $\Delta\hat{h}(\pm\Delta\theta_n)$ .

<sup>5</sup>[http://www.nvidia.com/dev\\_content/nvopengl/specs/GL\\_NV\\_occlusion\\_query.txt](http://www.nvidia.com/dev_content/nvopengl/specs/GL_NV_occlusion_query.txt)

one of the RGBA channels. This optimization allows us to exploit vector computation capabilities of fragment processors. As shown in Fig. 5, we also tile 11 height differences per contact in the depth buffer.

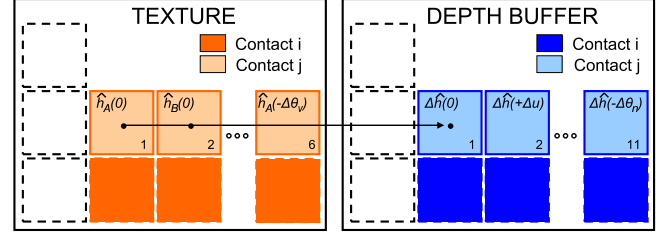


Figure 5: **Tiling in the GPU.** Tiling of multiple height functions and contacts to minimize context switches between target buffers.

### Multiple Simultaneous Contacts:

The computational cost of haptic texture rendering increases linearly with the number of contact patches between the interacting objects. However, performance can be further optimized. In order to limit context switches, we tile the height functions associated with multiple pairs of contact patches in one single texture  $t$ , and we tile the height differences in the depth buffer, as shown in Fig. 5. We also minimize the cost of “max search” operations by performing occlusion queries on all contacts in parallel.

## 6 RESULTS

We now describe the implementation details and results obtained with our haptic texture rendering algorithm, both in terms of force and motion characteristics, as well as performance.

### 6.1 Implementation Details

Our haptic texture rendering algorithm requires a preprocessing step. Input models are assumed to be 2-manifold triangle meshes with fine geometric details. We parameterize the meshes and create texture atlas storing surface positions. We also simplify the meshes to produce coarse resolution approximations which are used by the collision detection module. The parameterization must be preserved during the simplification process, and distortion must be minimized. Our implementation of parameterization-preserving simplification is based on existing approaches [25, 4].

As described in Sec. 3.3, before computing forces we perform collision detection between coarse-resolution models. We adapt the approach of Kim et al. [15] and decompose the models in convex pieces. Object interpenetration is considered to occur when objects are closer than a distance tolerance. In practice, by using this technique, penetration depth between the coarse resolution models is computed less frequently, thus accelerating collision detection.

For texture force computation, we compute each value of penetration depth between contact patches on a  $50 \times 50$ , 16-bit depth buffer. This resolution proved to be sufficient based on the results.

In our experiments we have used a 6-DoF *Phantom<sup>TM</sup>* haptic device, a dual Pentium-4 2.4GHz processor PC with 2.0 GB of memory and an NVidia GeForce FX5950 graphics card, and Windows2000 OS. The penetration depth computation on graphics hardware is implemented using OpenGL plus OpenGL’s ARB\_fragment\_program and GL\_NV\_occlusion\_query extensions. Our haptic texture rendering cannot be stalled by the visual display of the scene; hence, it requires a dedicated graphics card. We display the full resolution scene on a separate commodity PC. The force update of the haptic device takes place at a frequency of 1kHz, but the haptic simulation is executed in a separate thread that updates the force input to a stabilizing virtual coupling [1] asynchronously.

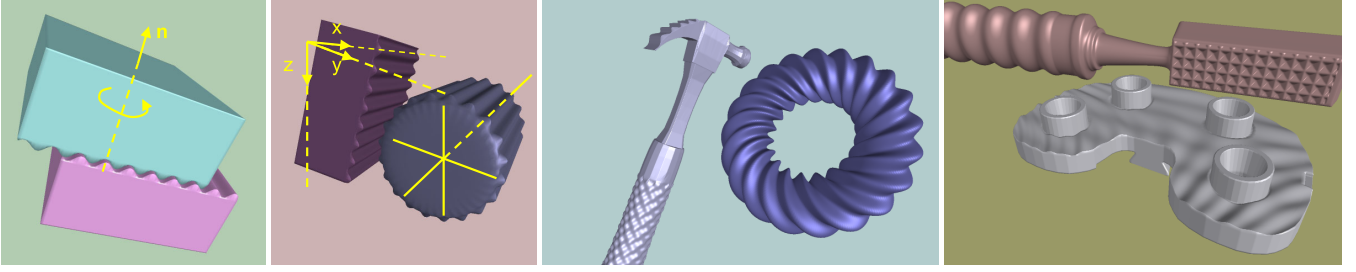


Figure 6: **Benchmark Models.** From left to right: (a) textured blocks, (b) block and gear, (c) hammer and torus, (d) file and CAD part.

## 6.2 Benchmarks

In our experiments we have used the models shown in Fig. 6. The complexity of full resolution textured models and their coarse resolution approximations is listed in Table 1.

Models	Full Res. Tris	Low Res. Tris	Low Res. Pcs
Block	65536	16	1
Gear	25600	1600	1
Hammer	433152	518	210
CAD Part	658432	720	390
File	285824	632	113
Torus	128000	532	114

Table 1: **Complexity of Benchmark Models.** Number of triangles at full resolution (Full Res. Tris) and low resolution (Low Res. Tris), and number of convex pieces at low resolution (Low Res. Pcs).

Notice the drastic simplification of the low resolution models. At this level all texture information is eliminated from the geometry, but it is stored in  $1024 \times 1024$ -size floating point textures. The number of convex pieces at coarse resolution reflects the geometric complexity for the collision detection module. Also notice that the *block* and *gear* models are fully convex at coarse resolution. The interaction between these models is described by one single contact, so they are better suited for analyzing force and motion characteristics in the simulations.

## 6.3 Conveyance of Roughness

We have performed experiments to test the conveyance of roughness with our haptic texture rendering algorithm.

**Roughness under Translation:** The gear and block models present ridges that interlock with each other. One of our experiments consisted of translating the block in the 3 Cartesian axes, while being in contact with the fixed gear, as depicted in Fig. 6-b. Fig. 7 shows the position of the block and the force exerted on it during 1500 frames of interactive simulation (approx. 3 seconds).

Notice that the force in the  $x$  direction, which is parallel to the ridges, is almost 0. Our model successfully yields this expected result, because the derivative of the penetration depth is 0 along the  $x$  direction. Notice also the staircase shape of the motion in the  $z$  direction, which reflects how the block rests for short periods of time on the ridges of the gear. The motion in the  $y$  direction resembles a staircase as well, but with small overshoots. These reflect the state between two successive interlocking situations, when the ridges are opposing each other. The wide frequency spectrum of staircase motion is possible due to the fine spatial resolution of penetration depth and gradient computation. Last, the forces in  $y$  and  $z$  are correlated with the motion profiles.

**Roughness under Rotation:** We placed two identical striped blocks interlocking each other, as shown in Fig. 6-a. We then performed small rotations of the upper block around the direction  $\mathbf{n}$ , and observed the induced translation along that same direction. Fig. 8 shows the rotation and translation captured during 6000

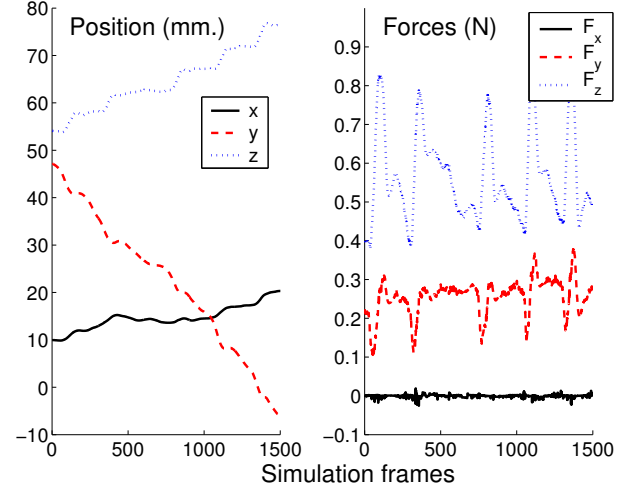


Figure 7: **Roughness under Translation.** Position and force profiles generated while translating the model of a textured block in contact with a gear model, as shown in Fig. 6-b. Notice the staircase like motion in  $z$ , and the correlation between force and position changes.

frames of interactive haptic simulation (approx. 12 seconds). Notice how the top block rises along  $\mathbf{n}$  as soon as we rotate it slightly, thus producing a motion very similar to the one that occurs in reality. Point-based haptic rendering methods are unable to capture this type of effect. Our force model successfully produces the desired effect by taking into account the local penetration depth between the blocks. Also, the derivative of the penetration depth produces a physically-based torque in the direction  $\mathbf{n}$  that opposes the rotation.

**Summary of Experiments:** From the experiments described above, we conclude that our force model successfully captures roughness properties of objects with fine geometric detail. We have also conducted informal experiments where subjects were asked to explore a textured plate with a virtual probe, while only the untextured coarse-resolution models were displayed graphically on the screen. Hence, the subjects could only recognize the texture patterns through haptic cues. The reported experiences were promising, as subjects were able to successfully describe regular patterns such as stripes, but had more difficulty with irregular patterns. This result is what we expect when real, physical textured models are explored.

## 6.4 Performance Tests

One of the key issues to achieve realistic haptic rendering is very high force update rate. High update rates enhance the stability of the system, as well as the range of stimuli that can be displayed. We have tested the performance of our haptic texture rendering algorithm and its implementation in scenarios where the coarse resolution models present complex contact configurations. These scenarios consist of a file scraping a rough CAD part, and a textured



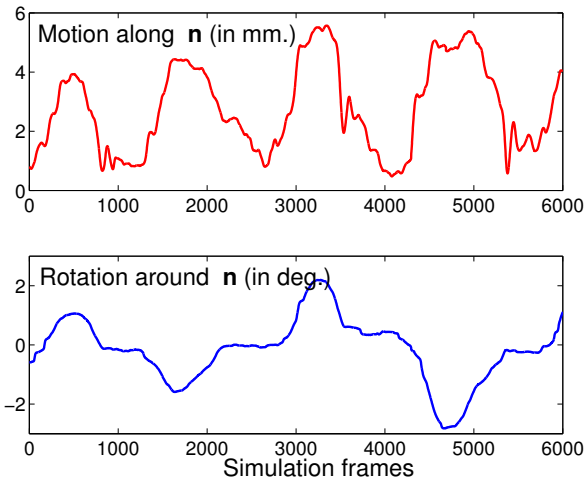


Figure 8: **Roughness under Rotation.** Motion profile obtained by rotating one textured block on top of another one, as depicted in Fig. 6-a. Notice the translation induced by the interaction of ridges during the rotational motion.

hammer touching a wrinkled torus. In particular, we show timings for 500 frames of the simulation of the file interacting with the CAD part in Fig. 9. The graph reflects the time spent on collision detection between the coarse-resolution models (an average of 2ms), the time spent on haptic texture rendering, and the total time per frame, which is approximately equal to the sum of the previous two. In this experiment we computed each value of penetration depth on a  $50 \times 50$  16-bit depth buffer (See Sec. 5.2). As shown in Sec. 6.3, this proved to be sufficient to display convincing roughness stimuli.

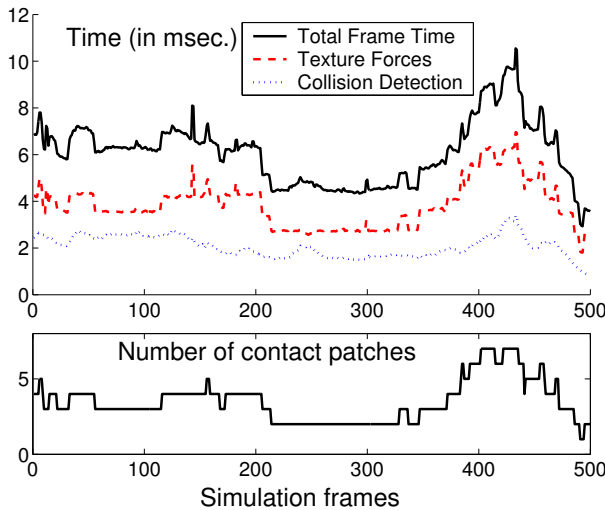


Figure 9: **Timings.** Performance analysis and number of clustered contact patches during 500 simulation frames of a file model scraping a CAD part, as shown in Fig. 6-d. In this complex contact scenario we are able to maintain a haptic frame rate between 100Hz and 200Hz.

In this particularly challenging experiment we were able to obtain haptic update rates between 100Hz and 200Hz. The dominant cost appears to be the haptic texture rendering, which depends nearly linearly on the number of contacts. The achieved force update rate may not be high enough to render textures with very high spatial frequency. However, as shown in Sec. 6.3, our proposed force model enables perception of roughness stimuli that were not captured previously by earlier methods. Moreover, in Fig. 9 we show performance results for a contact configuration in which large areas of the file at many different locations are in close proxim-

ity with the CAD part. In fact, collision detection using coarse-resolution models reports an average of 104 pairs of convex pieces in close proximity, which are later clustered into as many as 7 contacts. Using the full-resolution models, the number of contact pairs in close proximity would increase by several orders of magnitude, and simply handling collision detection would become challenging at the desired haptic rendering frame rates. Furthermore, as the support for programming on GPUs and capabilities of GPUs continue to grow at a rate faster than Moore's Law, we expect the performance of our algorithm to reach KHz update rates in the near future.

## 7 DISCUSSION AND ANALYSIS

In Sec. 6.3 we have analyzed forces and motion generated by our algorithm during actual haptic simulations. We have further analyzed the properties of the force model presented in Sec. 4 by simulating its behavior in experiments similar to the ones conducted in psychophysics studies [16]. Our main conclusion is that the acceleration produced by our force model matches qualitatively the behavior of roughness perception as a function of texture frequency. A detailed description of the experiments we have conducted can be found in [21].

Our force model and implementation present a few limitations, some of which are common to existing haptic rendering methods. Next we discuss these limitations.

### 7.1 Force Model

In some contact scenarios with large contact areas, the definition of a local and directional penetration depth is not applicable. An example is the problem of screw insertion. Situations also exist in which local geometry cannot be represented as height fields, and the gradient of directional penetration depth may not capture the effect of interlocking features.

As shown in Sec. 6, in practice our force model generates forces that create a realistic perception of roughness for object-object interaction; however, one essential limitation of penalty-based methods and impedance-type haptic devices is the inability to enforce motion constraints. Our force model attempts to do so by increasing tangential contact stiffness when the gradient of penetration depth is high. Implicit integration of the motion of the probe object allows for high stiffness and, therefore, small interpenetrations, but the perceived stiffness of rigid contact is limited through virtual coupling for stability reasons. New constraint-based haptic rendering techniques and perhaps other haptic devices [22] will be required to properly enforce constraints.

A very important issue in every force model for haptic rendering is its stability. Choi and Tan [3] have shown that even passive force models may suffer from a problem called *aliveness*. In our algorithm, discontinuities in the collision detection between low-resolution models are possible sources of aliveness.

### 7.2 Frequency and Sampling Issues

As with other sample-based techniques, our haptic texture rendering algorithm is susceptible to aliasing problems. Here we discuss different aliasing sources and suggest some solutions.

**Input textures:** The resolution of input textures must be high enough to capture the highest spatial frequency of input models, although input textures can be filtered as a preprocessing step to downsample and reduce their size.

**Image-based computation:** In the height function computation step, buffer resolution must be selected so as to capture the spatial

frequency of input models. Buffer size, however, has a significant impact in the performance of force computation.

**Discrete derivatives:** Penetration depth may not be a smooth function. This property results in an infinitely wide frequency spectrum, which introduces aliasing when sampled. Differentiation aggravates the problem, because it amplifies higher frequencies. The immediate consequence in our texture rendering approach is that the input texture frequencies have to be low enough so as to represent faithfully their derivatives. This limitation is common to existing point-based haptic rendering methods [18] as well.

**Temporal sampling.** Force computation undergoes temporal sampling too. The Nyquist rate depends on object speed and spatial texture frequency. Image-based filtering prior to computation of penetration depth may remove undesirable high frequencies, but it may also remove low frequencies that would otherwise appear due to the nonlinearity of the max search operation. In other words, filtering a texture with very high frequency may incorrectly remove all torque and tangential forces. Temporal supersampling appears to be a solution to the problem, but is often infeasible due to the high update rates required by haptic simulation.

## 8 CONCLUSION

We have introduced a new haptic rendering algorithm for displaying interaction between two textured models, based on localized directional penetration depth and its gradient. We have also presented an image-based implementation on programmable graphics hardware that enables interactive haptic display between complex textured models for the first time. We have further shown that, using a coarse geometric representation with haptic textures that encode fine surface details, it is possible to render contact forces and torques between two interacting textured models at haptic rates.

Several possible directions for future research remain, including but not limited to:

- Interactive haptic texture synthesis;
- Addition of constraint forces for fine motion and dexterous manipulation;
- Further analysis of human factors.

Finally, we would like to integrate our haptic rendering system with different applications, such as assisted technology, surgical training, and virtual prototyping.

## ACKNOWLEDGMENTS

This research is supported in part by a fellowship of the Government of the Basque Country, National Science Foundation, Office of Naval Research, U.S. Army Research Office, and Intel Corporation. We would like to thank Roberta Klatzky, Susan Lederman, Dinesh Manocha, Russ Taylor, Fred Brooks, Mark Foskey, Bill Baxter, the UNC Gamma group and the anonymous reviewers for their feedback on the earlier drafts of this paper.

## REFERENCES

- [1] R. J. Adams and B. Hannaford. A two-port framework for the design of unconditionally stable haptic interfaces. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.
- [2] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Dept. of CS, U. of Utah, December 1974.
- [3] S. Choi and H. Z. Tan. Aliveness: Perceived instability from a passive haptic texture rendering system. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.
- [4] J. Cohen, M. Olano, and D. Manocha. Appearance preserving simplification. In *Proc. of ACM SIGGRAPH*, pages 115–122, 1998.

- [5] D. Dobkin, J. Hersherberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9:518–533, 1993.
- [6] E.G. Gilbert and C.J. Ong. New distances for the separation and penetration of objects. In *Proceedings of International Conference on Robotics and Automation*, pages 579–586, 1994.
- [7] N. K. Govindaraju, B. Lloyd, W. Wang, M. C. Lin, and D. Manocha. Fast computation of database operations using graphics processors. *Proc. of ACM SIGMOD*, 2004.
- [8] L. J. Guibas and J. Stolfi. Ruler, compass and computer: the design and analysis of geometric algorithms. In R. A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, volume 40 of *NATO ASI Series F*, pages 111–165. Springer-Verlag, 1988.
- [9] V. Hayward and B. Armstrong. A new computational model of friction applied to haptic rendering. *Experimental Robotics VI*, 2000.
- [10] C.-H. Ho, C. Basdogan, and M. A. Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence*, 8(5):pp. 477–491, 1999.
- [11] D. Johnson and P. Willemsen. Six degree of freedom haptic rendering of complex polygonal models. In *Proc. of Haptics Symposium*, 2003.
- [12] S. S. Keerthi and K. Sridharan. Efficient algorithms for computing two measures of depth of collision between convex polygons. Technical Report, 1989.
- [13] Y. Kim, M. Lin, and D. Manocha. DEEP: an incremental algorithm for penetration depth computation between convex polytopes. *Proc. of IEEE Conference on Robotics and Automation*, pages 921–926, 2002.
- [14] Y. Kim, M. Otaduy, M. Lin, and D. Manocha. Fast penetration depth computation for physically-based animation. *Proc. of ACM Symposium on Computer Animation*, 2002.
- [15] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha. Six-degree-of-freedom haptic rendering using incremental and localized computations. *Presence*, 12(3):277–295, 2003.
- [16] R. L. Klatzky and S. J. Lederman. Perceiving texture through a probe. In M. L. McLaughlin, J. P. Hespanha, and G. S. Sukhatme, editors, *Touch in Virtual Environments*, chapter 10, pages 180–193. Prentice Hall PTR, Upper Saddle River, NJ, 2002.
- [17] W. McNeely, K. Puterbaugh, and J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. *Proc. of ACM SIGGRAPH*, pages 401–408, 1999.
- [18] M. Minsky. *Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display*. PhD thesis, Ph.D. Dissertation, Program in Media Arts and Sciences, MIT, 1995. Thesis work done at UNC-CH Computer Science.
- [19] A. M. Okamura and M. R. Cutkosky. Feature detection for haptic exploration with robotic fingers. *International Journal of Robotics Research*, 20(12):925–938, 2001.
- [20] M. A. Otaduy and M. C. Lin. Sensation preserving simplification for haptic rendering. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*, pages 543–553, 2003.
- [21] M. A. Otaduy and M. C. Lin. A perceptually-inspired force model for haptic texture rendering. *Proc. of Symposium APGV*, 2004.
- [22] M. Peshkin and J. E. Colgate. Cobots. *Industrial Robot*, 26(5):335–341, 1999.
- [23] D. Ruspinii and O. Khatib. A framework for multi-contact multi-body dynamic simulation and haptic display. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.
- [24] J. K. Salisbury. Making graphics physically tangible. *Communications of the ACM*, 42(8), 1999.
- [25] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. *Proc. of ACM SIGGRAPH*, pages 409–416, 2001.
- [26] J. Siira and D. K. Pai. Haptic textures – a stochastic approach. *Proc. of IEEE International Conference on Robotics and Automation*, pages 557–562, 1996.
- [27] M. Wan and W. A. McNeely. Quasi-static approximation for 6 degrees-of-freedom haptic rendering. *Proc. of IEEE Visualization*, pages 257–262, 2003.

# A Unified Treatment of Elastostatic Contact Simulation for Real Time Haptics

Doug L. James\* and Dinesh K. Pai\*†

University of British Columbia

## Abstract

We describe real-time, physically-based simulation algorithms for haptic interaction with elastic objects. Simulation of contact with elastic objects has been a challenge, due to the complexity of physically accurate simulation and the difficulty of constructing useful approximations suitable for real time interaction. We show that this challenge can be effectively solved for many applications. In particular global deformation of linear elastostatic objects can be efficiently solved with low run-time computational costs, using pre-computed Green's functions and fast low-rank updates based on *Capacitance Matrix Algorithms*. The capacitance matrices constitute exact force response models, allowing contact forces to be computed much faster than global deformation behavior. *Vertex pressure masks* are introduced to support the convenient abstraction of localized scale-specific point-like contact with an elastic and/or rigid surface approximated by a polyhedral mesh. Finally, we present several examples using the CyberGlove™ and PHANTOM™ haptic interfaces.

## 1 Introduction

Discrete linear elastostatic models (LEMs) are important physically-based elastic primitives for computer haptics because they admit a very high-degree of precomputation, or “numerical compression” [1], in a way that affords cheap force response models suitable for force feedback rendering of stiff elastic objects during continuous contact. The degree of useful precomputation is quite limited for many types of nonlinear and/or dynamical elastic models, but LEMs are an exception, mainly due to the precomputability of time-independent *Green's functions* (GFs) and the applicability of linear superposition principles. Intuitively, GFs form a basis for describing all possible deformations of a LEM. Thus, while LEMs form a relatively simple class of elastic models in which geometric and material linearities are an ultimate limitation, the fact that the model is linear is also a crucial enabling factor. We conjecture that LEMs will remain one of the best runtime approximations of stiff elastic models for simulations requiring stable high-fidelity force feedback.

A central idea for LEMs in computer haptics is the formulation of the boundary value problem (BVP) solution in terms of suitable precomputed GFs using *Capacitance Matrix Algorithms* (CMAs). Derived from the Sherman-Morrison-Woodbury formula for low-rank updating of matrix inverses (and factorizations), CMAs have a long history in linear algebra [30, 16], where they have been commonly used for static reanalysis [22], to efficiently solve LEM contact mechanics problems [12, 25] and more recently for interactive LEM simulations [6, 21].

For computer haptics, a fundamental reason for choosing to compute the LEM elasticity solution using a CMA formulation, is that

the *capacitance matrix*<sup>1</sup> is the main quantity of interest: it is the *compliance matrix* which relates the force feedback response to the imposed contact displacements. Also, the precomputation of GFs effectively decouples the global deformation and force response calculations, so that the capacitance matrix can be extracted from the GFs at no extra cost; this is the fundamental mechanism by which a haptic interface can efficiently interact with a LEM of very large complexity. The user can feel no difference between the force response of the complete system and the capacitance matrix, because none exists. Lastly, CMAs are direct matrix solvers whose deterministic operation count is appealing for real time applications.

The second part of this paper addresses the special case of point-like interaction. It has long been recognized that point contact is a convenient abstraction for haptic interactions, and the PHANTOM™ haptic interface is a testament to that fact. While it is possible to consider the contact area to be truly a point for rigid models, infinite contact pressures are problematic for elastic models and tractions need to be distributed over finite surface areas. We propose to do this efficiently by introducing nodal traction distribution masks which address at least two core issues. First, having a point contact with force distributed over a finite area is somewhat contradictory, and the traction distribution is effectively an underdetermined quantity without any inherent spatial scale. This is resolved by treating the contact as a single displacement constraint whose traction distribution enters as a user (or manipulator) specified parameter. The distribution of force on the surface of the model can then be consistently specified in a fashion which is independent of the scale of the mesh. Second, given the model is discrete, special care must be taken to ensure a sufficiently regular force response on the surface, since irregularities are very noticeable during sliding contact motions. By suitably interpolating nodal traction distributions, displacement constraints can be imposed which are consistent with regular contact forces for numerous discretizations.

The remainder of the paper is organized as follows. After a discussion of related work (§2), the notation and definitions for a wide class of linear elastostatic models used herein are given (§3). Fast CMAs for general BVP solution using precomputed GFs of a particular reference BVP type are described in detail in §4. Particular attention is given to the role of the capacitance matrix for the construction of globally consistent stiffness matrices for use in local haptic buffer models (§5). The special case of point-like contacts are considered in detail, and we introduce (runtime computable) vertex masks for haptic presentation of surface stiffness (§6). Some results are given for our implementations (§7) followed by conclusions and a discussion of future work (§8).

## 2 Related Work

While a significant amount of work has been done on interactive simulation of physically-based elastic models, e.g., in computer

\*Institute of Applied Mathematics

†Dept. of Computer Science, 2366 Main Mall, Vancouver, B.C., V6T 1Z4, Canada, {djames|pai}@cs.ubc.ca

<sup>1</sup>The term “capacitance” is due to historical convention [16].

graphics [14], only a relatively small subset of work has addressed the simulation requirements of force feedback computer haptics. Much of the computer haptics literature on deformable models has been concerned with surgical simulation, while our focus is more general. It is not our intent to survey all related haptic work here, but simply to mention some relevant work combining kinesthetic force feedback with elastic models.

## 2.1 Elastostatic Models

There are several instances in the literature of real time simulation of linear elastostatic models based on precomputed GFs methods. These models were used because of their low runtime costs, and desirable force feedback properties. To date only polygonal models based on FEM and BEM discretizations have been considered, although other variants are possible within the linear systems description presented in the following sections.

Of particular relevance is the work done by researchers at INRIA who have made extensive use of a real time elastostatic FEM model for liver related surgical simulations [6, 5, 9]. During a precomputation phase they have used condensation [35] as well as iterative methods [8] to compute displacement responses due to unit forces applied to vertices on the “free” boundary. At run time, they solve a small system of equations to determine the correct superposition of responses to satisfy the applied surface constraints, which may be identified as a case of the capacitance matrix approach. We note that the point-like contact approach used in [8] could benefit from pressure mask concepts (§6). Recently, they have used anisotropic material properties for the liver [29]. Other groups have also used the precomputed elastostatic FEM approach of [6] for surgical simulation, e.g., the KISMET surgical simulator [23] incorporates precomputed models to provide high-fidelity haptic force feedback.

A limitation of the GF precomputation strategy is that incremental runtime modifications of the model require extra runtime computations. While it may be too costly for interactive applications, this can also be efficiently performed using low-rank updating techniques such as for static reanalysis in the engineering community [22]. For surgical simulation, a practical approach has been to use a hybrid domain decomposition approach in which a more easily modified dynamic model is used in a smaller region to be cut [9, 17].

Finally, the authors presented an interactive animation technique in [21] which combined precomputed GFs of boundary element models with matrix-updating techniques for fast boundary value problem (BVP) solution. Although computer haptics was an intended application, no force feedback implementation was mentioned. This paper generalizes that approach with a broad GF-based linear systems framework that subsumes the discretization issues of both [21] and the FEM approaches of [6, 8].

## 2.2 Other Elastic Models

Various approaches have been taken to simulate dynamic elastic models, by addressing commonly encountered difficulties such as the computational complexity of time-stepping 3D models, and numerical time integration issues, e.g., stiffness and stability. In order to meet the intense demands of force feedback rendering rates, most have opted for a multirate simulation approach. It is worth noting explicitly that methods for interactively simulating soft dynamic objects are in many ways complementary to the CMA methods presented here for simulating relatively stiff LEM. A few notable examples are now mentioned.

*Local buffer models* were presented by Balaniuk in [2] for rendering forces computed by e.g., deformable object, simulators which can not deliver forces at fast rendering rates. An application of the technique was presented for a virtual echographic exam

training simulator in [10]. While we do not use the same approach here, the local buffer model concept is related to our capacitance matrix method for force computation.

Astley and Hayward [1] introduced an approximation for linear viscoelastic FEM models that also exploits linearity, in this case by precomputing multilevel Norton equivalents for the system’s stiffness matrix. By doing so, haptic interaction is possible by employing an explicit multirate integration scheme wherein a model associated with the contact region is integrated at a higher rate than the remaining coarser model.

Çavuşoğlu and Tendick [7] also use a multirate approach. Balanced model reduction of a linearization of their nonlinear dynamical lumped element model is used to suggest a spatially localized dynamic model approximation for force feedback rendering. While promising, the example considered is a very special case of a supported model, and it is unclear how the local model would be derived in more generic geometric cases, as well as in the presence of nonlocal influences such as for multiple changing contacts, e.g., with surgical tools.

Debunne et al. [11] presented a space-time approach for simulating a hierarchical multirate dynamic linear-strain model. Zhuang and Canny [34] use a dynamic lumped finite element model exhibiting nonlinear (Green’s) strain. It is capable of being time-stepped at graphics frame rates for sufficiently soft objects using an explicit integration scheme. Interactive simulation of dynamic elastic models exclusively for superquadric shapes was considered by Ramanathan and Metaxas [31]. Volumetric and voxel-based modeling approaches for surgical simulation have also been considered, e.g., by Gibson et al. [13].

## 3 Linear Elastostatic Boundary Model Preliminaries

Linear elastostatic objects are essentially three-dimensional linear springs, and as such they are useful modeling primitives for physically-based simulations. The unfamiliar reader might consult a suitable background reference before continuing [3, 18, 35, 4, 21].

In this section, background material for a generic discrete GF description for a variety of precomputed linear elastostatic models is provided. Conceptually, GFs form a basis for describing all possible deformations of a LEM subject to a certain class of constraints. This is useful because it (1) provides a common language to describe all discrete LEMs, (2) subsumes extraneous discretization details by relating only physical quantities, and (3) clarifies the generality of the force feedback algorithms described later.

Another benefit of using GFs is that they provide an efficient means for exclusively simulating only boundary data (displacements and forces) if desired. While it is possible to simulate various internal volumetric quantities (see §3.5), simulating only boundary data involves less computation. This is sufficient since we are primarily concerned with interactive simulations that impose surface constraints and provide feedback via surface deformation and contact forces.

### 3.1 Geometry and Material Properties

Given that the fast solution method is based on linear systems principles, essentially any linear elastostatic model with physical geometric and material properties is admissible. We shall consider models in three dimensions, although many arguments also apply to lower dimensions. Suitable models would of course include bounded volumetric objects with various internal material properties, as well as special subclasses such as thin plates and shells. Since only a boundary or interface description is utilized for specifying user interactions, other exotic geometries may also be easily



considered such as semi-infinite domains, exterior elastic domains, or simply any set of parametrized surface patches with a linear response. Similarly, numerous representations of the surface and associated displacement shape functions are also possible, e.g., polyhedral, NURBS or subdivision surfaces [32].

### 3.2 Nodal Displacements and Traction

Let the undeformed boundary be denoted by  $\Gamma$ . The change in shape of the surface is described by the surface *displacement field*  $\mathbf{u}(\mathbf{x})$ ,  $\mathbf{x} \in \Gamma$ , and the surface force distribution is called the *traction<sup>2</sup> field*  $\mathbf{p}(\mathbf{x})$ ,  $\mathbf{x} \in \Gamma$ . We will assume that each surface field is parametrized by  $n$  nodal variables (see Figure 1), so that the discrete displacement and traction vectors are

$$\mathbf{u} = [\mathbf{u}_1, \dots, \mathbf{u}_n]^T \quad (1)$$

$$\mathbf{p} = [\mathbf{p}_1, \dots, \mathbf{p}_n]^T, \quad (2)$$

respectively, where each nodal value is a vector in  $\mathbb{R}^3$ . This description admits a very large class of surface displacement and traction distributions.

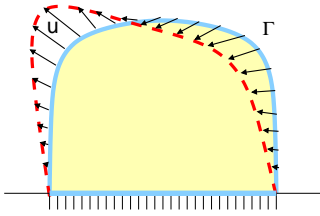


Figure 1: *Illustration of discrete nodal displacements*  $\mathbf{u}$  defined at vertices on the undeformed boundary  $\Gamma$  (solid blue line), that result in a deformation of the surface (to dashed red line). Although harder to illustrate, a similar definition exists for the traction vector,  $\mathbf{p}$ .

In order to relate traction distributions to forces, define a scalar function space,  $\mathcal{L}$ , on the model's boundary:

$$\mathcal{L} = \text{span} \{ \phi_j(\mathbf{x}), \quad j = 1 \dots n, \quad \mathbf{x} \in \Gamma \}, \quad (3)$$

where  $\phi_j(\mathbf{x})$  is a scalar basis function associated with the  $j^{\text{th}}$  node. The continuous traction field is then a 3-vector function with components in  $\mathcal{L}$ ,

$$\mathbf{p}(\mathbf{x}) = \sum_{j=1}^n \phi_j(\mathbf{x}) \mathbf{p}_j, \quad (4)$$

The force on any surface area is equal to the integral of  $\mathbf{p}(\mathbf{x})$  on that area. It then follows that the nodal force associated with any nodal traction is given by

$$\mathbf{f}_j = a_j \mathbf{p}_j \quad \text{where} \quad a_j = \int_{\Gamma} \phi_j(\mathbf{x}) d\Gamma_{\mathbf{x}} \quad (5)$$

defines the area associated with the  $j^{\text{th}}$  node.

Our implementation uses linear boundary element models, for which the nodes are vertices of a closed triangle mesh. The mesh is modeled as a Loop subdivision surface [24] to conveniently obtain multiresolution models for rendering as well as uniformly parameterized surfaces ideal for BEM discretization and deformation depiction. The displacement and traction fields have convenient vertex-based descriptions

$$\mathbf{u}_j = \mathbf{u}(\mathbf{x}_j), \quad \mathbf{p}_j = \mathbf{p}(\mathbf{x}_j),$$

<sup>2</sup>Surface traction describes force per unit area.

where  $\mathbf{x}_j \in \Gamma$  is the  $j^{\text{th}}$  vertex. The traction field is a piecewise linear function, and  $\phi_j(\mathbf{x})$  represents a “hat function” located at the  $j^{\text{th}}$  vertex with  $\phi_j(\mathbf{x}_j) = 1$ . Given our implementation, we shall often refer to node and vertex interchangeably.

### 3.3 Discrete Boundary Value Problem (BVP)

At each step of the simulation, a discrete BVP must be solved which relates specified and unspecified nodal values, e.g., to determine deformation and feedback forces. Without loss of generality, it shall be assumed that either position or traction constraints are specified at each boundary node, although this can be extended to allow mixed conditions, e.g., normal displacement and tangential tractions. Let nodes with prescribed displacement or traction constraints be specified by the mutually exclusive index sets  $\Lambda_u$  and  $\Lambda_p$ , respectively, so that  $\Lambda_u \cap \Lambda_p = \emptyset$  and  $\Lambda_u \cup \Lambda_p = \{1, 2, \dots, n\}$ . In order to guarantee an equilibrium constraint configuration we will require that there is at least one displacement constraint, i.e.,  $\Lambda_u \neq \emptyset$ . We shall refer to the  $(\Lambda_u, \Lambda_p)$  pair as the *BVP type*.

Typical boundary conditions for a force feedback loop consist of specifying some (compactly supported) displacement constraints in the area of contact, with “free” boundary conditions (zero traction) and other (often zero displacement) support constraints outside the contact zone. The solution to (7) yields the rendered contact forces and surface deformation.

Denote the unspecified and complementary specified nodal variables by

$$\mathbf{v}_j = \begin{cases} \mathbf{p}_j : j \in \Lambda_u \\ \mathbf{u}_j : j \in \Lambda_p \end{cases} \quad \text{and} \quad \bar{\mathbf{v}}_j = \begin{cases} \bar{\mathbf{u}}_j : j \in \Lambda_u \\ \bar{\mathbf{p}}_j : j \in \Lambda_p \end{cases}, \quad (6)$$

respectively. By linearity of the discrete elastic model, there formally exists a linear relationship between all nodal boundary variables

$$0 = \mathbf{A}\mathbf{v} + \bar{\mathbf{A}}\bar{\mathbf{v}} = \mathbf{A}\mathbf{v} - \mathbf{z} \quad (7)$$

where the BVP system matrix  $\mathbf{A}$  and its complementary matrix  $\bar{\mathbf{A}}$  are, in general, dense block  $n$ -by- $n$  matrices [18]. Body force terms associated with other phenomena, e.g., gravity, have been omitted for simplicity, but can be included since they only add an extra contribution to the  $\mathbf{z}$  term.

A key relationship between BVP system matrices  $(\mathbf{A}, \bar{\mathbf{A}})$  of different BVP types  $(\Lambda_u, \Lambda_p)$  is that they are related by exchanges of corresponding block columns, e.g.,  $(\mathbf{A}_{ij}, \bar{\mathbf{A}}_{ij})$ , and therefore small changes to the BVP type result in low-rank changes to the BVP system matrices (see §4.2.1).

While the boundary-only system matrices in (7) could be constructed explicitly, e.g., via condensation for FEM models [35] or using a boundary integral formulation (see next section), it need not be in practice. The discrete integral equation in Equation 7 is primarily a common starting point for later definition of GFs and derivation of the CMA, while GFs may be generated with any convenient numerical method, or even robotically scanned from real objects [28].

### 3.4 Example: Boundary Element Models

A simple closed-form definition of  $(\mathbf{A}, \bar{\mathbf{A}})$  is possible for models discretized with the boundary element method (BEM) [4, 21]; BEM discretizations are possible for models with homogeneous and isotropic material properties. The surface-based nodal quantities are related by the dense linear block matrix system

$$0 = \mathbf{H}\mathbf{u} - \mathbf{G}\mathbf{p} = \sum_{j=1}^n \mathbf{h}_{ij} \mathbf{u}_j - \sum_{j=1}^n \mathbf{g}_{ij} \mathbf{p}_j \quad (8)$$

where  $G$  and  $H$  are  $n$ -by- $n$  block matrices, with each matrix element,  $g_{ij}$  or  $h_{ij}$ , a 3-by-3 influence matrix with known expressions [4]. In this case, the  $j^{th}$  block columns of  $A$  and  $\bar{A}$  may be identified as column exchanged variants of  $G$  and  $H$ :

$$A_{:j} = \begin{cases} -G_{:j} & : j \in \Lambda_u \\ H_{:j} & : j \in \Lambda_p \end{cases} \quad (9)$$

$$\bar{A}_{:j} = \begin{cases} H_{:j} & : j \in \Lambda_u \\ -G_{:j} & : j \in \Lambda_p \end{cases} \quad (10)$$

While we use BEM models for our implementation, we reiterate that the CMA is independent of the method used to generate the GFs (explained next).

### 3.5 Fast BVP Solution with Green's Functions

GFs of a single BVP type provide an economical means for solving (7) for that BVP, and when combined with the CMA (§4) will also be useful for solving other BVP types. From (7), the general solution of a BVP type  $(\Lambda_u, \Lambda_p)$  may be expressed in terms of discrete GFs<sup>3</sup> as

$$v = \Xi \bar{v} = \sum_{j=1}^n \xi_j \bar{v}_j = \sum_{j \in \Lambda_u} \xi_j \bar{u}_j + \sum_{j \in \Lambda_p} \xi_j \bar{p}_j, \quad (11)$$

where the discrete GFs of the BVP system are the block column vectors

$$\xi_j = - (A^{-1} \bar{A})_{:j} \quad (12)$$

and

$$\Xi = -A^{-1} \bar{A} = [\xi_1 \xi_2 \cdots \xi_n]. \quad (13)$$

Equation (11) may be taken as the definition of the discrete GFs (and even (7)), since it is clear that the  $j^{th}$  GF simply describes the linear response of the system to the  $j^{th}$  node's specified boundary value,  $\bar{v}_j$ . Once the GFs have been computed for one BVP type, that class of BVPs may be solved easily using (11). An attractive feature for interactive applications is that the entire solution can be obtained in  $18ns$  flops<sup>4</sup> if only  $s$  boundary values (BV) are nonzero (or have changed since the last time step). Temporal coherence may also be exploited by considering the effect of individual changes in components of  $\bar{v}$  on the solution  $v$ .

Further leveraging linear superposition, each GF system response may be augmented with other additional information in order to simulate other precomputable quantities. Volumetric stress, strain and displacement data may also be simulated at preselected locations. Applications could use this to monitor stresses and strains to determine, e.g., if fracture occurs or that a nonlinear correction should be computed.

### 3.6 Precomputation of Green's Functions

Since the GFs for a single BVP type only depend on geometric and material properties of the deformable object, they may be precomputed for use in a simulation. This provides a dramatic speed-up for simulation by determining the deformation basis (the GFs) ahead of time. While this is not necessary a huge amount of work (see Table 2), the principal benefits for interactive simulations are the availability of the GF elements via cheap look-up table operations, as well as the elimination of redundant runtime computation when

<sup>3</sup>Note on GF terminology: we are concerned with discrete numerical approximations of continuous GFs, however for convenience these GF vectors will simply be referred to as GFs.

<sup>4</sup>counting both + and \*

computing solutions, e.g., using a haptic device to grab a vertex of the model and move it around simply renders a single GF.

Once a set of GFs for a LEM are precomputed, the overall stiffness can be varied at runtime by scaling BVP forces accordingly, however changes in compressibility and internal material distributions do require recomputation. In practice it is only necessary to compute the GF corresponding to nodes which may have changing or nonzero boundary values during the simulation.

## 4 Fast Global Deformation using Capacitance Matrix Algorithms (CMAs)

This section presents an algorithm for using the precomputed GFs of a relevant *Reference BVP* (RBVP) type to efficiently solve other BVP types. With an improved notation and emphasis on computer haptics, this section unifies and extends the approaches presented in [21] exclusively for BEM models, and for FEM models in, e.g., [6], in a way that is applicable to all LEMs regardless of discretization, or origin of GFs [28]. Haptic applications are considered in §5.

### 4.1 Reference Boundary Value Problem (RBVP) Choice

A key step in the GF precomputation process is the initial identification of a RBVP type, denoted by  $(\Lambda_u^0, \Lambda_p^0)$ , that is representative of the BVP types arising during simulations. For interactions with an exposed free boundary, a common choice is to have the uncontacted model attached to a rigid support as shown in Figure 2. The  $n$ -by- $n$  block system matrices associated with the RBVP are identified with a subscript as  $A_0$  and  $\bar{A}_0$ , and the corresponding GFs are hereafter always denoted by  $\Xi$ .

Note that the user's choice of RBVP type determines which type of nodal constraints (displacement of traction) are commonly specified (in order to define  $\Xi$ ), but is independent of the actual numerical boundary values  $\bar{v}$  used in practice. For example, there are no requirements that certain boundary values are zero, although this results in fewer summations (see (11)).

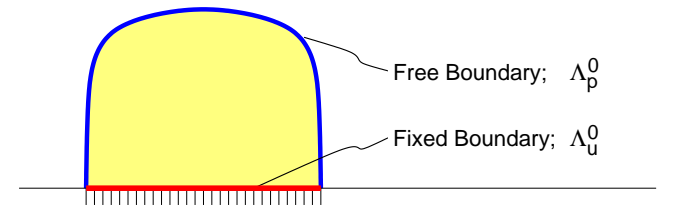


Figure 2: *Reference Boundary Value Problem (RBVP) example:* The RBVP associated with a model attached to a flat rigid support is shown with boundary regions having fixed ( $\Lambda_u^0$ ) or free ( $\Lambda_p^0$ ) nodal constraints indicated. A typical simulation would impose contacts on the free boundary via displacement constraints with the CMA.

### 4.2 Capacitance Matrix Algorithm (CMA) for BVP Solution

Precomputed GFs speed-up the solution to the RBVP, but they can also dramatically reduce the amount of work required to solve related BVP when used in conjunction with CMAs. This section describes the CMA and presents the derivation of related formulae.

#### 4.2.1 Relevant Formulae

Suppose the constraint-type changes, e.g., displacement $\leftrightarrow$ traction, with respect to the RBVP at  $s$  nodes specified by the list of nodal indices  $S = \{S_1, S_2, \dots, S_s\}$ . As mentioned earlier, it follows from (6) and (7) that the new BVP system matrices  $(A, \bar{A})$  are related to those of the RBVP  $(A_0, \bar{A}_0)$  by  $s$  block column swaps. This may be written as

$$A = A_0 + (\bar{A}_0 - A_0) EE^T \quad (14)$$

$$\bar{A} = \bar{A}_0 + (A_0 - \bar{A}_0) EE^T \quad (15)$$

where  $E$  is an  $n$ -by- $s$  block matrix

$$E = [I_{S_1} I_{S_2} \dots I_{S_s}]$$

containing columns of the  $n$ -by- $n$  identity block matrix,  $I$ , specified by the list of updated nodal indices  $S$ . Postmultiplication by  $E$  extracts columns specified by  $S$ . Throughout,  $E$  is used to write sparse matrix operations using dense data, e.g.,  $\Xi$ , and like the identity matrix, it should be noted that there is no cost involved in multiplication by  $E$  or its transpose.

Since the BVP solution is

$$v = A^{-1}z = -A^{-1}\bar{A}\bar{v}, \quad (16)$$

substituting (15) for  $\bar{A}$  and the Sherman-Morrison-Woodbury formula [15] for  $A^{-1}$  (using the GF definition  $\Xi = -A_0^{-1}\bar{A}_0$ ),

$$A^{-1} = A_0^{-1} + (I + \Xi)E(-E^T\Xi E)^{-1}E^T A_0^{-1}, \quad (17)$$

into (16), leads directly to an expression for the solution in terms of the precomputed GFs<sup>5</sup>. The resulting *capacitance matrix formulae* are

$$v = \underbrace{v^{(0)}}_{n \times 1} + \underbrace{(E + (\Xi E))}_{n \times s} \underbrace{C^{-1}}_{s \times s} \underbrace{E^T v^{(0)}}_{s \times 1} \quad (18)$$

where  $C$  is the  $s$ -by- $s$  *capacitance matrix*, a negated submatrix of  $\Xi$ ,

$$C = -E^T \Xi E, \quad (19)$$

and  $v^{(0)}$  is the response of the RBVP system to  $z = -\bar{A}\bar{v}$ ,

$$v^{(0)} = A_0^{-1}z = [\Xi(I - EE^T) - EE^T]\bar{v}. \quad (20)$$

#### 4.2.2 Algorithm for BVP Solution

With  $\Xi$  precomputed, formulae (18)-(20) immediately suggest an algorithm given that only simple manipulations of  $\Xi$  and inversion of the smaller capacitance submatrix are required. An algorithm for computing *all* components of  $v$  is as follows:

- For each new BVP type (with a different  $C$  matrix) encountered, construct and temporarily store  $C^{-1}$  (or LU factors) for subsequent use.
- Construct  $v^{(0)}$ .
- Extract  $E^T v^{(0)}$  and apply the capacitance matrix inverse to it,  $C^{-1}(E^T v^{(0)})$ .
- Add the  $s$  column vectors  $(E + (\Xi E))$  weighted by  $C^{-1}(E^T v^{(0)})$  to  $v^{(0)}$  for the final solution  $v$ .

<sup>5</sup>Similarly from [21] with  $\delta A_S = (\bar{A}_0 - A_0)E$ .

#### 4.2.3 Complexity Issues

Given  $s$  nonzero boundary values, each new capacitance matrix LU factorization involves at most  $\frac{2}{3}s^3$  flops, after which each subsequent solve involves approximately  $18ns$  flops ( $s \ll n$ ). This is particularly attractive when  $s \ll n$  is small, such as often occurs in practice with localized surface contacts.

An important feature of the CMA for interactive methods is that it is a direct matrix solver with a deterministic operation count. It is therefore possible to predict the runtime cost associated with each matrix solve and associated force feedback subcomputations (see §5), thus making CMAs predictable for real-time computations.

#### 4.3 Selective Deformation Computation

A major benefit of the CMA direct BVP solver is that it is possible to just evaluate selected components of the solution vector at runtime, with the total computing cost proportional to the number of components desired. This is a key enabling feature for force feedback where, e.g., contact forces are desired at different rates than the geometric deformations. Selective evaluation would also be useful for optimizing (self) collision detection queries, avoiding simulation of occluded or undesired portions of the model, as well as rendering adaptive level of detail representations.

In general, any subset of solution components may be determined at a smaller cost than computing  $v$  entirely. Let the solution be desired at nodes specified by the set of indices  $D$ , with the desired components of  $v$  extracted by  $E_D^T$ . Using (18), the selected solution components may be evaluated as

$$E_D^T v = E_D^T v^{(0)} + E_D^T (E + (\Xi E)) C^{-1} E^T v^{(0)}$$

using only  $\mathcal{O}(s^2 + s|D|)$  operations. The case where  $S = D$  is especially important for force feedback and is discussed exclusively in the following section.

### 5 Capacitance Matrices as Local Buffer Models

For force feedback enabled simulations in which user interactions are modeled as displacement constraints applied to an otherwise free boundary, the capacitance matrix has a very important role: it constitutes an exact contact force response model by describing the compliance of the contact zone. Borrowing terminology from [2], we say that the capacitance matrix can be used as a local buffer model. While the capacitance matrix is used in §4.2.2 to determine the linear combination of GFs required to solve a particular BVP and reconstruct the global deformation, it also has the desirable property that it effectively decouples the global deformation calculation from that of the local force response. The most relevant benefit for haptics is that the local contact force response may be computed at a much faster rate than the global deformation.

#### 5.1 Capacitance Matrix Local Buffer Model

From (18), the  $S$  components of the solution  $v$  are

$$\begin{aligned} E^T v &= E^T \left[ v^{(0)} + (E + (\Xi E)) C^{-1} E^T v^{(0)} \right] \\ &= E^T v^{(0)} + \underbrace{(E^T E)}_I C^{-1} E^T v^{(0)} + \underbrace{(E^T \Xi E)}_{-C} C^{-1} E^T v^{(0)} \\ &\downarrow \\ &= E^T v^{(0)} + C^{-1} E^T v^{(0)} - E^T v^{(0)} \\ &= C^{-1} (E^T v^{(0)}). \end{aligned} \quad (21)$$

Consider the situation, which naturally arises in haptic interactions, in which the only nonzero constraints are updated displacement constraints, i.e.,

$$\bar{\mathbf{v}} = \mathbf{E}\mathbf{E}^T\bar{\mathbf{v}} \Rightarrow \mathbf{v}^{(0)} = -\bar{\mathbf{v}} \quad (\text{using (20)}). \quad (22)$$

In this case, the capacitance matrix completely characterizes the local contact response, since (using (22) in (21))

$$\mathbf{E}^T\mathbf{v} = -\mathbf{C}^{-1}\mathbf{E}^T\bar{\mathbf{v}}. \quad (23)$$

This in turn parametrizes the global response since these components (not in  $\mathbf{S}$ ) are

$$\begin{aligned} (\mathbf{I} - \mathbf{E}\mathbf{E}^T)\mathbf{v} &= (\mathbf{I} - \mathbf{E}\mathbf{E}^T) \left[ \mathbf{v}^{(0)} + (\mathbf{E} + (\Xi\mathbf{E})) \mathbf{C}^{-1}\mathbf{E}^T\mathbf{v}^{(0)} \right] \\ &= (\mathbf{I} - \mathbf{E}\mathbf{E}^T)(\Xi\mathbf{E})(\mathbf{E}^T\mathbf{v}) \end{aligned} \quad (24)$$

where we have used (23) and the identity  $(\mathbf{I} - \mathbf{E}\mathbf{E}^T)\mathbf{E} = 0$ . Such properties allow the capacitance matrix and  $\Xi$  to be used to derive efficient local models for surface contact.

For example, given the specified contact zone displacements

$$\mathbf{u}_S = \mathbf{E}^T\bar{\mathbf{v}}, \quad (25)$$

the resulting tractions are

$$\mathbf{p}_S = \mathbf{E}^T\mathbf{v} = -\mathbf{C}^{-1}(\mathbf{E}^T\bar{\mathbf{v}}) = -\mathbf{C}^{-1}\mathbf{u}_S, \quad (26)$$

and the rendered contact force is

$$\mathbf{f} = \mathbf{a}_S^T \mathbf{p}_S = (-\mathbf{a}_S^T \mathbf{C}^{-1}) \mathbf{u}_S = \mathbf{K}_S \mathbf{u}_S, \quad (27)$$

where  $\mathbf{K}_S$  is the effective stiffness of the contact zone used for force feedback rendering,

$$\mathbf{a}_S = (a_{S1}, a_{S2}, \dots, a_{S_s})^T \otimes \mathbf{I}_3 \quad (28)$$

represents nodal areas (5), and  $\mathbf{I}_3$  is the scalar 3-by-3 identity matrix. A similar expression may be obtained for torque feedback. The visual deformation corresponding to solution components outside the contact zone is then given by (24) using  $\mathbf{p}_S = \mathbf{E}^T\mathbf{v}$ .

## 5.2 Example: Single Displacement Constraint

A simple case, which will be discussed in much greater detail in §6, is that of imposing a displacement constraint on single a node  $k$  which otherwise had a traction constraint in the RBVP<sup>6</sup>. The new BVP therefore has only a single constraint switch with respect to the RBVP, and so  $s = 1$  and  $\mathbf{S} = \{k\}$ . The capacitance matrix here is just  $\mathbf{C} = -\Xi_{kk}$  so that the  $k^{th}$  nodal values are related by

$$\mathbf{p}_k = -\mathbf{C}^{-1}\bar{\mathbf{u}}_k = (\Xi_{kk})^{-1}\bar{\mathbf{u}}_k \quad \text{or} \quad \bar{\mathbf{u}}_k = \Xi_{kk}\mathbf{p}_k.$$

The capacitance matrix can generate the force response,  $\mathbf{f} = \mathbf{a}_k\mathbf{p}_k$ , required for haptics in  $\mathcal{O}(1)$  operations, and for graphical feedback the corresponding global solution is  $\mathbf{v} = \xi_k\mathbf{p}_k$ .

## 5.3 Force Feedback for Multiple Displacement Constraints

When multiple force feedback devices are interacting with the model by imposing displacement constraints, the force and stiffness felt by each device are tightly coupled in equilibrium. For example, the stiffness felt by the thumb in Figure 3 will depend on how the other fingers are supporting the object. For multiple contacts like this, the capacitance matrix again provides an efficient force response model for haptics. Without presenting the equations in detail, we shall just mention that the force responses for each of the contact patches can be derived from the capacitance matrix in a manner similar to equations (25)-(28).

<sup>6</sup>This case occurs, for instance, when the tip of a haptic device comes into contact with the free surface of an object.

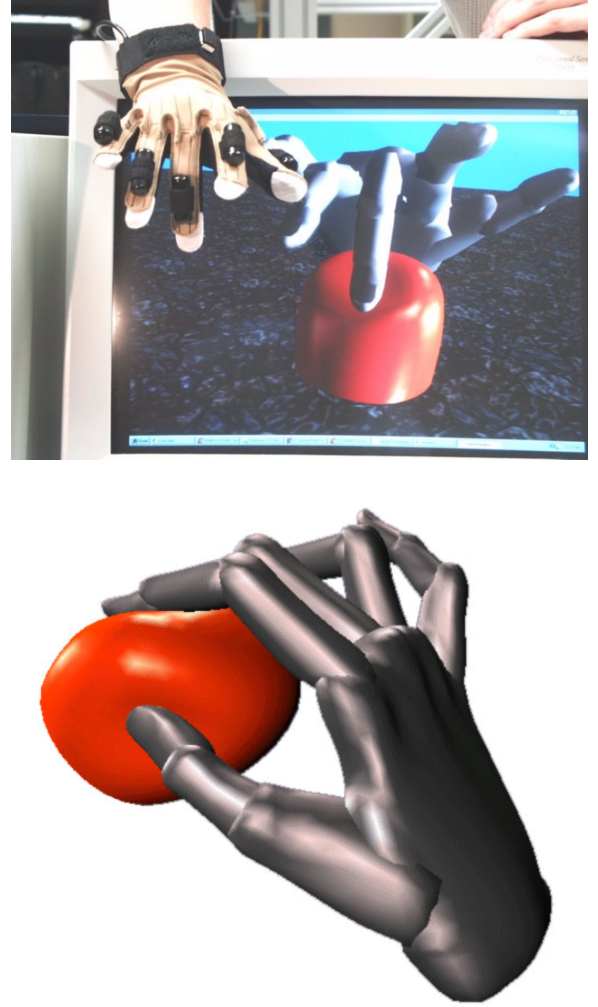


Figure 3: *Grasping simulation*: Using a CyberTouch data input device from Virtual Technologies Inc. (Top), a virtual hand (Bottom) was used to deform an elastostatic BEM model with approximately 900 surface degrees of freedom (dof) at graphical frame rates ( $> 30$  FPS) on a personal computer. The capacitance matrix algorithm was used to impose displacement constraints on an otherwise free boundary, often updating over 100 dof per frame. While force feedback was not present, the capacitance matrices computed could also have been used to render contact forces at a rate higher than that of the graphical simulation.

## 6 Surface Stiffness Models for Point-like Contact

The second part of this paper concerns a special class of boundary conditions describing point-like contact interactions. Such interactions are commonly in the haptics literature for *rigid* surface models [26, 19]. Unlike their rigid counterparts, special care must be taken with elastic models to define *finite contact areas* for point-like interactions since point-like contacts defined only as single-vertex (§5.2) or nearest neighbour [8] constraints lead to mesh-related artifacts, and ambiguous interactions as the mesh is refined (see Figure 4). However, the benefit of point-like contacts comes from the convenience of the point-like parameterization of the contact and not because the contact is highly concentrated or “pin-like”. We present an approach using *vertex pressure masks* which maintains the single contact description yet distribute forces on a specified



scale. This allows point contact stiffnesses to be consistently defined as the mesh scale is refined, and provides an efficient method for force feedback rendering of forces with regular spatial variation on irregular meshes.

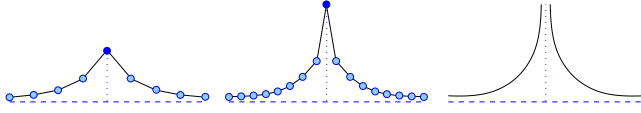


Figure 4: *Point contact must not be taken literally for elastic models:* This figure illustrates the development of a displacement singularity associated with a concentrated surface force as the continuum limit is approached. In the left image, an upward unit force applied to a vertex of a discrete elastic model results in a finite vertex displacement. As the model's mesh is refined (middle and right image), the same concentrated force load eventually tends to produce a singular displacement at the contact location, and the stiffness of any single vertex approaches zero (see Table 1). Such point-like constraints are mathematically ill-posed for linear models based on a small-strain assumption, and care must be taken to meaningfully define the interaction.

## 6.1 Vertex Pressure Masks for Distributed Point-like Contacts

In this section, the distribution of force is described using compactly-supported per-vertex pressure masks defined on the free boundary in the neighbourhood of each vertex.

### 6.1.1 Vertex Pressure Mask Definition

Scalar pressure masks provide a flexible means for modeling vector pressure distributions associated with each node. This allows a force applied at the  $i^{th}$  node to generate a traction distribution which is a linear combination of  $\{\phi_j(\mathbf{x})\}$  and not just  $\phi_i(\mathbf{x})$ .

In the continuous setting, a scalar surface density  $\rho(\mathbf{x}) : \Gamma \rightarrow \mathbb{R}$  will relate the localized contact force  $\mathbf{f}$  to the applied traction  $\mathbf{p}$  via<sup>7</sup>

$$\mathbf{p}(\mathbf{x}) = \rho(\mathbf{x})\mathbf{f}$$

which in turn implies the normalization condition

$$\int_{\Gamma} \rho(\mathbf{x}) d\Gamma \mathbf{x} = 1. \quad (29)$$

In the discrete setting, the piecewise linear surface density on  $\Gamma$  is

$$\rho(\mathbf{x}) = \sum_{j=1}^n \phi_j(\mathbf{x}) \rho_j \in \mathcal{L}, \quad (30)$$

and is parameterized by the discrete scalar vertex mask vector,

$$\rho = [\rho_1, \rho_2, \dots, \rho_n]^T.$$

Substituting (30) into (29), the discrete normalization condition satisfied becomes

$$a^T \rho = 1, \quad (31)$$

where  $a$  are the vertex areas from (5). Notice that the mask density  $\rho$  has units of  $\frac{1}{\text{area}}$ .

In practice, the vertex pressure mask  $\rho$  may be specified in a variety of ways. It could be specified at runtime, e.g., as the byproduct of a physical contact mechanics solution, or be a user specified quantity. We shall consider the case where there is a compactly

supported scalar function  $\rho(\mathbf{x})$  specified at each vertex on the free boundary. The corresponding discrete vertex mask  $\rho$  may then be defined using nodal collocation (see Figure 5),

$$\rho_j = \begin{cases} \rho(\mathbf{x}_j), & j \in \Lambda_p^0, \\ 0, & j \in \Lambda_u^0. \end{cases},$$

followed by suitable normalization,

$$\rho := \frac{\rho}{a^T \rho},$$

to ensure the satisfaction of (31).

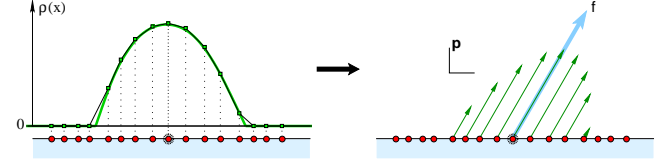


Figure 5: *Collocated scalar masks:* A direct means for obtaining a relative pressure amplitude distribution about each node, is to employ a user-specified scalar functional of the desired spatial scale. The scalar pressure mask is then given by nodal collocation (left), after which the vector traction distribution associated with a nodal point load is then computed as the product of the applied force vector and the (compactly supported) scalar mask (right).

In the following, denote the density mask for the  $i^{th}$  vertex by the  $n$ -vector  $\rho^i$ , with nonzero values being indicated by the set of masked nodal indices  $\mathcal{M}_i$ . Since the intention is to distribute force on the free boundary, masks will only be defined for  $i \in \Lambda_p^0$ . Additionally, these masks will only involve nodes on the free boundary,  $\mathcal{M}_i \subset \Lambda_p^0$ , as well as be nonempty,  $|\mathcal{M}_i| > 0$ .

### 6.1.2 Example: Spherical Mask Functionals

Spherically symmetric radially decreasing mask functionals with a scale parameter were suitable candidates for constructing vertex masks via collocation on smooth surfaces. One functional we used (see Figure 6 and 7) had linear radial dependence,

$$\rho^i(\mathbf{x}; r) = \begin{cases} 1 - \frac{|\mathbf{x} - \mathbf{x}_i|}{r}, & |\mathbf{x} - \mathbf{x}_i| < r, \\ 0, & \text{otherwise.} \end{cases},$$

where  $r$  specifies the radial scale<sup>8</sup>. The effect of changing  $r$  is shown in Figure 6.

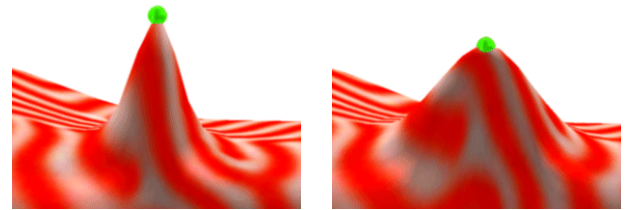


Figure 6: *Illustration of changing mask scale:* An exaggerated pulling deformation illustrates different spatial scales in two underlying traction distributions. In each case, pressure masks were generated using the linear spherical mask functional (see §6.1.2) for different values of the radius parameter,  $r$ .

<sup>7</sup>Tensor-valued masks for torque feedback can also be computed.

<sup>8</sup> $r$  may be thought of as the size of the haptic probe's tip.

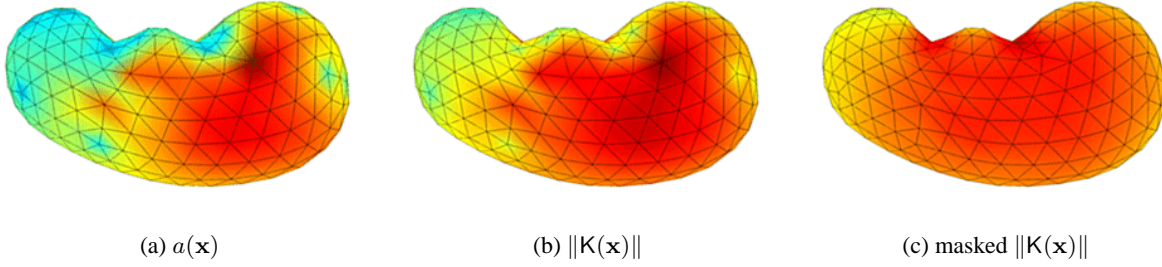


Figure 7: *Effect of pressure masks on surface stiffness:* Even models with reasonable mesh quality, such as this simple BEM kidney model, can exhibit perceptible surface stiffness irregularities when single-vertex stiffnesses are used. A plot (a) of the vertex area,  $a$ , clearly indicates regions of large (dark red) and small (light blue) triangles. In (b) the norm of the single-vertex surface stiffness,  $\|K(x)\|$ , reveals a noticeable degree of mesh-related stiffness artifacts. On the other hand, the stiffness plotted in (c) was generated using a pressure mask (collocated linear sphere functional (see §6.1.2) of radius twice the mesh’s mean edge length) and better approximates the regular force response expected of such a model. Masks essentially provide anti-aliasing for stiffnesses defined with discrete traction distributions, and help avoid “soft spots.”

## 6.2 Vertex Stiffnesses using Pressure Masks

Having consistently characterized point-like force loads using vertex pressure masks, it is now possible to calculate the stiffness of each vertex. In the following sections, these vertex stiffnesses will then be used to compute the stiffness at any point on model’s surface for haptic rendering of point-like contact.

### 6.2.1 Elastic Vertex Stiffness, $K^E$

For any single node on the free boundary,  $i \in \Lambda_p^0$ , a finite force stiffness,  $K_i \in \mathbb{R}^{3 \times 3}$ , may be associated with its displacement, i.e.,

$$\mathbf{f} = K_i \mathbf{u}_i, \quad i \in \Lambda_p^0.$$

As a sign convention, it will be noted that for any single vertex displacement

$$\mathbf{u}_i \cdot \mathbf{f} = \mathbf{u}_i \cdot (K_i \mathbf{u}_i) \geq 0, \quad i \in \Lambda_p^0$$

so that positive work is done deforming the object.

Given a force  $\mathbf{f}$  applied at vertex  $i \in \Lambda_p^0$ , the corresponding distributed traction constraints are

$$\mathbf{p}_j = \rho_j^i \mathbf{f}.$$

Since the displacement of the  $i^{th}$  vertex is

$$\mathbf{u}_i = \sum_{j \in \mathcal{M}_i} \rho_j^i \Xi_{ij} \mathbf{f},$$

therefore the effective elastic stiffness of the masked vertex is

$$K_i = K_i^E = \left( \sum_{j \in \mathcal{M}_i} \rho_j^i \Xi_{ij} \right)^{-1}, \quad i \in \Lambda_p^0. \quad (32)$$

Some examples are provided in Table 1 and Figure 7.

Therefore, in the simple case of a single masked vertex displacement constraint  $\mathbf{u}_i$ , the local force response model exactly determines the resulting force,  $\mathbf{f} = K_i \mathbf{u}_i$ , distributed in the masked region. The corresponding globally consistent solution is

$$\mathbf{v} = \zeta_i \mathbf{f} = \left( \sum_{j \in \mathcal{M}_i} \rho_j^i \xi_j \right) \mathbf{f}$$

where  $\zeta_i$  is the convolution of the GFs with the mask  $\rho$ , and characterizes the distributed force load. The limiting case of a single vertex constraint corresponds to  $\mathcal{M}_i = \{i\}$  with  $\rho_j^i = \delta_{ij}/a_i$  so that the convolution simplifies to  $\zeta_i = \xi_i/a_i$ .

Mesh Level	Vertices	$\ K\ _F$ , Single	$\ K\ _F$ , Masked
1	34	7.3	13.3
2	130	2.8	11.8
3	514	1.1	11.2

Table 1: *Vertex stiffness dependence on mesh resolution:* This table shows vertex stiffness (Frobenius) norms (in arbitrary units) at the top center vertex of the BEM model in Figure 10(a), as geometrically modeled using Loop subdivision meshes for three different levels of resolution. The stiffness corresponding to a single vertex constraint exhibits a large dependence on mesh resolution, and has a magnitude which rapidly decreases to zero as the mesh is refined. On the other hand, the stiffness generated using a vertex pressure mask (collocated linear sphere functional (see §6.1.2) with radius equal to the coarsest (level 1) mesh’s mean edge length) has substantially less mesh dependence, and quickly approaches a nonzero value.

### 6.2.2 Rigid Vertex Stiffness, $K^R$

For rigid surfaces a finite force response may be defined using an isotropic stiffness matrix,

$$K^R = k^{\text{Rigid}} I_3 \in \mathbb{R}^{3 \times 3}, \quad k^{\text{Rigid}} > 0.$$

This is useful for defining responses at position constrained vertices of a deformable model,

$$K_i = K^R, \quad i \in \Lambda_u^0, \quad (33)$$

for at least two reasons. First, while it may seem physically ambiguous to consider contacting a constrained node of a deformable object, it does allow us to define a response for these vertices without introducing other simulation dependencies, e.g., how the haptic interaction with the elastic object support is modeled. Second, we shall see in §6.3 that defining stiffness responses at these nodes is important for determining contact responses on neighbouring triangles which are not rigid.

### 6.3 Surface Stiffness from Vertex Stiffnesses

Given the vertex stiffnesses,  $\{K_i\}_{i=1}^n$ , the stiffness of any location on the surface is defined using nodal interpolation

$$K(\mathbf{x}) = \sum_{i=1}^n \phi_i(\mathbf{x}) K_i, \quad \mathbf{x} \in \Gamma, \quad (34)$$

so that  $(K(\mathbf{x}))_{kl} \in \mathcal{L}$ . Note that there are no more than three nonzero terms in the sum of (34), corresponding to the vertices of the face in contact. In this way, the surface stiffness may be continuously defined using only  $|\Lambda_p^0|$  free boundary vertex stiffnesses and a single rigid stiffness parameter,  $k^{\text{Rigid}}$ , regardless of the extent of the masks. The global deformation is then visually rendered using the corresponding distributed traction constraints.

For a point-like displacement constraint  $\bar{\mathbf{u}}$  applied at  $\mathbf{x} \in \Gamma$  on a triangle having vertex indices  $\{i_1, i_2, i_3\}$ , the corresponding global solution is

$$\mathbf{v} = \sum_{i \in \{i_1, i_2, i_3\} \cap \Lambda_p^0} \zeta_i \phi_i(\mathbf{x}) \mathbf{f}. \quad (35)$$

This may be interpreted as the combined effect of barycentrically distributed forces,  $\phi_i(\mathbf{x}) \mathbf{f}$ , applied at each of the triangle's three masked vertex nodes, which is consistent with (38).

### 6.4 Rendering with Finite Stiffness Haptic Devices

Similar to haptic rendering of rigid objects, elastic objects with stiffnesses greater than some maximum renderable magnitude (due to hardware limitations) have forces displayed as softer materials during continuous contact. This can be achieved using a *haptic vertex stiffness*,  $K_i^H$ , which is proportional to the elastic vertex stiffness,  $K_i^E$ . While the stiffnesses could all be uniformly scaled on the free boundary, this can result in very soft regions if the model has a wide range of surface stiffness. Another approach is to set

$$K_i^H = \eta_i K_i^E \quad \text{where} \quad \eta_i = \min \left( 1, \frac{\|K^R\|}{\|K_i^E\|} \right),$$

so that the elastic haptic model is never more stiff than a rigid haptic model. The surface's haptic stiffness  $K^H(\mathbf{x})$  is then determined using (34), so that  $\|K^H(\mathbf{x})\| \leq \|K^R\|, \forall \mathbf{x} \in \Gamma$ .

In accordance with force reflecting contact, the deformed elastic state corresponds to the haptic force applied at the contact location  $\mathbf{x}^C$ . This produces geometric contact configurations similar to that shown in Figure 8, where the haptic displacement  $\mathbf{u}^H$  can differ from the elastic displacement  $\mathbf{u}^E$ . The geometric deformation is determined from the applied force  $\mathbf{f}$  and equation (35). Note that when the haptic and elastic stiffnesses are equal, such as for soft materials, then so are the elastic and haptic displacements. In all cases, the generalized “god object” [36] or “surface contact point” [33] is defined as the parametric image of  $\mathbf{x}^C$  on the deformed surface.

## 7 Experimental Results

GFs were precomputed using the boundary element method (BEM) with piecewise linear boundary elements. Table 2 provides timings for the BEM precomputation stages as well as the submillisecond cost of simulating point-like deformations using GFs. Further timings of CMA suboperations are shown in Table 3, and reflect interactive performance for modest numbers of constraint type changes,  $s$ . All timings were performed using unoptimized Java code on a single processor Pentium III, 450MHz, 256MB computer with

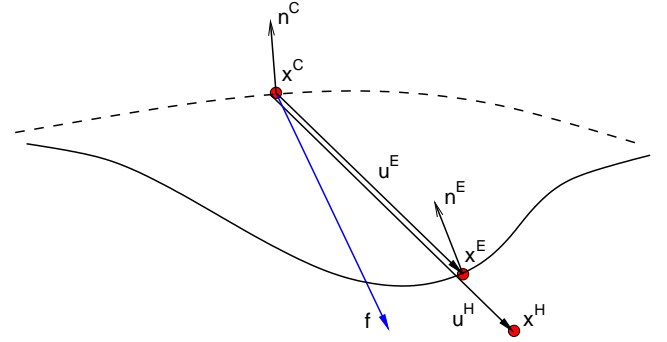


Figure 8: *Geometry of point-like contact*: The surface of the static/undeformed geometry (curved dashed line) and that of the deformed elastic model (curved solid line) are shown along with: applied force ( $\mathbf{f}$ ), static contact location ( $\mathbf{x}^C$ ), deformed elastic model contact location ( $\mathbf{x}^E$ ), haptic probe-tip location ( $\mathbf{x}^H$ ), haptic contact displacement ( $\mathbf{u}^H = \mathbf{x}^H - \mathbf{x}^C$ ), elastic contact displacement ( $\mathbf{u}^E = \mathbf{x}^E - \mathbf{x}^C$ ), static contact normal ( $\mathbf{n}^C$ ) and elastic contact normal ( $\mathbf{n}^E$ ). Once the contact is initiated by the collision detector, the sliding frictional contact can be tracked in surface coordinates at force feedback rates.

Sun's JDK 1.3 client JVM for Windows. These times can be significantly reduced by using hardware-optimized matrix libraries, and current computing hardware.

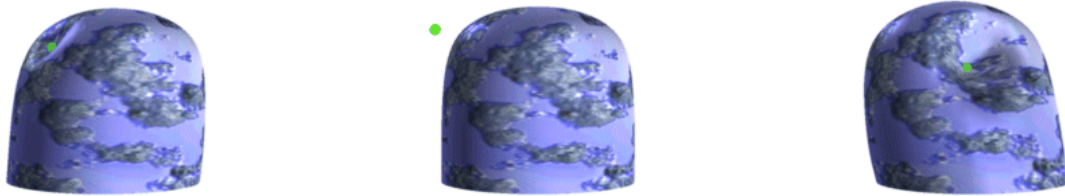
An application of the CMA for multiple distributed contacts with unilateral contact constraints was the grasping task illustrated in Figure 3 using the LEM from Figure 10(a). A short video clip is also available online [20].

Our current force feedback implementation is based on the point-like contact approach discussed in the previous section. Forces are rendered by a 3 dof PHANTOM™ haptic interface (model 1.0 Premium), on a dual Pentium II computer running Windows NT. The haptic simulation was implemented in C++, partly using the GHOST® toolkit, and interfaced to our ARTDEFO elastostatic object simulation written in Java™ and rendered with Java 3D™. The frictional contact problem is computed by the haptic servo loop at 1 kHz, which then prescribes boundary conditions for the slower graphical simulation running at 25–80 Hz. For a point-like contact, it was only necessary to perform collision detection on the undeformed model, so this was done using the GHOST® API. A photograph of the authors demonstrating the simulation is shown in Figure 9, and a number of screen shots for various models are shown in Figure 10. A short video clip is also available online [20].

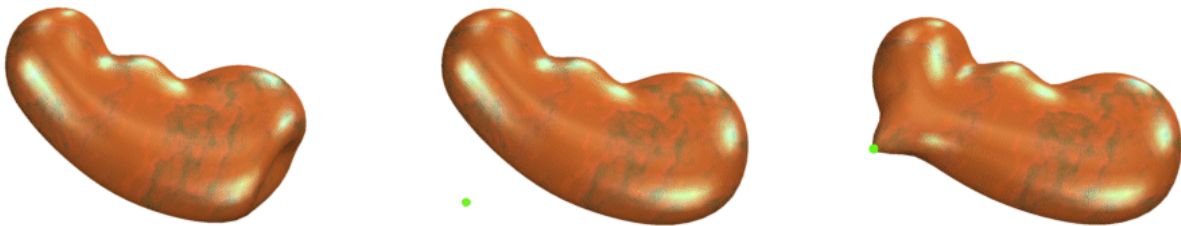
We observed that the vertex masks were successful in producing noticeable improvements in the smoothness of the sliding contact force, especially when passing over regions with irregular triangulations (see Figure 7). We have not conducted a formal human study of the effectiveness of our simulation approach. However, the haptic simulation has been demonstrated to hundreds of users at two conferences: the 10<sup>th</sup> Annual PRECARN-IRIS (Institute for Robotics and Intelligent Systems) Conference (Montreal, Quebec, Canada, May 2000) and in the ACM SIGGRAPH 2000 Exhibition (New Orleans, Louisiana, USA, July 2000). Users reported that the simulation felt realistic. In general, the precomputed LEM approach was found to be both stable and robust.

## 8 Summary and Discussion

We have presented a detailed approach for real time solution of boundary value problems for discrete linear elastostatic models (LEM), regardless of discretization, using precomputed GFs in con-



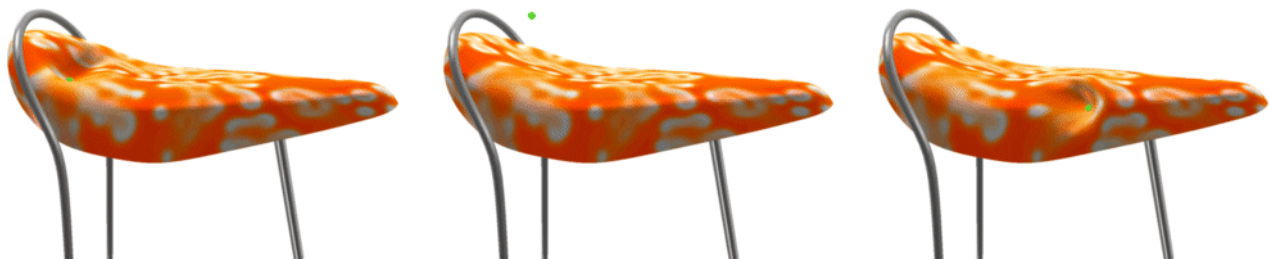
(a) A simple nodular shape with a fixed base region.



(b) A kidney-shaped model with position constrained vertices on part of the occluded side.



(c) A plastic spatula with a position constrained handle.



(d) A seemingly gel-filled banana bicycle seat with matching metal supports.

Figure 10: *Screenshots from real time haptic simulations:* A wide range of ARTDEFO models are shown subjected to various displacements using the masked point-like contacts of §6. For each model, the middle of the three figures is uncontacted by the user's interaction point (a small green ball).



Elastic Model	# Vertices ( $n$ )	# Faces	Precomp (min)	LUD %	Simulate (ms)
Nodule	130v (89 free)	256f	1.1	1.1%	0.05
Kidney	322v (217 free)	640f	7.7	3.1%	0.13
Spatula	620v (559 free)	1248f	45	5.7%	0.34
Banana Seat	546v (245 free)	1088f	25	20.0%	0.15

Table 2: *GF precomputation and simulation times* for the BEM models depicted in Figure 10. All GFs corresponding to moveable free vertices (in  $\Lambda_p^0$ ) were computed, and the precomputation time (Precomp) of the largest model is less than an hour. As is typical of BEM computations for models of modest size ( $n < 1000$ ), the  $\mathcal{O}(n^2)$  construction of the matrices (H and G in equation 8) is a significant portion of the computation, e.g., relative to the  $\mathcal{O}(n^3)$  cost of performing the LU decomposition (LUD %) of the  $A$  matrix. The last column indicates that submillisecond graphics-loop computations (Simulate) are required to determine the point contact deformation response of each model's free boundary.

# Updates, $s$	LU Factor (ms)	LU Solve (ms)	$(\Xi E)(E^T \bar{v})$ for $n = 100$ (ms)
10	0.54	0.03	0.38
20	2.7	0.15	0.74
40	19	0.58	1.7
100	310	5.7	5.7

Table 3: *Timings of CMA suboperations* such as LU decomposition (LU Factor) and back-substitution (LU Solve) of the capacitance matrix, as well as the weighted summation of  $s$  GFs (per 100 nodes) are shown for different sizes of updated nodal constraints,  $s$ .

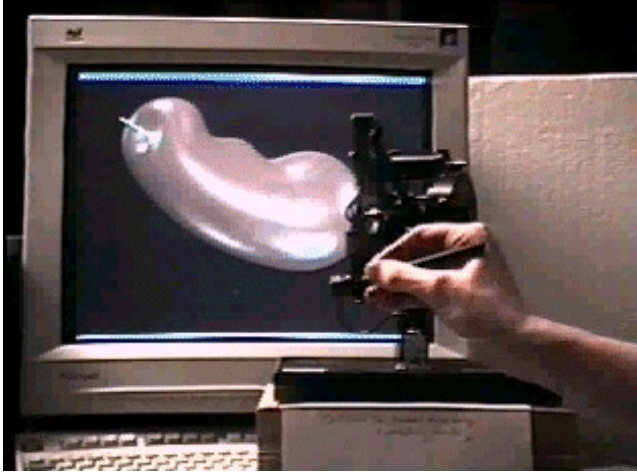


Figure 9: : *Photograph of simulation in use*: Users were able to push, slide and pull on the surface of the model using a point-like manipulandum. Additionally, it was possible to change the surface friction coefficient, as well as the properties of the pressure mask, with noticeable consequences. The PHANToM™ (here model 1.0 Premium) was used in all force feedback simulations, and is clearly visible in the foreground.

junction with capacitance matrix algorithms (CMAs). The data-driven CMA formulation highlights the special role of the capacitance matrix in computer haptics as a contact compliance useful for generating contact force and stiffness models, and provides a framework for extending the capabilities of these models.

Additionally, the important special case of point-like contact was addressed with special attention given to the consistent definition of contact forces for haptics. While this topic has been discussed before, we have introduced vertex masks to specify the distribution of contact forces in a way which leads to physically consistent force feedback models which avoid the numerical artifacts which lead to nonsmooth rendering of contact forces on discrete models, as well as ill-defined contacts in the continuum limit.

There are several issues to be addressed by future work on the simulation of LEMs for computer haptics.

One of the promises of linear GF models is that it should be possible to precompute and haptically touch large-scale models even if they are too large to be graphically rendered. However, the CMA presented here is very efficient for small models (small  $n$ ) and limited constraints (small  $s$ ), but further optimizations and required for precomputing, storing and simulating large-scale LEMs. Extending LEMs to accommodate geometric and material nonlinearities is also an area of study. Results addressing these problems will appear shortly in subsequent publications.

A key challenge for interactively rendering elastic models is the plausible approximation of friction in the presence of multiple distributed elastic-rigid and elastic-elastic contacts. While large contact areas are a potential problem for LEM haptics, i.e., due to large update costs, the accompanying collision detection and friction problems appear to be at least as difficult. Incorporation of LEMs into hybrid interactive dynamical simulations is also a relatively unexplored area.

Finally, the same issues (perceptible force regularity and spatial consistency) which motivated our approach for a single point-like contact, also arise for multiple point-like contacts and in general with multiple distributed contacts. The tight coupling of force stiffnesses between all contact zones, and therefore each (networked) force feedback device, can make this a difficult problem in practice.

## A Justification of Interpolated Traction Distributions for Point Contact

This section derives the nodal boundary conditions associated with a localized point contact at an arbitrary mesh location. The practical consequence is that the discrete traction distribution may be conveniently interpolated from suitable nearby nodal distributions or masks.

Given a continuous surface traction distribution,  $\mathbf{p}(\mathbf{x})$ , a corresponding discrete distribution  $\Phi(\mathbf{x})\mathbf{p}$  may be determined by a suitable projection into  $\mathcal{L}$  of each Cartesian component of  $\mathbf{p}(\mathbf{x})$ . For example, consider the projection of a scalar function on  $\Gamma$  defined as the minimizer of the scalar functional  $\mathcal{E} : \mathbb{R}^{3n} \mapsto \mathbb{R}$ ,

$$\mathcal{E}(\mathbf{p}) = \int_{\Gamma} [\|\mathbf{p}(\mathbf{x}) - \Phi(\mathbf{x})\mathbf{p}\|_2^2 + \|B\Phi(\mathbf{x})\mathbf{p}\|_2^2] d\Gamma \mathbf{x},$$

where  $B : \mathcal{L} \mapsto \mathbb{R}$  is some linear operator that can be used, e.g., to

penalize nonsmooth functions, and  $\Phi(\mathbf{x}) : \mathbb{R}^{3n} \mapsto \mathbb{R}^3$  is a nodal interpolation matrix defined on the surface,

$$\Phi(\mathbf{x}) = [\phi_1(\mathbf{x})\phi_2(\mathbf{x})\cdots\phi_n(\mathbf{x})] \otimes I_3, \quad \mathbf{x} \in \Gamma,$$

where  $I_3$  is the 3-by-3 identity matrix. The Euler-Lagrange equations for this minimization are

$$\begin{aligned} \sum_{j=1}^n (\int_{\Gamma} [\phi_i(\mathbf{x})\phi_j(\mathbf{x}) + (B\phi_i(\mathbf{x}))(B\phi_j(\mathbf{x}))] d\Gamma_{\mathbf{x}}) \mathbf{p}_j \\ = \int_{\Gamma} \phi_i(\mathbf{x})\mathbf{p}(\mathbf{x})d\Gamma_{\mathbf{x}} \quad i = 1, 2, \dots, n, \end{aligned}$$

which, in an obvious notation, is written as the linear matrix problem

$$\mathcal{A}\mathbf{p} = \mathbf{f} \quad (36)$$

to be solved for the nodal traction values  $\mathbf{p}$ . Note that  $\mathcal{A}$  has units of area.

The relevant traction distribution for point-like contact is a scale-independent concentrated point load

$$\mathbf{p}(\mathbf{x}) = \mathbf{p}^{\delta}(\mathbf{x}) = \mathbf{f}^{\delta}\delta(\mathbf{x} - \mathbf{x}^{\delta})$$

which models a force  $\mathbf{f}^{\delta}$  delivered at  $\mathbf{x}^{\delta} \in \Gamma$ . The force  $n$ -vector in equation (36) has components

$$f_i = \phi_i(\mathbf{x}^{\delta})\mathbf{f}^{\delta}$$

and the corresponding pressure distribution's nodal values are

$$\mathbf{p} = \mathcal{A}^{-1}\mathbf{f}.$$

For compactly supported basis functions,  $\phi_i(\mathbf{x})$ ,  $\mathbf{f}$  has only a small number of nonzero components for any given  $\mathbf{x}$ . Hence  $\phi_i(\mathbf{x}^{\delta})$  are the interpolation weights describing the contribution of the nearby nodal pressure distributions, here specified by the columns of  $\mathcal{A}^{-1}$ .

As an example, consider the important case where  $\mathcal{L}$  is a continuous piecewise linear function space with  $\phi_i(\mathbf{x}_j) = \delta_{ij}$ . This was the space used in our implementation. In this case, at most only three components of  $\mathbf{f}$  are nonzero, given by the indices  $\{i_1, i_2, i_3\}$  which correspond to vertices of the contacted triangle  $\tau^{\delta}$ , i.e., for which  $\mathbf{x}^{\delta} \in \tau^{\delta}$ . The values  $\phi_i(\mathbf{x}^{\delta})$  are the barycentric coordinates of  $\mathbf{x}^{\delta}$  in  $\tau^{\delta}$ . The pressure distribution's nodal values are then

$$\mathbf{p} = \mathcal{A}^{-1}\mathbf{f} \quad (37)$$

$$\begin{aligned} &= \sum_{k=1}^3 (\mathcal{A}^{-1})_{:i_k} f_{i_k} \\ &= \sum_{k=1}^3 \phi_{i_k}(\mathbf{x}^{\delta}) \left[ (\mathcal{A}^{-1})_{:i_k} \mathbf{f}^{\delta} \right] \\ &= \sum_{k=1}^3 \phi_{i_k}(\mathbf{x}^{\delta}) \mathbf{p}^{(i_k)}, \end{aligned} \quad (38)$$

where  $\mathbf{p}^{(i_k)}$  is the pressure distribution corresponding to the application of the load directly to vertex  $i_k$ , and  $(\cdot)_{:i_k}$  refers to block column  $i_k$  of the matrix. Therefore the *piecewise linear pressure distribution for a point load applied at a barycentric location on a triangle is equal to the barycentric average of the pressure distributions associated with the point load applied at each of the triangle's vertices*. This may be recognized as an elastic generalization of *force shading* [27] for rigid models.

Note that the  $j^{th}$  column of  $\mathcal{A}^{-1}$  is a vertex mask that describes the nodal distribution of the load applied to the  $j^{th}$  vertex. Modifying the penalty operator  $B$  results in masks with varying degrees of smoothness and spatial localization.

## References

- [1] Oliver Astley and Vincent Hayward. Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998.
- [2] Remis Balaniuk. Building a haptic interface based on a buffer model. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.
- [3] James R. Barber. *Elasticity*. Kluwer Press, Dordrecht, first edition, 1992.
- [4] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. *Boundary Element Techniques: Theory and Applications in Engineering*. Springer-Verlag, New York, second edition, 1984.
- [5] Morten Bro-Nielsen. Surgery simulation using fast finite elements. *Lecture Notes in Computer Science*, 1131:529–, 1996.
- [6] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, August 1996.
- [7] Murak Cenk Çavuşoğlu and Frank Tendick. Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000.
- [8] S. Cotin, H. Delingette, and N. Ayache. Realtime elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1):62–73, 1999.
- [9] Stephane Cotin, Herve Delingette, and Nicholas Ayache. Efficient linear elastic models of soft tissues for real-time surgery simulation. Technical Report RR-3510, Inria, Institut National de Recherche en Informatique et en Automatique, 1998.
- [10] Diego d'Aulignac, Remis Balaniuk, and Christian Laugier. A haptic interface for a virtual exam of a human thigh. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000.
- [11] Gilles Debunne, Mathieu Desbrun, Alan Barr, and Marie-Paule Cani. Interactive multiresolution animation of deformable models. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation and Simulation '99*, SpringerComputerScience, pages 133–144. Springer-Verlag Wien New York, 1999. Proceedings of the Eurographics Workshop in Milano, Italy, September 7–8, 1999.
- [12] Y. Ezawa and N. Okamoto. High-speed boundary element contact stress analysis using a super computer. In *Proc. of the 4<sup>th</sup> International Conference on Boundary Element Technology*, pages 405–416, 1989.
- [13] Sarah Gibson, Christina Fyock, Eric Grimson, Takeo Kanade, Rob Kikinis, Hugh Lauer, Neil McKenzie, Andrew Mor, Shin Nakajima, Hide Ohkami, Randy Osborne, Joseph Samosky, and Akira Sawada. Volumetric object modeling for surgical simulation. *Medical Image Analysis*, 2(2):121–132, 1998.
- [14] Sarah F. Gibson and Brian Mirtich. A survey of deformable models in computer graphics. Technical Report TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, November 1997.

- [15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore and London, third edition, 1996.
- [16] William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, June 1989.
- [17] K. V. Hansen and O. V. Larsen. Using region-of-interest based finite element modeling for brain-surgery simulation. *Lecture Notes in Computer Science*, 1496:305–, 1998.
- [18] Friedel Hartmann. *The mathematical foundation of structural mechanics*. Springer-Verlag, New York, 1985.
- [19] C.-H. Ho, C. Basdogan, and M. A. Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence*, 8(5):477–491, 1999.
- [20] Doug L. James. <http://www.cs.ubc.ca/~djames/deformable>.
- [21] Doug L. James and Dinesh K. Pai. ARTDEFO: Accurate Real Time Deformable Objects. *Computer Graphics*, 33(Annual Conference Series):65–72, 1999.
- [22] A. M. Abu Kassim and B. H. V. Topping. Static reanalysis: a review. *Journal of Structural Engineering*, 113:1029–1045, 1987.
- [23] U. Kühnapfel, H.K. Çakmak, and H. Maaß. 3d modeling for endoscopic surgery. In *Proceedings of IEEE Symposium on Simulation*, pages 22–32, Delft University, Delft, NL, October 1999.
- [24] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.
- [25] K. W. Man, M. H. Aliabadi, and D. P. Rooke. Analysis of Contact Friction using the Boundary Element Method. In M. H. Aliabadi and C. A. Brebbia, editors, *Computational Methods in Contact Mechanics*, chapter 1, pages 1–60. Computational Mechanics Publications and Elsevier Applied Science, 1993.
- [26] T. H. Massie and J. K. Salisbury. The phantom haptic interface: A device for probing virtual objects. In *ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Chicago, IL, Nov. 1994.
- [27] Hugh B. Morgenbesser and Mandayam A. Srinivasan. Force shading for haptic shape perception. In *Proceedings of the ASME Dynamics Systems and Control Division*, volume 58, 1996.
- [28] Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, and Som H. Yau. Scanning Physical Interaction Behavior of 3D Objects. In *Computer Graphics Proceedings (SIGGRAPH 2001)*. ACM Siggraph, 2001.
- [29] G. Picinbono, J. C. Lombardo, H. Delingette, and N. Ayache. Anisotropic elasticity and force extrapolation to improve realism of surgery simulation. In *Proceedings of IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000.
- [30] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*, chapter Sherman-Morrison and Woodbury, pages 66–70. Cambridge University Press, Cambridge, 1987.
- [31] R. Ramanathan and D. Metaxas. Dynamic deformable models for enhanced haptic rendering in virtual environments. In *IEEE Virtual Reality Conference*, 2000.
- [32] P. Schröder, D. Zorin, T. DeRose, D. R. Forsey, L. Kobbelt, M. Lounsbery, and J. Peters. Subdivision for modeling and animation. *SIGGRAPH 99 Course Notes*, August 1999.
- [33] Sensable Technologies, Inc. GHOST SDK, <http://www.sensable.com>.
- [34] Yan Zhuang and John Canny. Haptic interaction with global deformations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.
- [35] O. C. Zienkiewicz. *The Finite Element Method*. McGraw-Hill Book Company (UK) Limited, Maidenhead, Berkshire, England, 1977.
- [36] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, volume 1, pages 149–150, Chicago, IL (US), 1994.

# Scanning Physical Interaction Behavior of 3D Objects

Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, Som H. Yau

Department of Computer Science, University of British Columbia, Vancouver, Canada  
{pai,kvdoel,djames,jlang,lloyd,jrichmo,shyau}@cs.ubc.ca



(a) Real toy tiger. By design, it is soft to touch and exhibits significant deformation behavior.



(b) Deformable model of tiger scanned by our system, with haptic interaction.



(c) Real clay pot, with glazed regions. The pot exhibits a variety of contact textures and sounds.



(d) Virtual interaction with scanned model of pot; includes contact texture and contact sounds.

Figure 1: Examples of behavior models scanned by our system

## Abstract

We describe a system for constructing computer models of several aspects of physical interaction behavior, by scanning the response of real objects. The behaviors we can successfully scan and model include deformation response, contact textures for interaction with force-feedback, and contact sounds. The system we describe uses a highly automated robotic facility that can scan behavior models of whole objects. We provide a comprehensive view of the modeling process, including selection of model structure, measurement, estimation, and rendering at interactive rates. The results are demonstrated with two examples: a soft stuffed toy which has significant deformation behavior, and a hard clay pot which has significant contact textures and sounds. The results described here make it possible to quickly construct physical interaction models of objects for applications in games, animation, and e-commerce.

**Keywords:** Behavioral Animation, Deformations, Haptics, Multimedia, Physically Based Modeling, Robotics, Sound Visualization

## 1 Introduction

Real 3D objects exhibit rich interaction behaviors which include how an object deforms on contact, how its surface feels

when touched, and what kinds of sounds it makes when one interacts with it. These aspects of visual, haptic<sup>1</sup>, and auditory behavior provide important interaction cues and increase the sense of presence in virtual environments such as games. Despite recent progress in deformation modeling (e.g., [37, 7, 18]), haptic rendering (e.g., [32]), and auditory displays (e.g., [12]), these aspects are either entirely missing from models used for computer graphics and interaction, or must be painstakingly added by highly skilled professionals. We believe that a major reason for this unfortunate situation is the difficulty of constructing these complex multimodal models by hand. In this paper we show how this problem can be solved by scanning the behavior of real objects.

Constructing behavior models requires not only acquiring measurements of real object behavior, but also designing mathematical models whose parameters can be effectively estimated from the measurements, and can be effectively used for realistic rendering. Each of these steps is important for successfully modeling real object behavior. We give detailed descriptions of how this can be done for three aspects of interaction behavior: (1) deformation models which can be rendered both visually and haptically; (2) contact texture models for capturing surface friction and roughness for haptic display and dynamics simulation; and (3) contact sound models for synthesizing interaction sounds, including sounds of continuous interaction.

Figure 1 shows some behavior models acquired by our system. Of course, it is somewhat difficult to show real time behavior on paper. The accompanying video demonstrates the behavior models better.

In this paper we also describe how the acquisition of these models can be automated using a highly integrated robotic measurement facility, and how behavior models can be registered relative to a geometric model of an object. Even though our facility is an expensive prototype and uses sophisticated robotics for interaction and behavior measurement, we believe it can be practical and economical for modeling

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

<sup>1</sup>haptics refers to the sense of touch.

because the facility can be shared by multiple users. The techniques we describe can be used to construct a *model foundry*, similar to a VLSI chip foundry but in reverse. Users could send real objects to such a foundry and receive in return a behavior model of the object.

It is helpful to compare the work presented here to 3D geometric modeling, from which we draw inspiration and instruction. 3D geometric models can be constructed by hand using 3D modeling software and for some special applications, this hand modeling and editing is essential. However, 3D scanning technology has dramatically changed the way such models are constructed. 3D geometry scanning makes it possible to capture fine details for unprecedented realism, but more importantly, empowers users with modest artistic talents with the ability to construct 3D models quickly and inexpensively. In a similar way, we expect that hand construction of interaction behavior will continue to be useful for some applications (e.g., those requiring fine creative control to match the context and narrative). But for many applications in games, animation, and e-commerce, the ability to construct highly detailed behavior quickly and inexpensively using the techniques we describe here could transform the way we construct models for 3D object interaction.

## 1.1 Related Work

We are not aware of any other system capable of scanning models of contact interaction behavior such as that described in this paper. However, each component of our system has many connections to previous work that we discuss in the relevant sections below. Here we briefly discuss related work in the general area of building models of 3D objects by measuring the real world.

Almost all the work in the area of modeling real world objects in computer graphics has focused on geometric modeling and reflectance modeling. In recent years there has been dramatic progress in scanning geometric models of 3D objects (e.g., [17, 8, 33, 22, 23]). In part because of the high accuracy of laser scanning, most of these techniques assume that the range data are given and focus on estimating good meshes from the data. Techniques have also been developed for geometry reconstruction from a few photographs [11] or even from a simple desktop system with a wand [2]. Acquiring the reflectance properties of existing objects has also been an active research area (e.g., [33, 9, 31, 15]). Our work has parallel goals with this type of automated model acquisition, but differs in terms of types of models acquired. Acquiring interaction models is more difficult because of the necessity of actually interacting and making contact with the objects to be measured.

Our work has some connections with image based techniques [4] and motion capture in that a recording can be considered a simple model of behavior, which can be edited and transformed (e.g., [26]). With few exceptions they have generally not been used for estimating parameters of physical models. They also do not account for inputs to the motion, and therefore can not be directly used for simulating the effects of new inputs.

Measuring the real world requires a certain amount of infrastructure and several groups have developed measurement facilities for this purpose. We mention the Cornell Light Measurement facility [15], the Columbia/Utrecht facility used for constructing a reflectance and texture database [9], the CMU Virtualized Reality<sup>TM</sup> laboratory [41], the Berkeley Light Stage [10]. Field-deployable systems include the Digital Michelangelo project [23] and the IBM Pietà

project [31].

Our own facility is perhaps the most highly automated and flexible system available today. It was designed to provide one-stop shopping for a large number of measurements rather than for highly accurate measurement. It required significant developments in robotics to control and coordinate the various measurement and actuation subsystems of the facility. We have previously described the robotics aspects of teleprogramming and motion planning for the system, and measurement techniques for sound and deformation (e.g., [25, 24]). However, the present paper is the first to describe the complete process of modeling interaction behaviors, from real object to rendered model.

## 2 A Framework for Modeling Behavior

To help understand how to construct useful models of real objects, it is helpful to break the task down into four steps: selection of model structure, measurement, parameter estimation, and rendering. In the following sections we detail how these steps are carried out for acquiring models of deformation, contact texture, and sound.

### Selection of model structure

This defines the fundamental class of mathematical models that will be used to represent real physical behavior. In this step we fix the structure and not the parameters of the model.

We emphasize that this step is a creative act of model design, in which the modeler balances the competing needs of the accuracy of the model's predictions, rendering speed, ease of acquiring measurements, stability of parameter estimation, etc. It is tempting to think that the model is dictated by Physics and can be "looked up" in a textbook, but this is far from the truth. In designing the model structure it is essential to realize that all models have a finite domain of applicability.

### Measurement

In this step we acquire the data from the real world to construct the model. This step is critical since all subsequent results depend on it. It is non-trivial for several reasons. First, we are interested in contact behavior which can not be simply observed but must be "excited" by physically interacting with the object. Thus we not only need to measure, say, the sound produced by an object, but we also need a way to hit the object in a carefully controlled manner to produce the sound. Second, traditional measurement techniques are designed for measuring material properties of small samples of objects, but we would like to build models of entire objects without destroying them or changing their essential global properties. Thus it is necessary to be able to move either the object or the measuring instruments or both, to access different parts of the object. Third, we need to register the different measurements relative to each other. We do this by registering all measurements relative to a geometric model of the object. For instance, this allows us to use the surface roughness of a particular point on the surface to drive the sound produced by scraping the surface at that point. However this means that we must first acquire a geometric model of the object prior to scanning the behavior. Finally, to interpret the raw data the instruments need to be calibrated. We do this using special calibration objects,



but auto-calibration techniques could also be used. To facilitate this kind of measurement we have built an integrated robotic measurement facility described in §3.

### Parameter estimation

Estimation connects measurements to models. It is important to realize that estimation problems are frequently ill-posed and can lead to very sensitive inverse problems. Therefore it is necessary to pre-process the raw data, use regularization (which can be done in a natural way using Bayesian priors), and to use robust estimation techniques. As an example, direct fitting of sound models to sound waveforms by least-squares techniques produces very poor results in the presence of noise. We describe robust techniques for estimation which we found work well for the various estimation problems described below.

### Rendering

Finally, the estimated models must be rendered, to produce deformations, haptic forces, and sounds. This step, of course, is the primary motivation for the previous steps and influences design decisions throughout. In particular, since we are interested in *interaction* behavior, it is important that the models are rendered at interactive rates. We describe implemented algorithms suitable for interactive rendering of deformation, forces, and sounds. The rendering step is also important for *validation* of the scanned model, i.e., determining whether the estimated model approximates reality sufficiently well for a given purpose. For computer graphics and interaction, it is difficult to design sensible quantitative metrics of model validity, and one must largely rely on user perception. We show results comparing the behavior of real objects to simulations of the scanned behavior models.

## 3 Measurement System for Interaction Behavior

Carefully acquiring the measurements required for model building is often the most challenging part of the modeling process. For acquiring the kind of contact measurements we need, we have developed the UBC Active Measurement facility (ACME), a highly automated robotic measurement facility. The robotics aspects of the facility have been previously described in detail [25]. For completeness, we briefly outline it here. We note, however, that the techniques described below do not depend on the use of this particular facility which was designed to provide a large variety of measurements; a simpler measurement set up could be constructed to build specific kinds of models. As with 3D geometry scanning, we expect that in the future it may be possible to build portable or desktop versions of such a measurement system.

Fig. 2 shows the facility which consists of a variety of sensors and actuators, all under computer control. Its major subsystems include: a 3 DOF Test Station (bottom of image) used for precise planar positioning of the test object; a Field Measurement System (shown at the top left) consisting of a Triclops trinocular stereo vision system, a high resolution RGB camera, and a microphone, all mounted on a 5 DOF positioning robot; and a Contact Measurement System (CMS, shown on the right) consisting of a Puma 260 robot equipped with a force/torque sensor, mounted on a linear stage. The CMS is the key subsystem for contact measurements as it is used to interact with the object to



Figure 2: ACME Facility Overview

be modeled. The entire facility can be controlled from any location on the Internet. We will describe the use of the sensors and actuators for measurement in the relevant sections below.

## 4 Geometric Modeling

The geometric models required for registering other measurements can be produced by any geometry scanning and mesh generation method (e.g., [17, 8, 33, 28, 23]). While the focus of this paper is not on geometric modeling, for completeness we will briefly outline our approach to geometric model construction starting from stereo range measurements.

### Stereo range measurement

The main shape sensor in our measurement facility is a trinocular stereo vision system<sup>2</sup> which is capable of producing large amounts of viewpoint-registered range images at modest resolutions (approx. 2-3 mm for a typical viewpoint), in close to real time. Accurate ranging relies on image features for matching between the stereo cameras; additional features can be attached or projected onto the surface.

Stereo range data is notoriously noisy, and for best results it is filtered in several ways. First, range is calculated with variable mask sizes in a voting scheme. We also utilize the checks and filters of the Triclops stereo library. Further processing of the data by volumetric reconstruction requires approximate surface normals at range data points. The normals are estimated by plane fitting in local image neighborhoods. Further filtering of the range-data is performed based on the quality of fit, the number of valid range-data per neighborhood and the viewing angle of the plane.

### Multiresolution mesh construction

An initial triangle mesh is generated from the filtered range data using a volumetric approach by reconstruction software provided courtesy of NRC of Canada [28]. The number and quality of triangles depends on the surface sampling density. Further processing is required to arrive at a useful geometric model since this mesh may not be watertight; it may include some of the supporting Test Station surface; and there may

<sup>2</sup>A Color Triclops from Point Grey Research.

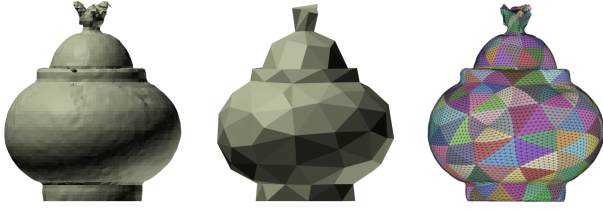


Figure 3: *Clay Pot Mesh Reconstruction*: (left) raw scan mesh (13150 vertices, 26290 faces) with various external stray polygons; (middle) watertight base level (level 0) mesh (127 vertices, 250 faces) generated via simplification of the raw scan mesh; (right) level 3 subdivision connectivity mesh (8002 vertices, 16000 faces) generated using displaced subdivision surface style piercing of the raw scan mesh.

be erroneous surfaces due to noisy data. An example is shown in the left image of Figure 3.

Finally, we construct watertight subdivision connectivity triangle meshes at several resolutions  $l = 0, 1, 2, \dots, L$  because they are desirable for later modeling. Here we exploit a common property of raw meshes produced from range data: while the region exterior to the object geometry may contain undesirable mesh features, the interior of the mesh is a fair representation of the scanned surface. This allows the construction of multiresolution meshes by expanding and refining a surface inside the object using a normal piercing process similar to the construction of displaced subdivision surfaces [21] and normal meshes [16]. The coarse resolution mesh is generated by simplifying the raw mesh (using QSlim [13]) and possibly some minor editing to ensure that the mesh is free of defects such as in regions with poor range data. To ensure that the mesh expansion process is bounded, the raw mesh is manually closed if the NRC package produces a large hole on the unimaged underside of the object; this was addressed for the tiger mesh by simply inserting a large horizontal triangle.

While this approach to mesh construction is not robust, it produces meshes of sufficient quality that we were able to proceed with our main physical modeling objectives.

## Texture mapping

Texture mapping is accomplished using calibrated color images from the stereo vision system. The vertices of the meshes are projected into the color images using the pinhole camera calibration matrix, and triangles are tagged as visible if their vertices and centroid are all visible. We select the imaged triangular area as the texture map if the product of the image area times the cosine between view direction and triangle normal is maximum in all our images. We also adjust the relative global brightness of all color texture images. The texture maps could be readily improved by blending of the local textures over the entire map eliminating abrupt brightness changes at edges, as in [27].

## 5 Deformation Modeling

### Selection of model structure

We use linear elastostatic models for deformable objects since these models can be simulated at interactive rates [18], and linearity makes parameter estimation easier. Most im-

portant, linear elastic models can be characterized using Green's functions, which capture the input-output behavior of the object's boundary for a given set of boundary conditions; therefore there is no need to observe or estimate quantities in the interior of the object.

It is a practical consideration that the model must be fixtured for it to be deformed by ACME's robots. Therefore we start by assuming that the physical object will be measured (and rendered) while attached to a support, like the Test Station. The linear elastostatic model then approximates the displacement response  $\mathbf{u} = \mathbf{u}^{(l)}$  of the resolution  $l$  mesh vertices due to *applied* surface tractions<sup>3</sup>  $\mathbf{p} = \mathbf{p}^{(l)}$  by

$$\mathbf{u} = \mathbf{U}\mathbf{p} \quad \text{or} \quad \mathbf{u}^{(l)} = \mathbf{U}^{(l)}\mathbf{p}^{(l)}. \quad (1)$$

Here  $\mathbf{u}$  and  $\mathbf{p}$  are block vectors of length  $n$  with 3-vector elements, where the displacement and traction at the  $k^{\text{th}}$  vertex is  $\mathbf{u}_k$  and  $\mathbf{p}_k$ , respectively;  $\mathbf{U}$  is a square  $n^{(l)}$ -by- $n^{(l)}$  block matrix with 3-by-3 matrix elements. The  $k^{\text{th}}$  block column  $\mathbf{U}_{:,k}$  describes the displacement response contribution from the  $k^{\text{th}}$  applied traction  $\mathbf{p}_k$ . The diagonal blocks  $\mathbf{U}_{kk}$  describe the self-compliance of the  $k^{\text{th}}$  vertex, and play an important role in defining vertex stiffnesses for force-feedback rendering [19]. The traction vector at a vertex is the force over the area of the one-ring of a vertex. The force is distributed linearly over the area as a hat function located at the vertex.

The key behavior of the model is characterized by the displacement response of the unfixtured free surface due to forces applied to the free surface. This corresponds to the only interesting measurable portion of the  $\mathbf{U}$  matrix, which in turn is related to the discrete *Green's function matrix* [19] for the free boundary. For instance, rows of  $\mathbf{U}$  corresponding to vertices attached to the fixtured support necessarily have zero displacement values; these same vertices correspond to columns which are not exposed and therefore can not be actively inspected.

Finally, once the Green's function matrix for a fixture configuration is known, deformations of the object can be simulated efficiently using fast matrix updating techniques [19].

### Measurement

Objects to be measured, such as the stuffed toy tiger shown in Figure 1, are fixed to the Test Station. Deformation measurements record surface displacements and contact forces resulting from active probing with ACME's CMS robot arm. During deformation the position of the contact probe can be continuously monitored at 100 Hz. The robot arm's wrist force sensor is capable of recording the six-dimensional force-torque wrench at the tip of the probe at 1000 Hz.

The robot probes surface locations corresponding to vertices of the geometric model at the desired reconstruction resolution<sup>4</sup>  $l$ . The position of the contact probe measures the displacement of the contacted vertex  $\mathbf{u}_k$ , while the force-torque sensor measures the contact force. The effective contact traction  $\mathbf{p}_k$  is the force divided by the effective vertex area (one third of the sum of adjacent triangle areas). Since there are no other *applied* tractions on the free or fixed surfaces, the complete traction vector  $\mathbf{p}$  is zero except for the single contact traction  $\mathbf{p}_k$ . In the following we describe how the deformation of the free surface is measured and mapped onto the vertices in order to estimate  $\mathbf{u}$ .

<sup>3</sup>Traction is the force per unit area, similar to pressure.

<sup>4</sup>To avoid measurement artifacts the scale of the mesh is larger than the measurement probe's contact area.

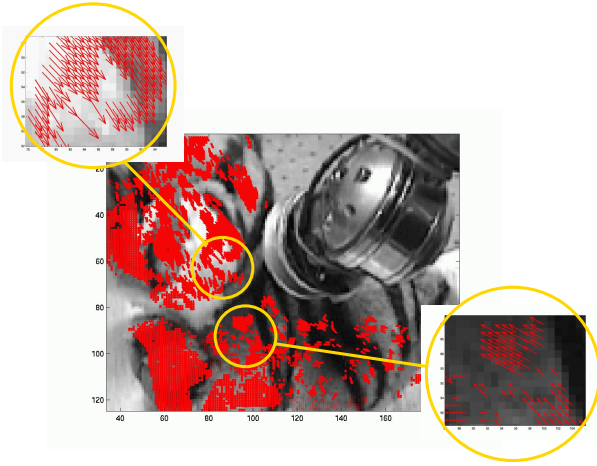


Figure 4: 2D Flow on tiger, poked near the neck.

The surface deformation of the object is measured visually using the Triclops trinocular stereo-vision system in the Field Measurement System (FMS) of ACME. The measurement is based on tracking visual surface features in three spatial dimensions in a range-flow [42] approach. Several methods for calculating dense range-flow from stereo and optical flow exist [41, 43, 34].

Our method utilizes redundancy in the imagery from the trinocular stereo-system to increase robustness [20]; It is summarized as:

- Segment image into “object surface” and “other,” based on the geometric model of the undeformed object.
- Calculate range from stereo.
- Calculate simultaneous optical flow in images from each camera for the “object surface”.
- Combine optical flow by voting and map the result into three dimensions based on range data.
- Use the “range-flowed” object surface to segment the next image in the sequence and continue with stereo calculations as above.

The optical flow during deformation measurement is shown in Figure 4. In our approach, the surfaces need sufficient visual texture because of the reliance on stereo vision and optical flow. Most objects have sufficient visual texture for stereo matching but if not non-permanent visual texture may be applied (e.g., using pins, stickers, or water soluble paint).

The next step in surface deformation measurement is the mapping of range flow vectors to displacements of vertices of a chosen mesh level. Flow vectors are associated with a start position on a triangular patch of the undeformed mesh. We estimate the displacement of a vertex with a robust averaging process rather than just using the closest flow vector to a vertex (see, for example, [1] for a discussion on robust motion estimation). The flow vectors associated with triangular patches joined at a vertex are median filtered. This is followed by a weighted average based on distance from the vertex. The flow vectors have to be dense enough on the surface relative to the mesh level for this process to be robust. In our set-up, we get high density by positioning

the camera close to the object ( $\approx 0.8m$ ). The measured displacement  $\mathbf{u}$  covers surface area visible from a chosen view point. In the estimation section below, we discuss how to combine multiple measurements for the same contact vertex.

### Parameter estimation

Our approach to the estimation of the matrix  $\mathbf{U}$  from the measurements of displacement  $\mathbf{u}$  and traction  $\mathbf{p}$  addresses two main issues: (1) noise in the form of outliers in the displacement measurement and (2) incomplete measurements. Outliers can, on occasion, still be observed in the displacement measurement despite the above described filtering process. These outliers originate from consistent incorrect flow vectors due to mismatches in the range-flow or in the stereo processing over an image area. Incomplete measurements arise from partially observed object surfaces, from reachability limits of the CMS and from anisotropic responses of the object to probing.

A single block element  $\mathbf{U}_{ij}$  describes the relationship between the displacement  $\mathbf{u}_i$  and a single applied traction  $\mathbf{p}_j$ :

$$\mathbf{u}_i = \mathbf{U}_{ij} \mathbf{p}_j.$$

For each element, we obtain  $m \leq M$  measurements by probing each vertex location  $M$  times. We arrive at a standard least squares problem to be solved for  $\mathbf{U}_{ij}^T$ :

$$[\mathbf{p}_j^1 \mathbf{p}_j^2 \dots \mathbf{p}_j^m]^T \mathbf{U}_{ij}^T = [\mathbf{u}_i^1 \mathbf{u}_i^2 \dots \mathbf{u}_i^m]^T$$

We solve this least squares problem if our measured tractions are a response to a set of probe directions which span 3-space. This guarantees that we excite the model in all directions. However, because of the possibility of (in general) noncompliant responses combined with the limited resolution of the measurements, this does not guarantee that the solution  $(\mathbf{U}_{ij})^T$  has full rank. Therefore, we calculate the solution by means of the truncated singular value decomposition (TSVD). We truncate the singular values at the approximate resolution of the traction measurements. Furthermore, we select a subset of measurements if we observe an unsatisfactory fit of the estimated  $(\mathbf{U}_{ij})^T$  and repeat. This subset selection process is performed using the least trimmed squares (LTS) method [29], implemented efficiently as in [30]. This process is robust even if  $(M-4)/2$  measurements are spoiled.

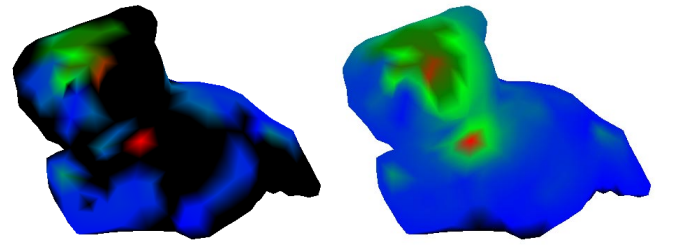


Figure 5: *Plots of estimated displacement responses:* (left) Missing observations result in unestimated response components (shown in black); the remaining nodes are color coded with red indicating the greatest displacement and blue the least. (right) These values are estimated by an interpolating reconstruction to obtain the final deformation responses.

At this stage of the estimation process, the measured displacement field columns of  $\mathbf{U}$  still contain unestimated



elements due to missing observations of the deformed surface (shown in Figure 5). This problem can be minimized by obtaining more measurements but not entirely avoided. Scattered data reconstruction is used to fill in elements for each column individually. We currently interpolate missing displacements by solving Laplace’s equation over the set of unestimated vertices, but better methods are currently being investigated. The result of this interpolation process is shown in Figure 5.

Finally, in order to improve rendering quality and reduce measurement and estimation time we exploit the multiresolution mesh structure to optionally infer Green’s function responses for unmeasured vertices. This is done by actively poking the model at a resolution  $(l - 1)$  one level coarser than the resolution  $l$  used to estimate displacement fields (illustrated in Figure 6). The  $k^{th}$  odd vertex on level  $l$  has a response  $U_{:k}$  inferred if both even vertices  $(k_1, k_2)$  of its parent edge have responses. If so, the  $k^{th}$  response  $U_{:k}$  is linearly interpolated from the two parent responses,  $(U_{:k_1}, U_{:k_2})$ . The local responses,  $U_{kk}$  and  $U_{jk}$  when vertex  $j$  is a one-ring neighbor of  $k$ , are handled differently.

Unlike long range displacement influences which are smoothly varying, these local values are associated with a cusp in the displacement field. Simple interpolation for these values is biased and leads to incorrect contact forces during rendering. Instead, the local values are computed as the weighted average of parent responses which have had their local parameterizations smoothly translated from even vertex  $k_*$  to odd vertex  $k$ , e.g.,  $U_{kk}$  is linearly interpolated from  $(U_{k_1 k_1}, U_{k_2 k_2})$  not  $(U_{k k_1}, U_{k k_2})$ . This shifting of the parent’s local response before averaging yields a good estimator of the local response at vertex  $k$ . The resulting displacement field  $U_{:k}$  is also linearly independent of  $U_{:k_1}$  and  $U_{:k_2}$ .

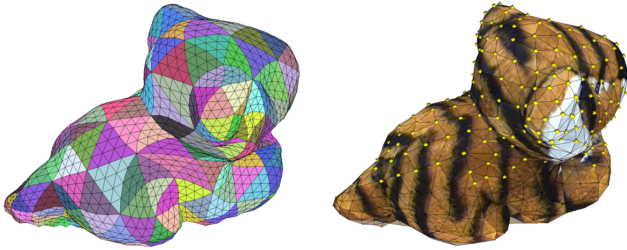


Figure 6: *Multiresolution Mesh and Contact Sampling Pattern*: (left) Coarse  $l = 0$  parameterization of model, used for active contact measurement, displayed on finest  $l = 2$  displaced subdivision surface mesh; (right) yellow points drawn on the  $l = 1$  resolution mark the nodes at which the system’s displacement response to applied tractions was either measured (even vertices) or inferred (odd vertices).

## Rendering

By design, the Green’s function models can be rendered at interactive rates using the algorithm described in [18, 19]. Contact forces are rendered using a PHANToM force-feedback interface with contact force responses computed using vertex pressure masks [19]. The multiresolution deforming surface is also displacement mapped using *displaced subdivision surfaces* [21] to add extra geometric detail to the model. Figure 1 and the accompanying video (and the CAL demonstration) show interaction with scanned tiger model using the PHANToM. In general the results are quite sat-

isfactory, capturing non-local effects such as the movement of the head when the back of the tiger is poked. The model does show some of the limitations of the linear model structure for large input displacements, with somewhat exaggerated deformations. For moderate input displacements<sup>5</sup>, the scanned model behaves quite realistically.

## 6 Contact Texture

### Selection of model structure

Contact texture denotes the way an object feels when it is rubbed or scraped. The two principal aspects we focus on in this paper are friction and surface roughness.

For modeling friction, we use the standard “textbook” Coulomb friction model

$$\mathbf{f}_f = -\mu \|\mathbf{f}_n\| \mathbf{u}_m \quad (2)$$

where  $\mathbf{f}_f$  is the frictional force,  $\mu$  is the coefficient of friction (and the model parameter to be determined),  $\mathbf{f}_n$  is the normal force, and  $\mathbf{u}_m$  is a unit vector in the direction of motion.

Surface roughness is a more elusive property and whole books have been written on how to model it [38]. Roughness is usually associated with small-scale variations in the surface geometry, which create variations in the tangential contact force proportional to the normal force. These tangential force variations can be modeled as local variations  $\tilde{\mu}$  in the coefficient of friction. Combined with  $\mu$ , this yields an *effective* coefficient of friction  $\mu_e$  for a given displacement  $x$  along some particular surface direction:

$$\mu_e(x) = \mu + \tilde{\mu}(x).$$

This formulation has the advantage of unifying haptic rendering of friction and roughness, particularly with commercial haptic devices like the PHANToM which implement their own contact and friction algorithms which may not correspond to the textbook model of Coulomb friction.

Forces due to roughness tend to have some randomness but often also contain periodic components, particularly in human artifacts. We assume that the roughness is isotropic and that people are sensitive only to statistical features of the roughness force variation, and can not discern the specific waveform. To capture both randomness and periodicity we model the friction variations  $\tilde{\mu}(x)$  as an autoregressive AR( $p$ ) process, driven by noise, so that

$$\tilde{\mu}(x) = \tilde{\mu}(k\Delta) \equiv \tilde{\mu}(k) = \sum_{i=1}^p a_i \tilde{\mu}(k-i) + \sigma \epsilon(k)$$

where  $k$  is the sample index,  $\Delta$  is the spatial discretization,  $\sigma$  is the standard deviation of the input noise, and  $\epsilon(k)$  is a zero-mean noise input with standard deviation of one. The model parameters to be determined are the  $a_i$  and  $\sigma$ .

The AR model is very suitable for real-time simulation and rendering, and typically one needs only a few parameters to reflect the roughness properties (as illustrated by Fig. 7). An AR(2) model is often sufficient in practice because it allows the modeling of a random sequence combined with one principal frequency.

<sup>5</sup>approximately  $< 15\%$  of the tiger’s diameter

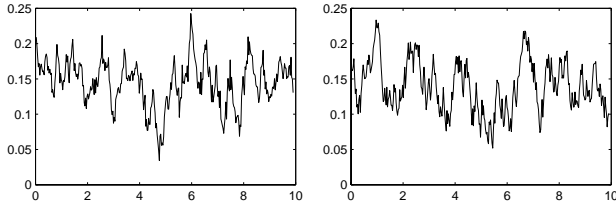


Figure 7: (left) Measured values of  $\mu_e$  for a 10 mm displacement along a rough section of the clay pot shown in Fig. 8. (right) Simulation of  $\mu_e$  with  $\mu = 0.142$  and  $\tilde{\mu}$  reproduced by an AR(2) model with  $a_1 = .783$ ,  $a_2 = .116$ , and  $\sigma = 0.0148$ .

## Measurement and Estimation

Friction and surface roughness are noticeable in terms of the forces they produce on a contacting instrument. Hence we can measure them in the same way: the robotic system performs a series of local rubs over the object surface with a probe (attached to a 6 DOF force sensor; Fig. 8) and the resulting force profiles are then analyzed.

The object’s surface mesh representation is used to plan “where and how” to do the rubbing. At present, the system assigns a contact texture model to each mesh vertex. This is determined either by explicit measurement, or by the interpolation of models at nearby vertices. The system employs a process of “active exploration”, in which models are initially sampled over the object surface at a low resolution, with further, higher resolution sampling in areas where the model parameters change significantly.

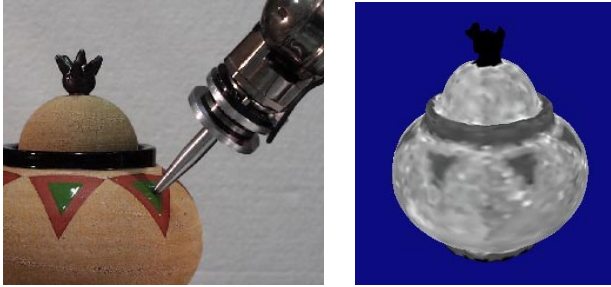


Figure 8: (left) Robotic system rubbing the pot to determine contact texture. (right) Surface friction map for the clay pot, in which the brightness is scaled according to the local value of  $\mu$  on the surface (with white corresponding to the maximum value of  $\mu = 0.5$ ). The enameled portions of the pot, with  $\mu \approx 0.09$ , are clearly visible. The ornament on top of the pot was not sampled.

The nominal friction coefficient  $\mu$  is estimated first. If the surface normal  $\mathbf{n}$  were known accurately, one could directly determine  $\mathbf{f}_n$  and  $\mathbf{f}_f$  in Eq.(2) and use this to solve for  $\mu$ . However,  $\mathbf{n}$  is known only approximately (due to uncertainty in the surface mesh and the actual contact location), and also varies along the path. To compensate for this, we stroke the surface *twice*: once in a forward direction and once in a reverse direction. At any point along the path, we then have a force value  $\mathbf{f}^+$  which was measured during the forward motion, and another value  $\mathbf{f}^-$  which was measured during the reverse motion.

$\mathbf{f}^+$  and  $\mathbf{f}^-$  each have components parallel to the surface

normal  $\mathbf{n}$ , along with friction components  $\mathbf{f}_f^+$  and  $\mathbf{f}_f^-$  which lie opposite to the motion directions and are perpendicular to  $\mathbf{n}$ . Now even if  $\mathbf{n}$  is unknown, and the magnitudes of  $\mathbf{f}^+$  and  $\mathbf{f}^-$  differ,  $\mu$  can still be estimated from the angle  $\theta$  between  $\mathbf{f}^+$  and  $\mathbf{f}^-$ :

$$\mu = \tan(\theta/2).$$

This calculation is quite robust, as it is independent of travel speed, the probe contact force and orientation, and of course the surface normal itself. By averaging the values of  $\mu$  obtained at various points along the path, a reasonable estimate for  $\mu$  over the whole path may be obtained. Our ability to determine  $\mu$  reliably is illustrated in Fig. 8.

The values of  $\mathbf{n}$  at each path point can also be estimated from the direction of  $(\mathbf{f}_f^+ + \mathbf{f}_f^-)$  and used to produce a smooth (typically quadratic) model of  $\mathbf{n}(x)$  along the path.

To calculate the effective friction, we use  $\mathbf{n}(x)$  to relate  $\mu_e$  to the observed force values  $\mathbf{f}$  acting on the probe tip:

$$\mathbf{f} = f_n[\mathbf{n}(x) - \mu_e \mathbf{u}_m(x)],$$

where  $\mathbf{u}_m(x)$  is the direction of motion along the path and  $f_n$  is the magnitude of the normal force. The unknowns in this equation are  $f_n$  and  $\mu_e$ . Solving for  $\mu_e$  at every path point  $x$  yields  $\mu_e(x)$ . An AR model is then fitted to  $\tilde{\mu}(x) = \mu_e(x) - \mu$ , using autoregressive parameter estimation via the covariance method (e.g., the `arcov` function in MATLAB).

## Rendering

To demonstrate the rendering of contact texture, we used a PHANToM haptic device to implement a virtual environment in which a user can rub an object with a point contact (represented as a red ball in the video). The GHOST software supplied with the PHANToM was used to perform collision detection and to generate the corresponding contact and frictional forces.

The friction value at the point of contact is generated by weighting the AR parameters at each vertex by the barycentric coordinates in the triangle. The distance traveled along the surface divided by the spatial discretization  $\Delta$  of the measurements determines the number of values to generate using the AR model. The last 2 values generated are then interpolated to obtain the effective coefficient of friction  $\mu_e$ . This value is passed to both the static and dynamic friction parameters in GHOST.

The resulting contact textures are quite convincing; it is easy to distinguish between the different surface preparations of the clay pot using the haptics alone. These results are best evaluated using the PHANToM haptic device (e.g., in our CAL demo) though Figs. 7 and 8 give a good indication of the sensitivity of our measurement technique.

## 7 Sound Modeling

### Selection of model structure

We model the contact sounds of an object by filtering an excitation (the “audio-force”) through a modal resonator bank which models the sonic response of the object. The details of this technique are explained in [39]. For this we need to acquire both a modal resonance model of the object, which will depend on its shape and internal composition, and an excitation model, which will depend mainly on the surface structure.

The modal model  $\mathcal{M} = \{\mathbf{f}, \mathbf{d}, \mathbf{A}\}$ , consists of a vector  $\mathbf{f}$  of length  $N$  whose components are the modal frequencies in Hertz, a vector  $\mathbf{d}$  of length  $N$  whose components are the (angular) decay rates in Hertz, and an  $N \times K$  matrix  $\mathbf{A}$ , whose elements  $a_{nk}$  are the gains for each mode at different locations. The modeled response for an impulse at location  $k$  is given by

$$y_k(t) = \sum_{n=1}^N a_{nk} e^{-d_n t} \sin(2\pi f_n t), \quad (3)$$

for  $t \geq 0$  (and is zero for  $t < 0$ ).

The surface texture generating the audio-force can be modeled in real-time by filtering an appropriate noise source with the location dependent autoregressive filter models obtained from the surface measurements described in Section 6.

For audio we need to know the audio surface texture at a much higher resolution than for haptics texture modeling. In the future we plan to measure the surface properties at higher resolutions and use  $\text{AR}(p)$  models acquired automatically. We have verified that such an approach yields good sound models but have not yet integrated this with the rest of the system. For now, we acquire audio-resolution surface properties by hand.

For the pot example shown in the accompanying video, we manually segment the surface into areas of substantially different textures and generate an excitation force from recordings made with a contact microphone at a reference speed and force and store them in wave-tables, just like audio signals. This approach is analogous to image-based rendering; as described below, the recorded excitation can be transformed by a few run-time interaction parameters to produce a realistic audio-force [39].

## Measurement

One way to estimate the modal model is to excite (i.e., hit) the object with an arbitrary force and measure both the audio response and the input force at the same high rate, and deconvolve the input force from the audio signal. This is the approach followed in [6]. However this can be delicate because measuring forces at audio frequencies requires very stiff force sensors, since otherwise the force signal can be contaminated by the resonances of the sensor itself. Deconvolution is also a numerically sensitive inverse problem. We have chosen instead to build a device for applying a light, highly peaked force which is a good finite approximation of an impulsive force; the measured audio signal can then be treated as the impulse response and used directly for parameter estimation. The device consists of a small push-type solenoid mounted at the tip of the robot arm; the solenoid is activated for a brief period so that the small plunger moves and hits the object ballistically and bounces off. The far field sound is recorded at 44.1 KHz using microphones mounted on the field measurement system. Fig. 9 shows the device pinging the clay pot. The robot systematically pings the object at the vertices of the base mesh. Several recordings are made at each mesh vertex for better estimation.

## Parameter estimation

We have developed a technique for estimating the modal model  $\mathcal{M}$  from the recorded impulse responses. The number of modes to extract is manually set to a large value and we discard the modes with very low gain which will not contribute to the sound. Precisely how many modes we want



Figure 9: Contact sound measurement

to use for synthesis depends on factors such as the desired accuracy of the reconstruction.

The modal frequencies are first estimated from the average power spectrum of the recordings (corrected for background noise) using peak identification with a quadratic interpolation of the discrete windowed Fourier transform. For a typical window size of 20ms this gives frequencies with errors of about 50Hz. This initial estimate is then refined by performing a phase reconstruction by fitting complex frequency trajectories of the windowed Fourier transforms of the signals with a sum of a small number of damped exponentials using the Steiglitz-McBride algorithm [36]. This will provide us with the estimated couplings  $\mathbf{A}$ , the dampings  $\mathbf{d}$ , and corrected estimates of the frequencies  $\mathbf{f}$ . In some cases very closely spaced modes arise because of approximate symmetries of the object which we resolve by fitting each trajectory with multiple complex exponentials. This “phase unwrapping” procedure has been used before to obtain very accurate frequency estimates [3]. Our application differs in that we are interested also in the dampings and coupling amplitudes, and we also want to be able to resolve densely spaced frequencies into their separate components.

We have tested the accuracy of the parameter estimation on artificially constructed impulse responses in the form of Eq. 3 and found that the frequencies and dampings have errors no larger than 0.0001%, and the gains have errors of about 1%. See Fig. 10.

## Rendering

During simulation, an audio kernel filters the audio-force excitation — which is parameterized by contact parameters such as velocity and friction — through the modal resonator bank and produces audio in real time. The audio-force can be a short impulse for impacts, or a noise-like excitation for scraping. Filtering with a modal reson bank can be computed very efficiently with an  $O(N)$  algorithm [14, 5, 40] for a model of  $N$  modes. Details of our contact sound rendering techniques are described in [39].

The geometrical locations on the surface of the object are mapped to points in the “timbre-space” of the object, which we define as the space spanned by the gain vectors  $\mathbf{a}$ . This is done by associating gains  $a_{nk}$  with each mesh vertex  $k$  at which the sounds were sampled during the measurement. In this manner we model the observed timbre shifts in the sound



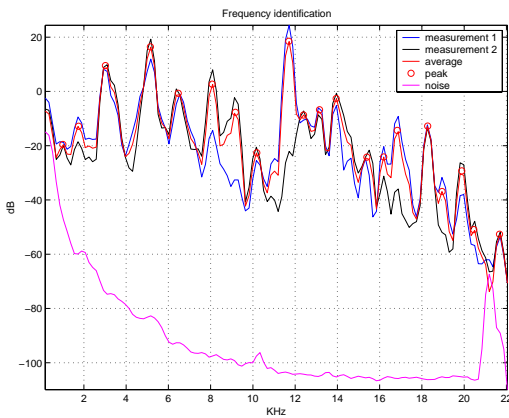


Figure 10: The power spectrum of two recorded impulse responses, their average, and the power spectrum of the background noise. The 20 most important peaks are indicated on the graph. The “best” peaks are those considered to stand out from their surrounding most clearly by “local height”

when an object is excited at different locations. Sudden jumps in the timbre during scraping can be heard clearly if we switch gain vectors discretely. We therefore utilize a form of “audio anti-aliasing”: at locations between mesh vertices we smoothly interpolate the gains from the vertex gains, using the barycentric coordinates of the location in the triangle. Note that because the gains  $a_{nk}$  at different vertices share the same modal frequencies, there is no need for frequency interpolation.

If the AR(2) filters turn out to be resonances [35], we can scale the resonance frequency measured at a reference contact speed with the actual contact speed in the simulation. This produces the effect of a shift in “pitch” dependent on the sliding velocity. If the AR(2) filters are not resonances (i.e., if their poles are real), which will occur if there is no prominent characteristic length scale in the surface profile, this model does not produce the illusion of scraping at a changing speed. The perceptual cue for the contact speed seems to be contained in the shifting frequency peak. We have found that higher order AR models in such cases will find these peaks, but we have not completed this investigation at the time of writing.

If an audio-force wave-table is used, it is pitch-shifted using linear sample interpolation to correspond to the actual simulation contact speed and the volume is adjusted proportional to the power-loss as determined by friction and speed [39].

## 8 Conclusions

We have described a system for modeling the interaction behavior of 3D objects by scanning the behavior of real objects. Modeling interaction behavior is essential for creating interactive virtual environments, but constructing such models has been difficult. We show how a variety of important interaction behaviors, including deformation, surface texture for contact, and contact sounds can be effectively scanned. We provided a description of the complete modeling process which could be used to construct these types of models. We also described our own measurement facility which automates many of the steps in measuring contact interaction behavior using robotics.

We believe that the techniques described in this paper could greatly improve the way virtual environments and animations are created. In addition to geometry and appearance, our methods will allow behaviors to be essential and easily obtained properties of virtual objects. Our methods make it feasible to build compelling interactive virtual environments populated with a large number of virtual objects with interesting behavior.

## References

- [1] M.J. Black and P. Anandan. The Robust Estimation of Multiple Motions: Parametric and Piecewise-smooth Flow Fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.
- [2] Y. Bouguet and P. Perona. 3D Photography on Your Desk. In *Proc. ICCV98*, pages 43–50, 1998.
- [3] J.C. Brown and M.S. Puckette. A High Resolution Fundamental Frequency Determination Based on Phase Changes of the Fourier Transform. *J. Acoust. Soc. Am.*, 94(2):662–667, 1993.
- [4] M. Cohen, M. Levoy, J. Malik, L. McMillan, and E. Chen. Image-based Rendering: Really New or Deja Vu? *ACM SIGGRAPH 97 Panel*, pages 468 - 470.
- [5] P.R. Cook. Integration of Physical Modeling for Synthesis and Animation. In *Proceedings of the International Computer Music Conference*, pages 525–528, Banff, 1995.
- [6] P.R. Cook and D. Trueman. NBody: Interactive Multi-directional Musical Instrument Body Radiation Simulations, and a Database of Measured Impulse Responses. In *Proceedings of the International Computer Music Conference*, San Francisco, 1998.
- [7] S. Cotin, H. Delingette, J-M Clement, V. Tasseti, J. Marescaux, and N. Ayache. Geometric and Physical Representations for a Simulator of Hepatic Surgery. In *Proceedings of Medicine Meets Virtual Reality IV*, pages 139–151. IOS Press, 1996.
- [8] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH 96 Conference Proceedings*, pages 303–312, 1996.
- [9] K.J. Dana, B. van Ginneken, S.K. Nayar, and J.J. Koenderink. Reflectance and Texture of Real-world Surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.
- [10] P. Debevec, T. Hawkins, C. Tchou, H-P Duiker, W. Sarokin, and M. Sagar. Acquiring the Reflectance Field of a Human Face. In *SIGGRAPH 2000 Conference Proceedings*, pages 145–156, 2000.
- [11] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. In *SIGGRAPH 96 Conference Proceedings*, pages 11–20, 1996.
- [12] T.A. Funkhouser, I. Carlbom, G. Pingali, G. Elko, M. Sondhi, and J. West. Interactive Acoustic Modeling of Complex Environments. *J. Acoust. Soc. Am.*, 105(2), 1999.
- [13] M. Garland and P.S. Heckbert. Surface Simplification Using Quadric Error Metrics. In *SIGGRAPH 97 Conference Proceedings*, pages 209–216, 1997.

- [14] W.W. Gavett. Synthesizing Auditory Icons. In *Proceedings of the ACM INTERCHI 1993*, pages 228–235, 1993.
- [15] D.P. Greenberg, K.E. Torrance, P. Shirley, J. Arvo, J.A. Ferwerda, S. Pattanaik, E.P.F. LaFortune, B. Walter, S. Foo, and B. Trumbore. A Framework for Realistic Image Synthesis. In *SIGGRAPH 97 Conference Proceedings*, pages 477–494, 1997.
- [16] I. Guskov, K. Vidimce, W. Sweldens, and P. Schroder. Normal Meshes. In *SIGGRAPH 2000 Conference Proceedings*, pages 95–102, 2000.
- [17] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise Smooth Surface Reconstruction. In *SIGGRAPH 94 Conference Proceedings*, pages 295–302, 1994.
- [18] D.L. James and D.K. Pai. ARTDEFO, Accurate Real Time Deformable Objects. In *SIGGRAPH 99 Conference Proceedings*, pages 65–72, 1999.
- [19] D.L. James and D.K. Pai. A Unified Treatment of Elastostatic Contact Simulation for Real Time Haptics. *Haptics-e, The Electronic Journal of Haptics Research* ([www.haptics-e.org](http://www.haptics-e.org)), 2001. (To appear).
- [20] J. Lang and D. K. Pai. Estimation of Elastic Constants from 3D Range-Flow. In *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, 2001.
- [21] A. Lee, H. Moreton, and H. Hoppe. Displaced Subdivision Surfaces. In *SIGGRAPH 2000 Conference Proceedings*, pages 85–94, 2000.
- [22] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *SIGGRAPH 98 Conference Proceedings*, pages 95–104, 1998.
- [23] M. Levoy, et al. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *SIGGRAPH 2000 Conference Proceedings*, pages 131–144, 2000.
- [24] D. K. Pai, J. Lang, J. E. Lloyd, and J. L. Richmond. Reality-based Modeling with ACME: A Progress Report. In *Proceedings of the Intl. Symp. on Experimental Robotics*, 2000.
- [25] D. K. Pai, J. Lang, J. E. Lloyd, and R. J. Woodham. ACME, A Telerobotic Active Measurement Facility. In *Proceedings of the Sixth Intl. Symp. on Experimental Robotics*, 1999.
- [26] Z. Popovic and A. Witkin. Physically Based Motion Transformation. In *SIGGRAPH 99 Conference Proceedings*, pages 11–20, 1999.
- [27] C. Rocchini, P. Cignoni, C. Montani, and P. Scopigno. Multiple Textures Stitching and Blending on 3D Objects. In *10th Eurographics Workshop on Rendering*, pages 173–180, Granada, Spain, 1999.
- [28] G. Roth and E. Wibowoo. An Efficient Volumetric Method for Building Closed Triangular Meshes from 3-D Image and Point Data. In *Proc. Graphics Interface*, pages 173–180, 1997.
- [29] P.J. Rousseeuw. Least Median of Squares Regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984.
- [30] P.J. Rousseeuw and K. Van Driessen. Computing LTS Regression for Large Data Sets. Technical report, University of Antwerp, 1999.
- [31] H. Rushmeier, F. Bernardini, J. Mittleman, and G. Taubin. Acquiring Input for Rendering at Appropriate Levels of Detail: Digitizing a Pietà. *Eurographics Rendering Workshop 1998*, pages 81–92, 1998.
- [32] D.C. Ruspini, K. Kolarov, and O. Khatib. The Haptic Display of Complex Graphical Environments. In *SIGGRAPH 97 Conference Proceedings*, pages 345–352, 1997.
- [33] Y. Sato, M.D. Wheeler, and K. Ikeuchi. Object Shape and Reflectance Modeling from Observation. In *SIGGRAPH 97 Conference Proceedings*, pages 379–388, 1997.
- [34] H. Spies, B. Jähne, and J.L. Barron. Dense Range Flow from Depth and Intensity Data. In *International Conference on Pattern Recognition*, pages 131–134, 2000.
- [35] K. Steiglitz. *A Digital Signal Processing Primer with Applications to Digital Audio and Computer Music*. Addison-Wesley, New York, 1996.
- [36] K. Steiglitz and L.E. McBride. A Technique for the Identification of Linear System. *IEEE Trans. Automatic Control*, AC-10:461–464, 1965.
- [37] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically Deformable Models. In *Computer Graphics (SIGGRAPH 87 Proceedings)*, volume 21, pages 205–214, 1987.
- [38] T.R. Thomas. *Rough Surfaces*. Imperial College Press, London, second edition, 1999.
- [39] K. van den Doel, P.G. Kry, and D.K. Pai. FoleyAutomatic: Physically-based Sound Effects for Interactive Simulations and Animations. In *SIGGRAPH 2001 Conference Proceedings*, 2001.
- [40] K. van den Doel and D.K. Pai. The Sounds of Physical Shapes. *Presence*, 7(4):382–395, 1998.
- [41] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional Scene Flow. In *International Conference on Computer Vision*, pages 722–729, 1999.
- [42] M. Yamamoto, P. Boulanger, J.-A. Beraldin, and M. Rioux. Direct Estimation of Range Flow on Deformable Shape from a Video Rate Range Camera. *PAMI*, 15(1):82–89, 1993.
- [43] Y. Zhang and C. Kambhamettu. Integrated 3D Scene Flow and Structure Recovery from Multiview Image Sequences. In *Computer Vision and Pattern Recognition*, volume 2, pages 674–681, 2000.

# The *AHI*: An Audio And Haptic Interface For Contact Interactions

*Derek DiFilippo and Dinesh K. Pai*

Department of Computer Science  
University of British Columbia  
2366 Main Mall.

Vancouver, BC V6T 1Z4, Canada

Tel: +1-604-822-6625

*{difilip | pai}@cs.ubc.ca*

## ABSTRACT

We have implemented a computer interface that renders synchronized auditory and haptic stimuli with very low (0.5ms) latency. The audio and haptic interface (AHI) includes a Pantograph haptic device that reads position input from a user and renders force output based on this input. We synthesize audio by convolving the force profile generated by user interaction with the impulse response of the virtual surface. Auditory and haptic modes are tightly coupled because we produce both stimuli from the same force profile. We have conducted a user study with the AHI to verify that the 0.5ms system latency lies below the perceptual threshold for detecting separation between auditory and haptic contact events. We discuss future applications of the AHI for further perceptual studies and for synthesizing continuous contact interactions in virtual environments.

**KEYWORDS:** User Interface, Haptics, Audio, Multimodal, Latency, Synchronization

## INTRODUCTION

When two objects collide, we have a contact interaction. Everyday lives are full of them: pulling a coffee cup across a table, tapping our fingers on a computer keyboard, etc. These contact interactions can produce characteristic sounds and forces that communicate information about our relationship with rigid objects and the surrounding environment. By using our ears and hands we can tell if the coffee cup was placed safely on the table or if the table is made of glass or wood, for example. Depriving someone of this sensory feedback from his or her interactions could

limit their ability to navigate and control their environment.

An effective computer interface for interacting with a simulated environment would allow one to tap and scrape on virtual objects in the same way one can tap and scrape on real objects. In addition to providing visual feedback, the interface would also create realistic auditory and haptic cues. These cues would be synchronized so that they appear perceptually simultaneous. They would also be perceptually similar -- a rough surface would both sound and feel rough. This type of interface could improve the amount of control a user could exert on their virtual environment and also increase the overall aesthetic experience of using the interface. For example, a system designer might wish to use this interface to represent remote objects to discerning user groups varying from NASA scientists to e-commerce customers. In this paper, we present an experimental audio and haptic interface (AHI) for displaying sounds and forces with low latency and sufficient realism for interactive applications.



Figure 1: The AHI in a typical configuration.

Figure 1 shows the AHI in a typical configuration. The user grips the handle with their right hand and moves it in

the plane as they would move a computer mouse. The left side of the monitor screen shows a Java graphical user interface for loading sound models and controlling interaction parameters and the right side shows a graphical window for viewing haptic forces and audio signals in real-time.

The novelty of the AHI lies in the tight synchronization of the auditory mode and the haptic mode. User interaction with the simulated environment generates contact forces. These forces are rendered to the hand by a haptic force-feedback device, and to the ear as contact sounds. This is more than synchronizing two separate events. Rather than triggering a pre-recorded audio sample or tone, the audio and the haptics change together when the user applies different forces to the object.

Building a device to a certain technical specification provides no guarantee for how a user will perceive the effect of the device. However, an over-specified device can be used to help establish lower bounds for human perception that give system builders a reliable design target. An analogy would be to the design of computer monitor hardware to support refresh rates of 60Hz. This refresh rate is well known to be sufficient for comfortable viewing and sufficient to simulate continuous motion. Our goal is to use the AHI to help establish similar perceptual tolerances for synchronized audio and haptic contact interactions.

The first part of this paper describes the AHI hardware for user input, models for calculating haptic and audio contact forces, and the control structure for rendering these forces. The second part of this paper presents experimental results that suggest the 0.5ms system latency of the AHI lies below the perceptual tolerance for detecting synchronization between auditory and haptic contact events. Establishing that our interface works below perceptual tolerance enables us to use it for psychophysical experiments for multimodal perception.

#### **HARDWARE**

User input to the AHI comes from a 3 degree of freedom (DOF) Pantograph device (Figure 2). The 5-bar mechanism is based on a design by Hayward [14] but extended to 3 DOF to our specification. It reads 3 DOF of position as user input, and renders 3 DOF of forces as output. The user can move the handle in the plane as well as rotate the handle. There are two large Maxon motors attached to the base of the Pantograph which apply forces on the handle via the 5-bar linkage. A small motor in the handle can exert a torque on the handle as well. The device, therefore, is complete for rigid motions in the plane, i.e., it can render the forces and torque due to any contact with a rigid body attached to the handle in a planar virtual world ("flatland"). We do not currently use the third rotational DOF for our work with the AHI.

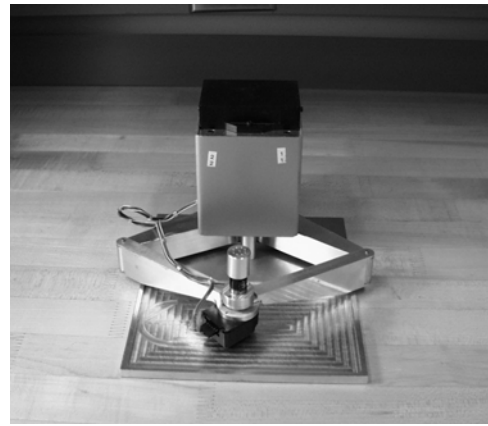


Figure 2: The Pantograph haptic device.

A dedicated motion control board (MC8, Precision Microdynamics), hosted in a PC running Windows NT, controls the AHI. The board has 14 bit analog to digital converters (ADCs) for reading the potentiometers attached to the base motors as well as quadrature decoders for reading the optical encoder which measures the handle rotation.

A SHARC DSP (Analog Devices 21061 running at 40MHz) on the MC8 synthesizes sounds and forces. Output voltages for controlling the Pantograph motors and for rendering audio are sent out through 14 bit digital to analog converters (DACs). The audio waveforms can be input directly to a sound system; in our set up they are sent to the soundcard of the computer for ease of capture and playback.

By using this specialized hardware we bypass the complications that arise from balancing the needs of real-time, deterministic response and ease of access from user-level software on a widely available operating system such as NT. The AHI control code is compiled for the DSP and has exclusive control over its resources. This allows us to precisely time our control algorithms as well as accurately diagnose inefficiencies and bugs. In particular, the overall system latency is 0.5ms for synchronized changes in audio and haptics.

#### **AUDIO SYNTHESIS**

We wish to simulate the audio response of everyday objects made out of wood, metal, ceramic, etc. Contact with these objects can be characterized by impulsive excitation of relatively few exponentially decaying, weakly coupled sinusoidal modes. Modal synthesis and impulse generation techniques have been developed for these types of percussive sounds [2]. We use the modal audio synthesis algorithm described in [20]. This algorithm is based on vibration dynamics and can simulate effects of shape, location of contact, material, and contact force. Model parameters are determined by solving a partial differential equation, or by fitting the model to empirical data [15].

The sound model assumes that the surface deviation  $y$  obeys a wave equation. We add a material-dependent decay coefficient to the wave equation to damp the sounds. The exponential damping factor  $d = f\pi\tan(\phi)$  depends on the frequency  $f$  and internal friction  $\phi$  of the material, and causes higher frequencies to decay more rapidly. The internal friction parameter is material dependent and approximately invariant over object shape. Equation 1 represents the impulse response of a general object at a particular point as a sum of damped sinusoids.

$$y(t) = \sum_{n=1}^{\infty} a_n e^{-d_n t} \sin(\omega_n t) \quad (1)$$

The sound model of an object consists of a list of amplitudes  $a_n$  and complex frequencies  $\Omega_n = \omega_n + id_n$ . Equation 2 shows how one complex frequency is computed. At time 0, the signal is the product of the frequency-amplitude  $a$ , and the contact force  $F(0)$ . At each successive time step (determined by the sampling frequency  $F_s$ ), the signal is the sum of a decayed version of the previous signal plus a new product of amplitude and contact force. The model responds linearly to input force  $F(k)$ . Once we have the model parameters, all we need to begin synthesizing sounds is a series of contact forces to plug into the right-hand side of the recursion. The output signal at time  $k$  is  $Re(\sum y_n(k))$ , with the sum taken over all computed frequencies.

$$\begin{aligned} y_n(0) &= a_n F(0) \\ y_n(k) &= e^{i \frac{\Omega_n}{F_s} k} y_n(k-1) + a_n F(k) \end{aligned} \quad (2)$$

This synthesis algorithm has two benefits. First, it is linear. The computed audio is the discrete convolution of the force history with the impulse response. There is a natural relationship between the input forces and the output signal; this relationship would not be as straightforward if we used sample-based synthesis. The linearity also makes it efficient. With a basis change, an audio signal can be computed with 2 multiplications and 3 additions per complex frequency, per sample. Second, the audio quality can degrade gracefully. If DSP time is running short we can compute fewer frequencies, resulting in a smooth loss of detail rather than sudden audio dropouts.

### HAPTIC FORCE SYNTHESIS

As the user moves the Pantograph handle we need to compute the contact forces resulting from these interactions, and then render them as forces on the handle of the Pantograph by exerting torques on its base joints. These computations take place in two coordinate frames. One is the world frame of xy-coordinates and the other is the Pantograph frame of joint angles. The simulated environment uses the world frame, but the control code

only knows about joint angles. We need a forward kinematic mapping that gives the xy-position of the handle as a function of base joint angles, as well as a differential kinematic mapping that gives the base joint torques as a function of applied force to the handle.

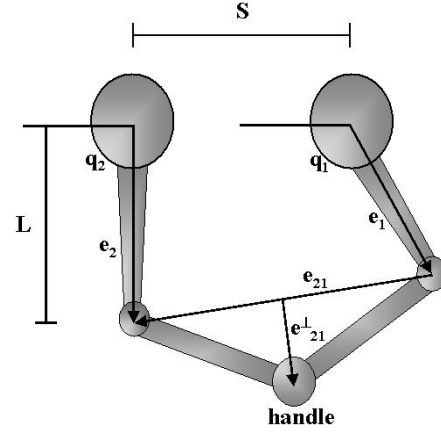


Figure 3: Pantograph kinematics.

For the forward kinematic mapping, we specify the base joint of motor 1 as the origin of the world frame. There is a geometric constraint that allows us to compute the position of the handle: the vector pointing from elbow 1 to elbow 2 ( $e_{21} = e_2 - e_1$ ) in the world frame is always perpendicular to vector pointing to the handle from the midpoint of  $e_{21}$ . If  $q_1$  and  $q_2$  are the base joint angles, then the elbows become  $e_1 = (L\cos(q_1), L\sin(q_1))$  and  $e_2 = (L\cos(q_2), L\sin(q_2))$  where  $L$  is the length of the proximal arms, and  $S$  is the separation between the two base joints. Setting  $e_{21}^\perp$  as the vector pointing from the midpoint of  $e_{21}$  to the handle  $h$ , we have  $h(q) = e_1 + 0.5e_{21} + e_{21}^\perp$ . This expression for  $h$  in terms of joint angles  $q$  has a simple geometric interpretation, as shown in Figure 3.

Once we have the handle coordinates, and compute a contact force  $F$ , we need to transform this force into base joint torques  $\tau$  for rendering. The Jacobian  $J = \partial h(q) / \partial q$  of the forward kinematic mapping relates forces to torques by  $J^T F = \tau$ . The details of constructing the Jacobian for the Pantograph are quite general and are covered in basic robotics texts [11]. In our particular implementation, we can avoid the expense of computing the partials of  $h(q)$  by exploiting the structure of the Jacobian. Details are removed here for the sake of brevity.

### NORMAL FORCES

For interactions normal to the surface of a plane, a spring/damper/impulse combination constrains the user to the surface by applying a penalty force. If the normal displacement past the surface is  $x_n$ , and the current normal velocity is  $v_n$ , then the haptic constraint force is  $F = Kx_n + Dv_n$  where  $K$  and  $D$  are spring and damping constants. For 10ms after a new contact, we add a unilateral impulse  $Pv_n$



to the spring/damper combination. This technique is known to increase the perception of haptic stiffness without introducing closed-loop instabilities that can occur with large spring coefficients [16].

### TANGENTIAL FORCES

For interactions tangential to the surface of a plane, we have implemented a stick-slip friction model to provide force feedback. Stick-slip models exhibit two states: *slipping* and *sticking*. Specifying a stick-slip model requires defining state transition conditions. In general, a virtual proxy point connects the real contact point to the surface. In the sticking state, the real contact point separates from the proxy and frictional force opposes further motion proportional to the separation. When a contact point is in a sliding state, the virtual proxy slides along with it.

Hayward's stick-slip model only uses displacements to determine state transitions [6]. If we define  $z = x_k - x_{proxy}$  as the displacement between the real contact point and the proxy and  $z_{max}$  as the maximum displacement then the update for the next proxy point is  $x_{proxy} = x_k \pm z_{max}$  if  $\alpha(z)/z/ > 1$  (slipping), or  $x_{proxy} = x_{proxy} + |x_k - x_{k-1}|/\alpha(z)/z/$  otherwise (sticking). Once the displacement between the proxy and real contact point passes a maximum the contact becomes fully tense and enters the slipping state. The proxy point and the real contact point move together, separated by  $z_{max}$ . For displacements less than the maximum the proxy point does not move much; this is the sticking state. The non-linear adhesion map  $\alpha(z)$  allows the proxy point to creep between these two regimes.

We have only selected two of many possible haptic force models that could be used by the AHI to represent surface properties. Stochastic models for haptic textures based on filtered noise and fractional Brownian motion are well known in the haptics community [5,18]. For example, perturbing the normal force by sampling from a Gaussian distribution generates haptic sandpaper. Friction models have a long history [1]. The algorithms and techniques we implement for planar normal and tangential forces are intended to be incorporated with more sophisticated applications that manage their own collision detection between complex polygonal geometries. After collision detection, these applications could use any suitable model for computing local normal and tangential forces.

### AUDIO FORCE SYNTHESIS

Naively using the raw normal forces to synthesize audio produces unsatisfying results. There are two properties of our synthesized normal forces that cause trouble. This section will describe how we filter out these two properties from the haptic force. The filtered result is the *audio force* that we convolve with the stored impulse response in Equation 2. The two properties of the haptic force that we wish to filter are as follows: (1) a spurious impulse results when the user breaks contact with the surface and the haptic

force discontinuously drops to zero, and (2) high frequency position jitter.

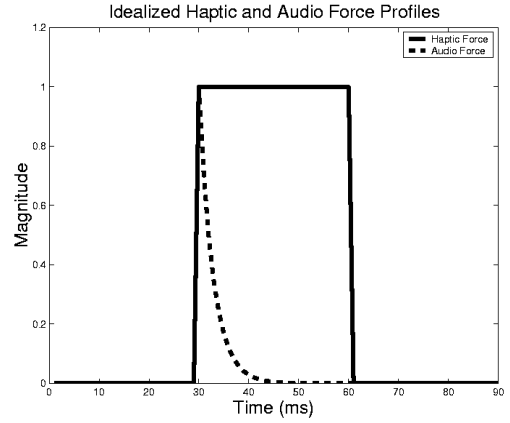


Figure 4(a): Idealized haptic and audio force profiles.

Figure 4(a) plots an idealized haptic contact force. At 30ms the user comes into contact with the surface and stays in contact for another 30ms. Convolution of this square wave profile with the impulse response of the surface will produce a spurious second “hit” when the user breaks contact. We introduce an attenuation constant  $\beta$  to allow the audio force to smoothly move to zero during sustained contact. If  $t$  is the elapsed time since contact, then the current audio force is the current haptic force attenuated by  $\beta^t$ . We have found that attenuating the audio force starting 10ms after a new contact with  $\beta = 0.85$  (half-life of 5ms) produces good results. Waiting 10ms before decaying improves the quality of impulsive contacts, whereas decaying the audio force immediately upon contact excessively reduces the overall amplitude and dynamic range of the resulting audio signal.

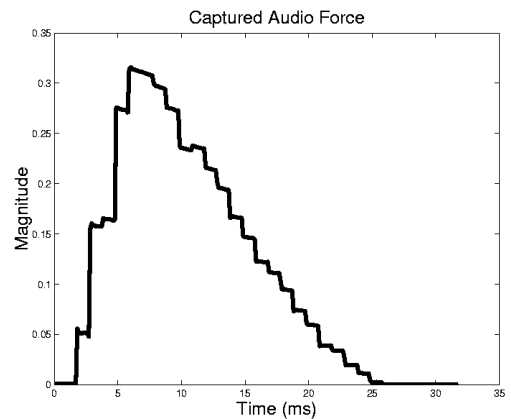


Figure 4(b): A captured audio force profile.

Haptic instabilities and signal noise generate sustained low amplitude, high frequency jitter in the position readings. These high frequencies are passed into the haptic force profile by the linear spring constant. Without filtering, this

noise becomes audible as crackling and popping while the user maintains a static contact with the surface. This low amplitude noise can be removed by truncation. Typically, we remove the 8 lowest order bits. Figure 4(b) plots a typical audio force profile. (The signal is not perfectly constant during each millisecond interval because it was captured as the input signal to a soundcard.) Using discrete optical encoders instead of potentiometers to read joint angles removes the need to truncate the position signals – the finite resolution of the encoders effectively truncates the signal for us.

### REAL-TIME SIMULATION

The basic control structure for the AHI real-time synthesis and simulation is interrupt-driven. There is a haptic interrupt service routine (HISR) that generates haptic and audio forces and an audio interrupt service routine (AISR) that convolves the audio force with the impulse response of the modelled object. Using these two separate interrupts we can synthesize the audio signal at a much higher rate than we generate haptic feedback. This section will describe the two interrupt routines shown in Figure 5.

The AISR and all DAC/ADC latches are synchronized to trigger at the audio control rate by using a programmable interval timer that counts at half the bus clock rate of 8.33 MHz. The AISR reads the Pantograph joint angles from

the ADCs and stores them in an array that contains a history of joint angle readings. Converting the DAC input to an equivalent floating point number requires 1 comparison, 2 multiplications, and 2 additions. The current audio force is then clipped to lie between 0.0 and 1.0 and truncated to remove low amplitude noise. This requires 2 comparisons and 2 multiplications. A discrete convolution step using this filtered audio force  $F(k)$  produces the output audio signal  $y_n(k)$ . This signal is placed in the DAC out. Computing the audio signal requires 2 multiplications and 3 additions per complex frequency, per sample. If the DSP is short on cycles, we can decrease the number of active frequencies. In our current scheme this isn't necessary -- there are no other competing processes for DSP time. Once a number of complex frequencies are selected the total amount of processing time is fixed and does not need to be adaptively adjusted. This would change if the DSP was also managing a complicated environment with graphics, collision detection, and rigid body dynamics.

Haptic interrupts trigger at an integer fraction of the audio control rate. The current joint angles are the mean of the array of joint angles captured during the AISR. From these filtered values, we use the forward kinematics of the Pantograph to compute the handle position. Since we only consider interactions with a plane, determining contact between the handle and the plane takes a sign check. If

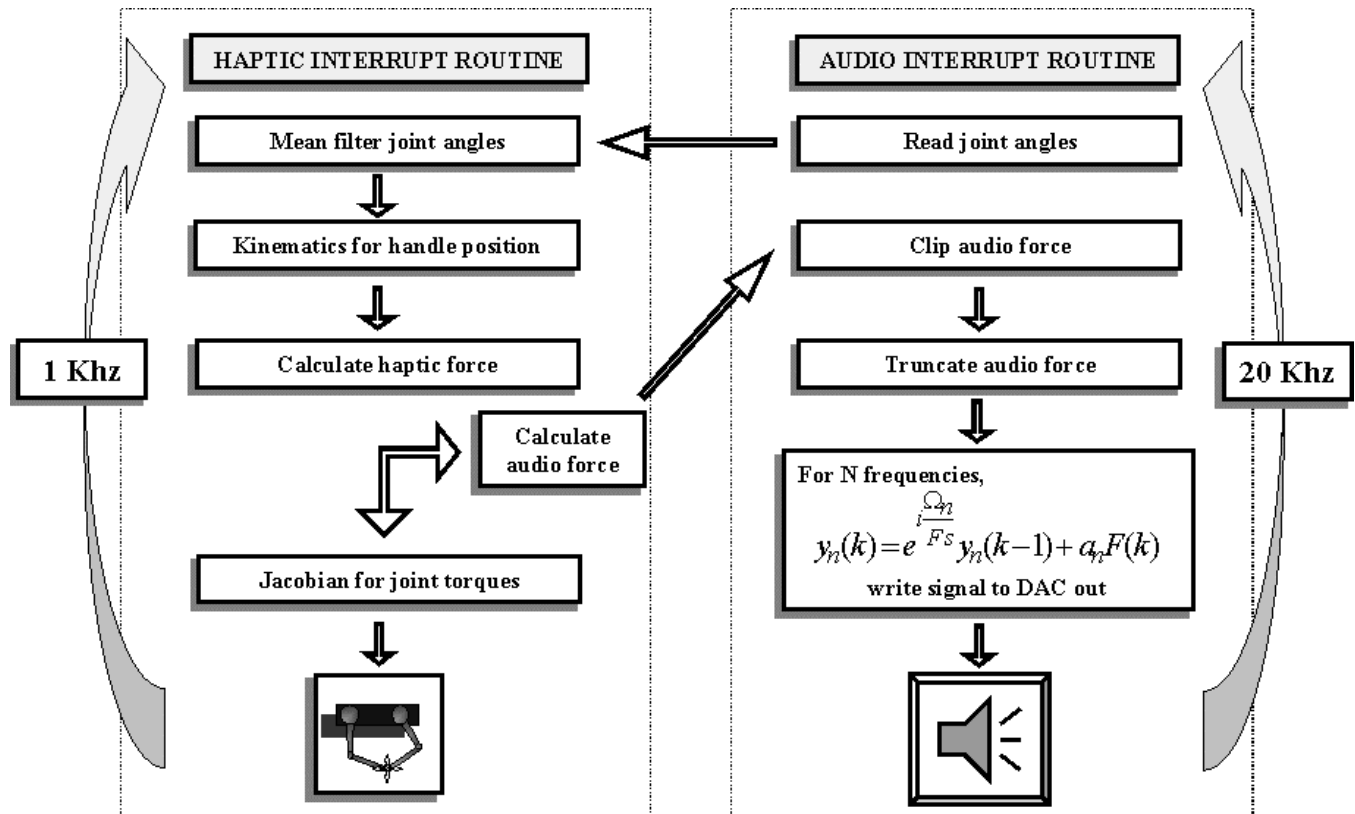


Figure 5: Flow of control for real-time synthesis and simulation.

there is contact, we compute the resulting normal force using a spring/damper/impulse model and a tangential force using a stick-slip friction model. Updating the current Jacobian takes 22 multiplications, 4 trigonometric calls, and 2 square roots. The Jacobian of the Pantograph translates the haptic force into motor torques. The voltages to generate these torques are written to the DACs. If there has been contact for  $(10+t)$  milliseconds, then  $\beta^t$  times the normal force plus the tangential force becomes the current audio force. The HISR writes the current audio force to a global variable shared with the AISR.

### CONTINUOUS CONTACT INTERACTIONS

We have experimented with some examples using the basic control structure just described to demonstrate how the AHI can generate continuous audio and haptic contact interactions. In the first example, the user scrapes the AHI handle across a sinusoidally modulated surface profile. In the second example, the user slides the handle across a surface with stick-slip friction. In both examples, we convolve the resulting audio force with the impulse response of a brass vase acquired from the University of British Columbia Active Measurement facility [15]. These examples have been informally tested in our laboratory. Haptic interrupts trigger at 1kHz and audio interrupts at 20kHz. Thus, there is a 1ms latency for changes in force and audio. Informally, the auditory and haptic stimuli are perceptually simultaneous.

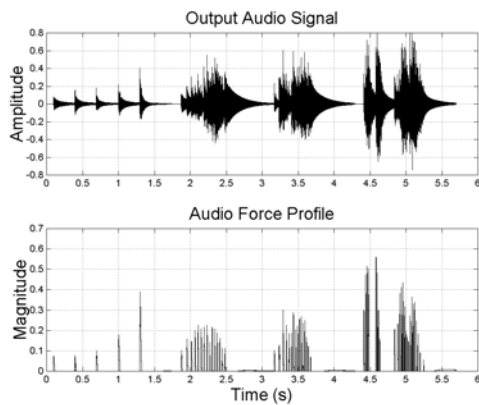


Figure 6: Output audio signal and audio force magnitude.

Figure 6 plots a captured audio force and the convolution of this force with the measured impulse response of the vase. We compute 20 modes for each audio sample. The user interaction in this example was five single strikes of increasing force normal to the surface, then tangential motion across the surface. The middle two bursts are slower scrapes back and forth, and the final two bursts are faster scrapes. The auditory signals produced in this fashion are satisfying. “Zipper” audio effects can be created by rapidly scraping on the surface. These synthesized audio signals compared favorably to live signals of tapping and

scraping along the ribbed surface of the vase with the tip of a pen.

Figure 7 shows captured audio signal and audio force magnitudes when interacting with the AHI and Hayward’s stick-slip friction model. Audio force decay was enabled for this captured signal, but only applied to the normal force component. The user interaction in this example was to slide the handle tangentially across a flat plane. A force hysteresis is apparent. The force increases as the displacement between the real and proxy contact point increases (sticking phase) and then discontinuously drops to zero during the slip phase. These discontinuities in the audio force create impulses in the audio signal. We used a similar model of a brass vase in this example to the one in Figure 6.

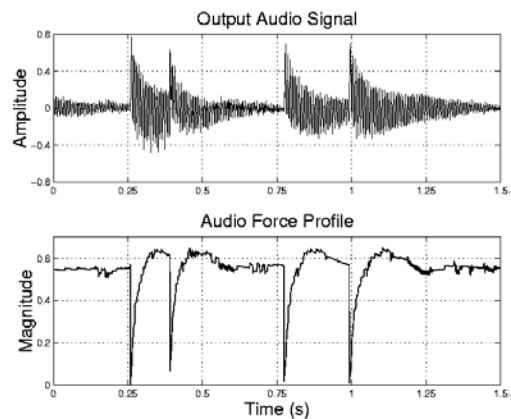


Figure 7: Output audio signal and audio force magnitude for interacting with the AHI and Hayward’s stick-slip friction model.

### USER STUDY

The remainder of the paper will describe a user study we conducted with the AHI. In this user study we tested the hypothesis that a 2ms latency lies below the perceptual tolerance for detecting synchronization between auditory and haptic contact events. We selected 2ms because it is larger than our 0.5 system latency, but a small enough interval to establish a lower bound. Briefly, a subject tapped on a virtual wall and received appropriate auditory and haptic stimuli, except that one of the two was delayed by 2ms. We tested the hypothesis that all subjects would perform at chance when asked to choose which stimulus came first.

### Participants

Twelve members of our department (2 females, 10 males) participated in the user study. Their mean age was 32 years, with a minimum age of 21 and a maximum age of 55. There was one left-handed participant. All twelve reported normal hearing and touch. The participants were not paid for their time.

## Apparatus and Stimuli

With a few software modifications, we used the AHI as described in the first half of this paper. The stimuli consisted of 24 contact events where the audio signal preceded the haptic by 2ms, and 24 contact events where the haptic signal preceded the audio signal by 2ms. The haptic control rate was 2kHz and the audio control rate was 20kHz, resulting in a 0.5ms system latency. We created precedence by delaying one of the two signals with a short circular buffer. Delaying the haptic signal did not cause any instability. We disabled tangential forces.

A vertical “wall” was positioned in the right half of the workspace. One set of 48 random locations of the wall within  $\pm 1\text{cm}$  were generated and used in the same order for all subjects. Haptic force-feedback was provided using the spring/damper/impulse combination. For audio, striking the wall was treated as striking a single point on an ideal bar, clamped at both ends. The contact point was at 0.61 of the length of the bar. For a given fundamental frequency, this simple geometry has an analytical solution for the frequencies and relative amplitudes of the higher partials. We used fundamental frequencies  $\omega_1$  of 1000Hz (High) and 316Hz (Low) and four higher partials.

As shown in equation 1, the decay of these frequencies are determined by a damping coefficient  $d = f\pi \tan(\phi)$ . We used two values of  $\tau_d = 1/(\pi \tan(\phi))$  that correspond to slow decay and fast decay: 300 (Fast), and 3 (Slow). These particular auditory stimuli were selected because they are a subset of those used for a study that connects variation of the coefficient  $\tau_d$  to auditory material perception [7].

## Experimental Design

The experiment used a two-alternative forced choice design. The subjects were asked to decide whether the audio signal preceded the haptic signal, or the haptic signal preceded the audio signal.

A three within-subject design was used with audio/haptic precedence, frequency, and damping as the within-subject variables. In total, there were 8 different stimuli presented to the user. Audio precedence coupled with one of four frequency/damping combinations (High + Fast, Low + Fast, High + Slow, Low + Slow), and haptic precedence coupled with the same four sound combinations. Six of each of these 8 different types were permuted once and used in the same order for all subjects for a total of 48 stimuli.

## Experimental Procedure

Subjects sat on a chair with the Pantograph on a desk in front of them. The Pantograph base was affixed to the desktop with a rubber sheet to minimize sliding and rotating. Subjects wore closed headphones (AKG K-240) for the audio signals and to minimize external sounds including those from the Pantograph device. We told the subjects that they would be striking a virtual wall, and that this would produce both haptic forces and audio signals.

They were told that, in each case, one of the stimuli preceded the other. We demonstrated how to hold the handle of the pantograph and how to make an impulsive strike to the wall. Then, the subjects were allowed to practice a few strikes with the headphones on. Finally, the experiment began.

Subjects were not told that there were equal numbers of stimuli types, nor were they told the number of repetitions in the experiment. No requirement on striking force was suggested -- subjects were free to strike the wall as hard or soft as they wished, as long as it was a single strike. There were no visual cues; however, the subjects were not blindfolded. After being read the the subjects were allowed to ask questions about the purpose of the experiment. If they expressed some concerns about their ability to discriminate between the two alternatives they were told that the discrimination task was designed to be difficult and to expect some ambiguity.

## RESULTS

Table 1 shows the mean number of correct responses, along with the maximum, minimum, and standard deviation of correct responses out of 48. Table 1 also shows the number of audio responses selected. Four different subjects were responsible for each of the maxima and minima.

	# Of Correct Responses	# Of Audio Responses Selected
Mean	21.08	23.58
Max	26	31
Min	17	16
Std	2.57	4.52

Table 1: Number of correct responses, and number of audio responses selected out of 48, for all 12 subjects.

We test the null hypothesis that the subjects perform at the chance level (each response is a pure guess), for each of the 12 subjects. By hypothesis, the mean number of correct responses  $\mu = 24$  and the standard deviation  $\sigma = 3.45$ . Using the normal approximation to the binomial distribution, we see that we can reject the hypothesis with a two-tailed test at the significance level  $\alpha = 0.05$  only if the sample mean is outside the interval  $\mu \pm 1.96\sigma = [17.21, 30.78]$ . Except for the lone subject with 17 correct responses, we cannot reject the hypothesis that the subjects are performing at chance. We note that a one-tailed test may be more appropriate since we want to know if the subjects can detect the precedence *better* than chance. With a one tailed test, we can not reject the hypothesis for any of the subjects.

## DISCUSSION

The results indicate that 2ms is a valid lower bound for the perceptual tolerance of synchronization latency of a contact interface like the AHI.

Other data suggests that the rate of decay of the auditory stimulus is a factor in the user responses, independent of the actual order of stimulus presentation. Table 2 contains the six most extreme number of correct responses, listed across stimuli. Sounds that decay more slowly are perceived as preceding the haptic stimulus and sounds that decay quickly are perceived as lagging the haptic stimulus, independent of the actual order of stimulus presentation. An increase in the audio decay rate (sounds decaying more quickly) reduces the total energy and total duration of the audio signal. It is possible that the subjects are using decay rate or total duration or total energy as a criterion for their response. The subject could be choosing the “loudest” signal (in terms of duration or energy) as the precedent stimulus. More studies are required to describe (and eventually understand) this effect.

Stimulus #	# Of Correct Responses	Stimulus Type
4	1	Audio + High + Fast
6	2	Haptic + High + Slow
11	10	Haptic + High + Fast
30	10	Haptic + Low + Fast
33	1	Haptic + Low + Slow
45	2	Haptic + Low + Slow

Table 2: The six most extreme values for number of correct responses, listed across stimuli. The number of correct responses is out of 12. Stimulus type is listed in order of precedence, frequency, and decay.

## FUTURE WORK

In this last section we consider further opportunities for using the AHI in perceptual studies of integrated audio and haptics, specifically for exploring multimodal texture and friction. One of the basic questions about cross-modal synchronization has been addressed recently, but many questions about similarity and synchronization between dynamic vibration stimuli remain.

Levitin, et al., have helped to identify the perceptual tolerance for synchronization between auditory and haptic contact events [10]. Our original plan was to conduct a very similar study. Nevertheless, the AHI is well suited to help establish similar perceptual tolerances for continuous audio and haptic contact interactions such as scraping and sliding over textured surfaces.

In the Levitin study, subjects manipulated a baton. They would strike a horizontal surface (containing a capacitor) with this baton. By tracking position, velocity, and acceleration, the experimenters could predict the time of actual impact. Using these predictions, a digitized sample of a stick striking a drum was played back to the subject at random temporal offsets varying between -200ms and 200ms. Subjects were asked to judge whether the baton strike and the audio sample occurred at the same or different times. The threshold where subjects considered the stimuli to be synchronous 75% of the time

corresponded to -19 and 38ms; that is, the interval between the audio preceding the haptics by 19ms, and the audio lagging the haptics by 38ms. When adjusted for response bias (using confidence ratings) the corrected thresholds for detecting synchrony are -25ms and 66ms. Levitin’s practical motivation for this study was to determine an upper limit for reliable perceptual synchronization that gives system designers a well-defined performance target. The 66ms 75% performance threshold is higher than what we expected partially based on our own experience with the AHI.

There may be an important difference in synchronization tolerances between active haptic devices with motors such as the AHI and passive devices such as the baton used in the Levitin study. The AHI’s motors do not operate quietly. Motor torques excite structural vibrations which produce sounds. It is known that the time resolution for successive audio clicks is on the order of 2ms and that this value is largely independent of frequency [13]. An intra-modal judgement would use both audio signals (one from the speakers, one from the motors) over the cross-modal judgement that compares the audio signal to the haptic signal. If this intra-modal judgement dominates the cross-modal judgements then it will be necessary to decrease the asynchrony to well below Levitin’s reported figure. Identifying and controlling asynchronies for active devices like the AHI does not directly address cross-modal perception, but given the preponderance of active haptic devices entering the market it would still be a valuable result to derive.

Our current simulation generates audio forces from haptic forces normal and tangential to a locally flat patch. Large scale surface features can consist of a collection of polygons which use our flat patch algorithms after collision detection. The spring/damper/impulse penalty method effectively parameterizes the normal component as surface hardness. Limited versions of small-scale surface features — friction and texture — have been implemented and described in this paper. We want simple and effective parameterizations of the force components as friction and roughness variables that remain relevant for auditory perception.

Hayward’s stick-slip model has been applied to the real-time physical modeling of violin bow-string interactions [17]. Bow-string interactions are some of the oldest studied examples of stick-slip friction. A good test for adding friction to our audio synthesis routine would be to simulate bowing a violin string and forcing it to resonate. Our AHI simulations of this sort of behaviour are not convincing yet — the audio signal in Figure 7 sounds more like a series of impacts than like the squeaking we associate with stick-slip phenomena. We might also use this model for synthesizing rough textures by imposing Gaussian noise perturbation on either the separation between the proxy and real contact, or on the spring coefficient that produces friction forces. Our

experience is that we will need another stage of force prefiltering to control the auditory roughness signal. Perhaps we could leverage work done in computer graphics on synthesizing fractional Brownian motion and fractals to gain more control over our signals [4].

Previous studies on the influence of auditory stimuli on haptic perception have used audio samples or tones triggered by contact events. The study by Miner, et al, used pitched tones and attack envelopes to simulate hard and soft sounds [12]. They found that “the auditory stimulus did not significantly influence the haptic perception”. The study by DiFranco, et al, triggered audio samples of contact events they recorded by hand [3]. They found that “sound cues that are typically associated with tapping harder surfaces were generally perceived as stiffer”. Both of these studies focus on the perception of hardness, which is a function of force on the user’s hand. These studies do not explore the perception of surface roughness, which in addition to being a function of force, can also be a non-trivial function of interaction speed and surface geometry. The spatial characteristics of the surface dominate roughness perception when using a bare finger. The speed of interaction in this case does not affect roughness perception. However, dynamic vibration effects as a function of speed are reported when interaction is mediated by a rigid probe [9]. These effects are complex and not fully understood. Increasing speed tends to render surfaces as smoother; however, unlike perception with the bare finger, the current effect tended to reverse itself as the interelement spacing increased.

A texture model for any haptic interface (not just the AHI) could use these perceptual results to inform their implementation. However, because the AHI tightly couples the auditory stimulus with the underlying physical process of collision, a simple grid produces haptic and auditory textures that vary as a function of both force and speed (Figure 6). Previous studies on perceiving auditory and haptic textures with the bare hand suggest that the subject will use the haptic texture before the auditory texture for roughness discrimination [8]. The similar and synchronized stimuli rendered by the AHI could be used to extend these results for multimodal vibration effects when a rigid probe mediates interaction.

Devising and verifying new multimodal friction and texture models will require several psychophysical studies. We believe the AHI in its current state provides an excellent platform for experimenting with and understanding these models.

## CONCLUSION

Designing compelling simulated environments is the high-level goal of this research. The representation and rendering of contact interactions comprises an essential component of any such simulation. Atomically representing the contact event as something that produces both sound

and force helps integrate auditory and haptic stimuli. We believe this is a natural way to think of representing and simulating contact. We have implemented a new interface that can render audio and haptic interaction atomically.

Our experimental results suggest that the AHI’s overall latency and synchronization between the auditory and haptic modes lies below the perceptible threshold. In the future, we will use the AHI to help explore perceptual synchronization and similarity between continuous auditory and haptic contact interactions.

## ACKNOWLEDGMENTS

Thanks to Susan Lederman for valuable discussions. Thanks to Paul Kry for timely debugging wisdom. This work was supported in part by grants from the IRIS NCE, and Natural Sciences and Engineering Research Council of Canada. Derek DiFilippo is supported by an NSERC scholarship and in part by the Advanced Systems Institute of British Columbia.

## REFERENCES

1. Armstrong, B., Dupont P. and Canudas de Wit C. A Survey of Models, Analysis Tools and Compensation Methods for the Control of Machines with Friction. *Automatica*. 30,7 (1994), 1083—1138.
2. Cook, P.R. Physically Informed Sonic Modeling (PhISM): Synthesis of Percussive Sounds. *Computer Music Journal*. 21,3 (1997), 38—49.
3. DiFranco, D.E. and Beauregard, G.L. and Srinivasan, M.A. The Effect of Auditory Cues on the Haptic Perception of Stiffness in Virtual Environments. *Proc. ASME Dynamic Systems and Control Division*, (1997).
4. Ebert, D.S., Musgrave, F.K., Peachey, D., Perlin, K. and Worley, K. *Texturing and Modelling*. AP Professional, New York NY, 1998.
5. Fritz, J.P. and Barner K.E. Stochastic models for haptic texture. *Proc. SPIE Int. Symp. on Intelligent Systems and Advanced Manufacturing*, Boston MA, November 1996.
6. Hayward, V. and Armstrong, B. A new computational model of friction applied to haptic rendering. *Experimental Robotics VI*, Lecture Notes in Control and Information Sciences, Springer-Verlag, NY, Vol. 250, 403—412.
7. Klatzky, R.L., Pai, D.K. and Krotkov, E.P. Hearing Material: Perception of Material from Contact Sounds. To appear in *Presence*, October 2000.
8. Lederman, S.J. Auditory Texture Perception. *Perception*. 8 (1979), 93—103.

9. Lederman, S.J., Klatzky, R.L., Hamilton, C.L. and Ramsay, G.I. Perceiving Surface Roughness via a Rigid Probe: Effects of Exploration Speed and Mode of Touch. *The Electronic Journal of Haptics Research*, 1, 1, (1999).
10. Levitin, D.J., MacLean, K. and Mathews, M. The Perception of Cross-Modal Simultaneity. To appear in *Int. Journal of Computing Anticipatory Systems*, 2000.
11. Murray, R.M., Li, Z. and Sastry, S.S. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Ann Arbor MI, 1994.
12. Miner, N., Gillespie, B. and Caudell, T. Examining the Influence of Audio and Visual Stimuli on a Haptic Display. *Proc. of the 1996 IMAGE Conf.*, Phoenix AZ, June 1996.
13. Pierce, J. Hearing in Time and Space. Chapter 8 of *Music, Cognition, and Computerized Sound*. The MIT Press, Cambridge MA, 1999.
14. Ramstein, C. and Hayward, V. The Pantograph: a large workspace haptic device for a multi-modal Human-computer interaction. *Conf. on Human Factors in Computing Systems ACM/SIGCHI*, Boston MA, April 1994.
15. Richmond, J.L. and Pai, D. K. Active Measurement of Contact Sounds. *Proc. of the 2000 IEEE Int. Conf. on Robotics and Automation*, San Francisco, April 2000.
16. Salcudean, S.E., and Vlaar, T. On the Emulation of Stiff Walls and Static Friction with a Magnetically Levitated Input-Output Device. *ASME J. Dynamic Systems, Meas., Control.* 119 (1997), 127—132.
17. Serafin, S., Vergez, C. and Rodet, X. Friction and Application to Real-time Physical Modeling of a Violin. *Int. Computer Music Conf.*, Beijing, October 1999.
18. Siira J. and Pai D.K. Haptic Textures — A Stochastic Approach. *IEEE International Conference on Robotics and Automation*, Minneapolis MN, April 1996.
19. Srinivasan, M.A. and Basdogan C., Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges. *Comput. & Graphics.* 4, 21 (1997), 393—404.
20. van den Doel, K. and Pai, D.K. The Sounds of Physical Shapes. *Presence* 7,4 (1998), 382—395.

# Haptics for Scientific Visualization

Russell M. Taylor II  
University of North Carolina at Chapel Hill  
<http://www.cs.unc.edu/~taylorr>

## The vision

The use of haptics in scientific visualization has barely begun, yet it already shows great promise. While haptics may be useful for pure visualization tasks, its true power comes from its ability to allow the user to simultaneously push on and be pushed by the computer. This allows direct and immediate control and sensing of parameters in a simulation.

Somewhere in the not-too-distant future... your simulation has been running for days on the supercomputer. You decided to step in and take a peek to make sure it is going in the right direction. Stepping into the VR station, you bring up a 3D, tracked view of isosurfaces on the data set. To your surprise, a bubble has formed in the wave front. You can't see anything obvious, so you use your force probe to feel the area around the bubble. Tiny vibrations at the site indicate instability in the simulation due to the simulation grid spacing being too sparse. You can fix the instability by re-meshing, but the bubble is still there. You need to push it back up to match the wave, but you don't want to add too much energy at any time step in order to avoid introducing more instability. By mapping the derivative of added energy to force, you feel the amount you are adding and are able to smoothly pull the simulation back on track.

These musings give glimpses of where haptic visualization may lead us. Let's take a look as where we are and how we got here. Along the way, we'll see concrete examples of the usefulness of haptics in scientific visualization.

## Haptic display history (highlights)

The history of haptic display and the study of human haptic and tactile perception is wide and varied, and covered in other sections of this book. Presented here are some highlights of this history that deal closely with the use of haptics for scientific visualization or form the basis for techniques that may be employed for this purpose.

### *The Sandpaper system for texture synthesis*

Margaret Minsky developed the *Sandpaper* system for synthesizing texture in a force-feedback display system, culminating in her 1995 dissertation at MIT on the subject. (Minsky, Ouh-young et al. 1990; Minsky 1995) This system was a 2D force-feedback joystick that allowed users to feel around on 2D textures that were computed or read from images. The "texture" in this system included both large-scale surface shape information and small-scale texture information. Lateral force was presented based on the local slope of the surface height map, with the joystick pushing in the direction that would be "down" on the surface. The amount of force was greater when the surface was more steeply sloped. Even though only lateral forces were presented, users perceived that they were moving a stylus up and down over bumps and dips in a surface.

Of interest for scientific visualization, screen-based sliders (adjusting the viscosity or spatial frequency of a computed texture, for example) could control Sandpaper's texture parameters. If the value of these parameters were mapped to spatially varying scalar or vector fields defined on a surface, the result would be a texture field whose properties depended on (and displayed) the underlying data sets. This has the potential to allow the display of multiple data sets on the same surface.

The user studies performed with the *Sandpaper* system can inform the selection of mappings from data values to texture parameters. Minsky explored the perception of surface roughness and found that for the case of small periodic ridges the roughness percept can be almost entirely predicted by the maximum lateral force encountered while feeling the simulation. She also proposed a framework for haptic models based on both physically-based and perceptually-based representations of the haptic properties of objects and situations. (Minsky 1995)



## Remote micro-machining

Collaboration between the University of Tokyo and George Washington University resulted in a system that provided local visual, haptic and auditory presentation of the action of a remote milling tool. (Mitsubishi, Hatamura et al. 1993) Their goal was the creation of a teleoperation system for remote control of a milling machine. Due to the latency of transmission and the small amount of available communication bandwidth, they used an intermediate model to provide force and auditory feedback. Furthermore, they pointed out that at very small scales, friction, viscosity and static charge may play a much more important role than inertial forces, so direct mapping of forces may be misleading and some translation may be required to allow “natural” operation by the user. This amounts to building a simulation of milling operation that the user interacts with, and whose parameters are driven from the actual, remote milling operation. Thus, their work gives an example of visualizing the behavior of a remote milling tool based on a local model.

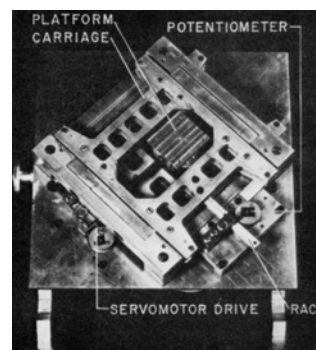
They performed averaging on the force signal to remove the strong 33.3 Hz component due to rotation of the cutting tip. They examined the offsets in the tool to determine whether chatter is occurring and simulated chatter at the user's end when it occurred. Because they were using prediction to overcome latency, and because prediction can produce incorrect motion, safeties were put in place on the device end to prevent over-force or other dangerous conditions at the tool end. When performing machining operations, the degrees of freedom of the tool were reduced relative to those of the user (the drill would only go up and down, for example) in order to increase precision over that of the human motor system. The user could also specify start and endpoints for a milling trajectory and then have the tool follow a nearest-neighbor path along this trajectory with speed controlled by the user.

Tool rotation speed was encoded and displayed to the user as a sound whose tone varied to indicate speed. They also encoded information in sound location, with sounds to the right meaning the tool was moving to the right. Discontinuous sound caught the user's attention and was used to emphasize rapid changes in velocity, which might indicate dangerous conditions.

## Display of force fields

An early haptic feedback application developed at the University of North Carolina at Chapel Hill allowed the user to feel the effects of a 2D force field on a simulated probe, and was used to teach students in an introductory Physics course. (Brooks, Ouh-Young et al. 1990) These experiments were performed using a 2D sliding-carriage device that used potentiometers for position and servomotors for force presentation. Experimental results showed that this feedback improved the understanding of field characteristics by students who were interested in the material.

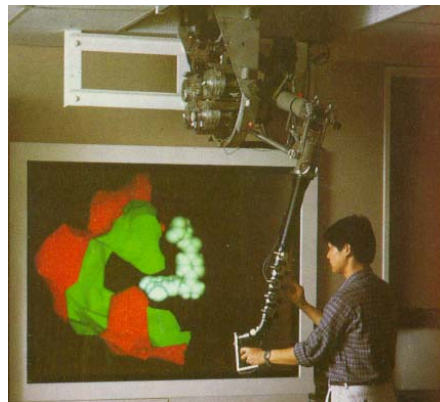
Students reported that using the haptic display dispelled previous misconceptions. They had thought that the field of a (cylindrical) diode would be greater near the plate than near the cathode, and they thought the gravitation vector in a 3-body field would always be directed at one of the bodies.



## Molecular modeling

Ming Ouh-young at UNC-CH designed and built a haptic feedback system to simulate the interaction of a drug molecule with its receptor site in a protein. (Brooks, Ouh-Young et al. 1990; Ouh-young 1990) This system, called the *Docker*, computed the force and torque between the drug and protein due to electrostatic charges and inter-atomic collisions. These forces were presented to the user, pulling the drug towards local energy minima. This task is very similar to that of other “lock and key” applications where the user moves one object and senses collisions with other objects in the environment.

The system presented the force and torque vectors both visually and using haptic feedback. Experiment showed that



chemists could perform the rigid-body positioning task required to determine the lowest-energy configuration of the drug up to twice as quickly with haptic feedback turned on compared to using the visual-only representations. (Ouh-young 1990) Scientists also reported that they felt like they had a better understanding of how the drug fit into the receptor site when they were able to feel the forces.

The Docker application, like other path-planning applications, required the presentation of both force and torque to the user. Since the drug molecule was not a point probe, different portions of it could collide with the protein at the same time. Extricating the drug from a collision sometimes required both translation and twisting. If the user was provided with only force (translation) information and no torque (twist) information, they could be led to move the drug in an improper direction.

### ***Haptic visualization for the blind (and the sighted)***

The Applied Science and Engineering Laboratories at the University of Delaware have been pursuing haptic visualization in the context of providing visualization systems that are suitable for use by the blind or visually impaired. (ASEL 1998) The haptic work was coordinated by Jason Fritz, who completed his master's thesis on haptic rendering techniques for scientific visualization in 1996. (Fritz 1996) In it, he describes several results, some of which are listed here.

Fritz found that the haptic equivalent to the grid lines on a 2D graph were very helpful in providing scale information and aid in navigation without being distracting. His implementation of this was to produce parallel planes evenly spaced in the volume that felt like thin walls through which the probe penetrates while moving through the volume where data is displayed. Fritz also discusses using friction and texture to make the simulated surface feel more realistic and to distinguish features in a data set. He describes a stochastic model for texture generation that can be used to create information-rich haptic textures for surfaces. (Fritz and Barner 1996)

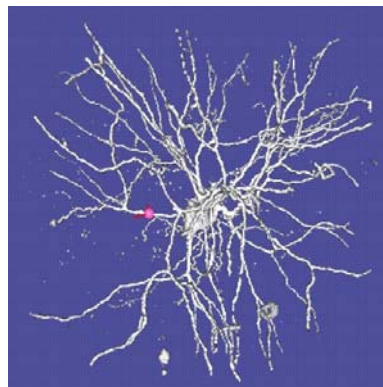
## **Haptic visualization circa 1998**

Several groups around the world are actively pursuing the haptic presentation of scientific data. These groups often include haptic feedback into systems that already use graphical or auditory data presentation. While care must be taken to avoid the effects of conflicting cues ((Srinivasan, Beauregard et al. 1996; DiFranco, Beauregard et al. 1997)), this can form a powerful combination. Haptics forms the only bi-directional channel between the user and the computer; each can push on the other. This allows the user to simultaneously sense the state of a system and control its parameters. Presented here is the work and results of some of these groups.

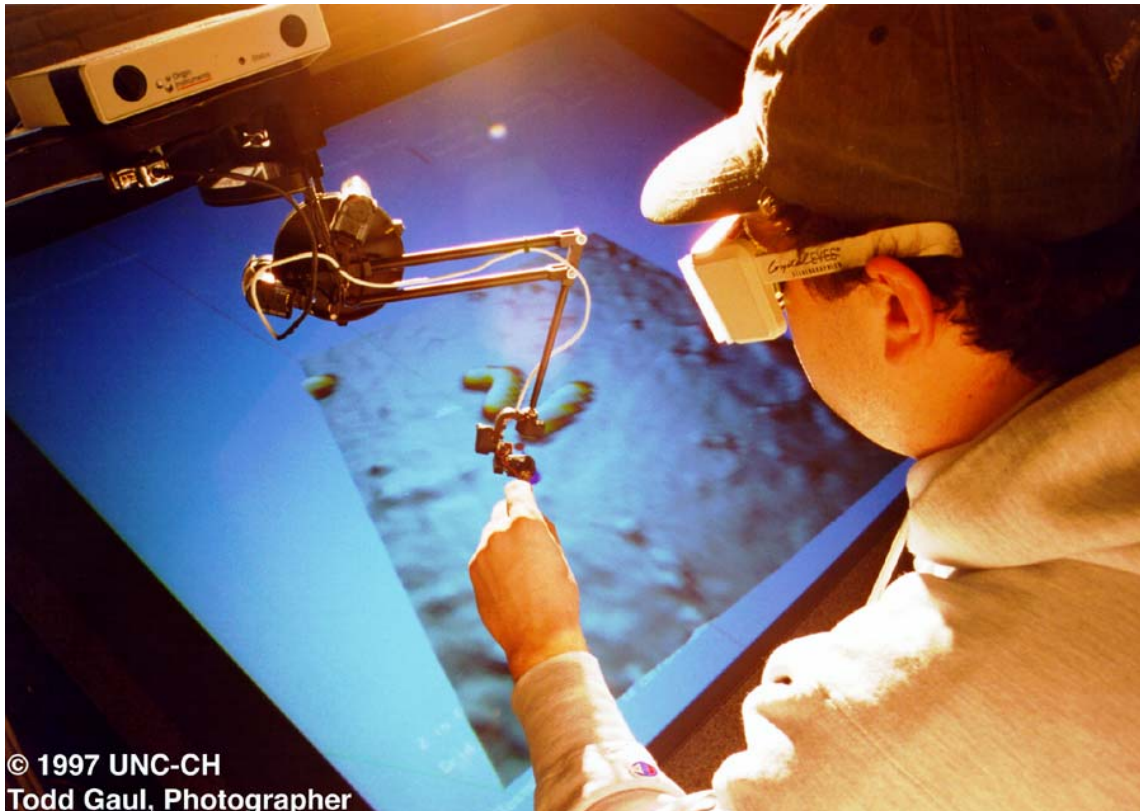
### ***Volume visualization***

Iwata and Noma at the University of Tsukuba built a haptic feedback force/torque sensor and HMD system to display volume data. (Iwata and Noma 1993) The system displays scalar data (the density function) as either torque about Z depending on density, force depending on density gradient, or both combined. They found that position error was reduced by a factor of two as compared to visual feedback alone when either or both of these forces were enabled. They describe one possibility for viewing multi-parameter data (fluid flow) by mapping flow velocity into force and one component of vorticity into torque around the direction of flow.

Avila and Sobierajski have developed a system that displays volume data sets both visually and haptically, and allows the user to modify the data sets. (Avila and Sobierajski 1996) Their system has been applied to medical visualization, art and scientific visualization. They have shown how 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.



## Microscope control



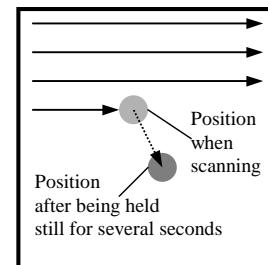
The nanoManipulator (nM) application provides an intuitive interface to scanning-probe microscopes, allowing scientists from a variety of disciplines to examine and manipulate nanometer-scale structures. (Taylor II, Robinett et al. 1993) The nM displays a 3D rendering of the data as it arrives in real time. Using haptic feedback controls, the scientist can feel the surface representation to enhance understanding of surface properties and can modify the surface directly. Studies have shown that the nM greatly increases productivity by acting as a translator between the scientist and the instrument being controlled. (Finch, Chi et al. 1995)

The haptic feedback component of our system has always been exciting to the scientists on the team; they love being able to feel the surface they are investigating. However, it is during modification that haptic feedback has proved itself most useful, allowing finer control and enabling whole new types of experiments. We describe here particular benefits received by adding haptic feedback to this application. Haptic feedback has proved essential to finding the right spot to start a modification, finding the path along which to modify, and providing a finer touch than permitted by the standard scan-modify-scan experiment cycle. (Taylor II, Chen et al. 1997)

### Finding the right spot

Due to time constants and hysteresis in the piezoceramic positioners used by SPMs to move the tip, the actual tip position depends on past behavior. The location of the tip for a given control signal is different if it is scanned to a certain point than if it is moved there and left constant. This makes it difficult to plan modifications accurately based only on an image made from scanned data.

Haptic feedback allows the user to locate objects and features on the surface by feel while the tip is being held still near the starting point for modification. Surface features marking a desired region can be located without relying only on visual feedback from the previous scan. This allowed a collaborator to position the





tip directly over an adenovirus particle, then increase the force to cause the particle to dimple directly in the center. It also allowed the tip to be placed between two touching carbon filaments in order to tease them apart.

### Finding the right path

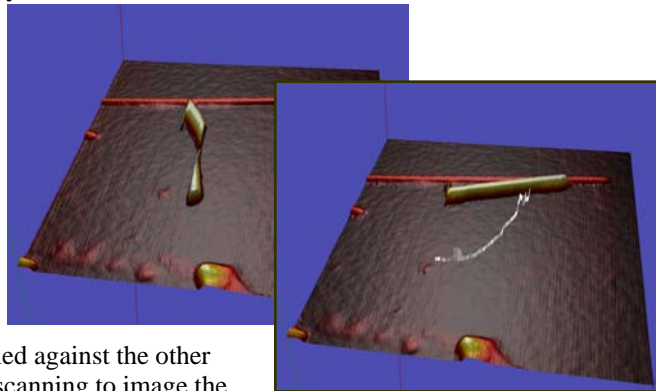
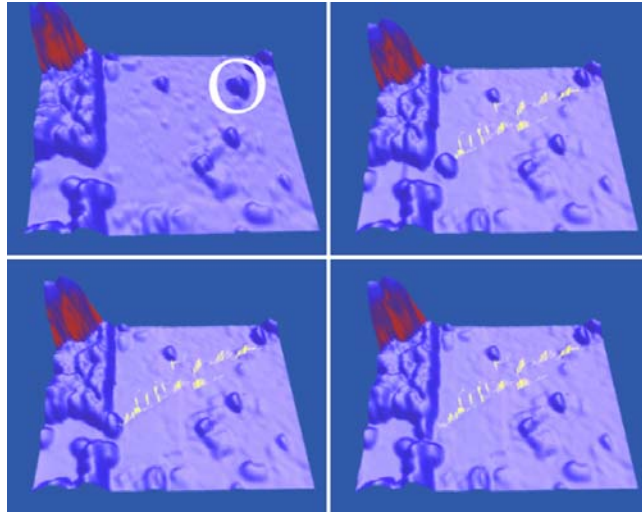
Even given perfect positioners, the scanned image shows only the surface as it was before a modification began. There is only one tip on an SPM: it can either be scanning the surface or modifying it, but not both at the same time. Haptic feedback during modification allows one to guide changes along a desired path.

This sequence shows haptic feedback being used to maneuver a gold colloid particle across a mica surface into a gap that has been etched into a gold wire. The yellow lines indicate where the user pushed with high force. This gap forms a test fixture to study the energy states of the ball. The colloid is fragile enough that it would be destroyed by getting the tip on top of it with modification force or by many pushes. This prevents attempts to move it by repeated programmed “kicks”. Haptic feedback allowed the user to tune the modification parameters so that the tip barely rode up the side of the ball while pushing it. This allowed the guidance of the ball during pushing so that only about a dozen pushes were required.

Haptic feedback was also used to form a thin ring in a gold film. A circle was scraped to form the inside of the ring, leaving two “snow plow” ridges to either side. By feeling when the tip bumped up against the outside of the outer ridge, another slightly larger circle was formed. This formed a thin gold ring on the surface.

### A light touch: observation modifies the system

When deposited on the surface, carbon nanotubes are held in place by residue from the solution in which they are dispersed. On some surfaces, the tubes slide freely once detached from the residue until they contact another patch of residue or another tube. Even the light touch of scanning causes them to move. By using only touch mode and switching between imaging and modification force, we have been able to move and re-orient one carbon tube across a surface and into position alongside another tube. Once settled against the other tube, it was stable again and we could resume scanning to image the surface. Haptic feedback and slow, precise hand motion (“haptic imaging”) allowed us to find the tube at intermediate points when we could not scan. The fact that the surface cannot be imaged at intermediate stages prevents this type of experiment from being performed using the standard scan-modify-scan cycle.



## Haptic visualization moving forward

The routine application of haptic feedback has reached the stage where computer graphics was years ago: Phong shading and some basic texturing operations. Building on techniques suggested by Minsky, Fritz and others, it is time for applications to push further. They should make use of multiple surface characteristics to carry information about multiple data sets. Some options being investigated are:

- Compliance (stiffness) of the simulated surface

- Friction models (coulomb, viscous and drag)
- Adhesion
- Texture (image-based or procedural, stationary or probabilistic)
- Surface vibration

Again, the true potential for haptic feedback seems to lie in its unique position as the only bi-directional modality. It allows both feeling and modifying at the same time. This allows direct and immediate control over simulations and remote operations with direct and immediate sensing of the results.

At UNC, investigation into the appropriate linear mapping for each of these forces is being led by Mark Hollins. Early work in this area is described in (Seeger, Henderson et al. 2000), with the perceptual studies underlying this use described more fully in (Hollins, Seeger et al. 2004). It turns out that the interplay between dimensions (friction, stiffness, etc) is non-trivial, and care must be taken when more than one is presented at a time. (Hollins, Seeger et al. 2004)

## References

- ASEL (1998). <http://www.asel.udel.edu/sem/research/haptics/>.
- Avila, R. S. and L. M. Sobierajski (1996). A Haptic Interaction Method for Volume Visualization. Proceedings of IEEE Visualization '96, San Francisco. 197-204.
- Brooks, F. P., Jr., M. Ouh-Young, et al. (1990). Project GROPE - Haptic displays for scientific visualization. Computer Graphics: Proceedings of SIGGRAPH '90, Dallas, Texas. 177-185.
- DiFranco, D. E., G. L. Beauregard, et al. (1997). The effect of auditory cues on the haptic perception of stiffness in virtual environments. Proceedings of the ASME Dynamic Systems and Control Division, ASME. 17-22.
- Finch, M., V. Chi, et al. (1995). Surface Modification Tools in a Virtual Environment Interface to a Scanning Probe Microscope. *Computer Graphics: Proceedings of the ACM Symposium on Interactive 3D Graphics*, Monterey, CA, ACM SIGGRAPH. 13-18.
- Fritz, J. and K. Barner (1996). Stochastic models for haptic texture. Proceedings of the SPIE International Symposium on Intelligent Systems and Advanced Manufacturing - Telemanipulator and Telepresence Technologies III, Boston, MA.
- Fritz, J. P. (1996). Haptic Rendering Techniques for Scientific Visualization. Electrical Engineering, University of Delaware: 148.
- Hollins, M., A. Seeger, et al. (2004). "Haptic perception of virtual surfaces: Scaling subjective qualities and interstimulus differences." *Perception* **33**: 1001-1019.
- Iwata, H. and H. Noma (1993). Volume Haptization. Proc. IEEE 1993 Symposium on Research Frontiers in Virtual Reality. 16-23.
- Minsky, M. (1995). Computational Haptics: The *Sandpaper* System for Synthesizing Texture for a ForFeedback Display. Program in Media Arts and Sciences, MIT: 217.
- Minsky, M., M. Ouh-young, et al. (1990). Feeling and Seeing: Issues in Force Display. Proceedings ACM Siggraph Symposium on Interactive 3D Graphics.
- Mitsubishi, M., Y. Hatamura, et al. (1993). Auditory and Force Display of Key Physical Information in Machining/Handling for Macro/Micro Teleoperation. Proceedings of the 1993 IEEE International Conference on Robotics and Automation, Atlanta, Georgia. 137-169.
- Ouh-young, M. (1990). Force Display In Molecular Docking. Computer Science. Chapel Hill, University of North Carolina: Tech Report #90-004.
- Seeger, A., A. Henderson, et al. (2000). Haptic Display of Multiple Scalar Fields on a Surface. Workshop on New Paradigms in Information Visualization and Manipulation, ACM Press.
- Srinivasan, M. A., G. L. Beauregard, et al. (1996). The impact of visual information on the haptic perception of stiffness in virtual environments. Proceedings of the ASME Dynamics Systems and Control Division, ASME. 555-559.
- Taylor II, R. M., J. Chen, et al. (1997). Pearls Found on the way to the Ideal Interface for Scanned-probe Microscopes. Visualization '97, Phoenix, AZ, IEEE Computer Society Press. 467-470.
- Taylor II, R. M., W. Robinett, et al. (1993). The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope. *SIGGRAPH 93*, Anaheim, California, ACM SIGGRAPH. 127-134.

# DAB: Interactive Haptic Painting with 3D Virtual Brushes

Bill Baxter   Vincent Scheib   Ming C. Lin   Dinesh Manocha  
Department of Computer Science  
University of North Carolina at Chapel Hill  
{baxter,scheib,lin,dm}@cs.unc.edu  
<http://www.cs.unc.edu/~geom/DAB>

**Abstract:** We present a novel painting system with an intuitive haptic interface, which serves as an expressive vehicle for interactively creating painterly works. We introduce a deformable, 3D brush model, which gives the user natural control of complex brush strokes. The force feedback enhances the sense of realism and provides tactile cues that enable the user to better manipulate the paint brush. We have also developed a bidirectional, two-layer paint model that, combined with a palette interface, enables easy loading of complex blends onto our 3D virtual brushes to generate interesting paint effects on the canvas. The resulting system, DAB, provides the user with an artistic setting, which is conceptually equivalent to a real-world painting environment. Several users have tested DAB and were able to start creating original art work within minutes.

**Keywords:** Haptics, Human Computer Interaction, Painting Systems, Deformable Brush Model

## 1 Introduction

The *art* of painting refers to the aesthetic aspects of a painterly work. The *craft* of painting deals with the study of materials, including paint medium, tools, supports, and methods, i.e. the manipulation of materials to express an artist's intent and purpose [May70]. The art and craft of painting are closely related: an artist cannot divorce one from the other. Nevertheless, recent technological advances in computer graphics have largely centered around the art of painting, with little attention being given to the *craft*.

Commercial painting systems and recent research on the generation of painterly works have mainly emphasized the appearance of the final product. However, the word 'painterly' also describes a fusion of feeling and action, sight and touch, purpose and paint, beyond merely producing an image that gives an artistic impression [May70].

Rather than focus primarily on the rendered appearance, there may be equal merit in recreating the "sight, touch, action and feeling" of the artistic process itself. By designing a setting for artists to freely and creatively express themselves, as they would in a traditional painting environment, computer graphics can serve as a conduit to the craft as well.

### 1.1 Main Contribution

Our primary goal is to provide an expressive vehicle for interactively creating original painterly works with computer systems. We present a physically-based, deformable 3D brush model, which gives the user control of complex brush strokes intuitively. The haptic feedback enhances the sense of realism and provides tactile cues that enable the user to better manipulate the paint brush. We have



Figure 1: An original work created using DAB. (Rebecca Holmberg, artist)

also developed a bidirectional, two-layer paint model that, in combination with a palette interface, enables easy loading of complex blends onto our 3D brush model and generates interesting paint effects on the canvas.

We have attempted to provide a minimalistic interface that requires as few arcane buttons, key-presses, and complex controls as possible, yet still offers a great deal of expressive power. With our haptic painting system, *DAB*, most paintings can be created with just the force-feedback device and the space bar on the keyboard. In comparison to the existing computer painting programs, our approach offers the following advantages:

- Natural and expressive mechanisms for manipulating the painting tools, including brushes, palette, paint and canvas;
- Simple and easy loading of complex blends using 3D virtual brushes;
- Physically-based and realistic brush footprints generated automatically by the brush strokes;
- Intuitive and familiar feel of the painting process requiring little or no training.

Our haptic painting system, *DAB*, has been tested by a number of users. A novice user can start painting with just a few (typically less than ten) minutes of simple instruction. Fig. 1 shows a painting created by an amateur artist with *DAB*. Since *DAB* provides a familiar setting, conceptually equivalent to a real-world painting

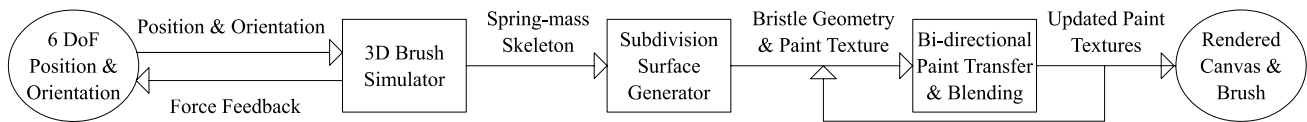


Figure 2: System Architecture

environment, an artist need only control the virtual brush as he or she would a real brush. This interface could also be combined with most of the existing interactive painting programs or used as an effective training system for painting.

## 1.2 Prior Work

**Computer-Generated Painting:** A number of researchers have developed automatic methods for transforming ordinary images into painterly or otherwise imaginative renderings [Her98, Lit97, Mei96]. Others have developed 2D methods for simulating the look of painting, from Alvy Ray Smith’s original “Paint” program [Smi78] to more recent physically-based approaches [CPE92, CAS<sup>+</sup>97]. Commercial packages such as COREL’s Painter [COR00] are able to achieve realistic looking simulations of natural media by clever use of 2D textures and compositing tricks. The amount of training required to proficiently use these commercial painting systems is large, as is the complexity involved in obtaining the precise strokes desired, even for skilled painters.

**Modeling of Brushes:** Several researchers have endeavored to accurately model the *appearance* of real brush strokes, but most techniques have been 2D heuristics. Strassmann modeled a brush as a one-dimensional array of bristles swept over a trajectory defined by a cubic spline curve [Str86]. This work was able to account for a number of effects achievable with an actual brush, such as varying color, width, and wetness. Wong and Ip [W100] defined a complex set of interrelated parameters to vary the density, opacity, and shape of a footprint in a way that takes into account the behavior of a three-dimensional round calligraphy brush. The resulting stroke appearances are *informed* by the physical behavior of the brush, but are not actually physically generated. The method as described is only partially interactive.

Our approach for brush modeling shares some similar themes with the work of Saito [SN99] on modeling a physical 3D brush for Japanese calligraphy and *sumie* paintings. However, our technique is more flexible in terms of brush shape, dynamics, and loading, and is able to take advantage of 3D graphics hardware as well.

**User Interface:** Hanrahan et al. allowed the user to paint directly onto a 3D model by using standard graphics hardware to map the brush from screen space onto the model [HH90]. Commercial systems, such as Z-Brush [Pix00] and Deep Paint [hem00], also allow users to paint directly on surfaces, but this is accomplished with standard 2D brush footprints that are projected onto the surface of the 3D object. The brush itself is not three-dimensional.

Several of the more advanced commercial tools, e.g. Painter, support pen-based input with sophisticated 5-DOF tablet devices, yet most still use only the position and pressure parameters and ignore the tilt. Further discussion on tablet systems is given in Section 6.

The idea of 3D painting has been explored in [ABL95, JTK<sup>+</sup>99, GEL00] using a simple, rigid 3D brush (tip) controlled by a 6-DOF input device to color 3D surfaces. All these 3D painting systems were restricted to monochrome brushes.

## 1.3 Organization

The rest of the paper is organized as follows. Section 2 gives an overview of our approach and the user interface. We present the modeling of the paint brushes in Section 3 and force display using haptic devices in Section 4. Section 5 describes our techniques for rendering acrylic or oil-like paint. Next, we briefly highlight the implementation of our prototype painting system with a haptic interface and demonstrate its features via the actual paintings of several volunteers in Section 6.

## 2 Approach

In this section we give an overview of our approach and the user interface of our haptic painting system.

### 2.1 Overview

We have developed a novel, physically-based, deformable 3D brush model integrated with a haptic interface. The haptic stylus serves as a physical metaphor for the virtual paint brush. It takes in the position and orientation of the brush and displays the contact force between the brush and the canvas to the user. The bristles of the brush are modeled with a spring-mass particle system skeleton and a subdivision surface. The brush deforms as expected upon colliding with the canvas. This framework allows for a wide selection of brush types to be made available to artists.

Our multi-layered paint model supports important features of paint, such as bidirectional paint transfer, blending, drying, and complex brush loading. The surfaces of the brush, canvas, and palette are coated with paint using this model. A schematic diagram is shown in Fig. 2 to illustrate how various system components are integrated.

### 2.2 User Interface

We use a SensAble Technologies’ PHANTOM as a haptic device and a dual-processor Pentium III PC with NVIDIA’s GeForce2 graphics card. One processor is dedicated to force display and the other is used to compute the brush dynamics and the paint transfer and blending. Fig. 3 shows the physical setup of our system.



Figure 3: Haptic Painting System Setup: An artist using a haptic stylus to paint directly on the virtual canvas using DAB.

Our haptic painting system allows the user to paint directly onto a virtual canvas displayed on the screen. Using the space bar as a toggle, the user can bring up the virtual palette for paint mixing and brush cleaning, or put the palette aside to paint directly onto the canvas. The user is also presented with a wide selection of virtual brushes that mimic different types and shapes of brushes used in traditional painting. A simple menu is presented for saving and loading a clean or previously painted canvas, undoing a brush stroke, quickly drying the canvas partially or completely, etc. Fig. 4 shows a snapshot of our graphical user interface, consisting of the virtual brushes, the palette, and the canvas.

The paint brush deforms in a natural, physical way, as the user moves the brush across the canvas. The user can create strokes with the brush, which behaves much in the way a real brush would. The actual footprints of the brush and resulting strokes are generated based on the user’s manipulation of the 3D brush on the virtual canvas.


















Type	Examples	Model	Structure	Surface	Example Strokes
Round					
Flat/ Bright					
Filbert					

Table 1: We show some real brushes, our model for each (skeletal structure and surface mesh), and example strokes generated with each.



Figure 4: *Graphical User Interface: The virtual canvas with the brush rack and a portion of the palette (LEFT); the brush rack and the palette for color mixing (RIGHT).*

### 3 Modeling of 3D Brushes

Paint brushes are often regarded as the most important tools at an artist's disposal. A good set of brushes can enable a competent artist to create virtually any effect he or she can imagine, from the intricate detail of cresting waves, wispy billowing clouds, to the subtly blended shifting hues in a sunset. In this section, we describe our techniques for modeling 3D virtual brushes.

#### 3.1 Introduction to Brushes

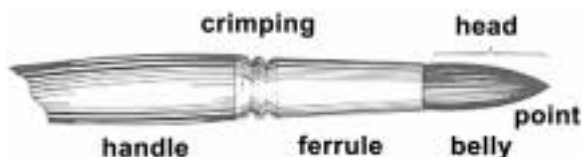


Figure 5: *Basic Brush Anatomy*

Fig. 5 shows the anatomy of a typical brush. Brush heads are made with a variety of bristles, natural soft animal hair, and synthetic materials. Some of the most common styles for brushes used in oil-like painting [May70] are:

- **Rounds.** Have a simple tubular shape with a semi-blunt point, allowing for a great variety of strokes.
- **Flats.** Thinner and wider than rounds with bristles squared off at the point. Flats are typically longer than they are wide.
- **Brights.** The same shape and construction as flats but typically shorter, with width nearly equal to length.
- **Filberts.** Have a thicker collection of bristles that increase ability to hold paint. Filberts usually have oval-shaped heads.

There are other types of specialty brushes, such as fans and blenders, but the four above are the most versatile and widely used. The second column of Table 1 shows images of each type.

#### 3.2 Overview of Modeling Approach

To model a 3D paint brush requires developing both a geometric representation and a physics-based model for its dynamic behavior. The requirements of an interactive haptic painting system place constraints on the design: the brush dynamics must run at interactive rates and remain stable under all types of user manipulation.

We model the brush head as a subdivision surface mesh wrapped around a spring-mass particle system skeleton. The particle system reproduces the basic motion and behavior of a brush head, while the deformable mesh skinned around this skeleton represents the actual shape of the head. We also derive an approximated implicit integration method based on an existing numerical technique for cloth simulation [DSB99] to take large integration steps while maintaining stability. Although our brush model may appear simplistic at first, it is designed to capture the essential quality of physical brushes to maintain interactivity at minimal computational costs.

Based on our generalized 3D brush model, we are able to adjust some key parameters to generate the different types and shapes of brushes and mimic their physical behavior. In Table 1, we show the geometric structure used to construct each of the brushes described in Section 3.1. We also show the deformation of different brushes as they make contact with the canvas.

#### 3.3 Brush Dynamics

The difficulty in simulating the paint brushes used in acrylic and oil-like painting is that the brushes are numerically stiff dynamical systems, and suffer from numerical instability. Bristles have very little mass. As they bend, energy stored in them can induce large accelerations and velocities when they are abruptly released. The brushes also behave as highly damped systems and we use this property to improve the stability of our solver.

We have considered and evaluated several numerical methods for particle system simulation, but we found the approximated implicit integrator presented in [DSB99] to be most effective for this application, primarily because of its stability. We simulate some brushes with the approximated implicit integrator and others with a variation based on first-order dynamics.

##### 3.3.1 Newtonian Dynamics

The motion of the particle system representing the brush can be described mathematically by Newton's second law, a second order differential equation, decomposed here as a pair of coupled first-



order differential equations:

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ \mathbf{M}^{-1}\mathbf{f} \end{pmatrix} \quad (1)$$

In this equation,  $\mathbf{x}$  is a  $3n$  vector containing the spatial coordinates of  $n$  particles,  $\mathbf{v}$  is a  $3n$  vector of particle velocities, and  $\mathbf{f}$  is a  $3n$  vector of the forces on those particles.  $\mathbf{M}$  is a  $3n \times 3n$  diagonal matrix whose diagonal entries are of the form  $M_{ii} = m_{\lceil i/3 \rceil}$ , where  $m_j$  is the mass of particle  $j$ .

The semi-implicit method for simulation of deformable objects [DSB99] approximates a solution to the equations of motion in three stages:

1. Implicit integration of linear force components
2. Approximate post-correction to account for non-linear force components
3. Deformation constraint enforcement to prevent excessive stretch

The resulting solution is much less accurate than other large-step integration techniques, such as that presented by Baraff and Witkin [BW98], but it is both more stable and computationally less demanding. The speed advantage comes from separating out the linear force component, which allows one to solve the equations of motion using just a matrix-vector multiply each step. The integration step has the form:

$$\begin{pmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{v} \end{pmatrix} = \begin{pmatrix} h(\mathbf{v}_0 + \Delta \mathbf{v}) \\ \left( \mathbf{I} - h \frac{\partial \mathbf{f}_l}{\partial \mathbf{x}} \right)^{-1} h \mathbf{M}^{-1} \mathbf{f}_l(\mathbf{x}_0) \end{pmatrix} \quad (2)$$

Since only the linear force components are handled by the integration step,  $(\mathbf{I} - h \partial \mathbf{f}_l / \partial \mathbf{x})^{-1}$  becomes a constant matrix.

This method works well for cloth, which generally has weak bending forces, but for brush simulation, approximating the effect of the non-linear force components leads to local errors in angular momentum. We have observed that with some brush skeletons, the solver effectively ignores stiff bend forces, leading to brushes with too much angular motion. Achieving stiff brush behavior is possible, but depends upon the skeletal structure. Section 3.5 discusses our brush construction in more detail.

The final step in the method is the application of deformation constraints. In this step, particles are assumed to have traveled in the right direction, but perhaps too far, inducing excessive stretch in the material. This is corrected by simply altering the positions of particles, iteratively contracting the springs in the material until overstretch is eliminated. The deformation constraints play a major role in the overall stability of the system by ensuring that at the end of every step, every spring is in a physically plausible configuration.

For collision handling and contact response, particles found to be penetrating the canvas are projected up to the nearest point on the surface. We also add a frictional drag force to colliding particles for more realistic results. We model the frictional drag as

$$\mathbf{F}_{\text{friction}} = -\mu \|\mathbf{F}_{\text{normal}}\| \mathbf{v}_{\text{tangential}},$$

where  $\mu$  is the coefficient of friction.

### 3.3.2 Aristotelian Dynamics

Real bristles empirically obey the Aristotelian model of physics, which is characterized by the lack of inertia. In this model, objects move only for the duration that forces are applied. Inspired by [WB97], we use this simplified model to simulate most of our brushes. This has advantages for speed, stability, and in some cases usability. With the Aristotelian dynamics model, the motion of the particle system is represented by a single first-order differential equation:  $\dot{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{f}$ .

Since objects now stop moving instantly in the absence of forces, the result is motion that appears heavily damped, which

is precisely how we wish brushes to behave. In the second order model, to simulate this damping requires adding large damping forces to the system to cancel out the large velocities induced by stiff springs. Using a first-order physics model, however, we can circumvent this step entirely.

We modify the approximated implicit integration formula as follows for the first order model:

$$\Delta \mathbf{x} = \left( \mathbf{I} - h \frac{\partial \mathbf{f}_l}{\partial \mathbf{x}} \right)^{-1} h \mathbf{M}^{-1} \mathbf{f}_l(\mathbf{x}_0)$$

Since this equation is still in the same form as Eqn. 2, most of the integration technique remains unchanged. An exception is that we omit the frictional damping force during collisions and just modify velocity, since in the first order model the two have the same effect.

### 3.4 Brush Surface

We use subdivision surfaces as the geometric representation for the brush head because of their ability to represent arbitrary topology and vertices of arbitrary valence easily. The brush head subdivision surface is defined by control points anchored relative to the mass particles. It is possible to use either interpolating or approximating subdivision schemes for the brush surface.

An interpolating scheme eases the task of choosing reasonable control vertices for the rough mesh, since the limit surface is guaranteed to pass through each of them. In fact, since *all* vertices at *all* subdivision levels are on the limit surface, it also facilitates changing the tessellation level of the mesh. However, due to frequent appearance of high curvature in the resulting surface, often interpolating surfaces do not deform as smoothly as would be expected of a brush.

Approximating schemes generate surfaces that are generally smoother and fairer, but it is more difficult to place control points to achieve the desired surface. The extensions to the Loop approximating scheme presented by [HDD<sup>+</sup>94] would be useful for accurately modeling sharp features like the finely tapered point of a round brush.

In our implementation we chose to use a triangular base mesh and to subdivide with the interpolating Butterfly rule to make the task of generating the brush control mesh easier. Some example results can be seen in Table 1.

### 3.5 Brush Generation

Given these dynamical models for simulation, we synthesize a full set of brushes suitable for creating a wide variety of paintings. One type of brush is modeled as a single spine composed of a linear chain of  $n$  particles. With our integration method and this structure, we are able to model the softer style of brushes used in Japanese calligraphy, called *fude*. Our *fude* brushes work best with the first order dynamics model, which makes the brush appear more solid by eliminating oscillations.

We model stiffer brushes, like those used in oil and acrylic painting, by using a more complicated skeletal structure. The basic building block for our stiff brushes is five mass particles connected with springs to form a pyramidal truss. The round brush consists of one of these trusses. The four particles that form the base are rigidly fixed to the brush handle and are directly driven by the user's input. The fifth particle serves as the point of the brush.

Table 1 shows a summary of the brush models and gives examples of the strokes that can be generated with each. Wide brushes are formed from two trusses, and filberts are generated from four of them, the outer two being shorter than the inner two. We use each brush structure to define an entire family of brushes of different sizes by parametrically varying the scaling along the three cardinal axes.

## 4 Haptic Display

An important aspect of our 3D painting system is the ability to provide sufficiently good force feedback to emulate the sensation of applying brush strokes to a canvas. Our 6-DOF armature input

device also serves as a 3-DOF force output device. We align the virtual paintbrush with the physical 6-DOF stylus, and position it so that the point of 3-DOF force delivery coincides with the point where the head meets the handle on the virtual brush. In this section, we present our approach for force display.

#### 4.1 Decoupled Haptics

We separate the force computation from the brush deformation computation, since the two have different goals. For instance, the non-dynamical deformation constraints used by the approximated implicit solver are acceptable for approximating the visual aspects of brush behavior, but are not appropriate for force simulation. Furthermore, the force updates for haptic feedback need to be generated at close to 1kHz for smooth jitter-free output, but the deformation calculation only needs to proceed at visual update rates (around 30Hz). Consequently we decouple the force simulation from brush dynamics simulation, and simplify the force computation to run at kHz rates.

#### 4.2 Basic Force Model

The root of our force model is a simple piecewise linear function of the penetration depth of the undeformed brush point. If  $d_p$  is the penetration depth, and  $l_p$  is the length of the brush head projected onto the canvas normal,  $\mathbf{n}$ , then the force is modeled as:

$$\mathbf{f}_b(d_p) = \begin{cases} 0 & \text{if } d_p \leq 0 \\ \mathbf{n}(k_1/l_p)d_p & \text{if } 0 < d_p \leq l_p \\ \mathbf{n}(k_1 + (k_2/l_p)(d_p - l_p)) & \text{if } l_p < d_p \end{cases} \quad (3)$$

where  $k_1$  is a small positive constant that models the light spring of bristles and  $k_2$  is a larger positive constant that simulates collision of the actual brush handle with the canvas. The spring constants are normalized by  $l_p$  so that the same absolute force is delivered when the handle first hits the canvas, regardless of the brush length or orientation. The value of  $k_1$  can be changed to simulate brushes of varying stiffness.

#### 4.3 Compressive Effects

When a real brush contacts the canvas at close to a right angle, the stiff bristles initially act as strong compressive springs, transmitting an abrupt force to the handle. As more pressure is applied, the bristles buckle and the compressive force reduces as bending forces take over. When the brush makes a contact at an oblique angle, compressive effects play a lesser role in the force felt.

Therefore, we extend the piecewise linear function, Eqn. 3, to a piecewise Hermite curve. This curve is defined by a series of control tuples which contain the penetration depth and corresponding force magnitude, and the linear stiffness of the spring model at that point. We currently use a four-segment piecewise curve, which was derived from the empirical observation of how a brush head behaves under compression.

The initial segment of the piecewise curve models the compressive force. We assign the initial control tuple a fairly strong linear spring constant to simulate the initial strong compressive force. We modulate this compressive force according to the angle of contact, by multiplying the force value of the second control tuple by an angle-dependent coefficient between one and zero. Given  $\theta$ , the angle between the canvas normal and negated bristle direction vector, the factor we use is

$$\gamma = \begin{cases} \cos^2(2\theta) & \text{if } -\frac{\pi}{4} < \theta < \frac{\pi}{4} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

This results in a compressive force that is strongest when a brush contacts the canvas at a right angle and that tapers off to zero as the brush approaches a 45 degree angle to the canvas.

#### 4.4 Frictional Forces

The final component of the force delivered to the user is a small amount of tangential resistance. Though small in magnitude, frictional forces have a large effect on the user's perceived ability to

control the brush by damping small oscillations in the user's hand. We model friction  $\mathbf{f}_t$  simply, as a force opposite the current brush velocity,  $\mathbf{v}_b$ , which is added to the other feedback forces:

$$\mathbf{f}_t = k_t (\mathbf{v}_b - \mathbf{n}(\mathbf{n} \cdot \mathbf{v}_b))$$

where  $k_t$  is the coefficient of friction.

### 5 Paint Model

Complementing our expressive brushes and force feedback, we present a paint model capable of capturing complex effects interactively. Our paint model incorporates variable wetness & opacity, conservation of volume, and a hardware-accelerated bi-directional paint transfer algorithm. It supports the following operations and techniques expected from acrylic or oil painting, while maintaining complete interactivity.

- **Blending** – Mixing of multiple pigments to obtain the desired color.
- **Bi-directional transfer** – Transferring paint both from the brush to canvas, and back from the canvas to the brush.
- **Complex brush loading** – Filling the various portions of the brush head with different pigments.
- **Variable dryness** – Controlling the blending of new paint onto previous layers by allowing paint to partially dry.
- **Glazing** – Painting with translucent layers (veils) of colors over other opaque colors (i.e. *underpainting*).
- **Impasto** – Painting with thick volumes of paint without addition of any medium.

Users can also generate similar results using other advanced painting programs. However, with our paint model, they need only manipulate the virtual brushes similar to real ones, in order to automatically generate the intended paint effects.

#### 5.1 Bi-directional Paint Transfer

Paint information is stored on both the canvas and brush in multiple textures (described in Section 5.2). The brush subdivision surface is tessellated to a polygonal surface. When this surface intersects the canvas geometry, the brush is considered to be in contact with the canvas. The bi-directional transfer must correctly modify the paint textures to simulate paint volume being interchanged between the two surfaces. Figure 6 displays a brush stroke possible only with bi-directional paint transfer.



Figure 6: *Bi-directional paint transfer is demonstrated by dragging a yellow paint stroke through wet purple paint (LEFT). A purple glaze of paint has been thinly applied over dry paint (RIGHT).*

The paint transfer problem is first reduced to two dimensions to simplify computation while introducing only slight inaccuracies. In the general case, a projection plane would be chosen that maximizes the area projected by the intersecting curve between the brush and canvas surfaces. Currently we have implemented only a two dimensional canvas, and therefore use the canvas plane for the orthographic projection of the brush. This is achieved with polygon

rasterization hardware, for speed and ease of implementation. The projected textures of the brush are used as the brush footprint.

The textures must be updated to simulate paint transfer and mixing. This 2D blending of the footprint with the canvas is discussed in Section 5.3. The simulation of the brush produces discrete instances of the brush surface; to produce a continuous stroke the blending operation is performed over a line connecting the current footprint to the previous one. The centroids of the footprint polygons are used as endpoints. This method provides smooth strokes while the footprint is not changing dramatically.

After 2D blending is complete, the updated textures are reapplied to the surfaces. This is achieved by rendering a variation of the brush subdivision surface mesh. The surface vertices that were projected to the footprint are used as texture coordinates into the now updated footprint textures. The original surface texture coordinates are used as vertex locations to render back into the surface's texture maps.

## 5.2 Paint Representation

The 3D brush and transfer methods presented here can be combined with many media types such as paint, ink, or watercolor. The *DAB* system currently includes a model that approximates the acrylic and oil families of paint.

Each paint surface contains two color layers. These are referred to as the 'surface' and 'deep' layers. Conceptually, the surface is covered by only a thin surface layer, and more thoroughly by the underlying deep layer. The surface layer is the boundary at which paint transfer between objects occurs. Surface layers are completely *wet*. The canvas's deep layer represents the paint that is completely *dry*. The brush's deep layer represents the reservoir of paint contained within the bristles. The paint transfer between surface layers occurs upon a collision between two objects (i.e. the brush and canvas). Transfer from the brush's reservoir layer to the surface is performed whenever the surface layer is no longer saturated (and paint remains in the reservoir layer). Drying paint from the canvas's surface layer to the dry layer occurs on a timed interval or as requested by the user.

The surface and deep layers are stored in color textures. A representation of the volume of paint in each layer is stored in an attribute texture. The surface layers and brush reservoir layer use fixed point representations, while the dry layer of the canvas is a specialized relative height field, and is described in Section 5.5.

## 5.3 Paint Mixing

The amount of volume transferred between surface layers is dependent on the volume of paint within each layer. The volume leaving,  $V_l$ , is computed from the initial volume,  $V_i$ , and transfer rate,  $R$ , over the elapsed time,  $T$ , by the equation,  $V_l = V_i \cdot T \cdot R$ . The resulting paint color,  $C_{new}$ , is computed by the weighted portions of remaining volume and color,  $V_r = V_i - V_l$  and  $C_i$ , and incoming volume and color from the other surface,  $V'_l$  and  $C'_i$ :

$$C_{new} = V_r \cdot C_i + V'_l \cdot C'_i$$

This essentially additive compositing formula is easy to work with, and gives predictable results, but does not model the way in which colloidal suspensions of pigment actually mix. The Kubelka-Munk model is the best known model available for accurately compositing pigments, but comes with significantly higher computational cost. See for example [CAS<sup>+</sup>97].

## 5.4 Optical Composition

To generate realistic paint effects, the wet and dry layers of the canvas are composited together with an embossing of the paint volume. This allows for glazing effects, as illustrated in Fig. 6. The volume of the wet layer,  $V_w$ , is multiplied by the optical thickness,  $O_t$ , of the paint, and then used for alpha blending the wet and dry layer colors,  $C_w$  and  $C_d$ .

$$C_{displayed} = \alpha \cdot C_w + (1 - \alpha) \cdot C_d; \quad \alpha = \min(V_w \cdot O_t, 1)$$

## 5.5 Drying the Canvas

Our paint model also supports variable wetness as shown in Fig. 7. Variable wetness is accomplished by gradually moving paint from the completely wet surface layer of the canvas to the completely dry deep layer.

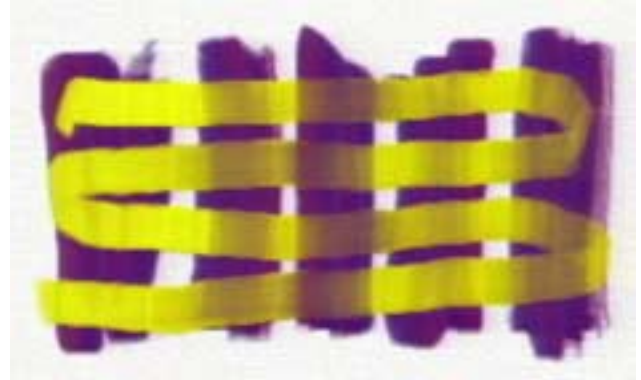


Figure 7: Variable wetness is displayed as yellow paint has been painted over the purple color stripes of 100%, 50%, 0%, 75%, 25% dryness (left to right).

The composited color of the paint must not change during drying. The optical blending function is used with this constraint to solve for the new dry layer color,  $C'_d$ , when some volume,  $\delta\alpha$ , is removed from the wet layer.

$$C'_d = \frac{\alpha \cdot C_w + (1 - \alpha) \cdot C_d - \alpha' C_w}{(1 - \alpha')}; \quad \alpha' = \alpha - \delta\alpha.$$

The dry layer of the canvas uses a relative height field to allow for unlimited volume of paint to be added, with a constraint only on the relative change in height between texels. An embossing of the height field is also computed. We use additive blending to combine this embossing and the color buffer to create the final rendered image of the paint.

## 6 Implementation Results

We have developed a prototype painting system, *DAB*, which combines 3D virtual brushes with a haptic interface and our paint model, as described in this paper. As mentioned in Section 2, the graphical user interface consists of three main elements: the canvas, the palette and the brush rack.

In the absence of a 3D stereo display, we have introduced shadows in our graphical display to enable the users to infer the relative position of the paint brush to the virtual canvas.

### 6.1 Discussion

A painter's palette not only "lists" available colors, but also allows a painter to mix and create a nearly unlimited number of new ones, and it presents both possibilities simultaneously through a simple, unified interface. Furthermore, creating complex color "gradients" on a painter's palette is just as easy as creating a single color: simply mix the constituent colors less thoroughly. In short, a real palette is a natural interface for color choosing, but one which has not been taken advantage of in previous computer painting systems.

To take best advantage of a painter's palette interface requires a 3D virtual brush like the one presented in this paper. With a 3D virtual brush, loading the complex blends created on the palette is simple, as is creating strokes that use those blends. Combined with an appropriate 3D input device, *DAB* offers a powerful yet simple interface for painting.

We chose to use a Desktop PHANTOM for input and haptic feedback because it provides true 6-DOF input with excellent precision and low noise, while offering fully programmable 3DOF force output. Other input devices such as tablets offer at most 5-DOF input (lacking a degree of freedom for twist), and have rather large noise in the tilt measurements.

On a pragmatic level, the force feedback is useful in that it enables the user to detect and maintain contact with the canvas better than if just shadow cues are provided. A tablet gives a physical surface that serves the same purpose, but it always gives the sensation of a rigid pen on a hard surface, rather than a soft, flexible brush on a canvas. Nearly all the users who have used both a tablet system and a haptic device preferred the soft feel of force feedback for brush simulation. Finally, with fully programmable forces, we are also able to change the feel of the brush at will, making it softer or harder for instance.

We are currently planning a detailed user study to thoroughly evaluate and assess the value of force feedback in creating the “right feel” for the artists. Using a programmable force feedback device with a true 3D workspace further enables the possibility to expand our system in a number of exciting directions covered in the next section.

## 6.2 User Feedback

More than fifteen users have painted with our system. This group of users includes amateurs and art students, both males and females, with ages ranging mostly from early 20’s to late 30’s. Some have prior experience with other computer painting programs and various user interfaces. All the users were able to pick up the haptic stylus and start painting immediately, with little training or detailed instruction. A small selection of their artistic creations is shown in Figs. 8 to 14. Additional images of art works created by our users, and detailed screen shots, are available as supplemental materials on the CD-ROM and on the project website.

Among users who have worked with other painting programs and interfaces, most found our painting system to be more intuitive. For artists with prior painting experience, our painting system was substantially easier to adapt to than other painting programs, while offering similar capabilities, such as undoing brush strokes, drying paint, etc. We attribute this to the fact that *DAB* offers a painting environment that takes advantages of skill transfer. *DAB* also seems to have an appeal for people with an artistic bent, but who would not normally consider painting, as well as for painters who would not normally use a computer painting system.

## 7 Future Work

Users of all types found *DAB* compelling to work with, however there are many aspects of the system which can be extended.

For improved accuracy in the brush deformation simulation, we continue to investigate the use of other efficient integration and simulation methods such as [BW98]. We are also interested in simulating a greater range of haptic phenomena from the feel of paint textures, to the variation in sensation when using different types of brush fibers, from painting on different backings, or with different mediums. Another natural step would be to go from painting 2D surfaces to painting 3D geometric models.

Our current paint model can be extended to depict more advanced characteristics of oil painting such as: gouging effects from bristle marks, anisotropic BRDFs, multiple wet layers of paint, and lighting-independent rendering of paint bumps. We are also interested in a real-time implementation of the Kubelka-Munk model for compositing. Expanding the set of virtual tools to include more types of brushes and other artistic tools is also of interest.

Our initial observations taken from a relatively small group of amateur artists, art students, and novices indicate that our approach is effective. We plan to conduct a more thorough and extensive formal user study over a larger group of users to confirm this observation, as well as to evaluate the effectiveness of various contributing factors in our interface design.

## 8 Acknowledgements

We are grateful to all the people who evaluated our system. Several of their paintings drawn using *DAB* are included in this paper. We would also like to thank the anonymous reviewers for their

useful comments. This research was supported in part by ARO DAAG55-98-1-0322, DOE ASCII Grant, NSF NSG-9876914, NSF DMI-9900157, NSF IIS-982167, ONR Young Investigator Award, and Intel.

## References

- [ABL95] M. Agrawala, A. Beers, and M. Levoy. 3D painting on scanned surfaces. In the Proc. of 1995 Symposium on Interactive 3D Graphics, pages 145–150. ACM SIGGRAPH, April 1995.
- [BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. *Proc. of ACM SIGGRAPH*, pages 43–54, 1998.
- [CAS<sup>+</sup>97] C. Curtis, S. Anderson, J. Seims, K. Fleischer, and D. Salesin. Computer-generated watercolor. *Proc. of SIGGRAPH’97*, pages 421–430, 1997.
- [COR00] COREL. Painter. <http://newgraphics.corel.com/products/painter6.html>, 2000.
- [CPE92] T. Cockshott, J. Patterson, and D. England. Modelling the texture of paint. *Computer Graphics Forum (Eurographics ’92 Proc.)*, 11(3):C217–C226, 1992.
- [DSB99] M. Desbrun, P. Schroder, and A. Barr. Interactive animation of structured deformable objects. *Proc. of Graphics Interface ’99*, 1999.
- [GEL00] A. Gregory, S. Ehmann, and M. C. Lin. *inTouch*: Interactive multiresolution modeling and 3D painting with a haptic interface. *Proc. of IEEE VR Conference*, 2000.
- [HDD<sup>+</sup>94] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of ACM SIGGRAPH*, pages 295–302, 1994.
- [hem00] RIGHT hemisphere. Deep paint. <http://www.us.deeppaint.com/>, 2000.
- [Her98] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. *Proc. of ACM SIGGRAPH’98*, pages 453–460, 1998.
- [HH90] P. Hanrahan and P. Haeblerli. Direct WYSIWYG painting and texturing on 3D shapes. In *Computer Graphics (SIGGRAPH ’90 Proceedings)*, volume 24, pages 215–223, August 1990.
- [JTK<sup>+</sup>99] D. Johnson, T. V. Thompson II, M. Kaplan, D. Nelson, and E. Cohen. Painting textures with a haptic interface. *Proceedings of IEEE Virtual Reality Conference*, 1999.
- [Lit97] P. Litwinowicz. Processing images and video for an impressionist effect. *Proc. of SIGGRAPH’97*, pages 407–414, 1997.
- [May70] R. Mayer. *The Artist’s Handbook of Materials and Techniques*. The Viking Press, 1970.
- [Mei96] B. Meier. Painterly rendering for animation. In *SIGGRAPH 96 Conference Proceedings*, pages 477–484. August 1996.
- [Pix00] Pixologic. Z-brush. <http://pixologic.com>, 2000.
- [Smi78] Alvy Ray Smith. Paint. TM 7, NYIT Computer Graphics Lab, July 1978.
- [SN99] S. Saito and M. Nakajima. 3D physically based brush model for painting. *SIGGRAPH99 Conference Abstracts and Applications*, page 226, 1999.
- [Str86] S. Strassmann. Hairy brushes. *Computer Graphics (SIGGRAPH’86 Proc.)*, 20:225–232, 1986.
- [WB97] A. Witkin and D. Baraff. *Physically Based Modeling: Principles and Practice*. ACM Press, 1997.
- [WI00] H. Wong and H. Ip. Virtual brush: A model-based synthesis of chinese calligraphy. *Computers & Graphics*, 24, 2000.



Figure 8: A painting by Eriko Baxter (LEFT); by Rebecca Holmberg (RIGHT)





Figure 9: *A painting by Rebecca Holmberg*

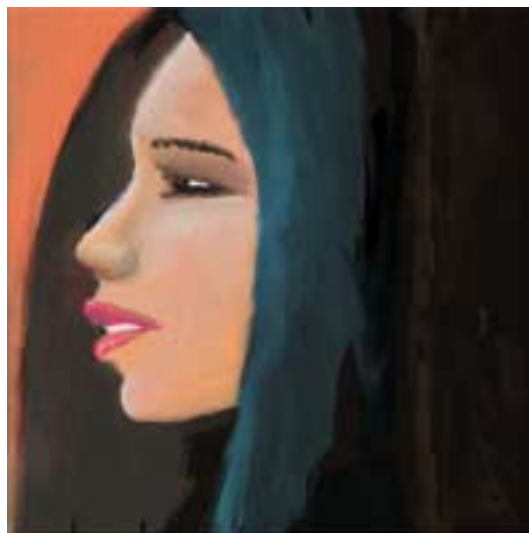


Figure 12: *A painting by Andrei State*



Figure 10: *A painting by Rebecca Holmberg*



Figure 13: *A painting by Lauren Adams*



Figure 11: *A painting by Rebecca Holmberg*



Figure 14: *A painting by Sarah Hoff*

# ArtNova: Touch-Enabled 3D Model Design

Mark Foskey   Miguel A. Otaduy   Ming C. Lin

University of North Carolina at Chapel Hill

{foskey,otaduy,lin}@cs.unc.edu

<http://www.cs.unc.edu/~geom/ArtNova>

## Abstract

We present a system, ArtNova, for 3D model design with a haptic interface. ArtNova offers the novel capability of interactively applying textures onto 3D surfaces directly by brush strokes, with the orientation of the texture determined by the stroke. Building upon the framework of inTouch [GEL00], it further provides an intuitive physically-based force response when deforming a model. This system also uses a user-centric viewing technique that seamlessly integrates the haptic and visual presentation, by taking into account the user's haptic manipulation in dynamically determining the new viewpoint locations. Our algorithm permits automatic placement of the user viewpoint to navigate about the object. ArtNova has been tested by several users and they were able to start modeling and painting with just a few minutes of training. Preliminary user feedback indicates promising potential for 3D texture painting and modeling.

**Keywords:** Haptics, Modeling, 3D Painting, Textures

## 1 Introduction

Designing 3D digital models is an important part of the production process in VR, computer game design, entertainment and education. Model design involves both 3D geometry and surface appearance, and each component offers its own challenges. Designing 3D shapes is difficult when one's input tool has only two degrees of freedom; and painting a surface is further complicated by the fact that the screen is flat while the object being painted can be curved and non-convex. With a physical model one could simply shape it in three dimensions and then paint on its surface. We propose to make model design easier by emulating that kind of experience as faithfully as possible, while offering users more flexibility and power via digital media.

In physical sculpting, the sense of touch is essential. Force display is now making it possible to add some degree of haptic feedback to the 3D sculpting experience. As an early attempt to provide touch-enabled modeling features, a non-commercial plug-in to Alias|Wavefront's Power Animator software package was developed at SensAble Technologies [Mas98], but force update rates were too low to provide a realistic feel. Only recently have commercial haptic sculpting systems, such as *FreeForm*<sup>TM</sup> [ST99], been introduced. They allow artists and designers to express their creativity with *3D-Touch*<sup>TM</sup>.



**Figure 1.** A turtle modeled and painted using ArtNova. Notice the patterns on its back and legs.

### 1.1 Main Results

Our goal is to develop a digital model design system that supports geometric modeling and texture painting with a direct 3D interface via force-feedback devices. Our system offers 3D texture painting on arbitrary polygonal meshes with the haptic stylus as an “electronic paintbrush”. Unlike most of the existing haptic sculpting systems [MQW01, RE99, ST99], we use subdivision surfaces as the underlying geometric representation with a physically-based force model for deforming the models. The turtle in Figure 1 was created and texture-painted by ArtNova. Notice the shell pattern on the back, and the mottling on the legs and face.

In this paper, we present ArtNova, an integrated system for 3D texture painting and multiresolution modeling with haptic interface and user-centric viewing. Our system has the following characteristics:

- **Interactive 3D Texture Painting** – Given true 3D interaction via a force feedback device, we can interactively apply predefined textures directly onto the surfaces of the model without object registration problems, in addition to painting monochrome colors.
- **Dynamically Adjusted Viewing** – Viewpoint locations have a direct impact on the quality of the graphical display accompanying haptic editing. In addition to the typical 3D object grabbing capability, our system offers automatic repositioning of the object to place a selected point near the center of the field of view, without the user having to switch between haptic editing and camera repositioning. Also, it provides incremental viewpoint navigation based on the user's gestures to provide proper views of the regions under haptic manipulation.

- **Physically-based Force Response** – A spring-based force model is used to emulate the surface tension when the user is pulling or pushing to edit the subdivision meshes.

Several users have been able to use *ArtNova* to create interesting models with just a few minutes of training. In addition to painting with monochrome colors [GEL00], it has also been used to apply textures onto a 3D model’s surface as well. The resulting meshes can be used directly for rendering and simulation without any format conversion. Preliminary user feedback suggests promising potentials of haptic interfaces for 3D painting and modeling.

## 1.2 Related Work

**Haptic Interaction:** Several real-time virtual environment systems have incorporated a haptic interface to enhance the user’s ability to perform interaction tasks [HCT<sup>+</sup>97, MRF<sup>+</sup>96, MS94, RKK97]. Gibson [Gib95] and Avila and Sobierajski [AS96] have proposed algorithms for object manipulation including haptic interaction with volumetric objects and physically-realistic modeling of object interactions. Recently, SensAble Technologies developed the *FreeForm*<sup>TM</sup> modeling system to create and explore 3D forms using volumetric representations [ST99].

**Geometric Modeling:** There is an abundant wealth of literature on geometric modeling, interactive model editing, and deformation methods applied to curves and surfaces. Geometric formulations for model editing can be classified as pure-geometric representations such as NURBS [PT97], free-form deformation (FFD) [SP86], or physically-based modeling techniques such as D-NURBS [QT96]. With a similar mathematical framework to hierarchical editing [FB88], subdivision methods allow modeling of arbitrary topology surfaces [SZ98], while supporting multiresolution editing [DKT98, HDD<sup>+</sup>94, KS99, SZMS98, ZSS97]. There are also other sculpting techniques based on volumetric modeling methods [GH91, MQW01, RE99, PF01]. The haptic modeling system *inTouch* uses subdivision surfaces as its underlying geometric representation [GEL00], but it lacks a physically-based force model to generate a realistic feedback sensation.

**3D Texture Painting:** By using standard graphics hardware to map the brush from screen space to texture space, Hanrahan et al. allowed the user to paint directly onto the model instead of into texture space [HH90]. This approach has been applied to scanned surfaces using 3D input devices, such as data gloves and a Polhemus tracker [ABL95]. However, the painting style of both systems can be awkward, due either to the difficulty in rotating an object for proper viewing during painting, or to the deviation in paint location introduced by the registration process.

General texture mapping approaches, such as [BVG91, MYV93], are powerful but require user input to generate the map. The approaches used in the Chameleon system [IC01] are more appropriate for casual, quick painting and not for highly detailed models. There are also commercial

3D painting systems [Hem00, COR00]. Most of them often use awkward and non-intuitive mechanisms for mapping 2D textures onto 3D objects, or require that a texture for the model be provided. None offers the natural painting style desired by artists and designers.

Johnson et al. introduced a method for painting a texture map directly onto a trimmed NURBS model using a haptic interface [JTK<sup>+</sup>99]. Its simplicity and intuitive interface support a natural painting style. However, its parameterization technique is limited to NURBS and does not apply to polygonal meshes, which are more commonly encountered in computer graphics and animation.

Our approach for haptic painting is similar to that presented in [GEL00] which can only paint colors. Our texture painting bears some resemblance to lapped textures [PFH00]. That work determines the orientation of the texture for overlapping patches based on user-specified vector fields, whereas ours applies textures *interactively* to a local region by brush strokes, with the orientation of the texture determined directly by the stroke.

**Camera Placement:** Camera control is a fundamental problem for 3D graphics applications. Several techniques on user interfaces for camera control have been proposed, including orbiting techniques mapping 2D motion into 3D interaction [CMS88, PBG92, Wer94, ZF99], use of image plane constraints [GW92, PFC<sup>+</sup>97], and direct camera manipulation using a 6DOF input device [WO90]. Our approach differs from many of the existing techniques that use 2D input devices to directly manipulate the viewpoint. Our main focus in *ArtNova* is to achieve automatic placement of viewpoint via *implicit control* based on the user’s manipulation of the haptic device. Our camera positioning techniques are also useful for touch-enabled exploration of predefined scenes and massive models [OL01].

## 1.3 Organization

The rest of the paper is organized as follows. Section 2 gives an overview of our system and its user interface. We present our new texture painting algorithm in Section 3. Multiresolution modeling based on subdivision surfaces with a spring-based force computation is presented in Section 4. Section 5 describes a novel feature for dynamically adjusting the viewpoint, as the objects are manipulated. We briefly highlight the implementation of our prototype system with a haptic interface and demonstrate its features via the actual artistic creations of several users in Section 6.

# 2 Overview

In this section we give an overview of the system architecture and the user interface of our system.

## 2.1 System Architecture

The overall system consists of a haptic server and a graphical client application, connected using VRPN [VRPN], a library for distributed virtual reality applications. As with *inTouch*, a copy of the model is retained on both the haptic server

and graphical client, and calculations that deform the model are duplicated on both applications, so that the changes in position of numerous vertices need not be passed over the network. An overview of the system architecture is given in Figure 2.

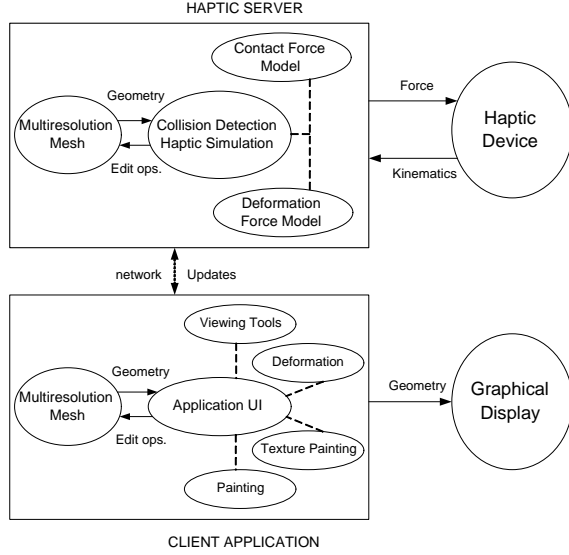


Figure 2. System Architecture.

## 2.2 User Interface

*ArtNova* allows the user to edit the geometry and the surface appearance of a model by sculpting and painting with a haptic interface. The user sees the model being edited, the tool being used, and a menu that can be operated using either the haptic tool or a mouse. Each type of model manipulation is indicated by a different tool. For example, the user moves the object with a mechanical claw, paints with a paintbrush, and deforms with a suction cup.

As an alternative to the claw tool for moving the object, the user’s viewpoint can be changed using the viewing techniques described in Sec. 5. An *automatic repositioning* feature lets the user move the last touched point on the model to the center of the viewing area using a single keystroke, and there is a “flying mode” controlled by the haptic device.

For painting there are a continuous color picker, sliders for brush width and falloff of paint opacity, and choice of textures for texture painting. The width of the stroke can also be changed by adjusting the pressure applied when painting.

A basic undo feature is provided for deformations and painting, and there are provisions for saving models and screen shots. A snapshot of the user interface is shown in Figure 3.

## 3 Texture Painting

In addition to modeling, *ArtNova* allows the user to paint textures and colors onto the model using a virtual paintbrush. Arbitrary polygonal models can be painted, and each stroke has a configurable *falloff*, fading into the background

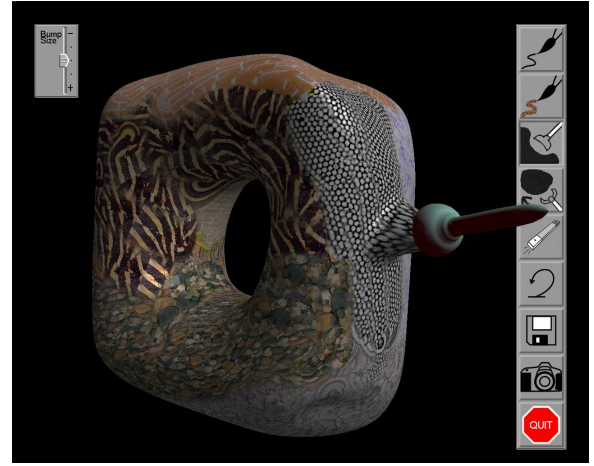


Figure 3. The graphical user interface. The user is performing a deformation on a painted toroidal base mesh.

near the boundaries of the stroke. The detailed algorithm for painting monochrome colors was described in [GEL00]. We briefly summarize it here before describing our novel texture painting method.

### 3.1 Painting Algorithm

Each stroke of the brush is decomposed into a sequence of *stroke segments*, which are represented as 3-space vectors linking the positions of the tool tip at successive frames. The brush radius is computed at the stroke endpoints (based on the force exerted by the user) and linearly interpolated across the length of the vector. This radius determines a volume of influence in 3D. Triangles are painted starting with the one containing the tail of the stroke segment. When one triangle has been painted, any neighbors that also extend into the volume of influence are then painted.

We paint each triangle by rasterizing the corresponding triangle in texture space. As the triangle is rasterized, for each texel  $p_t$  we determine the corresponding point  $p_w$  in world space on the surface of the model. We then calculate  $D$  as

$$D = \frac{\|p_w - q\|}{R_s(q)},$$

where  $q$  is the point on the stroke segment nearest  $p_w$ , and  $R_s(q)$  is the stroke radius at  $q$ . Once we have  $D$ , we compute the color of  $p_t$  by blending the color being painted with the background according to a falloff function depending on  $D$ .

Because we use a straight vector to represent a portion of a stroke that actually follows a curved surface, there can be artifacts if the surface curvature relative to the length of the stroke segment is appreciable. Typically, however, the stroke segments are quite short and this problem does not arise.

### 3.2 Texture-Mapped Paint

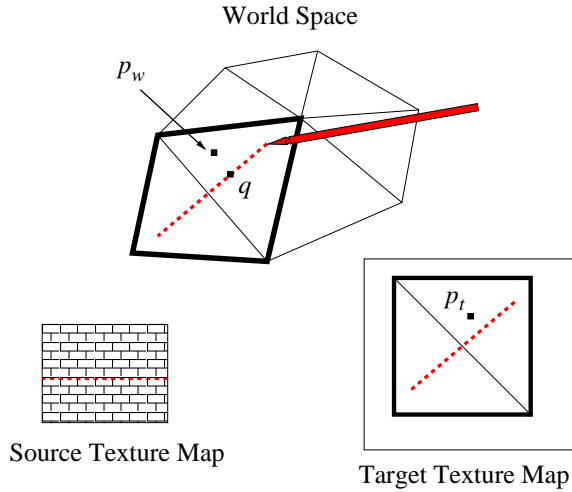
Texture-mapped paint builds upon our algorithm for painting monochrome colors. For clarity we will refer to the



texture map into which colors are drawn as the *target texture map*, and the predefined texture array as the *source texture map*. For each texel being rasterized in the target texture map, we compute the corresponding point  $p_w$  in world space, and its nearest neighbor  $q$  on the stroke segment (see Figure 4). We then compute two coordinates  $s$  and  $t$  that will be used to look up colors in the source texture map. The  $s$  coordinate represents length along the stroke, and  $t$  represents signed distance from the stroke. We compute  $s$  and  $t$  as follows: Let the current stroke segment be represented by a vector  $\vec{v}$  with tail  $p_v$ . Then, for  $s$ , we maintain the total length of all previous segments of a given stroke, and add the distance  $\|q - p_v\|$ . The sign of  $t$  is equal to the sign of

$$((p_w - p_v) \times \vec{v}) \cdot \vec{n}$$

where  $\vec{n}$  is the stored normal vector of the triangle containing  $p_w$ . The magnitude of  $t$  is just  $\|p_w - q\|$ .



**Figure 4.** Texture painting.

Before  $s$  and  $t$  are used to look up a color, they are scaled by a user-adjustable factor so that the texture features will be the desired size on the model. Texture lookup is performed modulo the dimensions of the texture patch, so it is important that the texture be periodic. Note that, although the texture will repeat along a stroke, the user can break up the periodicity by starting new strokes and altering stroke orientation.

At the boundary between stroke segments, the point  $q$  on the stroke vector nearest to  $p_w$  may be an endpoint. If  $q$  is the tail end of the stroke segment, texture is not applied for that point. If  $q$  is the front, then  $t$  is still taken as the distance from  $p_w$  to  $q$ , with sign given as above, while  $s$  is fixed. This has the effect of replicating pixels from the trailing stroke to fill gaps between roughly rectangular strips. Figure 5 shows an example of a textured cow model.

### 3.3 Accelerated Texture Painting

The original algorithms for color and texture painting rasterized the color data into the texture memory in software. We



**Figure 5.** Cuts of beef indicated by different textures using ArtNova.

have developed an improved algorithm that relies on modern graphics hardware to copy the texture. For each stroke segment, we first accumulate a list of triangles that extend into the volume of influence of the segment. Then the region of the texture parameter domain covered by those triangles is rendered with a fixed  $z$  coordinate in an orthogonal projection. If there are multiple patches in the target texture, the relevant triangles from each patch are rendered into a different region of the frame buffer. These triangles are textured, using coordinates generated dynamically that index into the source texture memory. The resulting image is then copied into the target texture memory with alpha blending. To produce the appropriate falloff function, the source texture has been prepared with the alpha values determined by the falloff. We can paint simple colors by choosing a monochromatic source texture.

### 3.4 Texture Coordinates

The above algorithms assume that texture coordinates for all triangles in the model have been determined. If they are not provided, we use a simple algorithm for generating texture coordinates. Since the user is painting on the model, not the texture map, we need not concern ourselves with contiguity of the image in texture memory. Also, there is relatively little concern with distortion, for the same reason. If the choice of texture coordinates distorts the image, the distorted version of the image will appear in the texture memory, not on the model. The algorithm we currently use groups triangles into pairs, which are mapped to squares in texture space. The areas of the squares are chosen to approximately reflect the initial areas of the triangles.

### 3.5 Comparison with other approaches.

One sophisticated program for painting on 3D models is DeepPaint [Hem00], a commercial product distributed by Right Hemisphere. Its interaction is fundamentally two dimensional, but the geometric model for applying a texture to the surface can be compared as an issue on its own. Deep

Paint offers two modes for texture painting. In one, the texture is copied directly to the texture memory, so that distortions in the parameterization are reflected in the appearance of texturing. For instance, the scale of the texture may be finer near the “north pole” of a sphere. In the other mode, the texture is applied in screen space, and then remapped as it is transferred to texture memory. This is the same approach that Chameleon [IC01] uses to apply solid color. This approach yields results that are independent of the parameterization of the object being painted, but it creates a distinctive “stretching” of the texture near the silhouette of the figure, which becomes noticeable when object is rotated to a different position. Because our approach determines temporary texture coordinates locally in the neighborhood of a portion of a single stroke, the overall scale of the texture is not dependent on where it is placed, making the painting more predictable for the user.

## 4 Multiresolution Mesh Editing

Our model editor is based on *inTouch*’s geometric framework [GEL00], which is strongly influenced by Zorin et al. [ZSS97]. For completeness, we briefly describe the framework here and then describe the new simplified force model used during deformation in *ArtNova*.

### 4.1 The Mesh Data Structure

We use a subdivision framework to represent our geometry. We store a coarse, triangular *base mesh*  $M_0$  and several meshes at finer resolutions  $M_i$  ( $i > 0$ ). By a single stage of Loop subdivision, each mesh  $M_i$  uniquely determines a finer mesh  $M_{i+1}^{\text{sub}}$ .  $M_{i+1}^{\text{sub}}$  is used as a reference mesh for the definition of  $M_{i+1}$ . Every vertex in the actual mesh  $M_{i+1}$  corresponds to a vertex of  $M_{i+1}^{\text{sub}}$ , but differs from it by a *displacement vector* stored with the vertex. In this way we can choose to edit at a specific resolution by moving vertices of a given mesh  $M_i$ . Vertices at finer levels retain their displacement vectors and are thus carried along by the motion of the subdivided surface.

In principle we could modify  $M_i$  without changing  $M_{i-1}$  at all, since the vertices of  $M_i$  are different from the vertices of  $M_i^{\text{sub}}$  (gotten by subdividing  $M_{i-1}$ ). However, we also perform a smoothing step using a method given by Taubin [Tau95] to modify coarser levels. In this way, for instance, an accumulation of edits at a high resolution, all tending to raise up one side of the model, can result in a repositioning of the coarser level vertices to better reflect the new overall geometry of the model.

### 4.2 Deformation Algorithms

To edit the model, the user simply places the tool against the model, presses the PHANTOM button, and moves the tool. As the surface is edited, the user can feel a resisting force and see the surface deform. The edit resolution (the choice of mesh  $M_i$  to modify directly) is presented to the user as a “bump size.”

#### 4.2.1 Surface Modification

Surface deformation is performed by moving a single triangle of the edit mesh  $M_i$ . When the user begins a deformation, the point of contact with the surface determines a unique triangle at the edit resolution, and a unique reference point on that triangle. For each frame, the successive positions of the tool tip define a motion vector  $\vec{m}$ , which is used to move the three vertices of the selected triangle. Each vertex is moved in the direction of  $\vec{m}$ , and by a distance scaled so that vertices nearer the current point of the tool tip are moved farthest. More precisely, the distance  $d_i$  from the reference point to each vertex  $v_i$  is computed, and the movement vector  $\vec{m}_i$  for each vertex is given by

$$\vec{m}_i = \left(1 - \frac{d_i}{d_0 + d_1 + d_2}\right) \vec{m}.$$

#### 4.2.2 Force Model

When the user places the tool against the model, there is a restoring force generated by the haptic rendering library, based on collision information from H-Collide [GLGT00]. When the user begins deforming the surface, the restoring forces are turned off, and the initial 3-space location of the tool tip,  $p_0$ , is recorded. The user is then free to move the tool in any direction. To provide feedback, a Hooke’s law spring force is established between the tool tip and  $p_0$ , given by

$$f = -k(p_{\text{tip}} - p_0),$$

where  $p_{\text{tip}}$  is the location of the tool tip and  $k$  is a small spring constant.

The spring constant is chosen so that the user can move the tip a sizable distance in screen space before the force becomes a substantial hindrance. When the user releases the button, the spring force is turned off and the usual restoring forces are turned on with the surface in its new position.

Because our force model is based on the initial position of the tool, the force computation is decoupled from the position of the surface. This provides a smoother feel to the user than computing the force from the instantaneous distance to the surface that is being moved, because computing the surface deformation is much slower than the 1 kHz haptic response rate.

## 5 Dynamic Viewing Techniques

As the user performs painting or modeling tasks over the entire model, the user will need to edit back-facing portions of the model from time to time. Typically the user repositions the model by performing a “grab and turn” operation using the application. We have developed several novel *user-centric* viewing techniques that make this task easier and more efficient. In addition to using the force feedback device for haptic manipulation, we also use it implicitly, and simultaneously, as a mechanism for viewpoint placement. These techniques allow users to express their intentions in a natural way, with a minimum of switching between editing and changing the view.

Our approach to the problem is to reposition the camera based on the configuration (i.e. position and orientation) of the haptic probe so that the region of interest on the model surface is placed at the center of the view. We call our techniques “user-centric” because the haptic tool indicates where the user wants to view the object from, rather than where the object should be.

### 5.1 Grabbing Operation

In the typical grabbing operation [RH92], the object is moved preserving the transformation between the virtual probe and the grabbing point. At the instant of grabbing, two points are picked:  $A$ , a point in the object, and  $B$ , the current position of the virtual probe. The transformation  $T_{BA}$  from the probe to the object is set constant as the virtual probe moves. Therefore, a displacement of the probe,  $\Delta T_B$ , produces a displacement of the object,  $\Delta T_g$ .

$$T_{BA} = T_A T_B^{-1},$$

$$\Delta T_g = T_{BA} \Delta T_B T_{BA}^{-1}.$$

where  $T_A$  and  $T_B$  are the transformations that represent the location and orientation of the object and the virtual probe.

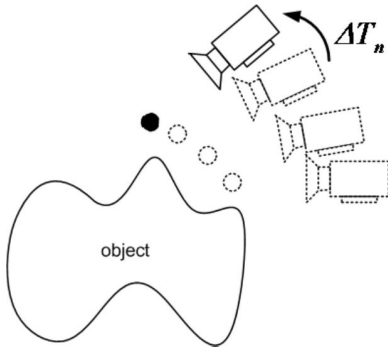
### 5.2 Viewpoint Navigation

We introduce a new concept called “viewpoint navigation”. This functionality is based on moving the viewpoint around the object to be modeled or painted, based on the gestures of the user. The position and orientation of the haptic device at each time are used as “commands”, which set a transformation,  $T_n$ , on the position of the virtual probe and the camera, as shown in Fig. 6.

In our system we apply the inverse transformation to the object that is being manipulated, producing the same visual effect. In addition, we apply this transformation incrementally, whenever the viewpoint navigation is enabled.

$$\Delta T_n = K \Delta T_H.$$

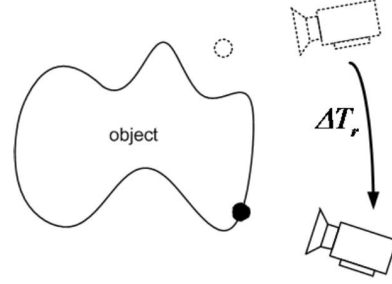
where  $\Delta T_H$  represents the variation of the position and orientation of the haptic device.



**Figure 6.** Viewpoint navigation based on user's intentions.

### 5.3 Automatic Repositioning

Another novel functionality called *automatic repositioning* allows the user to reorient the model quickly without interrupting the work flow to change tools. When the automatic repositioning key is pressed, the last point touched on the model is taken to define the region of interest. A transformation  $T_r$  is computed that moves the camera to a location along the normal direction of the object at the point of interest (as shown in Fig. 7). Then, we apply the inverse transformation to the object incrementally, so that the object appears to rotate to the new position.



**Figure 7.** Automatic repositioning of viewpoint.

### 5.4 Combining all the Functionalities

The transformation that sets the location and orientation of the object is updated every time step for better viewing to aid the model manipulation and editing tasks, depending on which functionality is enabled. We have:

$$\begin{aligned} T_{A,k+1} &= \Delta T_g T_{A,k}, \text{ if grabbing is enabled;} \\ T_{A,k+1} &= \Delta T_n^{-1} T_{A,k}, \text{ if viewpoint navigation is enabled;} \\ T_{A,k+1} &= \Delta T_r^{-1} T_{A,k}, \text{ if automatic repositioning is enabled.} \end{aligned}$$

## 6 Results

We have designed and implemented *ArtNova* as a proof-of-concept prototype system. In this section, we demonstrate the system capability and discuss issues related to the user interface design.

### 6.1 Prototype Demonstration

We use a dual-processor Pentium III PC as a haptic server, a SensAble Technologies' PHANTOM as a haptic device, a Silicon Graphics Inc. R12000 Infinite Reality for rendering, a large screen with a back projection system for graphical display, and UNC's VRPN library [VRPN] for a network-transparent interface between application programs and our haptic system. The system is written in C++ using the OpenGL and GLUT libraries. However, the design framework of our system is applicable to all types of haptic devices and libraries, as well as graphical display and computing platforms.

Several users with little or no experience in using modeling or painting systems were able to create interesting models using *ArtNova* with little training. Each user was given



**Figure 8.** A tree.

a selection of simple base meshes of various shapes (e.g. spherical, toroidal, etc.) and light gray in color. A few examples of their art work are given in Figures 1, 3, 8, and 9.

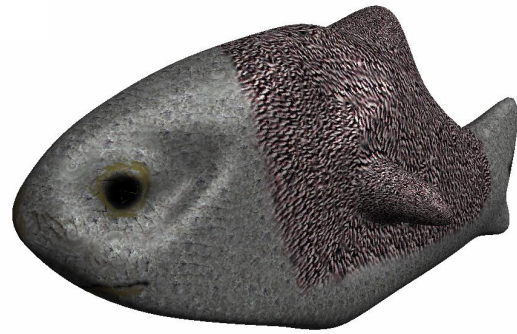
The accompanying videotape illustrates a few examples of the modeling and painting operations performed using *ArtNova*. Along with some more images of the models created by our users, the video clips are also available at:

<http://www.cs.unc.edu/~geom/ArtNova>.

## 6.2 User Experiences

We have asked several users to evaluate our system by creating and texture-painting a few models. We briefly summarize some of their comments here:

- The spring-based force model feels natural. Comparing the experiences of performing model deformation with and without the physically-based force model, most of the users found the newly added force model to be an improvement.
- The users commented that the force feedback was useful in detecting and maintaining contact with the model surfaces, when performing highly detailed painting.
- Users also felt that texture painting was easy to use, and had little trouble getting visually pleasing results.
- Users found automatic viewpoint adjustment and repositioning to be intuitive and natural.
- The overall graphical user interface with our new 3D tool metaphor was found to be intuitive and easy to understand.



**Figure 9.** A fish.

## 7 Summary

We have presented an integrated system for 3D texture painting and multiresolution modeling with a haptic interface and user-centric viewing. An artist or a designer can use this system to create and refine a three-dimensional multiresolution polygonal mesh, and further enhance its appearance by directly painting textures onto its surface. The system allows users to naturally create complex forms and patterns aided not only by visual feedback but also by their sense of touch. Based on preliminary user feedback, we believe these features considerably improve the ease and expressiveness of 3D modeling and texture painting. We are currently planning an extensive, formal user study to carefully evaluate and assess the contribution of various elements of this system on enhancing users' experiences in digital model design.

## 8 Acknowledgments

This research is supported in part by a fellowship of the Government of the Basque Country, NSF DMI-9900157, NSF IIS-9821067, ONR N00014-01-1-0067 and Intel. We would like to thank the following individuals for their suggestion and assistance: Arthur Gregory, Stephen Ehmann, Michael Rosenthal, Adrian Ilie, Sarah Hoff, Bryan Crumpler, Derek Hartman, Scott Cooper, and Joohi Lee.

## References

- [ABL95] Maneesh Agrawala, Andrew C. Beers, and Marc Levoy. 3D painting on scanned surfaces. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 145–150. ACM SIGGRAPH, April 1995.
- [AS96] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. *Proceedings of Visualization'96*, pages 197–204, 1996.
- [BVIG91] Chakib Bennis, Jean-Marc Vézien, Gérard Iglésias, and André Gagalowicz. Piecewise surface flattening for non-distorted texture mapping. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 237–246, July 1991.
- [CMS88] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-D rotation using 2-D control devices. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 121–129, August 1988.

- [COR00] COREL. Painter. <http://newgraphics.corel.com/products/painter6.html>, 2000.
- [DKT98] T. DeRose, M. Kass, and T. Troung. Subdivision surfaces in character animation. *Proc. of ACM SIGGRAPH*, 1998.
- [FB88] D. Forsey and R.H. Bartels. Hierarchical B-spline refinement. In *Proc. of ACM Siggraph*, pages 205–212, 1988.
- [GEL00] A. Gregory, S. Ehmann, and M. C. Lin. *inTouch*: Interactive multiresolution modeling and 3d painting with a haptic interface. *Proc. of IEEE VR Conference*, 2000.
- [GH91] Tinsley A. Galyean and John F. Hughes. Sculpting: An interactive volumetric modeling technique. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 267–274, July 1991.
- [Gib95] S. Gibson. Beyond volume rendering: Visualization, haptic exploration, and physical modeling of element-based objects. In *Proc. Eurographics workshop on Visualization in Scientific Computing*, pages 10–24, 1995.
- [GLGT00] A. Gregory, M. Lin, S. Gottschalk, and R. Taylor. Real-time collision detection for haptic interaction using a 3-dof force feedback device. *Computational Geometry: Theory and Applications, Special Issue on Virtual Environments*, 15(1-3):pp. 69–89, February 2000.
- [GW92] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 331–340, July 1992.
- [HCT<sup>+</sup>97] J. Hollerbach, E. Cohen, W. Thompson, R. Freier, D. Johnson, A. Nahvi, D. Nelson, and T. Thompson II. Haptic interfacing for virtual prototyping of mechanical CAD designs. *CDROM Proc. of ASME Design for Manufacturing Symposium*, 1997.
- [HDD<sup>+</sup>94] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of ACM SIGGRAPH*, pages 295–302, 1994.
- [Hem00] Right Hemisphere. Deep paint. <http://www.us.deeppaint.com/>, 2000.
- [HH90] Pat Hanrahan and Paul E. Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 215–223, August 1990.
- [IC01] T. Igarashi and D. Cosgrove. Adaptive unwrapping for interactive texture painting. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 209–216, 2001.
- [JTK<sup>+</sup>99] D. Johnson, T. V. Thompson II, M. Kaplan, D. Nelson, and E. Cohen. Painting textures with a haptic interface. *Proceedings of IEEE Virtual Reality Conference*, 1999.
- [KS99] A. Khodakovsky and P. Schröder. Fine level feature editing for subdivision surfaces. *Proceedings of ACM Symposium on Solid Modeling and Applications*, 1999.
- [Mas98] Thomas Massie. A tangible goal for 3D modeling. *IEEE Computer Graphics and Applications*, May/June, 1998.
- [MQW01] K. McDonnell, H. Qin, and R. Wlodarczyk. Virtual clay: A real-time sculpting system with haptic interface. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 179–190, 2001.
- [MRF<sup>+</sup>96] William Mark, Scott Randolph, Mark Finch, James Van Verth, and Russell M. Taylor II. Adding force feedback to graphics systems: Issues and solutions. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 447–452, 1996.
- [MS94] T. M. Massie and J. K. Salisbury. The phantom haptic interface: A device for probing virtual objects. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 1:295–301, 1994.
- [MYV93] Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive texture mapping. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 27–34, August 1993.
- [OL01] M. Otaduy and M. Lin. User-centric viewpoint computation for haptic exploration and manipulation. *Proc. of IEEE Visualization*, 2001. To appear.
- [PBG92] C. Phillips, N. Badler, and J. Granieri. Automatic viewing control for 3D direct manipulation. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 71–74, 1992.
- [PF01] R. Perry and S. Friskin. Kizamu: A system for sculpting digital characters. *Computer Graphics (ACM SIGGRAPH'01)*, 2001.
- [PFC<sup>+</sup>97] J. Pierce, A. Forsberg, M. Conway, S. Hong, R. Zeleznik, and M. Mine. Image plane interaction techniques in 3D immersive environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 39–44, 1997.
- [PFH00] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. *Proc. of ACM SIGGRAPH*, pages 465–470, 2000.
- [PT97] L. A. Piegl and W. Tiller. *The NURBS Book*. Springer Verlag, 1997. 2nd Edition.
- [QT96] Hong Qin and Demetri Terzopoulos. D-NURBS: A physics-Based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, March 1996. ISSN 1077-2626.
- [RE99] A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. *ACM Symposium on Solid Modeling and Applications*, 1999.
- [RH92] Warren Robinett and Richard Holloway. Implementation of flying, scaling, and grabbing in virtual worlds. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pages 189–192, March 1992.
- [RKK97] D.C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH*, pages 345–352, 1997.
- [SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 151–160, August 1986.
- [ST99] SensAble Technologies Inc. *freeform<sup>TM</sup>* modeling system. <http://www.sensable.com/freeform>, 1999.
- [SZ98] P. Schröder and D. Zorin. Subdivision for modeling and animation. *ACM SIGGRAPH Course Notes*, 1998.
- [SZMS98] T. Sederberg, J. Zheng, M. Sabin, and D. Sewell. Non-uniform recursive subdivision surfaces. *Computer Graphics (ACM SIGGRAPH'98)*, 1998.
- [Tau95] Gabriel Taubin. A signal processing approach to fair surface design. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 351–358. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.
- [VRPN] Virtual Reality Peripheral Network. <http://www.cs.unc.edu/research/nano/manual/vrpn>.
- [Wer94] Josie Wernecke. *The Inventor Mentor*. Addison-Wesley, 1994.
- [WO90] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 175–183, 1990.
- [ZF99] Robert Zeleznik and A. Forsberg. Unicam 2D gestural camera controls for 3D environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 169–173, 1999.
- [ZSS97] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Computer Graphics (ACM SIGGRAPH'97)*, 1997.



# **RECENT ADVANCES IN HAPTIC RENDERING AND APPLICATIONS**

## **Supplementary Course Notes**

### **PART II – Presentation Slides**

**Ming C. Lin**  
University of North Carolina at Chapel Hill

**Miguel A. Otaduy**  
ETH Zurich





## Introduction and Overview

Ming C. Lin  
UNC-Chapel Hill  
<http://www.cs.unc.edu/~lin>  
lin@cs.unc.edu

Miguel A. Otaduy  
ETH-Zurich  
<http://graphics.ethz.ch/~otmiguel>  
otaduy@inf.ethz.ch



## Definition

Haptic Rendering: *display of information through force or tactile cues.*

Haptic Rendering

+

Visual Display

+

Auditory Feedback

- Intuitive interaction with virtual environments
- Increase presence

## History



- Used in master-slave telerobotic systems
- First suggested by Sutherland [1965] for interaction with VEs
- Project GROPE in UNC:
  - 2D force field simulation [1971]
  - 3D molecular docking [1990]

## History



- [Minsky et al. 1990]: rendering of 2D textures.



2D height field

Compute  $F_x, F_y$   
forces using  
gradients

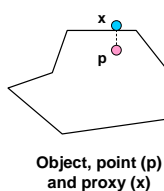


2DOF device

## History



- [Zilles and Salisbury 1995; Ruspini et al. 1997]: point-object interaction.



Constrain x to the  
surface  
Force given by  
 $\text{dist}(x, p)$

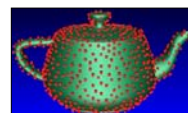


3DOF device

## History



- [McNeely et al. 1999]: interaction between rigid objects. Also called 6DOF haptic rendering.



Sample objects

Per-sample collision  
test  
Net force and torque



6DOF device

## Description of the Problem



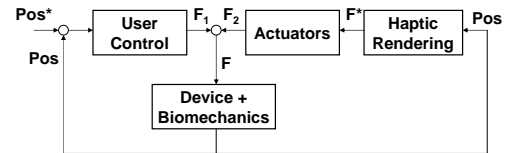
1. Move grabbed object according to input
2. Compute contact forces between objects
3. Apply net force and torque to grabbed object

Similar to Rigid Body Simulation

## Description of the Problem



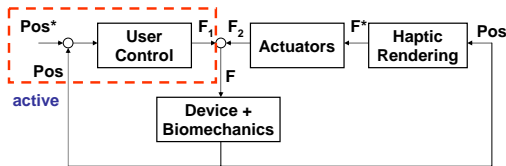
- Control engineering analysis:



## Description of the Problem



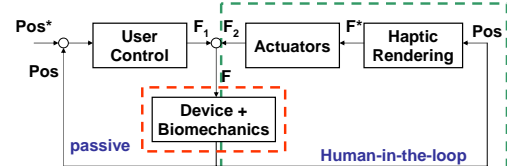
- Control engineering analysis:



## Description of the Problem



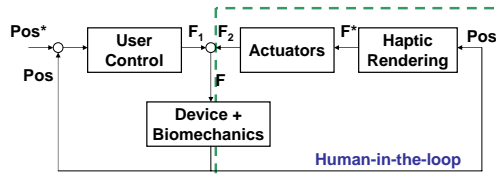
- Control engineering analysis:



## Description of the Problem



- Control engineering analysis:



- High sensitivity to instabilities!!
- High update rates required!! (kHz for high stiffness)

## Description of the Problem



- 2 major components:
  - Collision detection / contact determination
  - Contact response model and force computation

## Haptic Perception and Implications for Design

Roberta Klatzky

Carnegie Mellon University

Dept. of Psychology

Human-Computer Interaction Institute

Center for the Neural Basis of Cognition



Carnegie Mellon



## Haptic Perception and implications for design



- What is haptic perception?
- Based on **touch** receptors
  - in skin, muscles, tendons, joints
- Usually involves active seeking of information (exploration)

## Outline of Talk



- Neural Coding of Touch Primitives
- Functions of Peripheral Receptors
- Haptic Primitives and their Encoding
- Complementary Functions of Haptics and Vision
- Emergent Issues for Design

## 1. Neural Coding of Touch Primitives



- Touch Receptors
  - Mechanoreceptors and their Function
  - Other skin receptors: thermal, pain
- Pathways from receptors to brain

## Touch Receptors



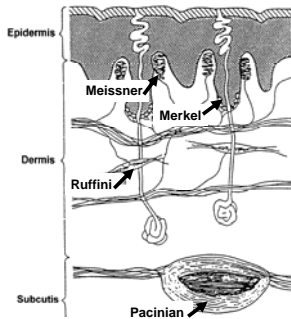
- Touch sensations are mediated by receptors that respond to pressure, vibration, and heat flow.
- The receptors are found in two regions:
  - Within skin: **cutaneous sensing**
    - ➔ pressure, temperature, pain
  - Beneath skin in muscles, tendons, joints: **kinesthetic sensing**
    - ➔ limb position and movement

## Classes of Receptors



- Receptors that respond to pressure and vibration are called **mechanoreceptors**
- Mechanoreceptors are found in skin (cutaneous) and muscles, tendons, and joints (kinesthesia)
- The skin also includes other receptors
  - that signal skin warming and cooling (**thermo-receptors**)
  - that signal pain (**nociceptors**)

## Skin Mechanoreceptors have specialized endings



### Functional Name:

Fast adapting Type I

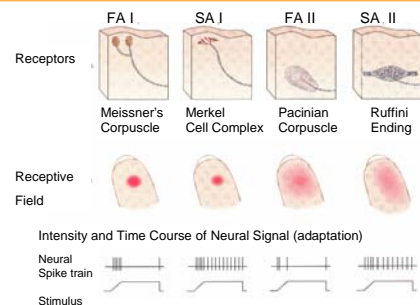
Slow adapting Type I

Slow adapting Type II

Fast adapting Type II

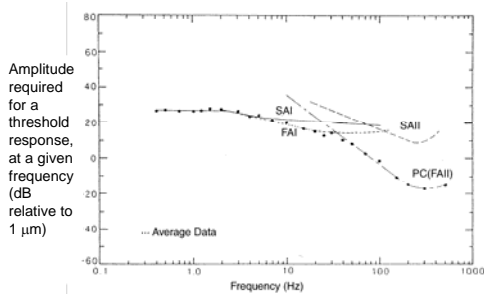
Johansson & Valbo, 1983

## Functional characteristics of Skin Mechanoreceptors: Receptive field size and adaptation rate



Kandel et al., 2000

## Frequency Sensitivity: One-channel-per-mechanoreceptor model



Bolanowski et al., 1988

## Thermal Receptors

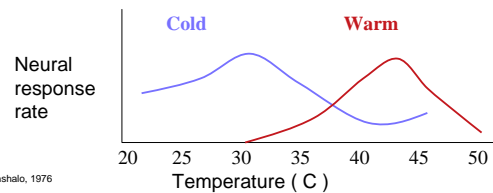


Thermo-receptors lack specialized endings.

Two populations of nerve fibers: warm and cold

Each has its own response range, with some overlap.

Note: body temperature in Centigrade = 37°.



Kenshalo, 1976

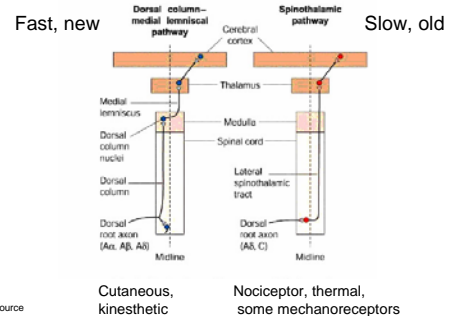
## A variety of neural fibers lead from periphery to spinal cord.



4 types, varying in conduction speed and associated receptor populations

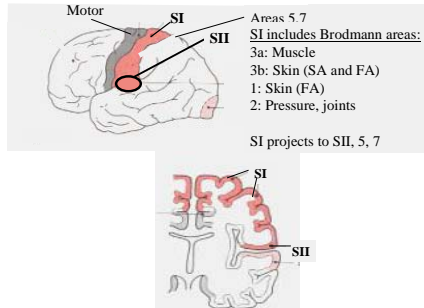
Fiber	Speed	Associated receptors
A-α	fast	kinesthetic
A-β	moderately fast	kinesthetic & cutaneous mechano.
A-δ	moderately fast	thermo- & nociceptor (sharp pain)
C	slow	thermo- & nociceptor (burning pain)

## Two pathways from cord to the brain

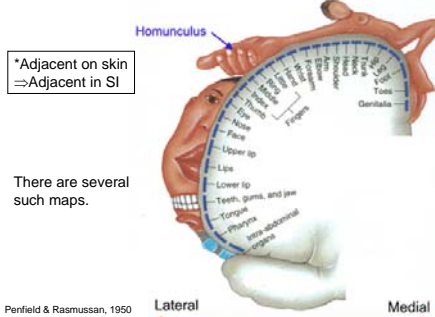


Unidentified source

## The brain: Primary and Secondary Somatosensory Cortex



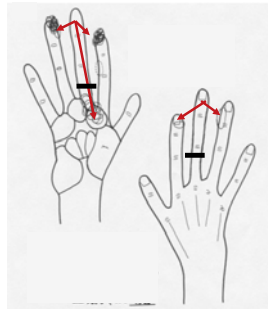
## Somatosensory "Homunculus" in SI results from somatotopic mapping\*



## Somatosensory areas in brain are plastic: Re-assignment of receptive fields after amputation of a digit



Areas in SI that once responded to 3rd fingertip are now activated by finger 2 and 4, plus base of 3



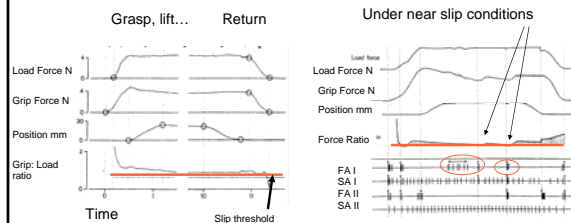
Merzenich et al., 1984

## 2. Functions of peripheral receptors



- Cutaneous mechanoreceptors provide array (tactile) sensing.
- Kinesthetic mechanoreceptors provide a sense of limb position, movement.
- They support different human abilities.

## Grasping is supported by FAI mechanoreceptors that detect incipient slip

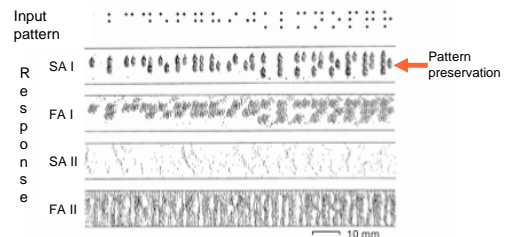


Johansson & Westling, 1984; Westling, 1986

## Tactile Pattern Perception is based on SA I mechanoreceptors



Spatial plot of the response of a skin mechanoreceptor to a Braille pattern swept through its receptive field: Each tick mark is a neural impulse given contact from that stimulus location. SA I have resolution of ~.5 mm at fingertip.

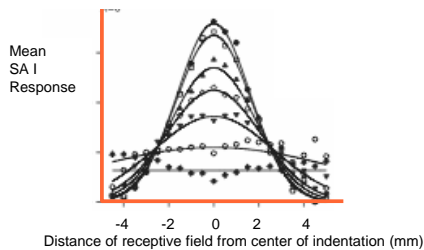


Phillips, Johansson & Johnson, 1990

## Curvature perception also reflects SA I responses



Data are shown for 7 curves, ranging from radius zero to radius 1.44 mm

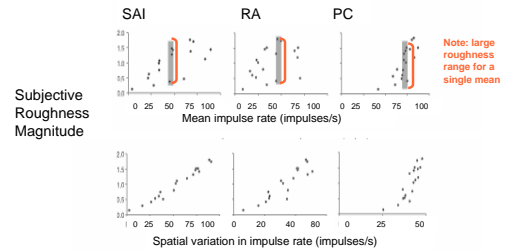


Goodwin et al., 1995

## Roughness perception is directly related to spatial variation in SA I responses

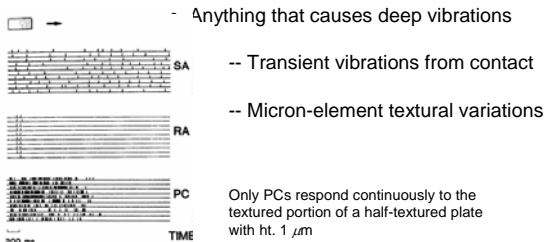


Mean impulse rate does not predict roughness magnitude.



Connor, Hsiao, Phillips, Johnson, 1990

## What do PC receptors signal?

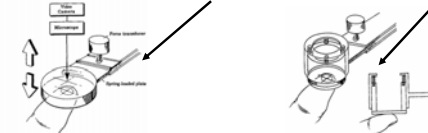


Srinivasan, Whitehouse, & LaMotte, 1990

## Kinesthetic mechanoreceptors are involved in softness perception



Softness discrimination of rubber specimens vs. rigidly covered springs



- Pure **kinesthetic** condition: anesthetize fingertips -- no cutaneous cues
- Pure **cutaneous** condition: passively press stimulus against fingerpad -- no muscle/tendon/joint involvement
- **Dual-cue** condition: normal, active touch
- Deformable surface (rubber) could be judged from cutaneous cues alone.
- Rigid (spring-loaded) surfaces required kinesthesia + cutaneous cues.

Srinivasan & LaMotte, 1995

## 3. Haptic primitives and their encoding



- Haptically perceived properties of objects and surfaces
- Link between exploration and perception of haptic properties

## What are haptically perceptible properties of objects?



- Geometry: Size (2-D, 3-D), Shape, Pointiness..
- Surface: Roughness, Slipperiness, Warmth...
- Substance: Compliance, Weight





## How do people perceive these properties?



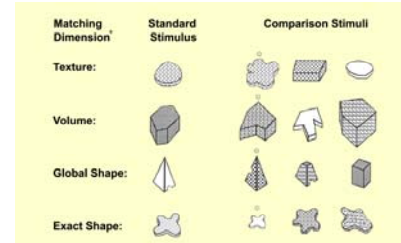
- Contact
  - Sufficient for some information
- Purposive Exploration
  - Highly informative
  - Highly stereotyped
  - Directly linked to the information that is desired

## Match-to-sample task shows links from properties to exploration



Task:  
Pick best match to standard stimulus on given dimension

Data: How people explored



Other properties not shown: temperature, weight, hardness  
Lederman & Klatzky, 1987

## Exploratory Procedures and Links to Properties



This exploration optimizes output of underlying neural systems and computational processes.

But, Non-optimal exploration still delivers some information.



Lederman & Klatzky, 1987

## Optimal and non-optimal exploration



EP	Property							Duration (s)
	Text	Hard	Temp	Wt	Vol	Global Shape	Exact Shape	
Lateral Motion	■							3
Pressure		■						2
Static Contact			■					<1
Unsupp. Holding				■				2
Enclosure					■			2
Contour Follow						■	■	11

□ Chance    □ Sufficient    ■ Optimal    ■ Necessary

Row: EP used to discriminate  
Column: Property discriminated  
Diagonal: Predicted EP/property link

Lederman & Klatzky, 1987;  
Klatzky & Lederman, 1993

## 3. Complementary functions of haptics and vision



- Material vs. Geometric Properties
- Differential Accessibility of these Properties
  - Preferences for free exploration
  - Speed of perceptual access
- Integration across modalities

## Material vs. Geometric Properties



- Geometric Properties
  - Depend on particular sampled object
  - Describe size, structure
  - Relatively accessible to Vision
- Material Properties
  - Independent of object's form (within extremes)
  - Describe surface and substance
  - Examples: warmth, roughness, ...
  - Relatively accessible to Haptics

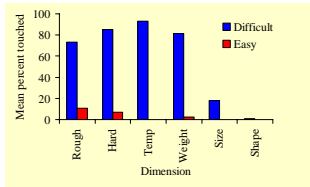
## Preferences under free exploration with vision and touch



- Subjects asked, which of two objects is rougher, harder...
- Some questions were easy (roughness of sandpaper vs. silk), some difficult (size of golf ball vs. marshmallow)

Touch is used for difficult judgments of material properties.

Otherwise, vision used for perceptual comparison or to trigger memory retrieval.

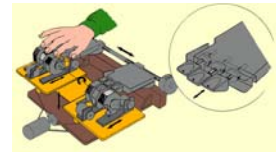


Klatzky, Lederman, & Matula 1993

## Speed of access to properties by touch favors material over geometry



- Subjects search for a target object presented to one finger, while other fingers get 0-5 distractors



Lederman & Klatzky, 1997; apparatus developed at Queen's University

## Speed of access, continued



- Judgments of several types of properties



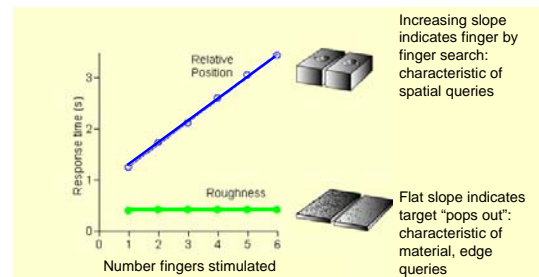
Material

Edge content

Spatial Relations

Lederman & Klatzky, 1997

## Speed of access, continued



Lederman & Klatzky, 1997

## Haptic/visual integration



- If material properties are relatively accessible to touch,
- And geometric properties are relatively accessible to vision,
- How do the two modalities interact?
- Apparently -- often -- optimally!
  - Ernst and Banks

## Maximum Likelihood Model



- Vision, touch assumed to input to a common integrator
- Integrator weights signals by reliability: inversely related to variance
- Integrator adds weighted signals

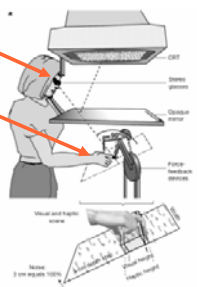
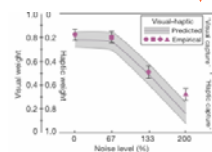
## Applying the model to roughness perception



Visual input from stereo goggles  
Noise added by eliminating stereo from some locations

Haptic input from PHANTOM force feedback device

Results: Optimal weighting of vision, depending on noise



Ernst & Banks, 2002

## 5. Emergent Issues for Design



- Implications of encoding pathways
- Implications of intersensory integration
- The “holy grail” of haptic interface design

## Implications of haptic encoding pathways



- Multiple receptors enable touch to encode a variety of neural primitives
  - A map of deformation on the fingertip
  - Vibration, skin stretch, temperature
  - Also enable fine manipulation
- Peripheral primitives, combined with active exploration, richly depict objects and surfaces
  - Haptics favors material over geometry
  - Haptics and Vision are complementary

## Implications of intersensory integration



- Inputs from other sensory systems are integrated with haptics
- Integration is sensitive to sensory reliability
  - Vision can compensate for relatively coarse spatial coding by haptics
  - Haptics can compensate for relatively coarse material coding by vision
  - One sense can be “fooled” by the other,
    - e.g., coarse visual textures make surfaces feel rougher;
    - visual curves make haptic edges feel rounder

## The “holy grail”



- Not just your grandmother’s force-feedback device
- Need dense, robust, array stimulators
- Thermal stimulation highly useful
- Exploration should be maximally enabled
- Add other modalities to capitalize on human multi-sensory abilities

## Introduction to 3-DoF Haptic Rendering

Ming C. Lin

Department of Computer Science  
University of North Carolina at Chapel Hill

<http://www.cs.unc.edu/~lin>

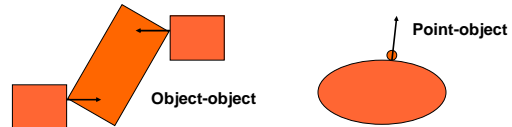
[lin@cs.unc.edu](mailto:lin@cs.unc.edu)



## 3DOF Haptics: Introduction



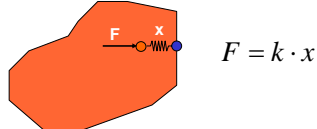
- Output: 3D force  $\rightarrow$  3DOF haptics
- Limited to applications where point-object interaction is enough.
  - Haptic visualization of data
  - Painting and sculpting
  - Some medical applications



## 3DOF Haptics: Basic approach



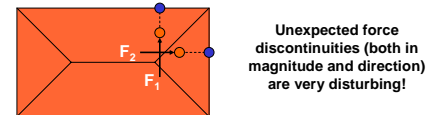
- Check if point penetrates an object.
- Find closest point on the surface.
- Penalty-based force.



## 3DOF Haptics: The problems



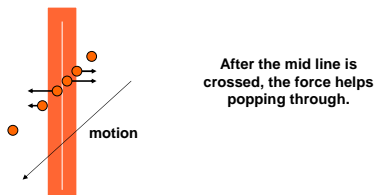
- Force discontinuities when crossing boundaries of internal Voronoi cells.



## 3DOF Haptics: The problems



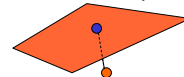
- Pop-through thin objects.



## 3DOF Haptics: Point-to-plane



- Mark et al., 1996.
- For distributed applications. The simulator sends the equation of a plane to the haptic loop. The update of the plane is asynchronous.
- Forces are computed between the haptic point and the plane.
- Possible stiffness is limited, because too stiff would bring a jerky behavior at low update rates.



### 3DOF Haptics: God-object

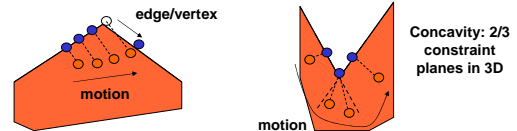


- Zilles and Salisbury, 1995.
- Use the position of the haptic interface point (HIP) and a set of local constraint surfaces to compute the position of god-object (GO).
- Constraint surfaces defined using heuristics.
- Compute GO as the point that minimizes the distance from HIP to the constraint surfaces. Lagrange multipliers.

### 3DOF Haptics: God-object



- Constraint surfaces:
  - Surfaces impeding motion
  - GO is outside (orientation test) and in the extension of the surface.
  - The HIP is inside the surface.



### 3DOF Haptics: God-object



- Constraint plane equations:
 
$$A_i \cdot x + B_i \cdot y + C_i \cdot z + D_i = 0 \quad 3 \text{ planes at most}$$
- Energy function that will account for the distance.
 
$$E = \frac{1}{2} \cdot k \cdot ((x - x_{HIP})^2 + (y - y_{HIP})^2 + (z - z_{HIP})^2)$$
- Define cost function using Lagrange multipliers.
 
$$C = \frac{1}{2} \cdot k \cdot ((x - x_{HIP})^2 + (y - y_{HIP})^2 + (z - z_{HIP})^2) + \sum_{i=1}^3 \lambda_i \cdot (A_i \cdot x + B_i \cdot y + C_i \cdot z + D_i)$$
- Minimize, solving for  $x, y, z, \lambda_1, \lambda_2$  and  $\lambda_3$ .

### 3DOF Haptics: God-object



- Partial derivatives on  $x, y, z, \lambda_1, \lambda_2$  and  $\lambda_3$  yield 6 linear equations:

$$\begin{pmatrix} 1 & 0 & 0 & A_1 & A_2 & A_3 \\ 0 & 1 & 0 & B_1 & B_2 & B_3 \\ 0 & 0 & 1 & C_1 & C_2 & C_3 \\ A_1 & B_1 & C_1 & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & 0 & 0 \\ A_3 & B_3 & C_3 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} x_{HIP} \\ y_{HIP} \\ z_{HIP} \\ D_1 \\ D_2 \\ D_3 \end{pmatrix}$$

- In case of less than 3 planes, the problem has a lower dimension.

### 3DOF Haptics: Virtual proxy

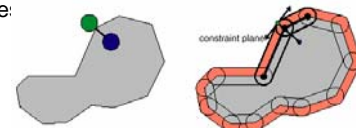


- Ruspini et al., 1997.
- Based on god-object.
- Virtual proxy is a small sphere, instead of a point. Use configuration-space obstacles (C-obstacles), from robotics.
- More formal definition of constraint planes.
- Implementation of additional features, based on relocation of the virtual proxy.

### 3DOF Haptics: Virtual proxy



- C-obstacles: for a spherical object, is reduced to computing offset surfaces at a distance equal to the radius of the sphere.
- Check the HIP against the offset surface.
- This is done to avoid problems with small gaps in the me:

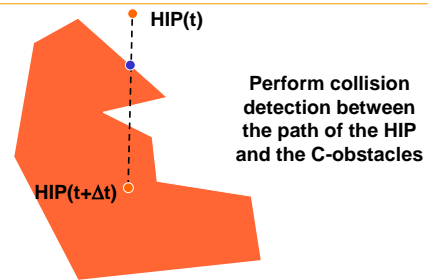


### 3DOF Haptics: Virtual proxy

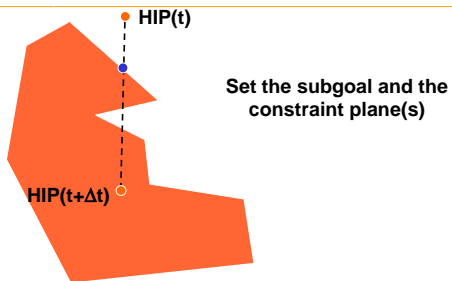


- Finding the virtual proxy is based on an iterative search.
- Basically, find subgoals based on the same distance minimization as for the god-object.
- At each subgoal, all the planes that go through that point are potential constraints. The minimum set of active constraints is selected.
- If the subgoal is in free space, set as new subgoal the HIP. The path might intersect the C-obstacles. Add the first plane intersected as a constraint and the intersection point as the current subgoal.
- The process ends when the virtual proxy becomes stable.

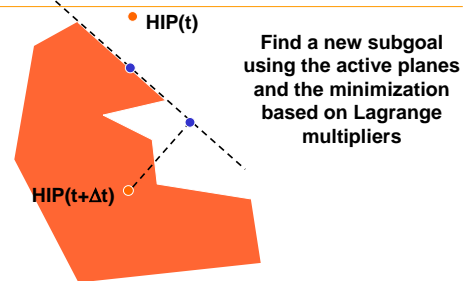
### 3DOF Haptics: Virtual proxy



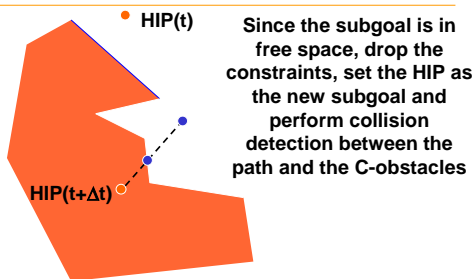
### 3DOF Haptics: Virtual proxy



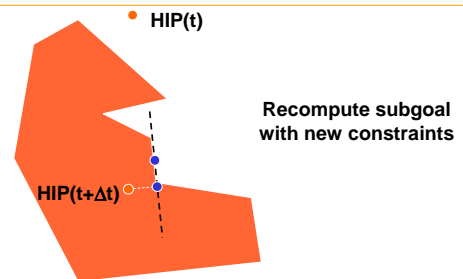
### 3DOF Haptics: Virtual proxy



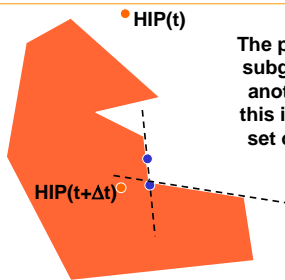
### 3DOF Haptics: Virtual proxy



### 3DOF Haptics: Virtual proxy

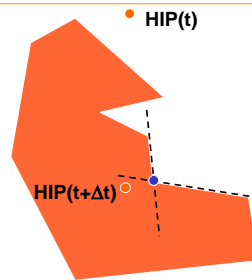


### 3DOF Haptics: Virtual proxy



The path to the new subgoal intersects another plane, so this is added to the set of constraints

### 3DOF Haptics: Virtual proxy



Compute active constraints (in 2D there are only 2) and find subgoal

For this example, this is the final position of the virtual proxy

### 3DOF Haptics: Virtual proxy



- Quadratic programming approach:

- The constraint planes define an open convex region (bounded by the plane at infinity).
- The function to minimize is the distance from the haptic device (HIP) to the new subgoal ( $VP_{i+1}$ ):

$$C = \|VP_{i+1} - HIP(t + \Delta t)\| \quad \text{Quadratic function}$$

- The translation from the current location to the new subgoal cannot intersect the constraint planes. Define linear constraints based on the normals of the planes.

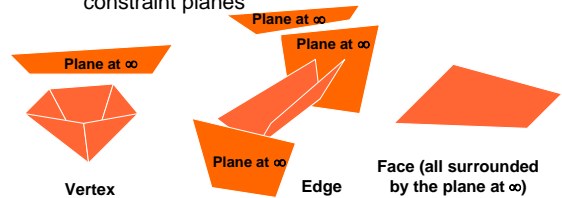
$$N_i \cdot (VP_{i+1} - VP_i) \geq 0 \quad \text{Linear constraints}$$

### 3DOF Haptics: Virtual proxy



- Special case: simplified approach.

- Let's look at the convex region defined by the constraint planes



### 3DOF Haptics: Virtual proxy

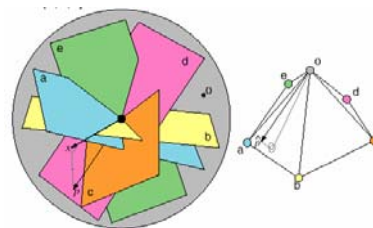


- Normals ( $n$ ) become points ( $-n$ ) in a dual space (Gauss map).
- The plane at infinity is the origin ( $o$ ).
- Points ( $-n$ ) are joint together if the associated planes intersect.
- The position of the virtual proxy also has a dual:

$$P = \frac{HIP - VP_i}{\|HIP - VP_i\|}$$

- The vertices of the closest feature to  $P$  are the duals of the active constraint planes that are used for the minimization with Lagrange multipliers.

### 3DOF Haptics: Virtual proxy

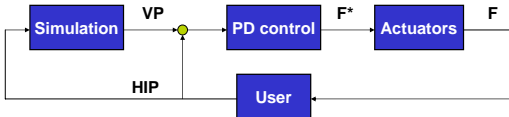




### 3DOF Haptics: Virtual proxy



- Force output: PD (proportional- derivative) control. Produces a force that will try to keep the VP and the HIP at the same position.



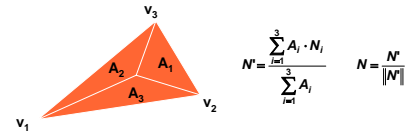
$$F^* = K_p \cdot (VP - HIP) + K_d \cdot \frac{d(VP - HIP)}{dt}$$

It's like a spring+damper, but the authors look at it from a control engineering approach

### 3DOF Haptics: Additional features



- Force shading by Basdogan et al., 1997.
  - Interpolate per-vertex normals using barycentric coordinates

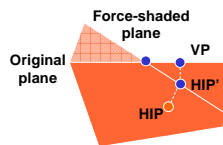


- Effect: edges and vertices seem 'rounded'

### 3DOF Haptics: Additional Features



- Force shading by Ruspini et al., 1997.
  - Modify the position of the VP.
  - A subgoal (HIP') is computed changing the normal of the plane.
  - This subgoal replaces the HIP, and the final VP is computed.



### 3DOF Haptics: Additional Features



- Other effects by Ruspini et al., 1997.
  - Friction.
  - Textures.
  - Deformation.
- All of them are achieved performing operations that modify the position of the VP.

### 3DOF Haptics: Additional Features



- Friction by Hayward and Armstrong, 2000.
  - Model friction as an elasticity between the actual contact point (x) and a fictitious point, called adhesion point (w).
  - The adhesion point remains static or follows the actual contact depending on the 'state' (sliding, static...)



### 3DOF Haptics: H-Collide

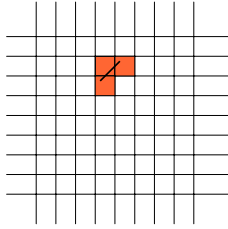


- What about the collision detection?
  - Spatial uniform decomposition of space. Locate the path traversed by the probe in that grid, using a hash table.
  - Test the path against an OBBTree. Optimized test.
  - Use frame-to-frame coherence, caching the previous contact, and perform incremental computation.
  - When the leaf intersects, compute the surface contact point (SCP). Pass this to GHOST (haptic rendering library used by the Phantoms).

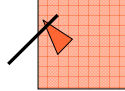
## 3DOF Haptics: H-Collide



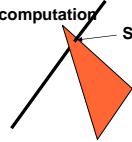
Spatial partitioning with hash table



Ray Vs. OBBTree test



SCP computation



## The End



For more information, see

<http://gamma.cs.unc.edu/interactive>

<http://gamma.cs.unc.edu/HCollide>

## Introduction to 6-DoF Haptic Display

Bill McNeely

Boeing Phantom Works

<http://www.boeing.com/phantom>

[bill.mcneely@boeing.com](mailto:bill.mcneely@boeing.com)



## Motivation

- Why six degrees of freedom?
  - Allowable motions of an extended rigid object
  - E.g., 3 Cartesian coordinates plus 3 Euler angles
- Basic to most real-world tasks
  - Virtual prototyping
  - Training

## Why is 6-DOF so difficult?



- Detect all contacts of interacting objects
  - Assume rigid bodies, but otherwise no *a priori* constraints on shape or motion
- Calculate forces and torques
  - For hard contact / contact avoidance
- 1000+ Hz haptic update rate
  - 10+ Hz graphic update rate

## Key decisions



- Collision Detection
  - Choice of representation and algorithm
- Interaction Paradigm
  - Penalty forces
  - Virtual coupling
  - Newtonian dynamics / Quasi-static approximation
  - Single user vs. collaboration

## Further decisions



- Decouple haptic and simulation loops?
  - Use intermediate representations?
- Force type and quality
  - How hard does hard contact feel?
  - How free does free-space feel?
    - Repulsive forces?
    - Force artifacts / stability considerations

## Three Approaches



- Voxmap PointShell™ (VPS)
  - W. McNeely, K. Puterbaugh, and J. Troy
- Sensation-Preserving Simplification
  - M. Otaduy and M. Lin
- Spatialized Normal-Cone Hierarchies
  - D. Johnson and E. Cohen

## Voxel Sampling for Six-DoF Haptic Rendering

Bill McNeely

Boeing Phantom Works

<http://www.boeing.com/phantom>

[bill.mcneely@boeing.com](mailto:bill.mcneely@boeing.com)



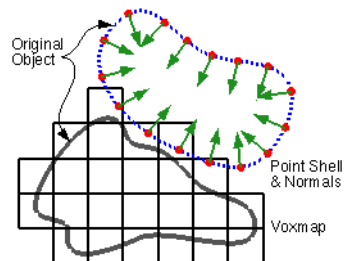
## VPS decisions

- Voxel representation; Point-voxel sampling
  - 3-level 512-tree voxel hierarchy
    - Voxels / chunks / hyperchunks
- Penalty forces; Virtual coupling
- Tactical degradation of force quality
  - Free-space viscosity artifact
- No decoupling of haptic and simulation loops

## VPS collision detection



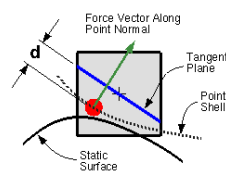
- Voxel sampling



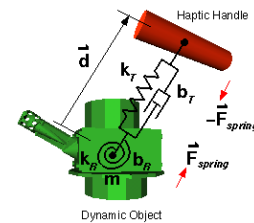
## VPS interaction paradigm



- Penalty Forces



- Virtual coupling



## Some myths about VPS



- “Cannot simulate multiple moving objects”
- “Not scalable to large scenarios”
- “VPS videos betray a stability problem”
- (Your favorite myth here)

## VPS progress since 1999

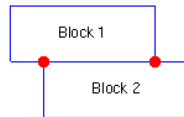


- 40x improvement in spatial accuracy
  - Applied to the design of Boeing 777-300ER
- Collaborative haptics
- See “Advances in Voxel-Based 6-Dof Haptic Rendering”

## Geometrical Awareness (2D)



- *Vertex-edge* contacts determine dynamically correct behavior of rigid polygons in 2D
  - “Edge” includes vertices



## Geometrical Awareness (3D)

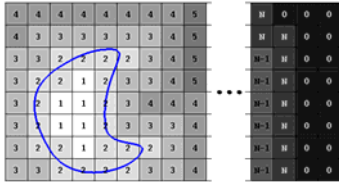


- *Vertex-surface* and *edge-edge* contacts determine dynamically correct behavior of rigid polyhedra in 3D
  - “Surface” includes edges
  - “Edge” includes vertices

## Voxel Based Distance Fields



- Voxels marked with distance-to-contact
- 3 types of fields: Surface, edge, vertex



## Temporal Coherence



- Anticipate force-generating collisions
- Sample fewer points using dead reckoning
  - Sampling rate varies inversely with distance-to-contact
- Slow down the motion if necessary
  - Prime directive: Avoid undetected contacts!
  - Accept a free-space viscosity artifact in trade

## Distance-to-contact queues



- Mark each point with distance-to-contact
- Order points into distance-specific queues
  - Points free to migrate between queues
- (Partially) traverse each queue every frame
- Extend this scheme to the voxel hierarchy
  - Hierarchical temporal coherence

## MaxTravel



**Definition:** *Maximum allowable travel of any point during current frame*

$$\text{MaxTravel} = (\text{nCapacity} - \text{nMandatory}) / \sum (n_i / (0.5 \text{ voxelsize} \cdot i))$$

where:

nCapacity = number of point-voxel tests CPU can perform in 1 ms  
 nMandatory = number of “mandatory” (contacting) points  
 i = index of “discretionary” (non-contacting) queue  
 n<sub>i</sub> = number of points currently occupying queue i

$$\text{Number of discretionary tests} = \text{MaxTravel} \cdot n_i / (0.5 \text{ voxelsize} \cdot i)$$

## What happens when $nMandatory$ exceeds $nCapacity$ ?



- Graceful degradation
  - If  $nMandatory$  exceeds  $(0.95 \cdot nCapacity)$ , then test every mandatory point plus  $(0.05 \cdot nCapacity)$  discretionary points
  - Allow frame rate to fall below 1000 Hz
    - Hopefully the haptic interface tolerates this situation
- Add more processing power?
  - Or use a larger voxel size

## Scalability characteristics



- CPU cycles per frame  $\sim O(1/voxelsize^2)$
- Amount of voxel data  $\sim O(1/voxelsize^3)$ 
  - $O(1/voxelsize^2)$  at currently realistic voxel sizes
- Multiple moving objects
  - CPU cycles  $\sim O(nMovingObjects^2)$

## Large-scale VPS haptics

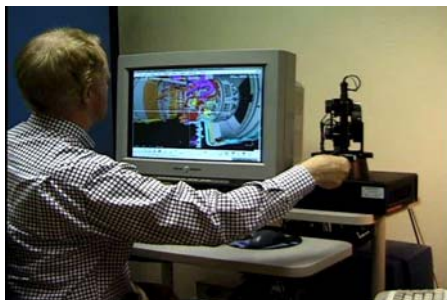


## VPS-enabled FlyThru®



- Client-server architecture
- 777-300ER main landing gear at  $\sim 1mm$  accuracy
  - 40,476 triangles moving (1.1 M points)
  - 2.7M triangles static (1.8 G voxels)
    - Dynamic pre-fetching of voxel data
  - SensAble Technologies PHANTOM® 1.5/6DOF
  - Bi-Xeon 2.8 GHz with 2 GB RAM

## Engine example



## VPS-enabled CATIA®/Haption® prototype



Interactive assembly simulation with force-feedback inside CATIA V5

*EuroHaptics 2004, Munich, Germany, June 5-7, 2004*

[jerome.perret@haption.com](mailto:jerome.perret@haption.com)

Haptic device: Virtuoso 6D35-45 by Haption  
Software solution: Catia V5 R11 using VPS/PBM  
CAD model: Courtesy of Renault



## CATIA/Haption demo



- 1.5mm voxel size
- 36,565 triangles static; 2,354 moving
- Voxelization time 1.5 minutes
- Bi-Xeon 2.4 GHz with 1 GB RAM
- Maximum force 35N, torque 3Nm



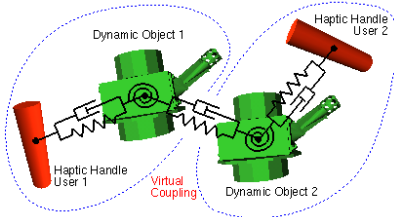
## Collaborative 6-DOF haptics



## Collaborative 6-DOF Haptics



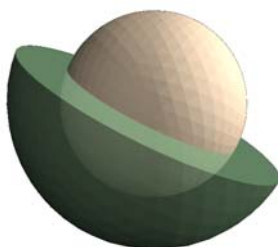
- “Type 2” cross-user virtual coupling
  - Ensures stability regardless of latency



## Virtual Swordplay



## Non-planar manifold contact



## Why is it a difficult scenario?



- Geometrical awareness loses effectiveness
- Extreme non-convexity is guaranteed
- Why not use formal kinematic constraints?
  - Tricky transitions between unilateral and quasi-permanent constraints
  - System complexity increases



## VPS experimental approach



- Allow slight surface interpenetration
- Retract pointshell slightly
  - E.g., by  $\frac{1}{2}$  voxel along inpointing surface normal
- Ball and socket example
  - At voxel size of  $(0.01 \cdot \text{diameter})$ , MaxTravel allows maximum rotational speed of 0.5 revolution/sec

## Sensation Preserving Simplification for 6-DoF Haptic Display

Miguel A. Otaduy

ETH-Zurich

<http://graphics.ethz.ch/~otmiguel>

otaduy@inf.ethz.ch



## Outline

- Motivation
- Sensation Preserving Simplification
- Multiresolution Collision Detection
- Rendering Pipeline
- Conclusion

## Bottleneck: Collision Detection

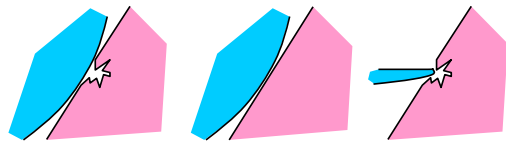


- Input:
  - Object A:  $m$  triangles      Object B:  $n$  triangles
- Query:
  - Return all pairs of triangles closer than  $d$
- Worst case scenario:  $O(mn)$ 
  - Parallel close proximity (large areas closer than  $d$ )
- Desired force update rate  $\sim 1\text{kHz}$

## Perceptual Motivation

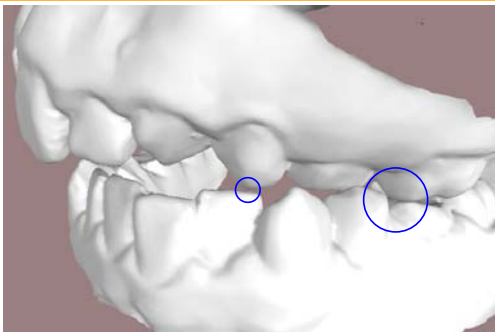


- Larger contact area  $\rightarrow$  Lower resolution

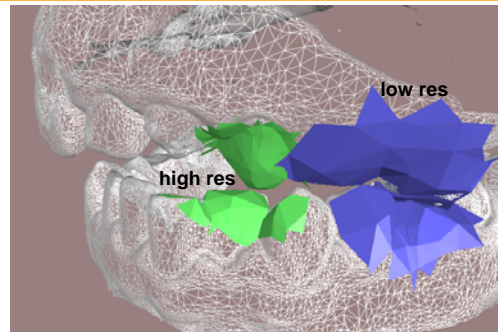


- Supported by studies on feature detection  
[Klatzky and Lederman 1995, Okamura and Cutkosky 1999]

## Goal: Adaptive Resolution



## Goal: Adaptive Resolution



## Levels of Detail



- Widely used in graphics [Hoppe 1996, Garland and Heckbert 1997, Lindstrom and Turk 1998...]
- Adaptive, view dependent [Luebke 1997, Hoppe 1997], but based on visual error metrics

## Bounding Volume Hierarchies



- Hierarchical data structures for fast pruning of non-colliding geometry
- Many different bounding volumes (AABBs, OBBs, spheres, k-dops, convex hulls...)

## Approach



- Compute contact information using multiresolution representations (LODs)
- Select LODs independently for each contact, based on local information
- Problem: Integration of LODs with BVHs for fast collision detection

## Contact Levels of Detail (CLODs)



- Unique hierarchy with dual functionality: LODs and BVH
  - Exploit hierarchical nature of both LODs and BVHs
  - Create LOD hierarchy and BVH simultaneously
  - Descend on BVH = Refine LODs

## Outline



- Motivation
- Sensation Preserving Simplification
- Multiresolution Collision Detection
- Rendering Pipeline
- Conclusion

## Creation of CLODs



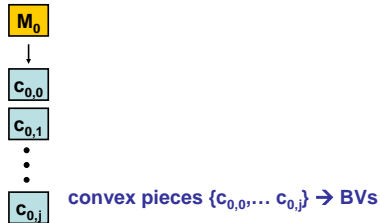
- Initial mesh

$M_0$

## Creation of CLODs



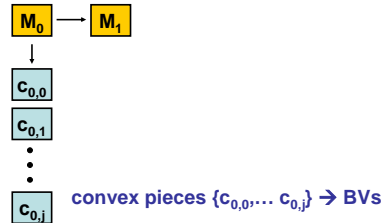
- Initialize convex decomposition



## Creation of CLODs



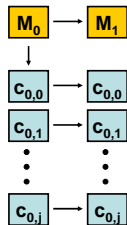
- Simplification + Filtering steps



## Creation of CLODs



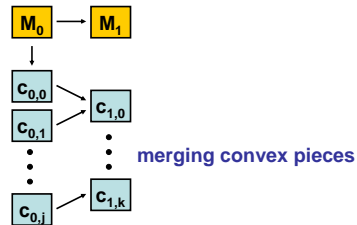
- Maintain convex decomposition



## Creation of CLODs



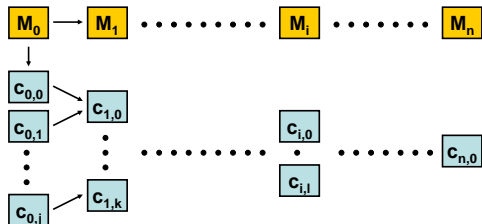
- Filtering → Convexification. Merge BVs



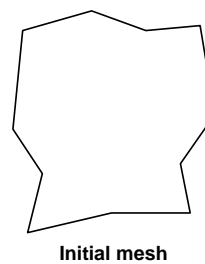
## Creation of CLODs



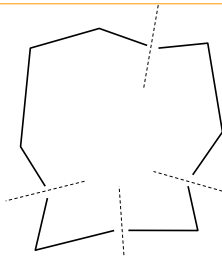
- Bottom-up construction



## 2D Example

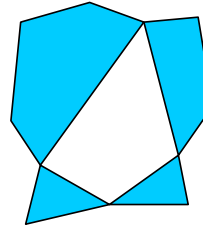


## 2D Example



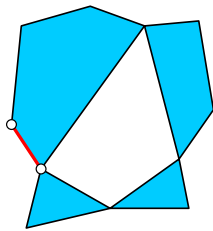
Convex surface decomposition

## 2D Example



BVH construction:  
convex pieces

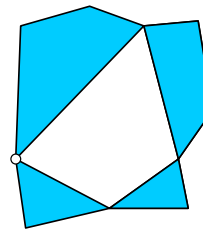
## 2D Example



Simplification



## 2D Example



Filtered edge collapse  
(to enforce convexity constraints)



## Filtered Edge Collapse



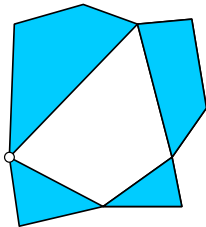
- Goals
  - Surface decimation
  - Filter detail
  - Convexification
- Constraints
  - Convexity constraints
  - Preserve topology

## Filtered Edge Collapse



1. Edge collapse initialization. Quadric error metrics [Garland and Heckbert 1997]
2. Unconstrained filtering [Guskov et al. 1999]
3. Optimization problem: minimize distance to unconstrained position subject to local convexity constraints
4. Bisection search: preserve topology and global convexity constraints

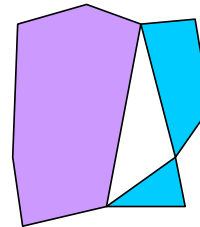
## 2D Example



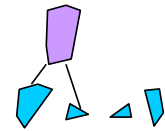
Filtered edge collapse  
(to enforce convexity constraints)



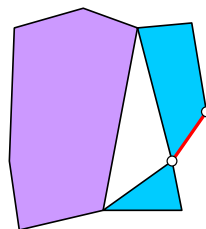
## 2D Example



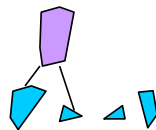
BVH construction:  
merge convex pieces



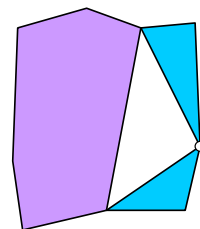
## 2D Example



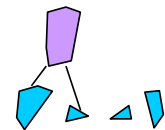
Simplification



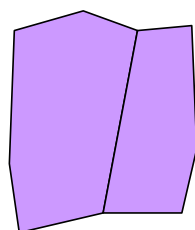
## 2D Example



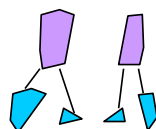
Simplification



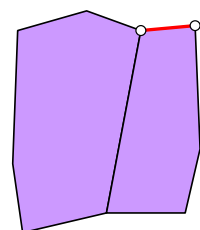
## 2D Example



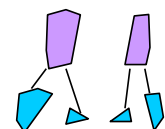
BVH construction



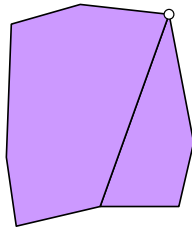
## 2D Example



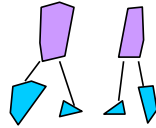
Simplification



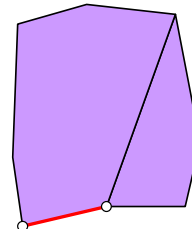
## 2D Example



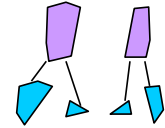
Simplification



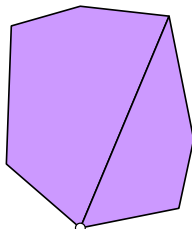
## 2D Example



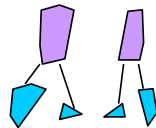
Simplification



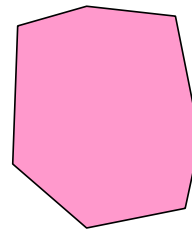
## 2D Example



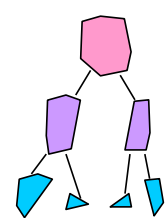
Simplification



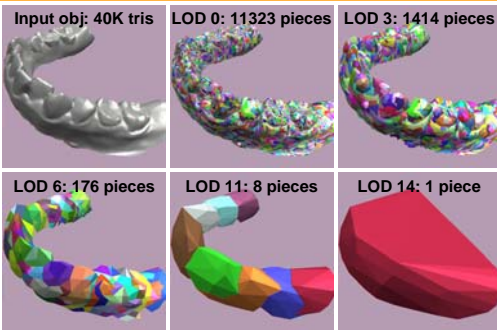
## 2D Example



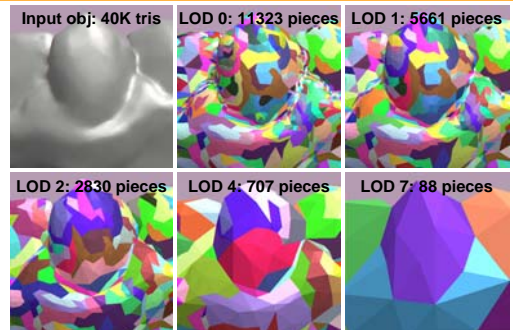
BVH construction



## Example of Hierarchy



## Example of Hierarchy



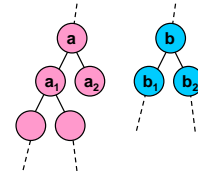


## Outline



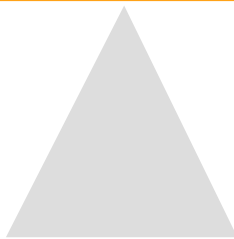
- Motivation
- Sensation Preserving Simplification
- Multiresolution Collision Detection
- Rendering Pipeline
- Conclusion

## Collision Detection: BVHs

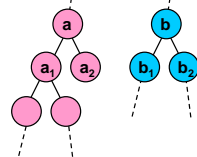


BVHs of objects  
A and B

## Collision Detection: BVHs

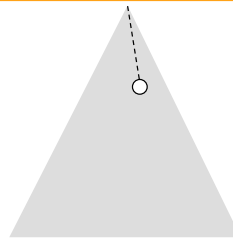


Bounding Volume  
Test Tree (BVTT)

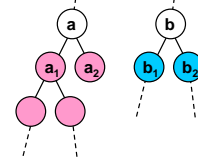


BVHs of objects  
A and B

## Collision Detection: BVHs



Bounding Volume  
Test Tree (BVTT)

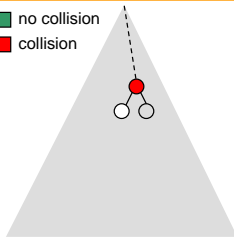


BVHs of objects  
A and B

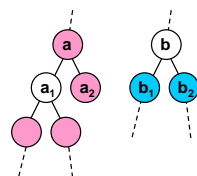
## Collision Detection: BVHs



- no collision
- collision



Bounding Volume  
Test Tree (BVTT)

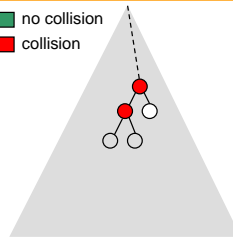


BVHs of objects  
A and B

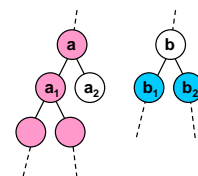
## Collision Detection: BVHs



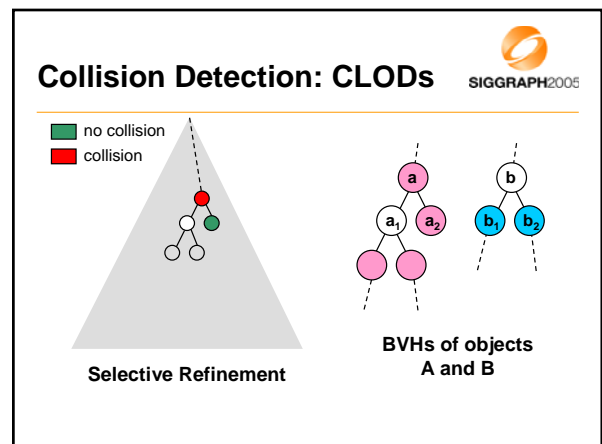
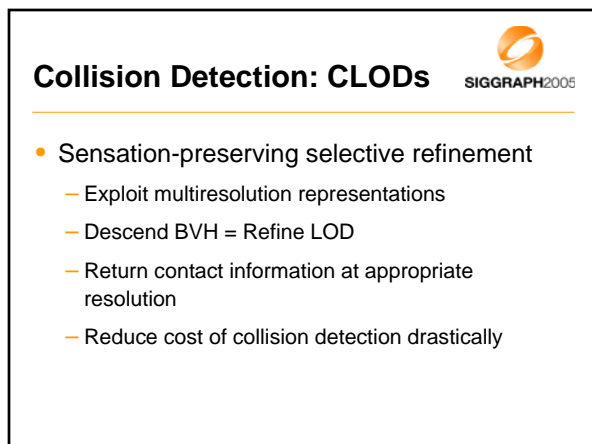
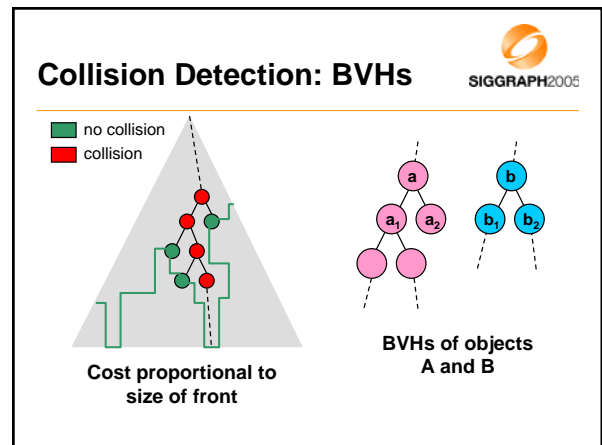
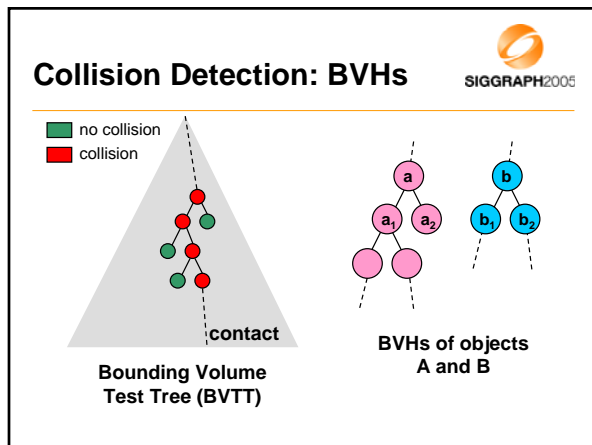
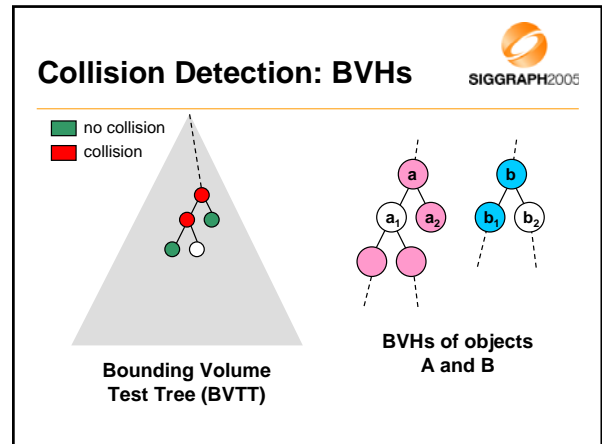
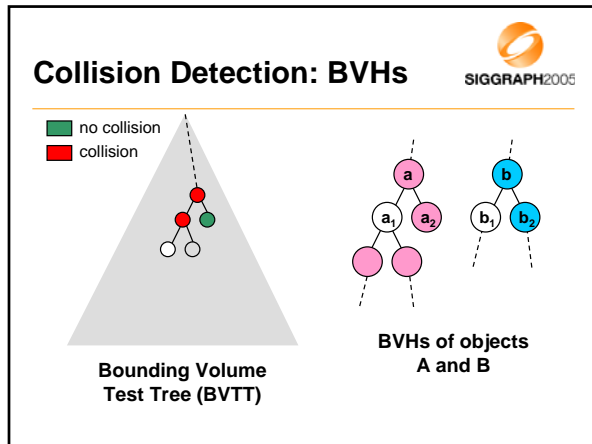
- no collision
- collision

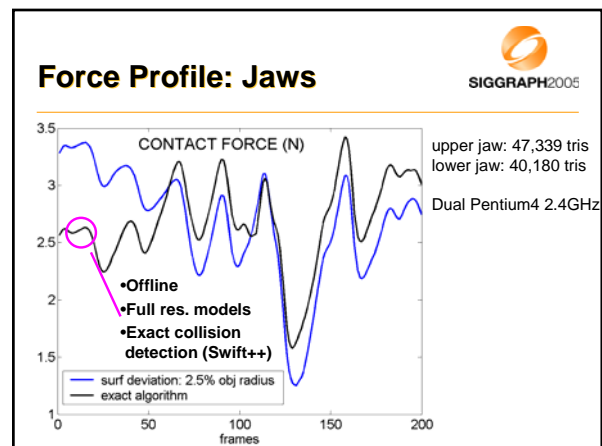
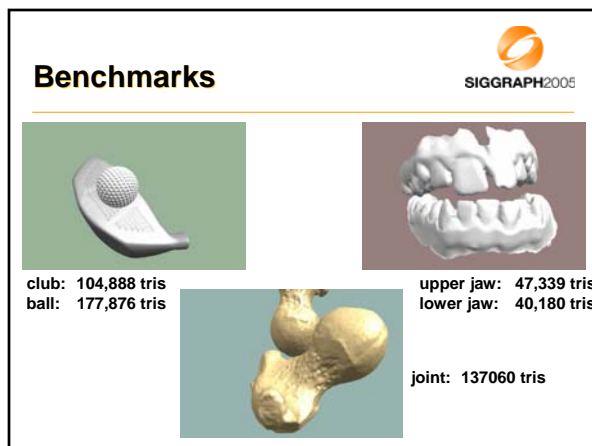
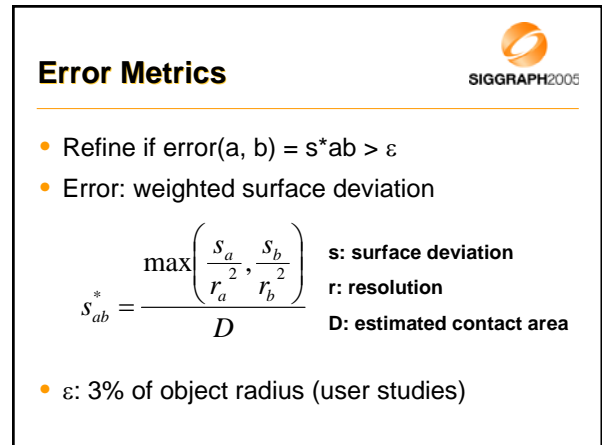
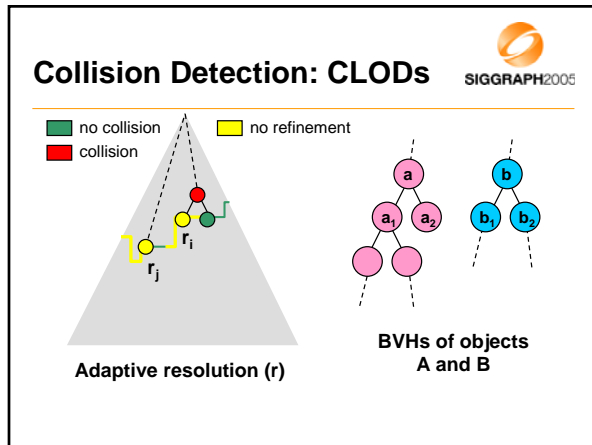
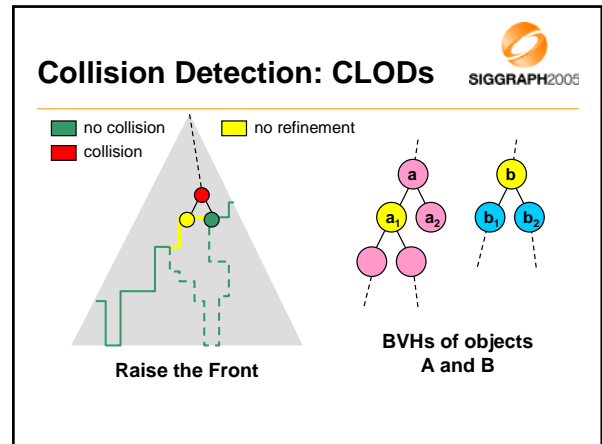
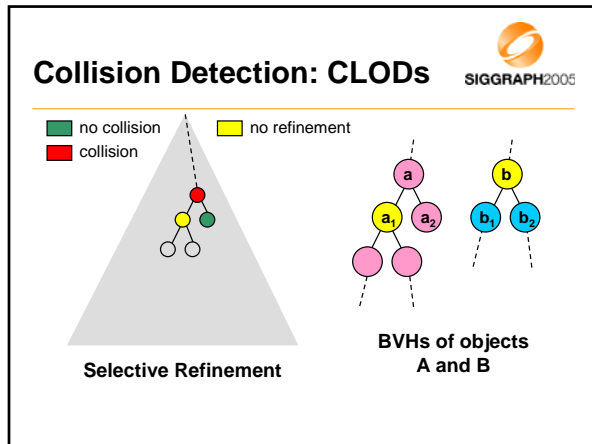


Bounding Volume  
Test Tree (BVTT)

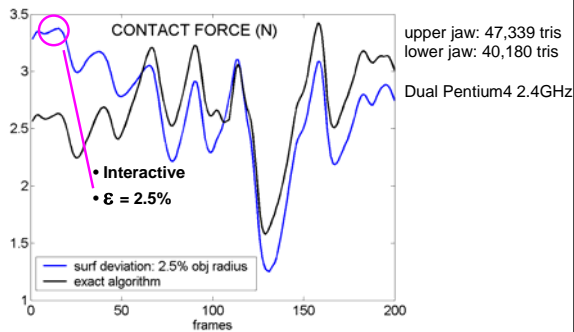


BVHs of objects  
A and B

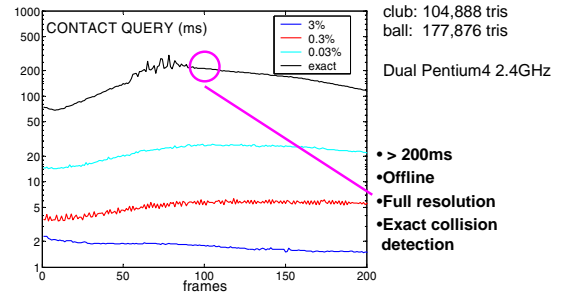




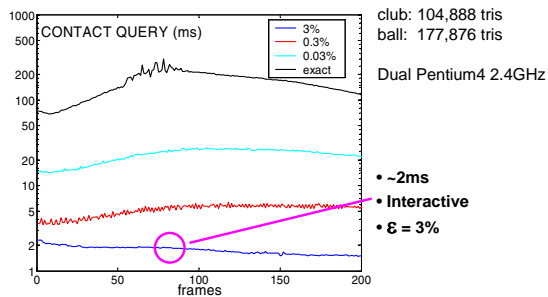
## Force Profile: Jaws



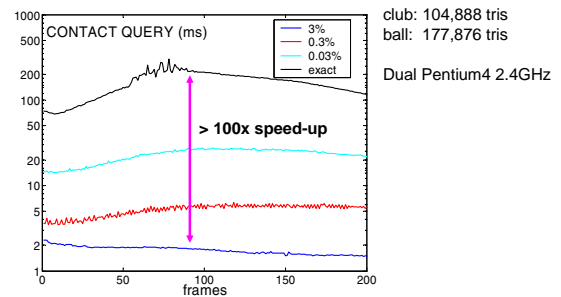
## Timings: Golf



## Timings: Golf



## Timings: Golf

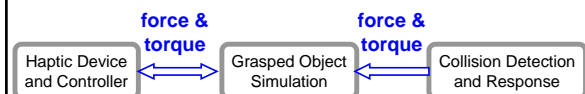


## Outline

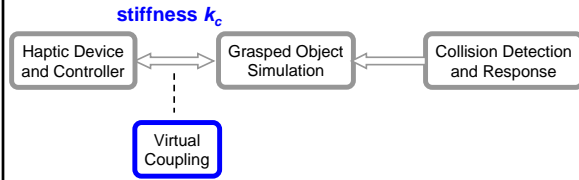


- Motivation
- Sensation Preserving Simplification
- Multiresolution Collision Detection
- Rendering Pipeline
- Conclusion

## Decoupled Modules

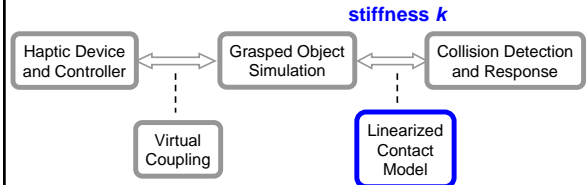


## Virtual Coupling



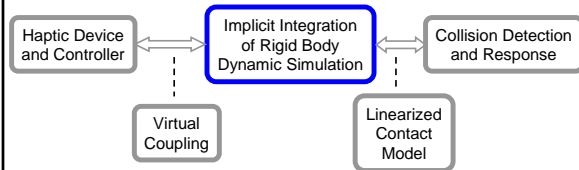
Decouple object simulation from synthesis of force feedback

## Linearized Contact Model



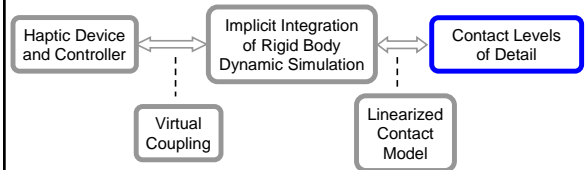
Decouple collision detection from simulation

## Grasped Object Simulation



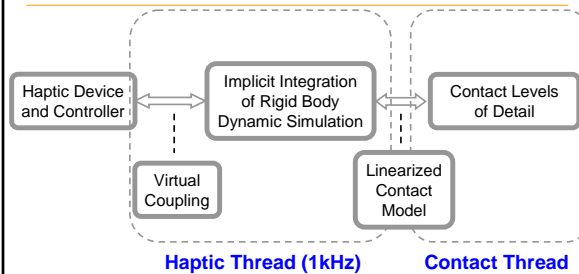
Larger range of stable stiffness values

## Multires. Collision Detection



Fast collision detection between complex objects

## Multirate Pipeline



## Stability and Responsiveness



## Outline



- Motivation
- Sensation Preserving Simplification
- Multiresolution Collision Detection
- Rendering Pipeline
- Conclusion

## Summary



- Contact levels of detail:
  - Multiresolution collision detection
  - Unifies LODs and BVHs
  - Per-contact adaptive selection of resolution
  - Error metrics based on perceptual studies

## Main Results



- Stable and responsive 6-DoF haptic rendering of complex models
  - Models: 40K – 170K (in complex contact scenarios)
  - Collision detection update rates: 300Hz – 500Hz
  - Collision detection error: 2.5% - 3% of object bounding radii
  - Speed-up: up to 100x

## Generalization



- CLODs data structure independent of BV
- Application to rigid body simulation
- Extension of sensation preserving simplification to other frameworks:
  - Voxel sampling [McNeely et al. 1999]
  - Normal cone hierarchies [Johnson and Cohen 2001]

## Limitations



- Lack of containment → virtual prototyping applications do not allow interpenetration
- Limited simplification aggressiveness due to topological constraints
- Static LODs and 'popping' effects

## Limitations



- Driving observation: small features cannot be perceived if the contact area is large
- Does not hold for:
  - Highly correlated features
  - Tangential motion
- Solution: 'Haptic Rendering of Textured Surfaces' (later in this course)

## References



*Sensation Preserving Simplification for Haptic Rendering.*

Miguel A. Otaduy and Ming C. Lin.

In Proc. of ACM SIGGRAPH 2003.

<http://gamma.cs.unc.edu/LODHaptics/>

*Stable and Responsive Six-Degree-of-Freedom Haptic Manipulation Using Implicit Integration.*

Miguel A. Otaduy and Ming C. Lin.

In Proc. of World Haptics Conference 2005.

<http://gamma.cs.unc.edu/SR6DoF/>

## Haptic Rendering of Polygonal Models Using Local Minimum Distances

David Johnson\*  
University of Utah  
<http://www.cs.utah.edu/~dejohnso>  
[dejohnso@cs.utah.edu](mailto:dejohnso@cs.utah.edu)



\*Joint work with Elaine Cohen and Pete Willemsen

## Motivation



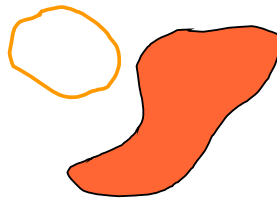
- Virtual Prototyping
  - Replace expensive and hard to make physical prototypes
  - Check assembly and accessibility
  - Test human ergonomics and aesthetics



## Haptic Computation



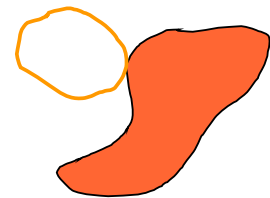
- Typically compute penetration depth



## Haptic Computation



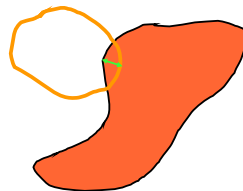
- Typically compute penetration depth



## Haptic Computation



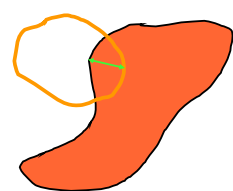
- Typically compute penetration depth
- Compute restoring force



## Haptic Computation



- Typically compute penetration depth
- Maintain restoring force

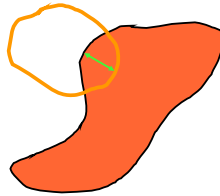




## Haptic Computation



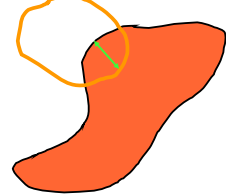
- Typically compute penetration depth
- Maintain restoring force



## Haptic Computation



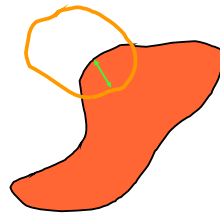
- Typically compute penetration depth
- Maintain restoring force



## Haptic Computation



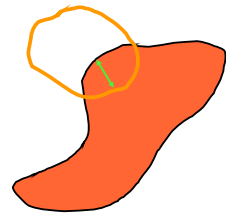
- Typically compute penetration depth
- Maintain restoring force



## Haptic Computation



- Typically compute penetration depth
- Maintain restoring force



## Difficulty Using Collision Status

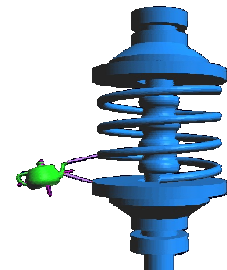


- Waiting until interpenetration violates real-world constraints
- In virtual prototyping application, we want to check whether models can fit – penetrating models reduces validity
- Penetration depth difficult

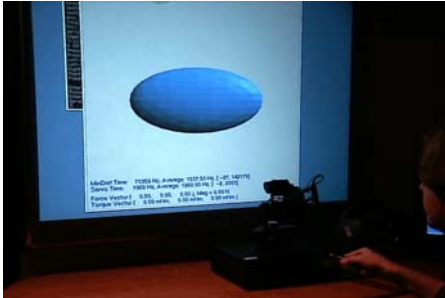
## Local Minimum Distance Approach



- Find local minimum distances between polygonal models
- Convert distances into forces (translational, torque)
- Forces guide objects to prevent interpenetration



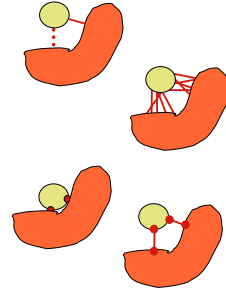
## Example



## Why are Local Closest Points Adequate?



- One global distance not enough
- All pairs within a cutoff distance too many
- Collisions are represented by pairs of contact points
- Move objects apart and pairs of points become local closest points



## Local minimum distance



- CAGD approach:

$$D^2(u, v, s, t) = F(u, v) - G(s, t)^2$$

Extrema when

$$\begin{aligned} (F - G) \cdot F_u &= 0 & (F - G) \cdot G_s &= 0 \\ (F - G) \cdot F_v &= 0 & (F - G) \cdot G_t &= 0. \end{aligned}$$

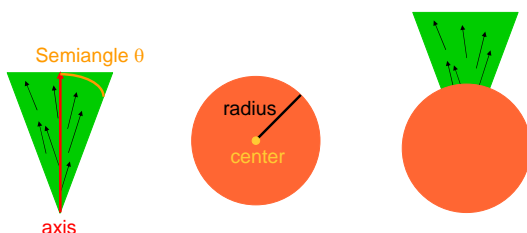
- Normals are the key to describing local minimum distance

## Computing LMD's between Polygonal Models



- Build bounding hierarchy
- Prune hierarchy based on normals, not distance
- Leaf tests between triangles

## Spatialized Normal Cone Hierarchy



## Constructing the hierarchy



- Use PQP package (UNC) to construct spatial bounding volume hierarchy
- At each node
  - find average normal and max spread to make normal cone

## Pruning the Hierarchy



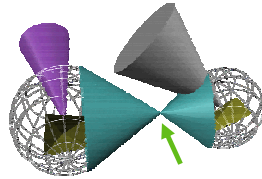
- Create double cone between bounding spheres

$$(F - G) \cdot F_u = 0$$

$$(F - G) \cdot F_v = 0$$

$$(F - G) \cdot G_s = 0$$

$$(F - G) \cdot G_t = 0$$



## Pruning the Hierarchy



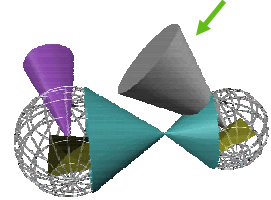
- Check normal cone of one model against double cone

$$(F - G) \cdot F_u = 0$$

$$(F - G) \cdot F_v = 0$$

$$(F - G) \cdot G_s = 0$$

$$(F - G) \cdot G_t = 0$$



## Pruning the Hierarchy



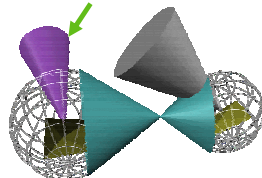
- Check normal cone of other model against double cone

$$(F - G) \cdot F_u = 0$$

$$(F - G) \cdot F_v = 0$$

$$(F - G) \cdot G_s = 0$$

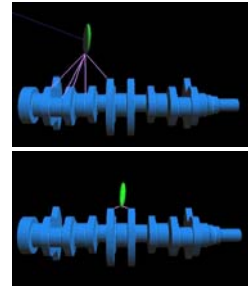
$$(F - G) \cdot G_t = 0$$



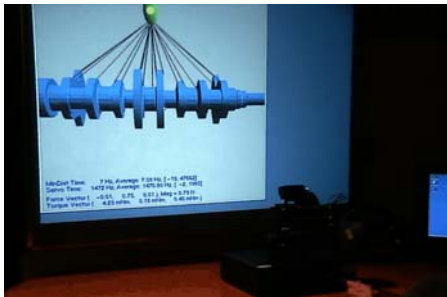
## Prune with Cutoff Distance



- We don't want all LMDs, just those nearby
- Modify computation to reject nodes that are further than cutoff
- This quickly prunes many nodes



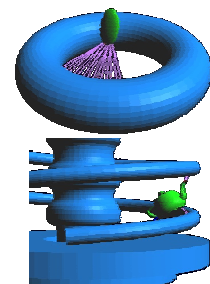
## Cutoff Example



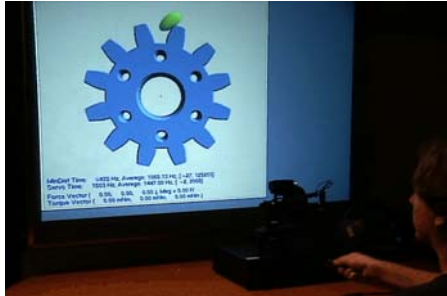
## Results



- Can handle arbitrary models
- Explore concave regions
- Provide sensations of contact and torque to guide model



## Example: Concave regions



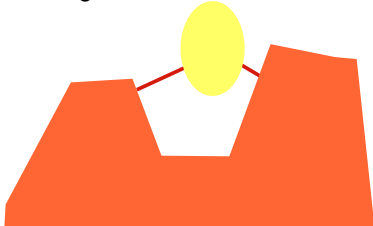
## Example: Non-mechanical Model



## Acceleration with Local Descent



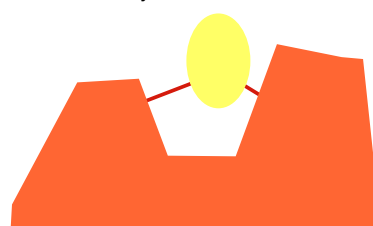
- First, do a global search



## Acceleration with Local Descent



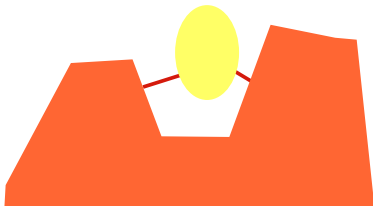
- Then, as the object moves, do local updates



## Acceleration with Local Descent



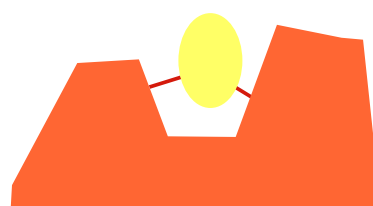
- Then, as the object moves, do local updates



## Acceleration with Local Descent



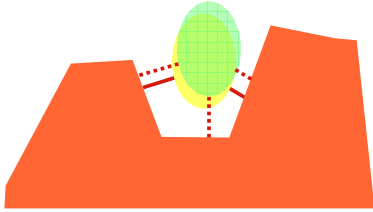
- Then, as the object moves, do local updates



## Acceleration with Local Descent



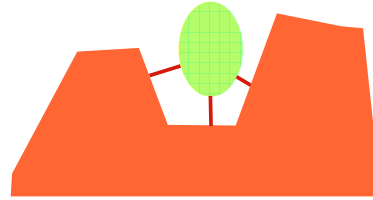
- Concurrently, repeat global search



## Acceleration with Local Descent



- Merge in new LMDs



## Application Design

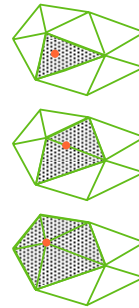


- Multi-threaded design
  - Global search
  - Local update and force computation
  - Graphics
- Communicate through shared data structures
- Dual 2.4GHz HT P4, shared memory, Linux/GNU computer

## Local Update



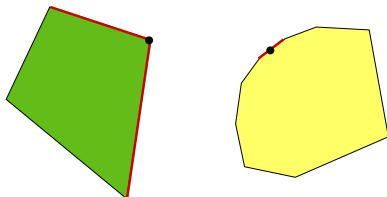
- Check neighbor triangles to move to a new local minimum
- Check triangles in one model's neighborhood against triangles in other model's neighborhood.
- Do this for each local minimum distance pair
- Can compute ~1,000,000 triangle distance pairs per second



## Local Update



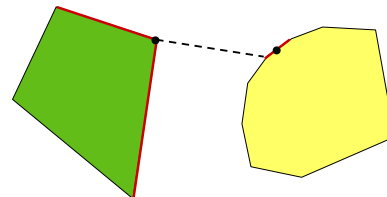
- Move and find neighborhood



## Local Update



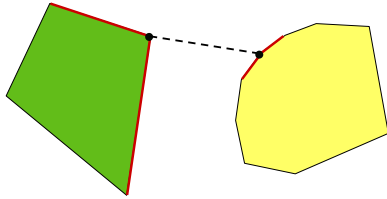
- Compute closest points between triangles



## Local Update



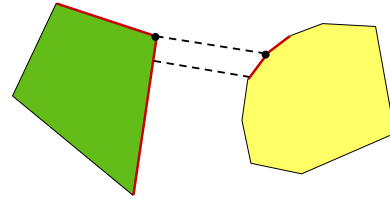
- Find new minimum and neighborhood



## Local Update



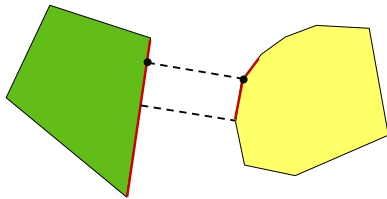
- Find new distances



## Local Update



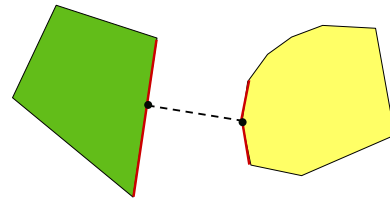
- Find new neighborhood and distances



## Local Update



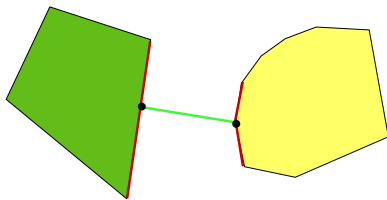
- Find new neighborhood and distances



## Local Update



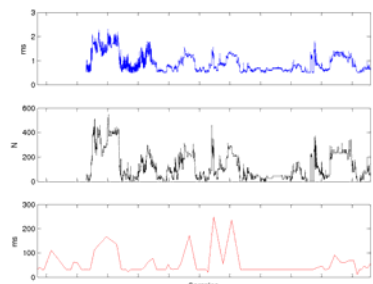
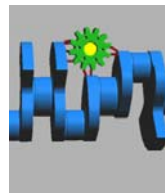
- Local search has converged



## Results



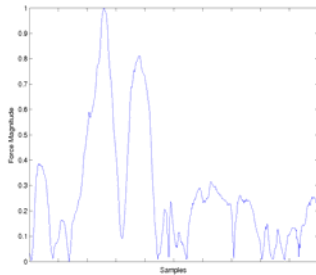
- Models with 10k-100k triangles



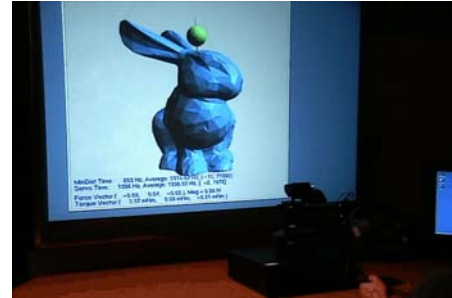
## More Results



- Haptic system is much smoother and more stable with local update



## Comparison – Just Global



## Comparison – Local Update

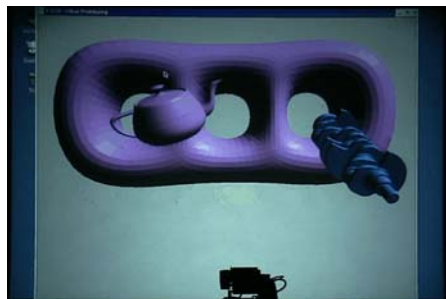


## Accessibility Application



- Save position and orientation at each time step
- If collision occurs
  - Revert to safe state several time steps before collision
  - Continue simulation
- Visualize path at end of trial
  - Sample saved positions
  - Draw model at each sample
  - Allow manipulation of scene to verify path

## Accessibility Application



## Conclusion



- High-performance haptic rendering
  - Using local techniques provides multiple orders of magnitude speedup
- General polygonal environment
- Accurate results
  - Adjustable clearance distances
- Keeps the human in the loop, but haptics provides guidance

## Haptic Rendering of Sculptured Models

Elaine Cohen\*  
University of Utah  
<http://www.cs.utah.edu/~cohen>  
[cohen@cs.utah.edu](mailto:cohen@cs.utah.edu)

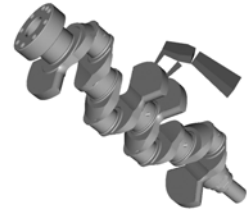


\* Joint work with Tom Thompson, Don Nelson, and David Johnson

## Why Sculptured Models?



- Prevalent format for manufacture and animation
- Compact and exact representation
- Surface normals vary smoothly



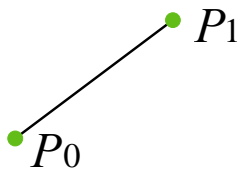
## Basic Spline Background



- Parametric function

— Simple example:  
interpolating two points

$$tP_0 + (1-t)P_1$$



## Basic Spline Background

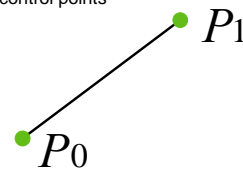


- Parametric function

— Simple example:  
interpolating two points

— Weighting control points

$$tP_0 + (1-t)P_1$$



## Basic Spline Background



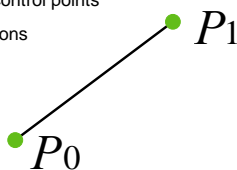
- Parametric function

— Simple example:  
interpolating two points

— Weighting control points

— Basis functions

$$tP_0 + (1-t)P_1$$



## Basic Spline Background

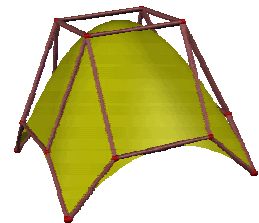


- Extends to

— higher-degree basis  
functions

— piecewise functions

— Higher dimensional  
domains

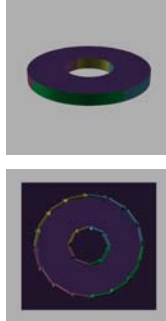




## Spline Background - Trims



- A trim is a curve in the domain
  - Often piecewise linear
- Removes part of domain and the corresponding surface
- Glues together multiple surfaces
  - Result of Boolean operations



## Distance Between a Point and Surface

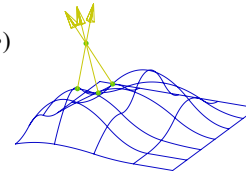


- Squared distance between point and surface

$$D^2(u, v) = (\mathbf{S}(u, v) - \mathbf{P}) \cdot (\mathbf{S}(u, v) - \mathbf{P})$$

- Extrema at simultaneous zeros of partials

$$\mathbf{F} = \begin{bmatrix} (\mathbf{S}(u, v) - \mathbf{P}) \cdot \mathbf{S}_u \\ (\mathbf{S}(u, v) - \mathbf{P}) \cdot \mathbf{S}_v \end{bmatrix}$$



## Finding Simultaneous Zeros



- Multidimensional Newton's Method

$$\mathbf{J} \cdot \Delta \mathbf{p} = -\mathbf{F}$$

- For minimum distance, this becomes

$$\begin{bmatrix} (\mathbf{S} - \mathbf{P}) \cdot \mathbf{S}_{uu} + \mathbf{S}_u \cdot \mathbf{S}_u & (\mathbf{S} - \mathbf{P}) \cdot \mathbf{S}_{uv} + \mathbf{S}_u \cdot \mathbf{S}_v \\ (\mathbf{S} - \mathbf{P}) \cdot \mathbf{S}_{uv} + \mathbf{S}_u \cdot \mathbf{S}_v & (\mathbf{S} - \mathbf{P}) \cdot \mathbf{S}_{vv} + \mathbf{S}_v \cdot \mathbf{S}_v \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = - \begin{bmatrix} (\mathbf{S} - \mathbf{P}) \cdot \mathbf{S}_u \\ (\mathbf{S} - \mathbf{P}) \cdot \mathbf{S}_v \end{bmatrix}$$

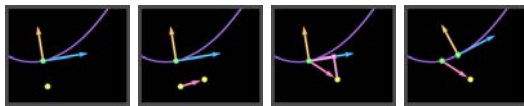
## Make Linear Approximation



- Increase speed
  - SARCOS arm on embedded system
- Good enough with high coherence



## Direct Parametric Tracking (DPT)



Initial State

Movement

Project

Compute parametric change and new surface position

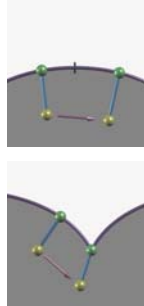
## Trimmed Models



- Trims add sharp edges in model
- Need to be able to transition from one surface to neighbor
- Need fast inside-outside test for trim loop

## Transitioning

- State change while tracing
- Three forms
  - Across surface boundary
  - Along surface intersection
  - Off of the model

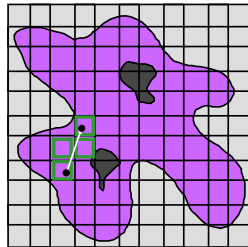


## Edge Tracing and Release

- Slide along trim in 3D to locally close point
  - Project probe onto current segment
  - If past endpoint slide
  - Continue in one direction until minimum reached
- Check for release onto surface
  - Evaluate surface and apply DPT
  - If point doesn't release try adjacent surface
  - If point still doesn't release make special normal

## Trim Intersection

- Discrete movement on surface
- Brute force approach not feasible
- Grid approach
  - Boundary on bounding box
  - Locality
  - Walk algorithm



## Grid Data

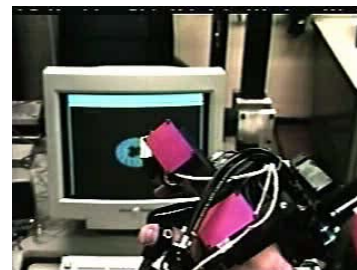
- |              |               |               |
|--------------|---------------|---------------|
| • Goblet     | • Gear        | • Crank Shaft |
| — 3 surfaces | • 22 surfaces | • 73 surfaces |
| — 254 segs   | • 1256 segs   | • 412 segs    |
| — 13 max     | • 15 max      | • 36 max      |
| — 3 mean     | • 4 mean      | • 4 mean      |
| — 89% empty  | • 92% empty   | • 89% empty   |



## Example



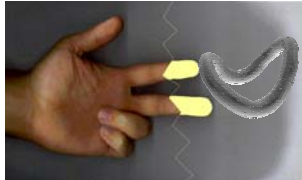
## Example2



## Model-model rendering



- Maintain penetration between two models
- Use Newton's method to initialize, then integrate to maintain

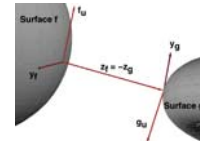


## Differential Surface Kinematics



- Using relations from differential geometry and robotics calibration, the general result for the relation between parametric contact coordinate velocity and body velocity is derived:

$$\dot{\mathbf{u}} = \mathbf{AB} \begin{bmatrix} \dot{\mathbf{q}}_f \\ \dot{\mathbf{q}}_g \end{bmatrix}$$



## Surface-to-Surface Tracing



Just integrate:

$$\dot{\mathbf{u}} = \mathbf{AB} \begin{bmatrix} \dot{\mathbf{q}}_f \\ \dot{\mathbf{q}}_g \end{bmatrix}$$

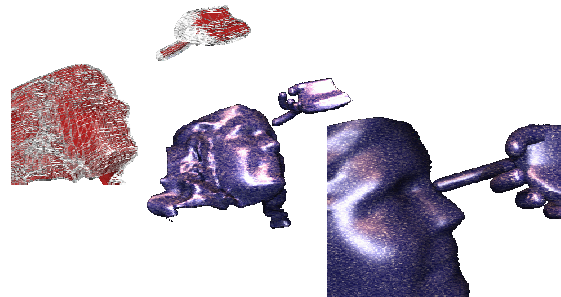
Provides closed form **local collision detection** updates.



## Surface-to-Surface Management



Three step tracking process



## Surface Contacts

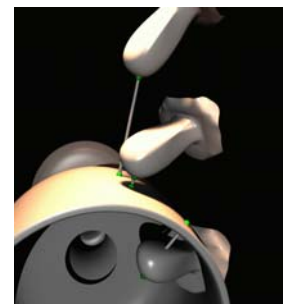


- Possible multiple contacts
  - Initializing method must track potential contacts



## Example

- Newton's method initializes closest points
- Near contact, stable integration takes over
- Maximum penetration is maintained



## Wearable Vibrotactile Displays

Hong Z. Tan  
Haptic Interface Research Laboratory  
Purdue University  
<http://www.ece.purdue.edu/~hongtan>  
[hongtan@purdue.edu](mailto:hongtan@purdue.edu)



## A Living Proof that Haptics Works!



Photograph by Hansi Durlach, 1980.

## Sensory Substitution: Text to Vibration



### Optacon



Photograph Courtesy of  
Telesensory Corp.

### Optohapt

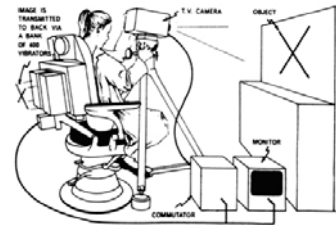


Geldard, F. A. (1966). Cutaneous coding of optical signals:  
The optohapt. *Perception & Psychophysics*, 1, 377-381.

## Sensory Substitution: Image to Vibration



### TVSS



White, B. W., Saunders, F. A., Scadden, L., Bach-Y-Rita, P., & Collins, C. C. (1970). Seeing with the skin. *Perception & Psychophysics*, 7(1), 23-27.

## Sensory Substitution: Speech to Vibration

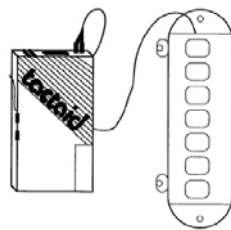


### "Felix"



Photograph by  
Alfred Eisenstaedt,  
1950.

### Tactaid VII



Weisenberger, J. M., & Percy, M. E. (1994). Use  
of the Tactaid II and Tactaid VII with children.  
*The Volta Review*, 96(5), 41-57.

## Considerations for Wearable Vibrotactile Displays

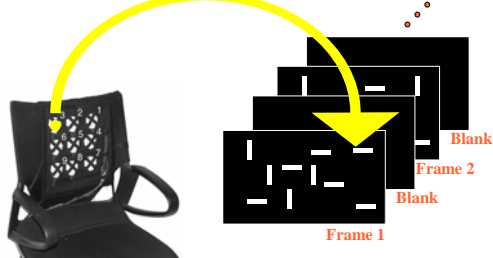


- Wearability
- Body site
- Intended User
- Training
- Sensory adaptation
- Integration with visual and auditory displays
- **What is the most effective way to communicate via a vibrotactile display?**

## Haptic Cueing of Visual Spatial Attention



Collaborators: Rob Grav, Jay Young



H. Z. Tan, R. Gray, J. J. Young, and R. Traylor, "A haptic back display for attentional and directional cueing," *Haptics-e: The Electronic Journal of Haptics Research*, 3(1), 2003.

## Applications



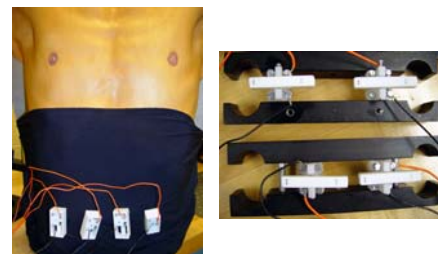
## Situation Awareness



Rupert, A. H. (2000, March/April). An instrument solution for reducing spatial disorientation mishaps - A more "natural" approach to maintaining spatial orientation. *IEEE Engineering in Medicine and Biology Magazine*, 19, 71-80.

Veen, H.-J. v., Spape, M., & Erp, J. v. (2004). Waypoint navigation on land: different ways of coding distance to the next waypoint. *Proceedings of EuroHaptics 2004*, 160-165.

## Military Silent Operation

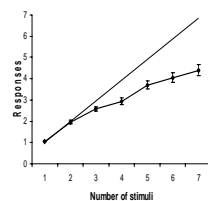


Jones, L. A., Nakamura, M., & Lockyer, B. (2004). Development of a tactile vest. *Proceedings of HAPTICS '04*, 82- 89.

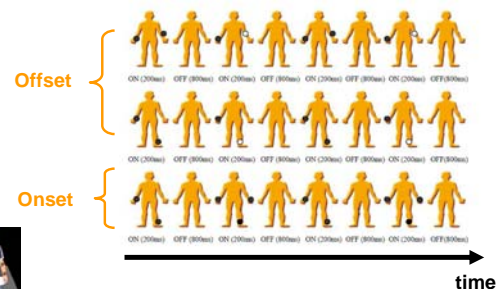
## Limitations in Tactile Information Processing



### Numerosity Judgments



## Tactile Change Detection



time

## Conclusions

---



- Haptics technology has reached a point where wearable vibrotactile displays are possible
- Many challenges remain in developing lightweight and robust actuators
- We've yet to develop a tactile vocabulary that match the sensory capability of the body surface
- More basic research is needed to explore the information-processing limitations in using wearable vibrotactile displays

## Towards Realistic Haptic Rendering of Surface Texture

Seungmoon Choi and Hong Tan  
Haptic Interface Research Laboratory  
Purdue University  
<http://www.ece.purdue.edu/~hongtan>  
[hongtan@purdue.edu](mailto:hongtan@purdue.edu)



SIGGRAPH2005

## Outline

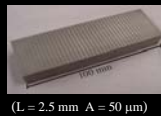
- Introduction
  - Texture
  - Perceived Instability
- Psychophysical Evaluations
- Sources of Perceived Instability
- Conclusions

## What is Texture?



SIGGRAPH2005

- What is texture?
  - Texture = micro geometry of a surface
  - Shape = macro geometry of a surface
- Why study haptic texture rendering?
  - To enrich the sensory attributes of virtual objects
  - To provide well-controlled stimuli for psychophysical studies



## Haptic Texture Rendering



SIGGRAPH2005

- How to render texture haptically?
  - Haptic texture information can be effectively conveyed through temporal cues alone (e.g., Katz' observations)
  - Point-contact probe-mediated force feedback devices can adequately render virtual textures
- Past work has focused on the developments of fast computational algorithms
  - Massie (1996) – *force magnitude*
  - Ho *et al.* (1999) – *force magnitude & direction*

## Perceived Instability



SIGGRAPH2005

- Definition: Any unrealistic sensation that can not be attributed to the physical properties of simulated textured surfaces (e.g., buzzing).
- Types of Perceived Instability
  - "Buzzing"
  - "Aliveness"
  - "Ridge Instability"
- Can be caused by both device instability and inadequate environment model

## Outline



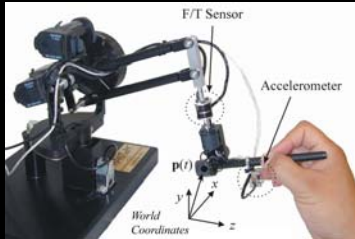
SIGGRAPH2005

- Introduction
  - About Texture
  - Perceived Instability
- Psychophysical Experiments
- Sources of Perceived Instability
- Conclusions

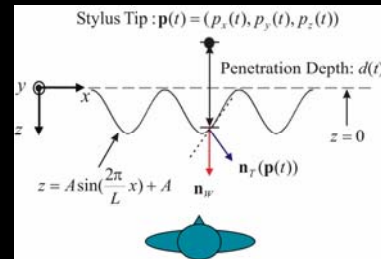
## Apparatus



- PHANToM 1.0A (SensAble Technologies, MA)



## Texture Model



- Free Exploration vs. Stroking

## Main Findings



- Largest stiffness values for “clean” texture: 0.01 to 0.78 N/mm (“soft corduroy”)
- Largest stiffness value for “clean” flat wall: 1.005 N/mm
- We need a better understanding of the types of perceived instabilities and their underlying causes
- The goal is to be able to render “harder” textures without any perceived instability

## Three Types of Perceived Instabilities



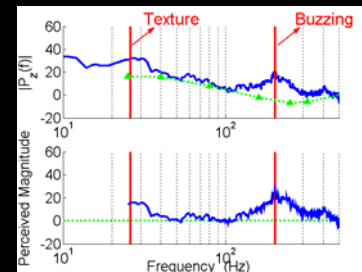
- Buzzing
  - Perceived as high-frequency noises (“vibrating” textures)
- Aliveness
  - Perceived as low-frequency force variations (“pulsating” textures) when the stylus is stationary
- Ridge Instability
  - When resting on the peak of a sinusoidal ridge, the stylus is being pushed into the valley.

## Outline



- Introduction
  - About Texture
  - Perceived Instability
- Psychophysical Experiments
- Sources of Perceived Instability
- Conclusions

## Buzzing



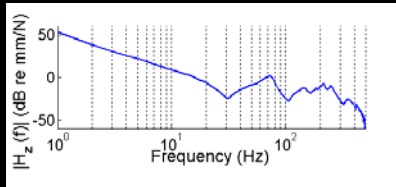
$|P_z(f)|$ : dB re 1 $\mu$ m peak, Perceived Magnitude: dB SL



## What Causes Buzzing?

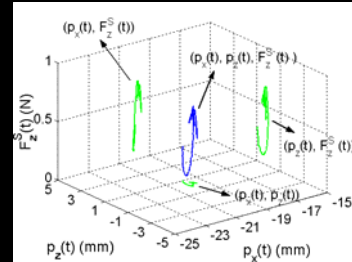


PHANToM Frequency Response



Measured at the origin of the PHANToM with the stylus pointing to the +z direction

## Aliveness



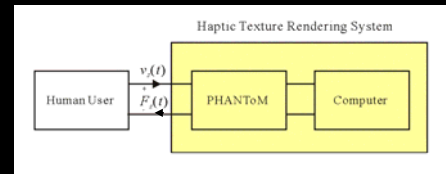
Signal duration = 400 ms

## What Causes Aliveness?



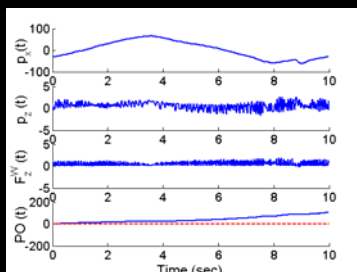
- Our hypothesis was that *Aliveness* came from the texture rendering algorithm, but not an unstable haptic interface
- To test our hypothesis, we implemented a Passivity Observer (Hannaford & Ryu, 2002) on measured force and position data

## Passivity Observer



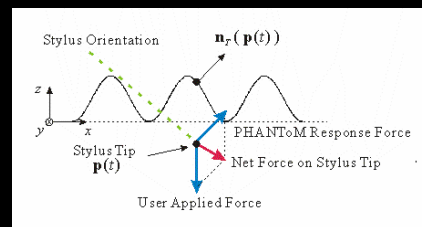
$$PO(t) = \sum_{i=1}^t F_z(i\Delta t) v_z(i\Delta t) \Delta t$$

## Passive Rendering with Aliveness



$p_x(t)$ : mm,  $p_z(t)$ : mm,  $F_z^W(t)$ : N,  $PO(t)$ : Nmm/sec

## What Causes Ridge Instability?



## Conclusions



- Many virtual textures contain perceived instability
- Stroking is the preferred method for texture perception because it has a larger stiffness threshold for perceived instability
- Perceived instability can occur even when the haptic interface system is passive
- Research based on both control and perception are necessary in order to achieve perceptually realistic texture rendering

## Haptic Rendering of Textured Surfaces

Miguel A. Otaduy

ETH-Zurich

<http://graphics.ethz.ch/~otaduy>

otaduy@inf.ethz.ch



## Outline

- Motivation
- Algorithm Overview
- Synthesis of the Force Model
- Penetration Depth on the GPU
- Experiments and Results
- Conclusion

## Introduction

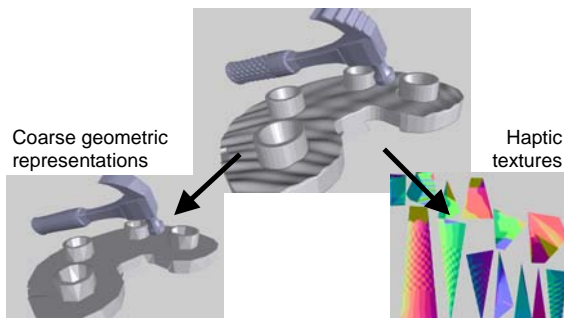


## Introduction



- Geometric surface texture:
  - Compelling cue to object identity
  - Strongly influences forces during manipulation
- Objects with rich surface texture information cannot be handled by state-of-the-art haptic rendering methods.

## Models



## Outline

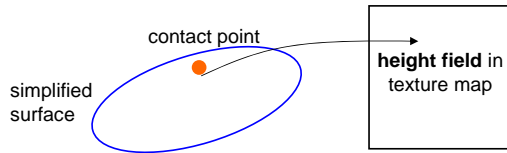


- Motivation
- Algorithm Overview
- Synthesis of the Force Model
- Penetration Depth on the GPU
- Experiments and Results
- Conclusion

### 3-DoF Texture Rendering



- 1 contact point on a textured surface
  - Minsky [1995], Ho et al. [1999]: high frequency forces based on gradient of height field



### 3-DoF Texture Rendering



- 1 contact point on a textured surface
  - Siira and Pai [1996]: stochastic model
  - Pai et al. [2001]: auto-regressive model for roughness and friction

### 6-DoF Texture Rendering



- Object-object interaction
  - Contact cannot be described as point-surface contact
  - Force and torque output has 6-DoF; point contact only has 3-DoF
- A different rendering algorithm is required

### Rendering Algorithm



- 1) Compute contact information between low-res models

### Rendering Algorithm



- 1) Compute contact information between low-res models
- 2) Refine contact information using detail geometry stored in textures

### Rendering Algorithm



- 1) Compute contact information between low-res models
- 2) Refine contact information using detail geometry stored in textures
- 3) Compute contact forces based on novel **texture force model**

## Force Model Overview



- Accounts for important factors identified by perceptual studies
- Based on the gradient of inter-object penetration depth
- GPU-based computation of directional penetration depth

## Outline



- Motivation
- Algorithm Overview
- Synthesis of the Force Model
- Penetration Depth on the GPU
- Experiments and Results
- Conclusion

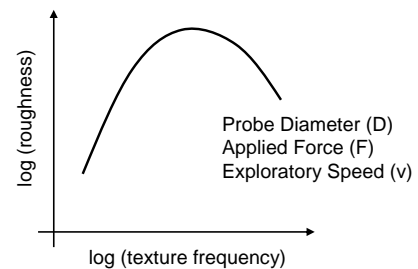
## Related Work: Perception & Psychophysics



- Studies on perception of textures through a rigid probe by Klatzky and Lederman [1999-present]
  - Analyze effects of probe diameter, applied force and exploratory speed
  - Inspiration for our force model

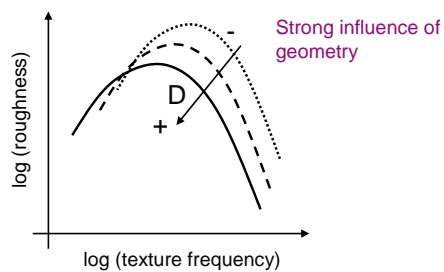
## Roughness Vs. Texture Spacing

[Klatzky and Lederman 1999-present]



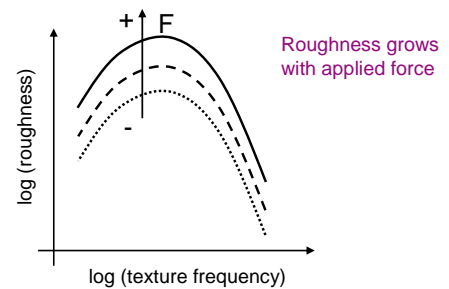
## Effect of Probe Diameter (D)

[Klatzky and Lederman 1999-present]



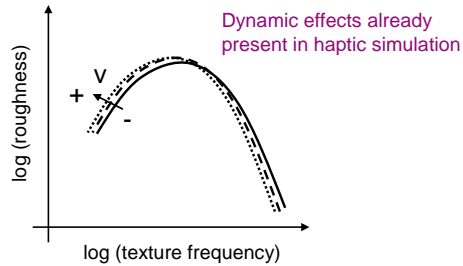
## Effect of Applied Force (F)

[Klatzky and Lederman 1999-present]



## Effect of Exploratory Speed ( $v$ )

[Klatzky and Lederman 1999-present]



## Offset Surfaces



Spherical probe  $\rightarrow$  trajectory = offset surface

Use offset surface as descriptor of vibration

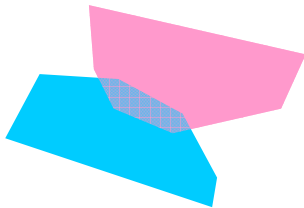
Arbitrary objects  $\rightarrow$  ???

How can we generalize offset surfaces?

## Penetration Depth: Definition



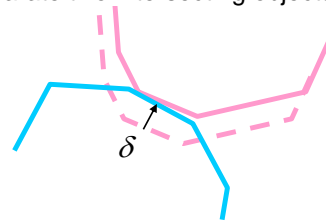
$\delta$  = Minimum translational distance to separate two intersecting objects



## Penetration Depth: Definition



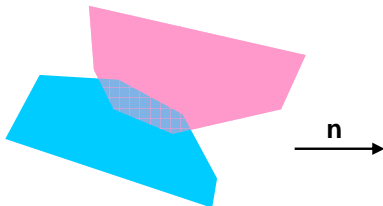
$\delta$  = Minimum translational distance to separate two intersecting objects



## Directional PD: Definition



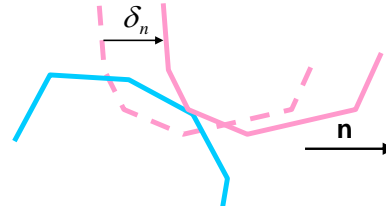
$\delta_n$  = Minimum translation along  $\mathbf{n}$  to separate two intersecting objects



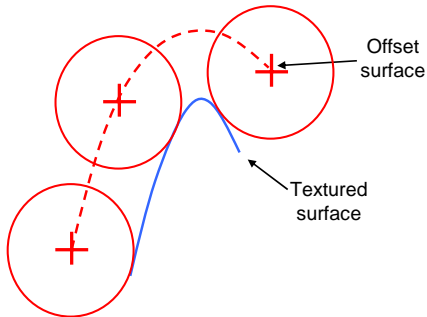
## Directional PD: Definition



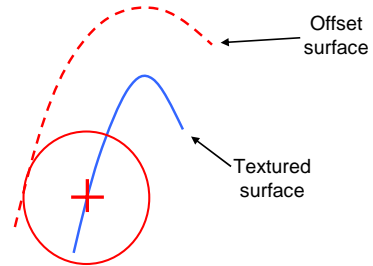
$\delta_n$  = Minimum translation along  $\mathbf{n}$  to separate two intersecting objects



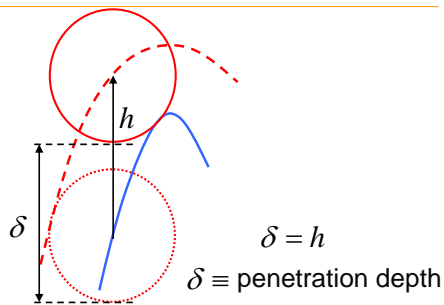
## Offset Surface and PD



## Offset Surface and PD



## Offset Surface and PD



## Force Model



- Penetration depth:
  - Applicable to arbitrary object-object interaction
  - Also used in previous single-point rendering methods

- Penalty-based potential field:

$$U = \frac{1}{2} k \delta^2$$

## Force Model



- Determine penetration direction  $\mathbf{n}$
- Force and Torque = Gradient of energy:

$$\begin{pmatrix} F_u & F_v & F_n & T_u & T_v & T_n \end{pmatrix} = -\nabla U = -k \delta_n (\nabla \delta_n) = -k \delta_n \begin{pmatrix} \frac{\partial \delta_n}{\partial u} & \frac{\partial \delta_n}{\partial v} & 1 & \frac{\partial \delta_n}{\partial \theta_u} & \frac{\partial \delta_n}{\partial \theta_v} & \frac{\partial \delta_n}{\partial \theta_n} \end{pmatrix}$$

## Effect of Geometry



- Force and torque proportional to gradient of penetration depth

$$\begin{pmatrix} F_u & F_v & F_n & T_u & T_v & T_n \end{pmatrix} = -k \delta_n (\nabla \delta_n)$$

High amplitude texture

→ High derivative of penetration depth

→ Large force/torque

Method validated by Minsky [1995]

## Effect of Applied Force



- Normal force:  $F_n = -k\delta_n$
- Other forces and torques:

$$F_u = -k\delta_n \frac{\partial \delta_n}{\partial u} = F_n \frac{\partial \delta_n}{\partial u}$$

Larger normal force  $\rightarrow$  Larger roughness effect

## Outline

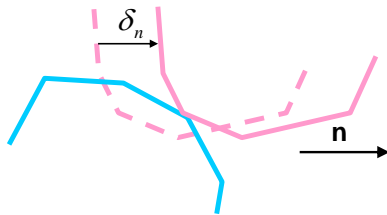


- Motivation
- Algorithm Overview
- Synthesis of the Force Model
- Penetration Depth on the GPU
- Experiments and Results
- Conclusion

## Directional PD: Definition



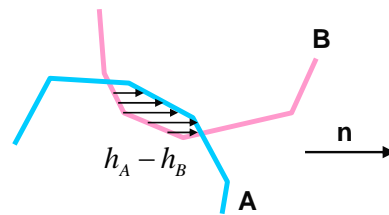
$\delta_n$  = Minimum translation along  $\mathbf{n}$  to separate two intersecting objects



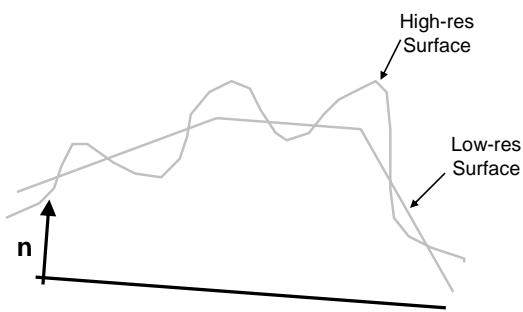
## Directional PD of Height Fields



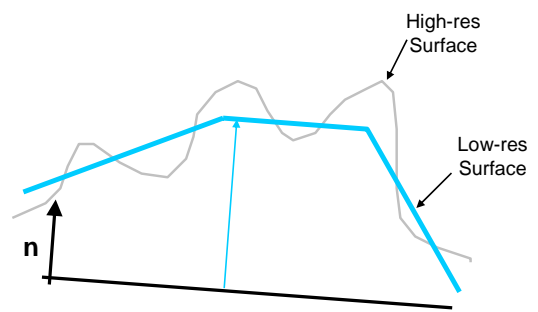
$$\delta_n = \max(h_A - h_B)$$



## PD with Texture Images

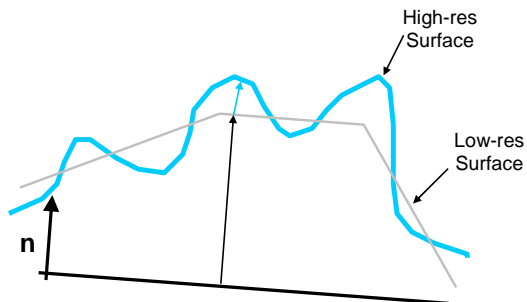


## PD with Texture Images

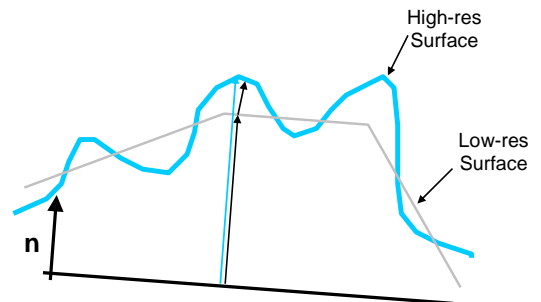




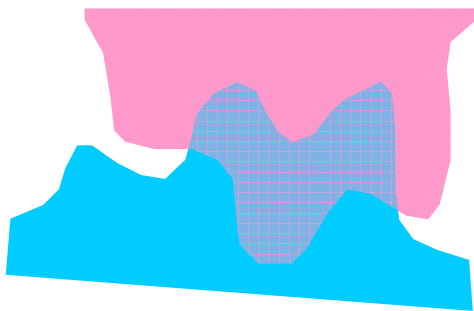
## PD with Texture Images



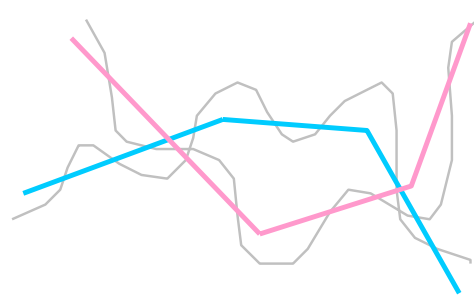
## PD with Texture Images



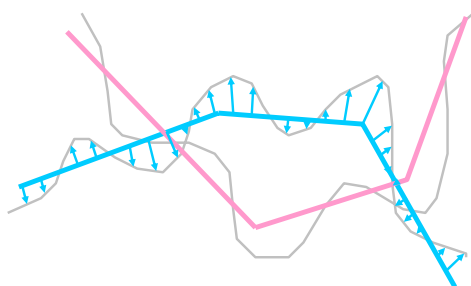
## PD Computation Algorithm



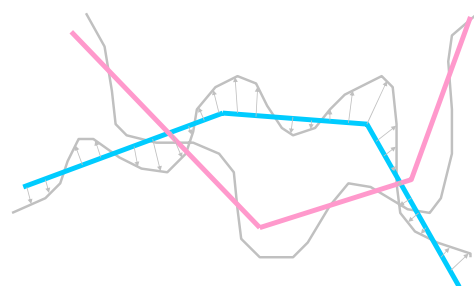
## Low-Resolution Models...



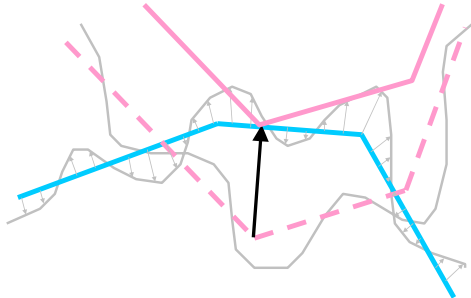
## ...+ Texture Images



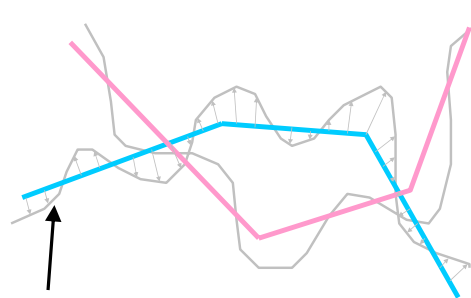
## Step 1: Approximate PD



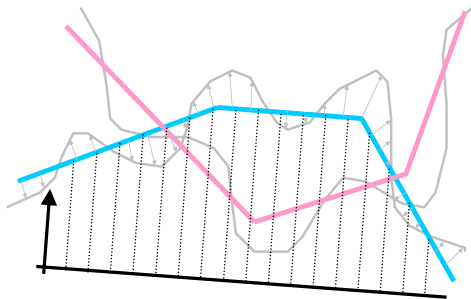
### Step 1: Approximate PD



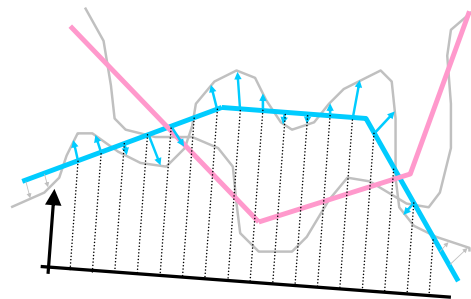
### Step 2: Refined PD



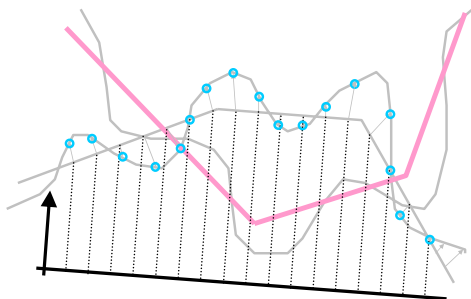
### Pass 1: Render Geometry



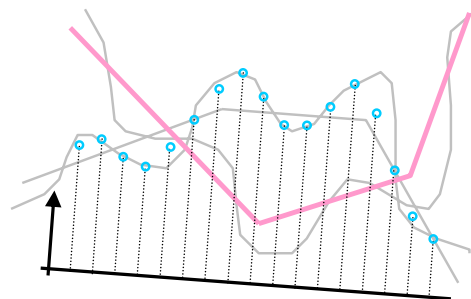
### Pass 1: Texture Mapping



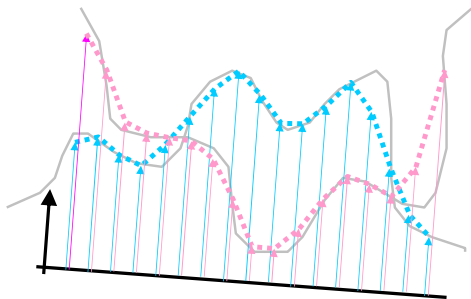
### Pass 1: Sample Surfaces



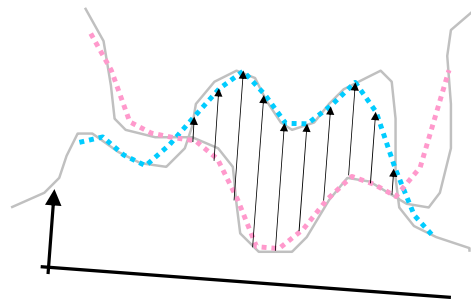
### Pass 1: Project to PD Direction



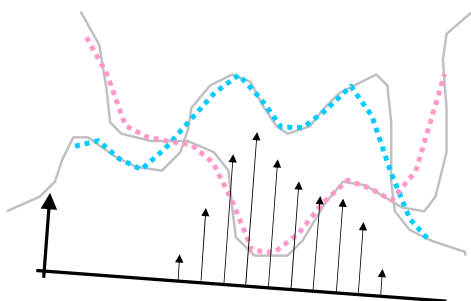
## Discrete Height Fields



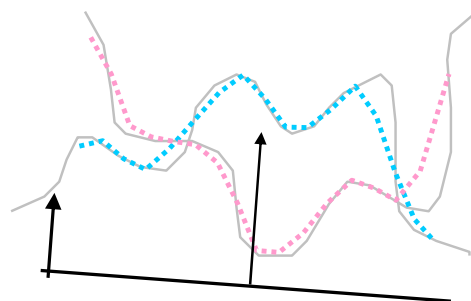
## Pass 2: Subtract Height Fields



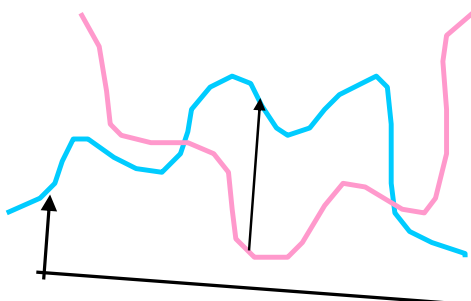
## Pass 2: Copy to Depth Buffer



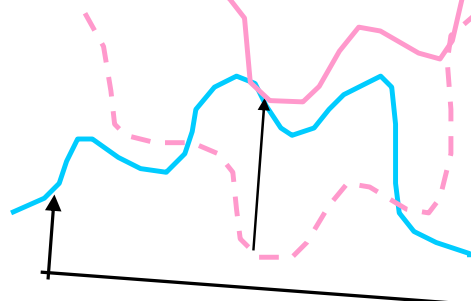
## Binary Search for Max = PD [Govindaraju et al. 2004]



## Test



## Test



## Gradient of PD



$$\begin{pmatrix} F_u & F_v & F_n & T_u & T_v & T_n \end{pmatrix} = -\nabla U = -k\delta_n (\nabla \delta_n) = -k\delta_n \begin{pmatrix} \frac{\partial \delta_n}{\partial u} & \frac{\partial \delta_n}{\partial v} & 1 & \frac{\partial \delta_n}{\partial \theta_u} & \frac{\partial \delta_n}{\partial \theta_v} & \frac{\partial \delta_n}{\partial \theta_n} \end{pmatrix}$$

## Gradient of PD



- Central differences

$$\frac{\partial \delta_n}{\partial u} \approx \frac{\delta_n(u + \Delta u, v, n, \dots) - \delta_n(u - \Delta u, v, n, \dots)}{2 \cdot \Delta u}$$

- Recompute PD at 2 new object configurations

## Outline



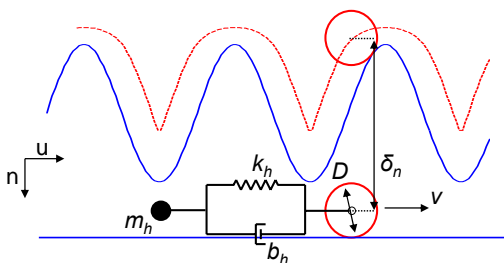
- Motivation
- Algorithm Overview
- Synthesis of the Force Model
- Penetration Depth on the GPU
- Experiments and Results
- Conclusion

## Experiments



- Offline analysis of force model
- Quality of texture effects
- Performance tests.

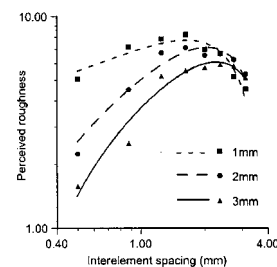
## Offline Experiments



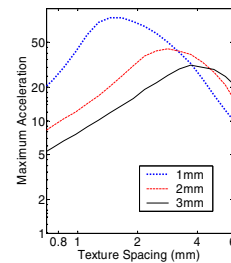
## Effect of Probe Diameter



Studies by Klatzky and Lederman



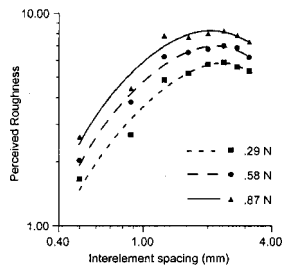
Simulation results



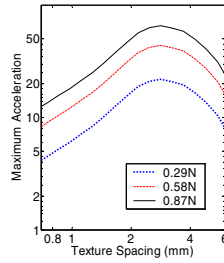
## Effect of Applied Force



Studies by Klatzky and Lederman



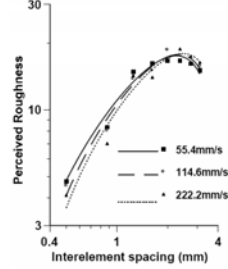
Simulation results



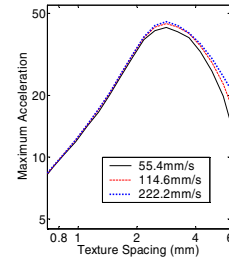
## Effect of Exploratory Speed



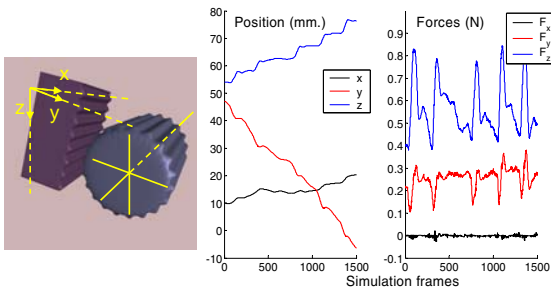
Studies by Klatzky and Lederman



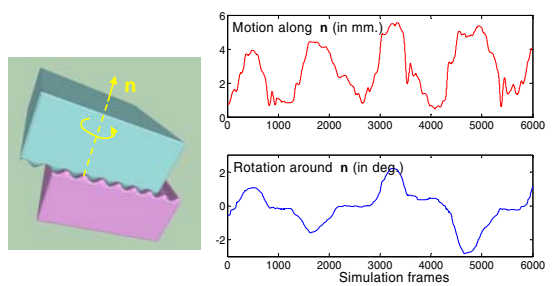
Simulation results



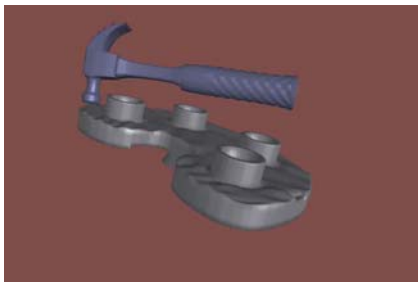
## Roughness under Translation



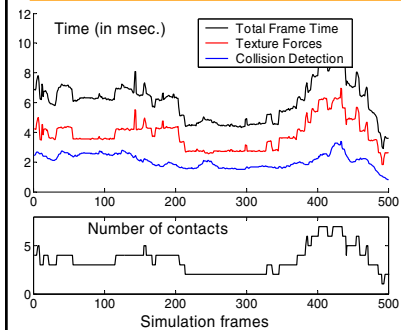
## Roughness under Rotation



## Complex Objects



## Timings: File and CAD Part



File full-res: 285Ktris  
File low-res: 632 tris  
CAD full-res: 658Ktris  
CAD low-res: 720tris  
  
Dual Pentium4 2.4GHz  
Nvidia FX5950

## Outline



- Motivation
- Algorithm Overview
- Synthesis of the Force Model
- Penetration Depth on the GPU
- Experiments and Results
- Conclusion

## Summary



- Haptic rendering algorithm using low-res models and texture images
- Force model inspired by psychophysics studies
- Image-space algorithm for PD computation (implemented on GPU)

## Results



- 500Hz force update rate with relatively simple models
- 100Hz-200Hz force update rate with complex models
- Haptic rendering of interaction between complex textured models

## Limitations



- Cannot handle surfaces that cannot be described as height fields in the contact region
- Possible sampling-related aliasing
- Limited stability with high PD gradient

## Future Work



- Higher frequency textures
- Deformable textured surfaces
- Analysis of human factors

## References



*Haptic Display of Interaction between Textured Models.*  
Miguel A. Otaduy, Nitin Jain, Avneesh Sud and Ming C. Lin.  
In Proc. of IEEE Visualization Conference 2004.

*A Perceptually-Inspired Force Model for Haptic Texture Rendering.* Miguel A. Otaduy and Ming C. Lin.  
In Proc. of the Symposium on Applied Perception in Graphics and Visualization 2004.

<http://gamma.cs.unc.edu/HTextures/>

## Modeling Deformable Objects for Haptics

Dinesh K. Pai  
(mostly in collaboration with Doug James)  
Rutgers University

## Elasticity

- Dynamics defined at infinitesimal scale
  - force -> stress
  - displacement -> strain
  - Hooke's law relates stress to strain
  - Newton -> Cauchy
- Hyperbolic partial differential equations
- PDE + boundary conditions  
= Boundary Value Problem (BVP)

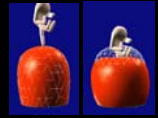
## Outline

- Basics of Elasticity
- Linear Elastostatic Models
- Green's Functions
- Capacitance Matrix Algorithms
- Haptic Interaction Issues
- Extensions

## Linear Elastostatic Models

- Simple but stable
- Captures many "global" aspects of deformation
- Sufficient to only deal with boundary discretization
- Significant precomputation
- But inaccurate for some materials, large deformation

$$G \sum_{k=1}^n \left( \frac{\partial^2 u_k}{\partial x_k^2} + \frac{1}{1-2\nu} \frac{\partial^2 u_k}{\partial x_i \partial x_i} \right) + b_i = 0$$



## Interactive Deformation



[James + Pai 99]

## Numerical Discretization

- Most problems must be solved numerically
  - Finite Differences
  - Finite Element Method (FEM)
  - Boundary Element Method (BEM)
- FEM => internal discretization; easy to handle anisotropy and inhomogeneity
- BEM => only boundary discretization; easy to handle mixed boundary conditions



## Key Idea: Green's Functions



Green's functions  $U$  relate  
 $u$ , vertex displacements to  
 $p$ , applied vertex tractions

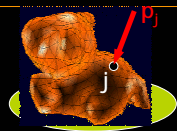
Can be computed analytically *if*  
 material distribution is known

Related work:

Cotin et al. 96 "Condensation"

Astley & Hayward 98 "Norton equivalents"

James and Pai 99 "Green's functions" via BEM



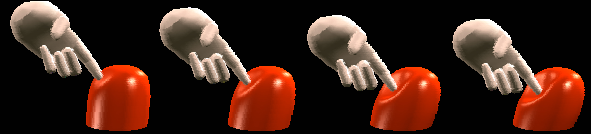
$$u = U p$$

Green's Functions

## Fast Solution to BVP with Green's Functions



- Usually few specified BV's are nonzero
- If  $s$  (out of  $n$ ) non-zero BVs,  
 $O(ns)$  time to evaluate  $v = \Xi \bar{v}$



## Example: Boundary Elements



$$N u + b = 0$$

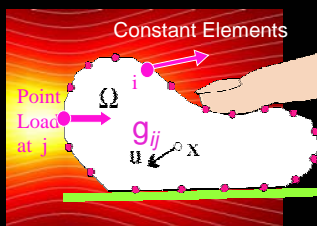
Weaken,  
 Integrate

$$c u + \int_{\Gamma} p^* u d\Gamma = \int_{\Gamma} u^* p d\Gamma$$

Discretize

$$H u = G p$$

$$G \sum_{k=1}^3 \left( \frac{\partial^2 u_l}{\partial x_k^2} + \frac{1}{1-2\nu} \frac{\partial^2 u_k}{\partial x_l \partial x_l} \right) + b_l = 0$$

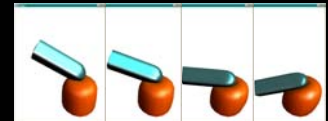


## Boundary Condition Type Changes



- Problem:
  - if BC changes, have to recompute  $\Xi$
  - $\Xi$  large and dense

- Idea: Exploit coherence



## Green's Functions for Discrete BVP (via BEM)



$$H u = G p$$

SPECIFY BC

$$H u = G p$$

REARRANGE

$$A v = -\bar{A} \bar{v}$$

INVERT LHS

$$v = -A^{-1} \bar{A} \bar{v} = \Xi \bar{v}$$

$$H u = G p$$

Red BV specified  
 Yellow BV unknown

$$A v = -\bar{A} \bar{v}$$

$$v = -A^{-1} \bar{A} \bar{v} = \Xi \bar{v}$$

Green's Functions

## Fast Elastostatic Deformation



- BC change swaps a block column of  $A$

$$H u = G p$$

$$A_s v = -\bar{A}_s \bar{v}$$

$$A_s = A_0 + \delta A_s E^T$$



## Notation



- E is  $n \times s$  submatrix of Identity  
M E "Extracts" columns from matrix M  
No cost
- Change in matrix  $A_s = A_0 + (\bar{A}_0 - A_0)EE^T$
- Changed columns  $\delta A_s = (\bar{A}_0 - A_0)E$   
 $A_s = A_0 + \delta A_s E^T = \text{[Diagram: A_0 matrix with a column added from delta A_s]}$



- video

ArtDefo  
Accurate Real Time  
Deformable Objects

Doug L. James  
Dinesh K. Pai  
Univ. of British Columbia  
Vancouver, Canada

[James, Pai SIGGRAPH 99]

## Sherman-Morrison-Woodbury



- Idea: Exploit coherence between BVPs
- If  $A_s = A_0 + \delta A_s E^T = \text{[Diagram: A_0 matrix with a column added from delta A_s]}$   
 $A_s^{-1} = A_0^{-1} - A_0^{-1} \delta A_s [I + E^T A_0^{-1} \delta A_s]^{-1} E^T A_0^{-1}$   
 $\text{[Diagram: Sherman-Morrison-Woodbury formula matrix structure]}$   
s-by-s capacitance matrix (small)

## Haptic Interaction [James & Pai 01]



- Need to update contact force at much higher rate (1 KHz)
  - Idea: use faster local model at contact point
  - Related work: [Astley & Hayward 98], [Balanuik 00], [Cavsolglu & Tendick 00]
- Need to support "point contact" abstraction (e.g., for PHANTom)

## Capacitance Matrix

### Algorithm [James + Pai 99,01]



- Exploit coherence using SMW formula
- When  $s$  nodes switch, extract and factor  
 $C = -E^T \Xi E = \text{s-by-s capacitance matrix}$   
E "extracts" the  $s$  columns  
Henceforth,  
 $v^{(0)} = [\Xi(I - EE^T) - EE^T] v^-$   
 $v = v^{(0)} + (I + \Xi)EC^{-1}E^T v^{(0)}$   
O( $s^3$ ) or better  
O( $n s$ )

## Local Models from the Capacitance Matrix Algorithm

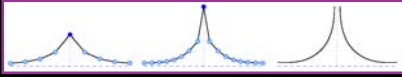


- Region of Interest
  - If output is needed only at  $d$  nodes (with extraction matrix  $E_D$ ),  
 $E_D^T v = E_D^T v^{(0)} + (E_D^T(I + \Xi)E)C^{-1}E^T v^{(0)}$   
in  $O(d s + s^2)$  time
- If these are the same  $s$  contacting nodes, simplifies dramatically!  
 $E^T v = -C^{-1}E^T v^-$   
 $\Rightarrow$  an **exact** "local buffer" for haptics

## Point Contact abstraction



- Real point contact produces infinite tractions (and therefore larger displacements on finer mesh)



- Use vertex masks to distribute displacement

[James & Pai 01]



## DyRT [James & Pai 02]



### Dynamic

physically-based modal deformation

### Response

to bone-based animation

### Textures

precomputed, sampled, and rendered almost entirely on graphics hardware



## Capacitance Matrix in Haptics



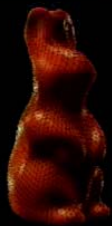
### Haptic Interaction with Linear Elastic Models

Doug L. James  
Dinesh K. Pai  
Univ. British Columbia  
April 2000

## DyRT movie



## Multiresolution Green's Functions



[James and Pai 02]

## Surgical Simulation



[James & Pai 02]

## Resources

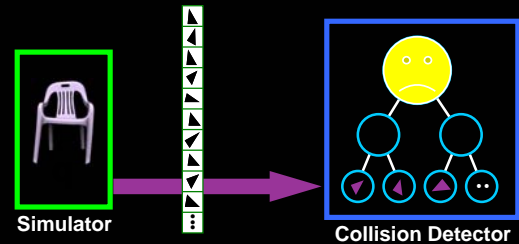


- Gibson and Mirtich "A Survey of Deformable Models in Computer Graphics" MERL Tech Report TR-97-19, 1997  
— [www.merl.com](http://www.merl.com)
- James & Pai SIGGRAPH 99 contains a review of elastic models  
— [www.cs.ubc.ca/~pai/papers/JamPai99.pdf](http://www.cs.ubc.ca/~pai/papers/JamPai99.pdf)
- James & Pai "A Unified Treatment of Elastostatic and Rigid Contact Simulation for Real Time Haptics", to appear (2001) in Haptics-e The Electronic Journal of Haptics Research  
— [www.haptics-e.org](http://www.haptics-e.org)

## Exploiting reduced coordinates for collision detection



*Traditional collision detectors don't care about reduced coordinates*



## Extensions

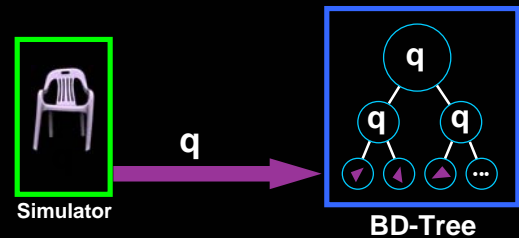


## BD-Tree [James&Pai SIGGRAPH 04]



Bounded Deformation tree

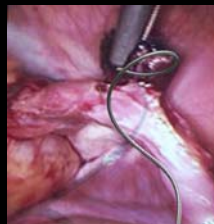
Idea: Parameterize bounding volumes by reduced coordinates



## Thin strands



- E.g., surgical sutures ("threads"), hair, etc.
- Cosserat model may be more efficient [Pai, Eurographics 02]
  - + reduces to a 1D problem
  - + handles geometric non-linearities
  - + fast (similar to rigid body chain)
  - useful for thin objects only



## Fast Sphere Update



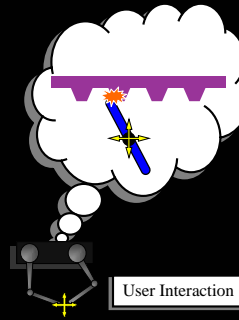
$$\begin{aligned} c' &= c + \bar{U}q \\ R' &= R + \Delta R \cdot |q| \end{aligned}$$

- Storage cost: 4 floats / mode
  - Amortized cost: 8 flops / mode
  - Spheres can be updated *in any order* and *without access to geometry*
- Output-sensitive collision detection



## Integrating Audio Haptic Displays

[DiFilippo and Pai UIST 00]



- Contacts are rendered atomically, with haptics *and* sound
- Same contact forces drive both haptics and sound
- Haptic force and sound synchronized to  $< 1\text{ms}$
- Using
  - Sound synthesis algorithm of [van den Doel & Pai]
  - Pantograph haptic device of [Hayward & Ramstein]
  - Custom DSP controller

## Reality-based Modeling for Haptics and Multimodal Displays

Dinesh K. Pai  
Rutgers University

## The AHI Audio-Haptic Interface



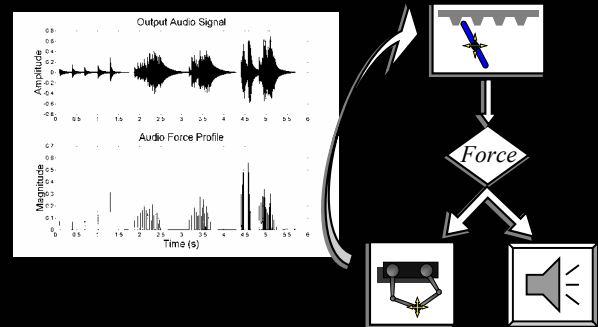
DiFilippo and Pai UIST 00

## Multisensory integration can improve haptic perception



e.g., [Doel, Kry and Pai, SIGGRAPH 01]

## Results



## Contact Interaction with Integrated Audio and Haptics

Derek DiFilippo  
Dinesh K. Pai

University of British Columbia

## Our Focus: Scanning Contact Interaction Behavior



## How to get models for contact simulation?

- Haptic interaction and, more generally, contact requires multisensory models
- Need parametrized models that support low-latency multisensory interaction
- Need to measure geometry, elastic deformation, sound response, friction, roughness, etc.

## Overview

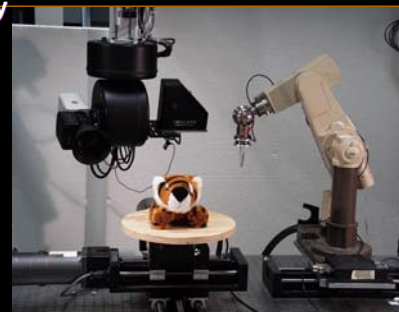
- ACME, a robotic facility for scanning contact behavior
- Scanning
  - contact deformation models
  - contact texture models
  - contact sound models



## Reality-based modeling of Contact

[Pai, van den Doel, James,  
Lang, Lloyd, Richmond, Yau  
SIGGRAPH 2001]

## ACME The UBC Active Measurement Facility



[Pai, Lang, Lloyd, Woodham '99]

## Preview: Scanning Contact Friction



Video

## Model Structure: Green's Functions



Linear Elastostatic Model

+Reference boundary conditions

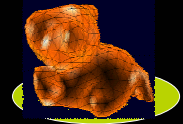
Green's functions  $U$  relate

$u$ , vertex displacements to

$p$ , applied vertex tractions

Can be computed analytically *if*  
material distribution is known

e.g., [JamesPai99, Cotin et al 96,...]

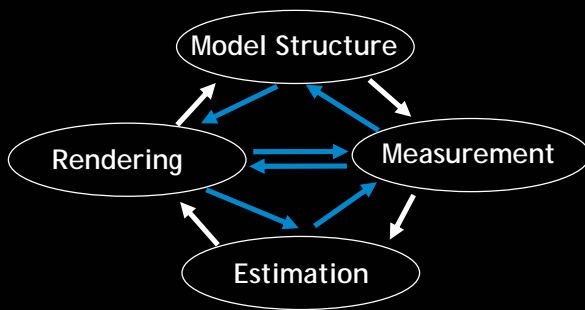


$$u = U$$

$$p =$$

Green's Functions

## A Framework for Scanning Reality-based Models



## Model Structure: Green's Functions



Linear Elastostatic Model

+Reference boundary conditions

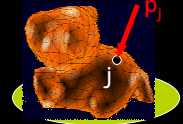
Green's functions  $U$  relate

$u$ , vertex displacements to

$p$ , applied vertex tractions

Can be computed analytically *if*  
material distribution is known

e.g., [JamesPai99, Cotin et al 96,...]



$$u = U$$

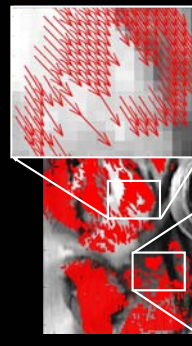
$$p =$$

Green's Functions

## Scanning Deformation Behavior



## Measurement



Robot arm measures  
contact force and local  
displacement

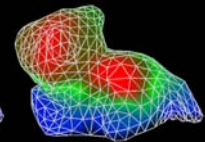
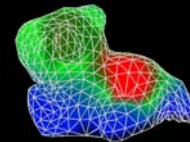
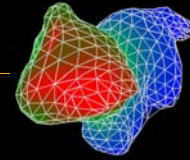
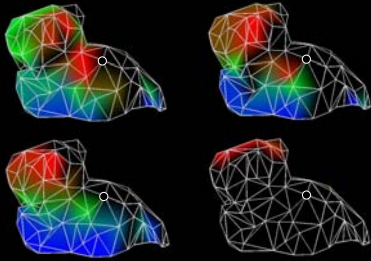
Global displacement  
measured with stereo  
vision and range flow  
Details in [LangPai01]



## Parameter Estimation



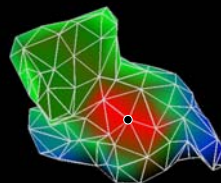
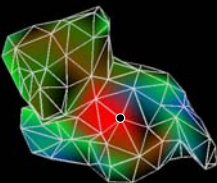
- Excite vertex  $j$  with several  $p_j^k$   
estimate vertex displacement  $u_j^k$  from range flow



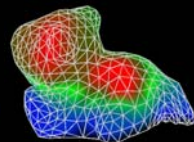
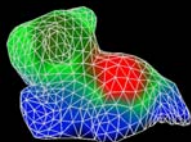
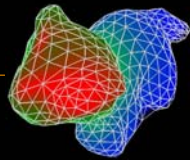
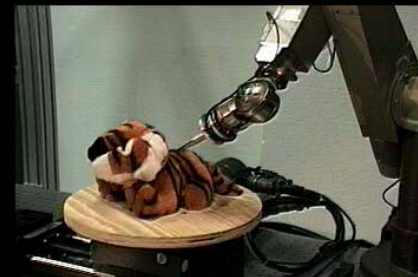
## Parameter Estimation



- Estimate  $U_j$  robustly using TSVD, TLS
- Interpolate missing observations



## Results



## Interactive Rendering Demo



Rendered using capacitance matrix algorithm  
with haptic force computation at 1KHz  
[JamesPai99, JamesPai01]



## Scanning Contact Texture



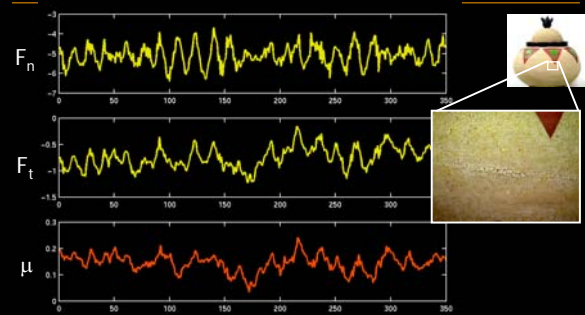
## Roughness

- Traditionally  $\approx$  small scale variation in surface geometry
- Our model: small scale variation in friction
  - Equivalent to traditional model, for frictional contact
  - Unifies friction and roughness haptic rendering

## What is Contact Texture?

- Physical parameters relevant to human perception of contact [Lederman Klatzky ...]
- Texture = friction + roughness + ...
- Model should support fast simulation for haptics (1 KHz) and audio (44.1 KHz)

## Example: Clay Pot

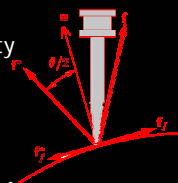


## Friction

- Model: **Coulomb Friction**  $f_t = -\mu |f_n| u$

### Measurement:

- Easy for small sample
- Hard for general object: uncertainty in surface normal, adaptation
- We use differential measurement technique for robust estimation
- Estimate: **normal and friction together**



## Roughness

- Model structure: **AR(p) autoregressive model**

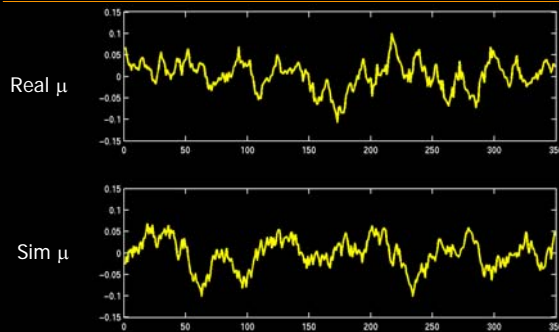
$$\mu_e(x) = \mu + \tilde{\mu}(x)$$

$$\tilde{\mu}_k = \sum_{i=1}^p a_i \tilde{\mu}_{k-i} + \sigma \varepsilon_k$$

- Captures randomness plus periodicities
- Small  $p$  sufficient for many surfaces [Thomas 82, Perlin]
- Estimate parameters: **using covariance method**
- Rendering: **Discrete convolution**
- **Extremely fast and simple**



## Real vs. Simulated Clay Pot



## Contact Sounds



- Provide cues of contact shape, location, force, and object material
- “Foley sounds” in radio and cinema
- Can be integrated with
  - Simulation and Interaction [O'BrienCookEssl, DoelKryPai, Friday]
  - Room acoustics [e.g., Tsingos et al., Friday]

## Scanning the Clay Pot

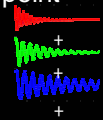


Contact Texture  
Modeling

## Model Structure



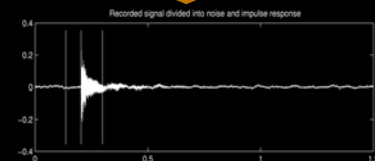
- Modal Synthesis [e.g., Doel&Pai 96-01, Cook 96]
- Models impulse response at surface point
 
$$p(x, t) = \sum_{i=1}^M a_i(x) e^{-d_i t} \sin(2\pi f_i t)$$
- - $f_i$  is frequency of a vibration mode
  - $d_i$  is frequency-dependent damping
  - $a_i(x)$  location dependent amplitude “texture”
- Continuous contact sounds generated by convolution with contact force



## Scanning Contact Sounds



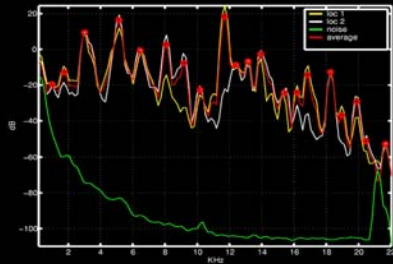
## Measure Impulse Response



## Estimate Parameters



First estimate modal frequencies  $f_i$



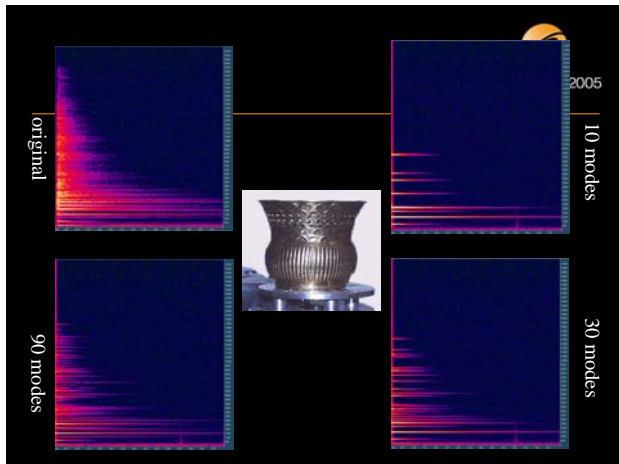
Refine frequencies + estimate  $a_{ik}$ ,  $d_i$

[SteiglitzMcBride65, BrownPuckette93]

## Scanning the Clay Pot



Video



## Conclusions



- It is now possible to scan multi-modal models of contact interaction behavior
- Scannable behavior includes
  - deformation (visual and haptic)
  - friction and roughness texture
  - sound response
- Can be automated with ACME, a robotic measurement facility

## Rendering



- Generate contact force at audio rates
  - depends on contact texture, nominal force, and velocity  
[see DoelKryPai paper on Friday]
- Convolve with audio impulse response
  - efficient using modal resonator filter bank (4 flops/mode/sample)
  - smoothly interpolate audio parameters  $a_i(x)$  from mesh vertices



## Applications in Scientific Visualization

Russell Taylor  
UNC-Chapel Hill  
<http://www.cs.unc.edu/~taylorr>  
[taylorr@cs.unc.edu](mailto:taylorr@cs.unc.edu)



## Outline of Taylor talk



- Eight scientific haptic applications
- Three haptic applications in more depth
  - Listing benefits found from haptics in each
  - Showing mechanism for coupling haptics thread
- (Brief) Haptics for multivariate display



## Molecular Dynamics Application



- VMD: Visual Molecular Dynamics

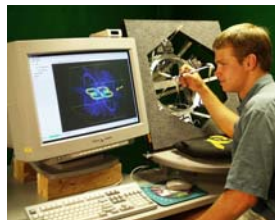


Humphrey, 1996

## Haptic Vector Field Application



- Lawrence, Lee, Pau, Roman, Novoselov
  - University of Colorado at Boulder
- 5 D.O.F. in
- 5 D.O.F. out

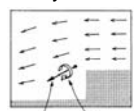
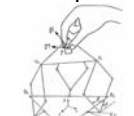


Lawrence, 2000

## Volume Flow Visualization User Study



- Iwata and Noma combined HMD and haptics to display volume data in 1993
  - Force based on density gradient
  - or Torque based on density
  - Either method improved positional accuracy
- Describe presenting flow field
  - Force for flow velocity
  - Torque for vorticity



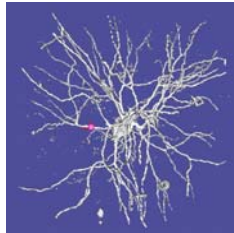
Iwata, 1993

## Medical Volume Exploration Application



- Avila and Sobierajski 1996

- Feeling neuron dendrites
  - Repulsion was too hard to follow for twisting dendrites
- Gentle attraction used
  - Opposite of gradient



Avila, 1996

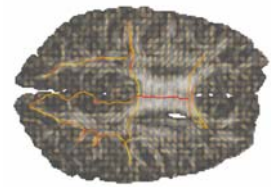
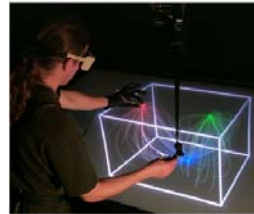
## Vector and Tensor Field Applications



- The Visual Haptic Workbench

Constrain to streamline

Anisotropic drag



Ikits, 2005

## When is Force Useful? UNC Experiences

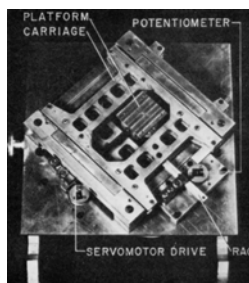


- Teaching physics potential fields
- Drug/protein docking
- nanoManipulator

## Force-Field Simulation for Training



- 2D sliding carriage device
- Presented magnetic, electric, and gravitational fields
- Motivated students learned more
  - Less motivated students didn't
- Dispelled misconceptions
  - Electric field in diode is not greater near the plate than near the cathode
  - Gravity field in 3-body system does not always point towards one of the bodies



Brooks, 1990

## Drug/Protein Docking



- Ming Ouh-Young
  - 6-DOF positioning task
  - “Lock and Key” problem
  - Hard surface + electrostatic
  - Got a better “feel” for docking
  - User study: NTE factor-of-2 speedup with haptics



Brooks, 1990

## nanoManipulator



A virtual environment  
interface to SPM

### The Goal:

- Remove boundaries between user and sample
- Can we make experiments on the molecular scale as easy as rolling a pencil or pushing a golf ball?



## nanoManipulator: Haptics

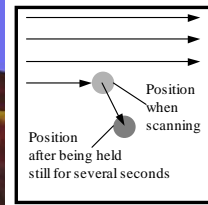
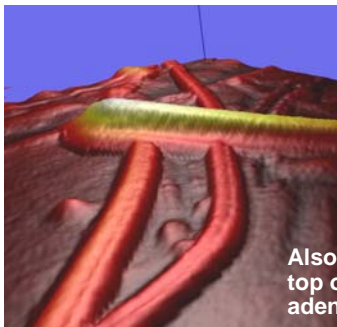


"It was really a remarkable feeling for a chemist to  
be running his hand over atoms on a surface"

— R. Stanley Williams, UCLA Chemistry

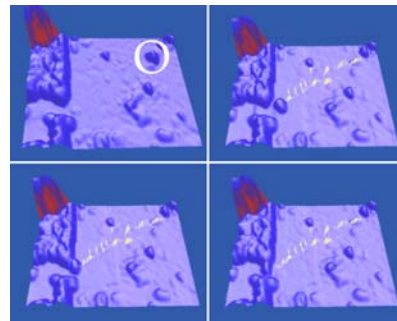
- Exciting and engaging, but what is it useful for?

## Finding the right spot



Also, finding the  
top of an  
adenovirus

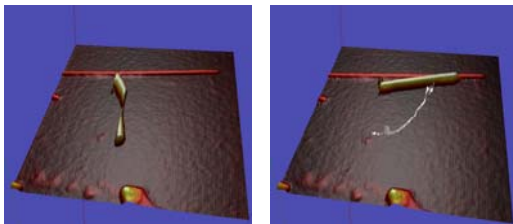
## Finding the right path



## Light touch (haptic imaging)



Observation modifies the system

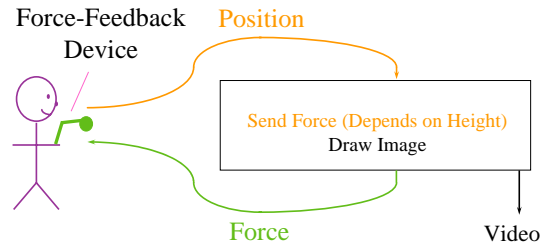


## Case Study of Multi-Threading

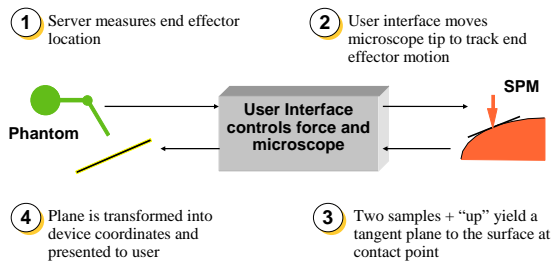


- Problem: Haptics must update at >500 Hz
  - Required for stable hard surfaces
  - Must be uninterrupted to prevent force discontinuities
- Solution: As Ken & Ming described
  - Separate force-feedback server
  - Pass **Intermediate representation** between threads

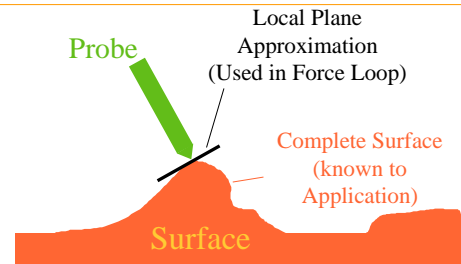
## nManipulator: The Old Way



## Decoupling surface update using intermediate rep.



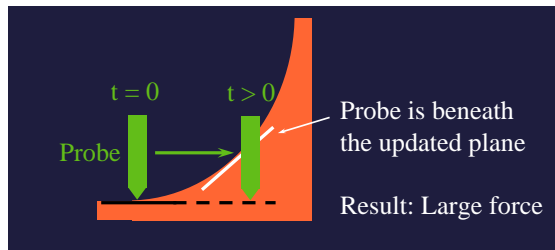
## Local Plane Equation



## Preventing Discontinuity



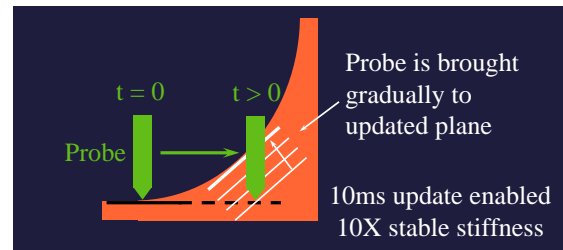
- Discontinuity when plane updated



## Preventing Discontinuity

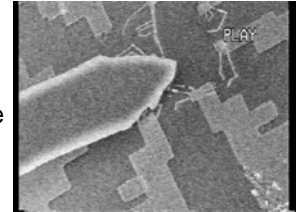


- Recovery time over several steps

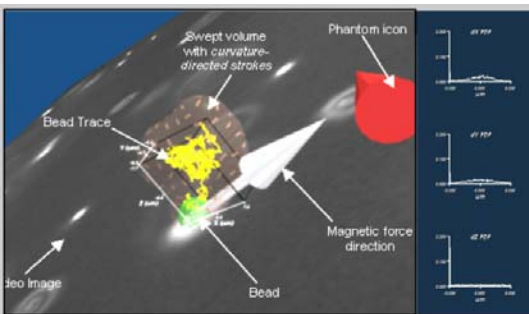


## SEM+AFM: See while touch

- SEM view of AFM tip
- Hand-controlled AFM
- Zooms in on nanotube
- “Twangs” nanotube



## 3D Magnetic Force Microscope



## Sandpaper: Haptic Textures

- Margaret Minsky dissertation (MIT, UNC)
- Displayed 3D surfaces on 2D haptic device by mapping slope to lateral force
  - People perceived 3D shape
- Displayed several properties
  - Viscosity
  - Spatial Frequency
- Vary properties based on data

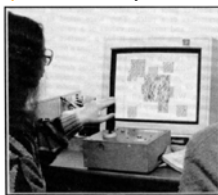


Figure 1 Photo of "Sandpaper" System In Use

Minsky, 2000

## Augmenting “Basic Haptics”

- Jason Fritz, University of Delaware
  - Adding haptic “grid planes” provide scale
  - Friction and texture
    - Produce more realistic-feeling surfaces
    - Can distinguish features in data sets
  - Produced stochastic texture model
    - Creates information-rich haptic textures

Fritz, 1996



## Haptics for Multi-Dimensional Display



- Modulate surface properties
  - friction, stiffness, bumpiness, vibration, adhesion
- User studies done by Mark Hollins at UNC to determine the perceptually-linear mapping for each
  - Map directly for the relevant physical parameter
  - Linearize for presentation of non-physical quantities
- Ongoing exploration of cross-talk between channels

Seeger, 2000

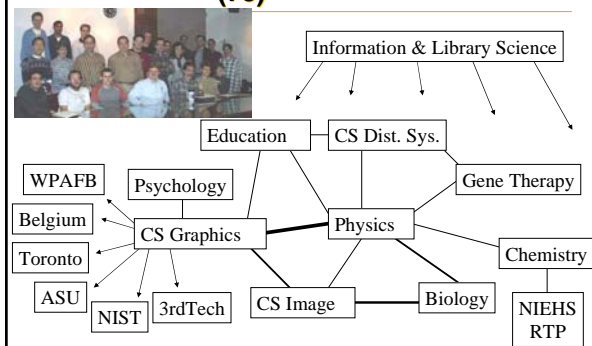


## Summary



- Useful for
  - Direct perception of force fields
  - Sensing while manipulating simulation or tele-operation
  - Tracing without occlusion in 3D
- Match application requirements
  - 2DOF in/out for force fields
  - 6DOF in/out for Docker
  - 6DOF in, 3DOF out for nM
- Separate force, graphics and control loops

## nanoManipulator Co-builders (78)



## Support provided by



- 3DMFM
    - NIH National Institute for Biomedical Imaging and Bioengineering
- University of North Carolina at Chapel Hill nanoManipulator System
- 
- NIH National Center for Research Resources
  - National Science Foundation
  - Army Research Office
  - Office of Naval Research

## References



- Avila, R. S. and L. M. Sobierajski (1996). "A Haptic Interaction Method for Volume Visualization," Proceedings of IEEE Visualization '96, San Francisco.
- Brooks, F. P., Jr., M. Ouh-Young, et al., "Project GROPE - Haptic displays for scientific visualization," Computer Graphics: Proceedings of SIGGRAPH '90, Dallas, Texas. 177-185. 1990.
- Fritz, J. and K. Barner (1996). Stochastic models for haptic texture. Proceedings of the SPIE International Symposium on Intelligent Systems and Advanced Manufacturing - Telemanipulator and Telepresence Technologies III, Boston, MA.
- William Humphrey, Andrew Dalke, and Klaus Schulten, VMD - Visual Molecular Dynamics," Journal of Molecular Graphics, 14:33-38, 1996.
- Milan Ikits and J. Dean Brederon, "The Visual Haptic Workbench," Visualization Handbook, eds. Charles Hansen and Christopher Johnson, Elsevier, 2005



## References



- Iwata, H.; Noma, "Volume haptization", 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in Virtual Reality, pp. 16 -23, 25-26 Oct. 1993.
- D. A. Lawrence, C. D. Lee, L. Y. Pao, and R. Y. Novoselov. "Shock and Vortex Visualization Using a Combined Visual/Haptic Interface," Proc. IEEE Conference on Visualization and Computer Graphics, Salt Lake City, UT, Oct. 2000.
- Minsky, M., M. Ouh-young, et al., "Feeling and Seeing: Issues in Force Display," Proceedings ACM Siggraph Symposium on Interactive 3D Graphics. 2000.
- Seeger, A., A. Henderson, et al., "Haptic Display of Multiple Scalar Fields on a Surface," Workshop on New Paradigms in Information Visualization and Manipulation, ACM Press. 2000.

## Haptic Interaction with Fluid Media

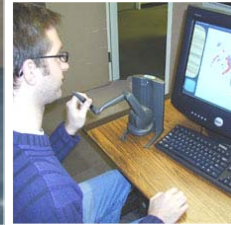
William Baxter and Ming C. Lin  
Department of Computer Science  
University of North Carolina at Chapel Hill  
<http://www.cs.unc.edu/~lin>  
{baxter,lin}@cs.unc.edu



## Goal



- Enable users to *FEEL* simulated fluid



SensAble's Phantom

## Goal



## Goal



## Haptics Overview

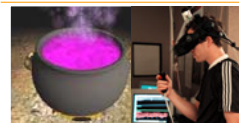


- Programmatic force feedback I/O
- Input
  - 6DOF – Position & Rotation
- Output
  - 3DOF – Forces
  - 6DOF – Forces & Torques
- Need kHz updates
  - For smooth output
- Allow users to feel and touch

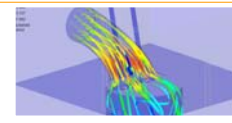


SensAble's Phantom

## Motivation



Virtual Reality



Scientific Visualization



Simulation / Training



Interactive Applications

## Motivation: Virtual Reality



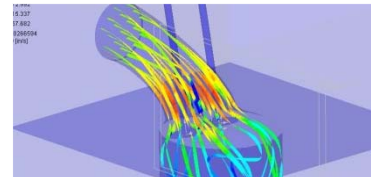
- Haptics add to the illusion of presence in VR [Insko 2001]



## Motivation: Scientific Visualization



- Immersive visualization
- Computational steering



## Motivation: Simulation/Training



- Interactive flight simulation
- Haptic cues for reducing Spatial Disorientation



## Motivation



- Entertainment
- Education
- Fluid-based painting



## Motivation: Interactive Painting



- dAb [SIGGRAPH '01]
- IMPaSTo [NPAR '04]
- Stokes Paint [CASA '04]
- Sim Brush [PG '04]



## Main Results



- Physically based force computation model for haptic interaction with fluid media
  - Based on stress tensor formulation
  - Low computational cost
- Haptic filtering technique
  - Based on FIR filter design

## Organization



- Previous work
- Overview
- Haptic Force Computation
- Haptic Filtering
- Results

## Organization



- Previous work
- Overview
- Haptic Force Computation
- Haptic Filtering
- Results

## Previous Work



- Fluid-Structure Interaction Offline Simulation
  - Foster & Metaxas '96
  - Ristow '99,'00
  - Carlson *et al.* '04

## Previous Work



- Haptic Rendering
  - Point interaction with rigid bodies  
Hollerbach *et al.* '97, Nahvi *et al.* '98
  - Point interaction with deformable bodies  
Ayache & Delingette '03
  - 6DOF interaction with rigid bodies  
Gregory '00
  - Haptic texture rendering  
Siira & Pai '96, Otaduy '04

## Previous Work



- Haptics of Vector Fields
  - Helman & Hesselink '90,'91
  - Durbeck *et al.* '98
  - Lawrence *et al.* '00
  - Mascarenhas *et al.* '01

## Organization



- Previous work
- Overview
- Haptic Force Computation
- Haptic Filtering
- Results

## Navier-Stokes Equations

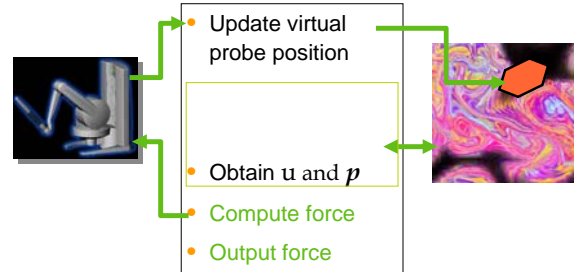


$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} + \mathbf{F}$$

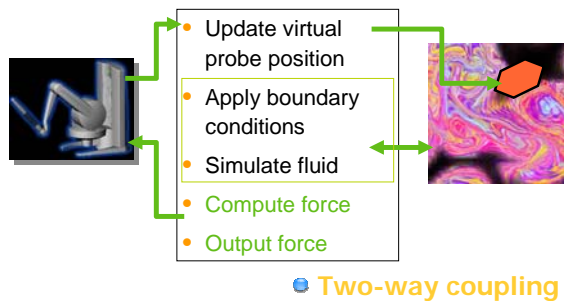
$$\nabla \cdot \mathbf{u} = 0$$

- Output:  
u velocity and  
p pressure

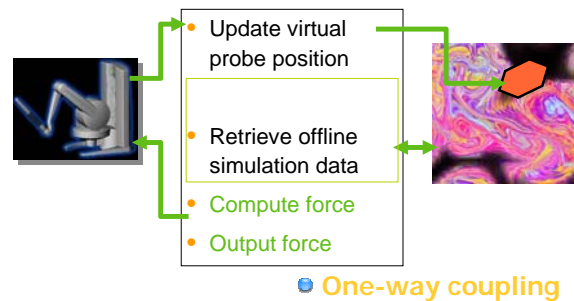
## Overview



## Overview – Online Simulation



## Overview – Offline Simulation



## Organization



- Previous work
- Overview
- Haptic Force Computation
- Haptic Filtering
- Results

## Common Force Approximations

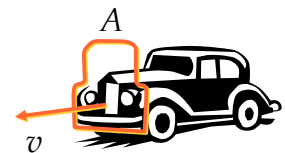


- Aerodynamic drag

$$F_{AD} = C_{AD} \rho v^2 A$$

- Viscous Drag

$$F_{VD} = C_{VD} \mu v A$$



## Fluid Force Computation

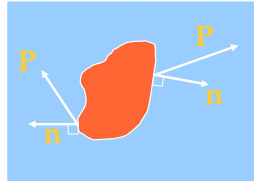


- Fluid stress tensor

$$\sigma_{ik} = -\underset{\text{pressure}}{p}\delta_{ik} + \underset{\text{viscosity}}{\mu} \left( \frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} \right)$$

- Force/Area

$$\mathbf{P} = \boldsymbol{\sigma}(\mathbf{x}) \cdot \mathbf{n}$$



## Fluid Force Computation



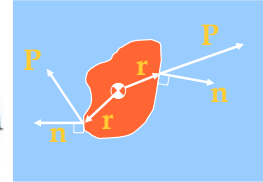
- Integrate over surface

— Force

$$\mathbf{F}_{\text{obj}} = \int_S \boldsymbol{\sigma} \cdot \mathbf{n} dA$$

— Torque

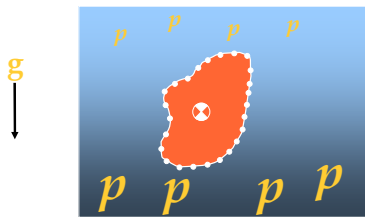
$$\boldsymbol{\tau}_{\text{obj}} = \int_S \mathbf{r} \times \boldsymbol{\sigma} \cdot \mathbf{n} dA$$



## Buoyancy?



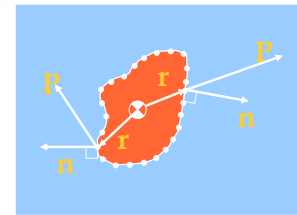
- Incorporated automatically



## Discretization



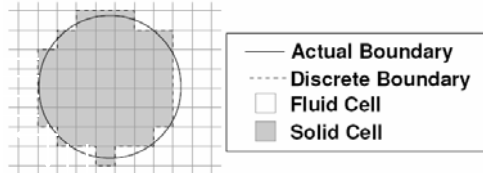
$$\mathbf{F}_{\text{obj}} = \int_S \boldsymbol{\sigma} \cdot \mathbf{n} dA \Rightarrow \sum \mathbf{P}_i \Delta A$$



## Grid-based Solution



- Boundary approximation

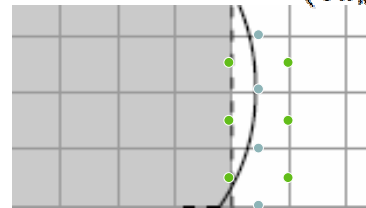


• All normals,  $\mathbf{n}$ , axis-aligned

## Grid-based Solution



$$\sigma_{ik} = -p\delta_{ik} + \mu \left( \frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} \right)$$



- One-sided finite difference derivatives

## Organization



- Previous work
- Overview
- Haptic Force Computation
- Haptic Filtering
- Results

## Haptic Display



- Problem:
  - Sim Generates 30-100Hz data
  - Want 1000Hz output
- One Solution:
  - Filtering



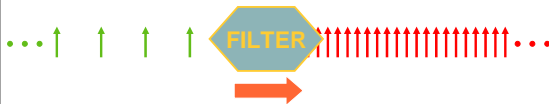
SensAble's Phantom

- Goal: smooth out abrupt changes

## Filtering



- Remove discontinuities
- Finite Impulse Response
- Bartlett-Hanning window filter



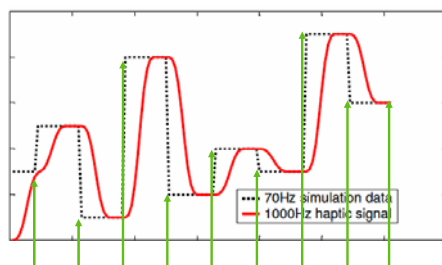
## Filtering - Reconstruction



- Analogous to image reconstruction



## Filtering



## Organization

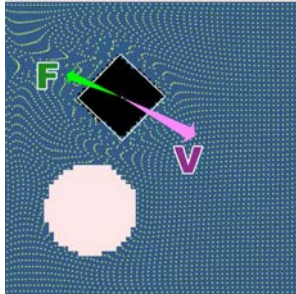


- Previous work
- Overview
- Haptic Force Computation
- Haptic Filtering
- Results

## Results



- 2D Navier-Stokes solver



## Results



- 3D Stokes fluid painting simulation (Baxter et al CASA 2004)



## Summary



- First fluid-haptic interaction
- Effective, inexpensive filtering
- Integration with interactive painting

## Conclusion



- Can enhance VR/Sim Apps
  - Users feel enriched experience
- General approach
  - Applies to both online & offline simulation
- Force computation is affordable
  - Bottleneck is fluid solver, not force

## Limitations



- Sampling of fluid solution
  - Many numerical methods have poor accuracy around boundary
- Filtering Forces
  - Can't create new information

## Future Work



- Higher order boundary treatment
  - e.g. GENSMAC
- Other interactive fluid methods (non-grid methods)
  - Smoothed Particle Hydrodynamics (SPH)
  - Moving Particle Semi-Implicit (MPS)
- Analytical requirements for smooth force output
- Criteria for force filters



## Acknowledgements

---



- Link Fellowship
- NVIDIA Fellowship
- Army Research Office
- Intel Corporation
- National Science Foundation
- Office of Naval Research

## The End

---



For more information, see

<http://gamma.cs.unc.edu/interactive>

<http://gamma.cs.unc.edu/DAB>

[http://gamma.cs.unc.edu/HAPTIC\\_FLUID/](http://gamma.cs.unc.edu/HAPTIC_FLUID/)