Proceedings of the Second PHANToM Users Group Workshop

October 19-22, 1997
Endicott House, Dedham MA
Massachusetts Institute of Technology, Cambridge, MA

Hosted by

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

Sponsored by

SensAble Technologies, Inc., Cambridge, MA

The Second PHANToM User's Group Workshop

On October 19-22, 1997, the Second PHANToM Users Group Workshop was held at the MIT Endicott House in Dedham Massachusetts. Designed as a forum for sharing results and insights, the workshop was attended by more than 60 participants from 7 countries.

Twenty presentations in diverse areas including rigid and compliant rendering, tool kits, development environments, techniques for scientific data visualization, multi-modal issues and a programming tutorial were enjoyed by all. In addition, numerous hands-on demonstrations were made available and provided concrete evidence of the enormous progress made by the PHANToM users community during the past year. Best presentation awards (and the coveted SensAble leather jackets) were given to John McLaughlin from CISRO (Australia) and Rob Playter from Boston Dynamics (USA).

We would like to express our sincere gratitude to all the participants for helping make the workshop a success and look forward to future meetings. We would also like to thank the staff of SensAble Technologies for their generous support and assistance in making the workshop run smoothly. Finally we extend our special thanks to Jacqui Taylor of MIT for her tireless, cheerful and expert assistance in pulling together all the logistical and administrative details which made the event a success.

Kenneth Salisbury
Mandayam Srinivasan
Cambridge, December 1997

The Second PHANToM User's Group Workshop
October 19-22, 1997
MIT Endicott House, Dedham MA

Program

Sunday, October 19

1:00-4:00 (Optional) Demonstrations at MIT (AI Lab and Touch Lab, RLE).
Rides available to Endicott House, departing 4:00 from AI Lab., ground floor entrance, by advance arrangement.

1:00-6:00 Arrival at Endicott House, Registration

6:00-7:00 Dinner

7:30-9:30 Welcome Reception and Refreshments

Monday, October 20

8:00-9:00 BREAKFAST (served in dining room for Endicott overnite guests. Conteniental breakfast available in Brook Center for all attendees.)

9:00-10:00 INTRODUCTION

Welcome to the Second PHANToM Users Group Workshop
Kenneth Salisbury and Mandayam Srinivasan
AI Lab and RLE, MIT
jks@ai.mit.edu, srini@mit.edu

The Year in Review at SensAble Technology
Bill Aulet, Thomas Massie
SensAble Technologies, Inc.
aulet@sensable.com

10:00-10:30 BREAK

10:30-12:30 RENDERING TECHNIQUES AND TOOLS

Haptic Rendering: Point- and Ray-Based Interactions
Chih-Hao Ho, Cagatay Basdogan, and Mandayam A. Srinivasan
MIT
srini@mit.edu

A Unified Approach To Haptic Rendering and Collision Detection
Craig Latimer
MIT
lat@ai.mit.edu

Adding Motion to Constraint Based Haptic Rendering Systems: Issues & Solutions
Diego Ruspini
Stanford University/Interval Corporation
ruspini@robotics.Stanford.edu

Extending the GHOST SDK Shape and Dynamic Classes
Ted Bardasz
SensAble Technology, Inc.
tbardasz@www.sensable.com

12:30-1:30 LUNCH

1:30-3:00 BREAKOUT SESSIONS

Breakout 1: GHOST: community feedback and collaboration
Breakout 2: TBD

3:00-4:30 REMOTE AND COMPLIANT RENDERING

Controlling Force Feedback Over a Network
Adam Seeger
UNC, Chapel Hill Computer Science Department
seeger@cs.unc.edu

Realistic Haptic Rendering of Flexible Objects
Franco De Angelis, Monica Bordegoni, Giancarlo Frugoli and Caterina Rizzi
Department of Industrial Engineering, University of Parma, Parma, Italy
fda@ied.eng.unipr.it

Haptic Rendering of Elastic and Plastic Surfaces
Chris Tarr
SensAble Technology, Inc.
ctarr@sensable.com

4:30-5:00 BREAK

5:00-6:30 BEYOND FORCE

Hot and cold running VR: adding thermal stimuli to the haptic experience
Mark Ottensmeyer
MIT
canuck@ai.mit.edu

Texture Sensing and Stimulation Using the PHANToM: Toward remote sensing of soil properties
Donald F. Green
MIT
dfg@ai.mit.edu

Simulation of contact sounds for haptic interaction
Dinesh K. Pai and Kees van den Doel
University of British Columbia
pai@cs.ubc.ca

7:00-8:00 DINNER

8:30-9:30 REFRESHMENTS

Tuesday, October 21

8:00 - 9:00 BREAKFAST (served in dining room for Endicott overnite guests. Conteniental breakfast available in Brook Center for all attendees.)

9:00 - 10:30 DEVELOPMENT ENVIRONMENTS

FLIGHT - A 3D Human-Computer Interface and Application Development Environment
Tom Anderson
Sandia National Labs.
tganders@u.washington.edu

LEGOLAND: A Multi-Sensory Environment for Virtual Prototyping
P. Young, T. Chen, D. Anderson, J. Yu(1); S. Nagata(2)
(1) Department of Electrical Engineering Colorado State University
(2) NHK Science & Technical Research Labs
pmy@engr.colostate.edu

Adding Force Feedback to Interactive Visual Simulations using Oxygen(tm)
Peter Larsson
Prosolvia Clarus AB, Sweden
peterl@prosolvia.se

10:30 - 11:00 BREAK

11:00 - 12:30 APPLICATIONS I

The Haptic Workbench Applied to Petroleum 3D Seismic Interpretation
Keith Nesbitt, Randall Gallimore, Bernard Orenstein(1); John McLaughlin(2)
(1) BHP Research
(2) CSIRO Mathematical and Information Sciences
gallimore.randall.rj@bhp.com.au,bernie@agents.com.au

Haptic Rendering of 3D Seismic Data
John McLaughlin(1); Bernard Orenstein(2)
(1) CSIRO Mathematical and Information Sciences
(2) BHP Research
John.McLaughlin@cmis.csiro.au

A Novel Virtual Reality Surgical Trainer with Force Feedback: Surgeon VS. Medical Student Performance
Rob Playter
Boston Dynamics, Inc.
playter@bdi.com

12:30 - 1:30 LUNCH

1:30 - 3:00 APPLICATIONS II

Some Calibration Information for a PHANToM 1.5 A.
Karl Reinig and Ryan Terry

University of Colorado Center for Human Simulation
karl@pixsun.uchsc.edu

Developing a PC-based Spine Biopsy Simulator
Kevin Cleary, Corinna Lathan, R. Craig Platenberg, Kevin Gary, Laura Traynor, Fred Wade
Georgetown University Medical Center
cleary@isis.imac.georgetown.edu

The Phantasticon: The PHANToM for Blind People
Calle Sjöström
Center for Rehabilitation Engineering Research, Lund University, Sweden
Calle.Sjostrom@certec.lth.se

3:00 - 4:30 BREAKOUT SESSIONS

4:30 - 5:30 WRAP UP AND DISCUSSION OF FUTURE DIRECTIONS

6:00 - 7:00 DINNER

7:30 - 9:00 REFRESHMENTS

Wednesday, October 22

7:30 - 8:30 BREAKFAST (in dinning room only)

9:00 - 11:00 DEPARTURE (Optional Bus to Airport by prior arrangement)

(revised 10/9/97)

Haptic Rendering: Point- and Ray-Based Interactions

Chih-Hao Ho, Cagatay Basdogan, and Mandayam A. Srinivasan

Laboratory for Human and Machine Haptics
Department of Mechanical Engineering and
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, MA, 02139
chihhao, basdogan, srini @mit.edu
http://touchlab.mit.edu

Abstract

In this paper, we summarize our progress on two interaction techniques (point-based and ray-based) for haptic rendering. First, we describe a new point-based rendering technique that is computationally less expensive than previously proposed methods. We use a localized search procedure to reduce the number of computations and make it independent of the number polygons of the polyhedron. Second, we discuss the implementation details of a ray-based technique together with its advantages. We have utilized this rendering technique to display the haptic interactions between 3D objects and the stylus modeled as a line segment.

In order to develop more effective multimodal VEs, we have experimented with multi-threading (on Windows NT platform) and multi-processing (on UNIX platform) techniques and have successfully separated the visual and haptic servo loops. Our experience is that both techniques are quite useful in achieving high haptic rendering rates and stable haptic interactions. Although creating a separate process for each modality requires more programming effort, it enables the user to display the graphics and/or haptics on any desired machine(s). On the other hand, programming with threads takes less effort, but they are not as flexible as processes. We have also developed a graphical interface that enables the user to construct virtual environments by means of user-defined text file, toggle stereo visualization, save the virtual environment and quit from the application. The user can load objects into the scene, assign simple visual and haptic properties to the objects using this text file. Following the construction of the scene using the text file, the user can interactively translate, rotate, and scale objects and the interface will automatically update the visual and haptic models.

Collision Detection

In haptic rendering, one of the important problems is the detection of collisions between the end-effector (or generic probe) of the haptic device and the 3D objects in the scene. A good collision detection algorithm not only increases the servo rate, but also helps in correctly displaying interaction forces to the human operator to make the virtual objects more realistic. The collision detection techniques used in haptic rendering slightly differ from the ones used in computer graphics. In computer graphics, the collision detection is used to detect if two objects overlap. On the other hand, the purpose of the collision detection in haptic rendering is not only to check collisions between objects, but also to calculate the appropriate interaction forces to convey to the user a tactual sensation of 3D objects. The calculated interaction force increases with the penetration depth and has the effect of preventing further penetration of stylus into object surfaces.

We propose two types of haptic interaction algorithms here: point-based and ray-based. For the point-based haptic interaction algorithm, the probe is simply modeled as a point. For the ray-based collision detection algorithm, the probe is modeled as a line segment. Both techniques have advantages and disadvantages. For example, it is computationally less expensive to render 3D objects using point-based technique. Hence, we achieve higher haptic servo rates. On the other hand, the ray-based haptic interaction technique handles side collisions and can provide additional haptic cues for conveying to the user the shape of objects.

Point-Based Haptic Interaction Algorithm

The point-based haptic interaction algorithm we have proposed is similar to the god-object algorithm (Zilles and Salisbury, 1995) but follows a different approach. We believe that our approach increases the servo rate, makes the haptic interactions more stable and the servo rate independent of the number of polygons. In this approach, the first step is to construct a hierarchical data base for the geometrical properties of the 3D objects. The data base is used to store geometrical information of the polygonal objects in the virtual environment. In the data base, we assume that the polyhedrons are made of three types of geometrical primitives: polygon, line, and vertex. Each polygon has neighboring primitives of lines and vertices. Each line has neighboring primitives of polygons and vertices, and each vertex has neighboring primitives of polygons and lines. In the data base, each primitive also has a normal vector attached to it. For a polygon primitive, the normal is the vector that is perpendicular to its surface and points outside. For a vertex or line primitive, its normal is the average of the normals of its neighboring polygons.

When exploring the virtual environments, we interact with objects through the Haptic Interface Point (HIP). At the same time, we also consider another point, ideal haptic interface point (IHIP) -same as the so called god-object(Zilles and Salisbury, 1995)- to follow the trace of the HIP. The HIP is not constrained, and because haptic interface devices have a finite maximum stiffness, it can penetrate the object. However, we can constrain the IHIP such that it doesn't penetrate any object. When the HIP is outside the virtual object, the IHIP will be coincident with the HIP. When the probe penetrates an object, the IHIP will stay on the surface of the object. When the user moves the HIP by manipulating the probe, the IHIP will also be moved to a new position depending on the situation. The rule for moving the IHIP is that the IHIP will move towards the probe and cannot penetrate any surface. Furthermore, we assume that there is a virtual spring between the HIP and the IHIP. The force applied to the HIP is calculated based on the distance between the HIP and the IHIP.

When the HIP is outside the object, we keep track of its path and check if this path penetrates any polygon surface. We construct a line segment from the previous (tail) and the current (tip) coordinates of the HIP and detect the collisions between the line segment and the polygons of the 3D object. If this line segment penetrates the object, then we choose the surface that is nearest to the previous HIP as the contacted geometric primitive. The IHIP will then stay on this surface. We then calculate the nearest distance from the current HIP to the contacted primitive and its neighboring primitives (We only check the contacted primitive and its neighboring primitives, not all of the primitives. For example, if the contacted primitive is polygon, then we check the distance from current HIP to the neighboring lines and vertices. This approach significantly reduce the number of computations). Finally, we set the primitive that has the shortest distance to the HIP (the priority of selecting the nearest primitive goes as vertex, line, and polygon) as the contacted primitive and move the IHIP to the point that is on this primitive and nearest to the HIP. After setting the new position of IHIP, we construct a vector from the IHIP to HIP. If this vector and the normal of the contacted primitive are in the same direction, the HIP is no longer inside the object.

```
if (collision == FALSE)
{
    if (the path of HIP penetrates a polygon)
        {
            1. set the polygon as the contacted primitive
            2. move the IHIP to this surface
        }
}
else // there is a collision
{
    primitive1 = the contacted primitive of the previous
        loop;
    distance1 = nearest distance from current HIP to
        primitive1
    for (i=1 : number of neighboring primitives of
        primitive1)
    {
        primitive2 = the i^th neighboring primitive of
                primitive1
        distance2 = distance from HIP to primitive2
```

```
            if(distance2 < distance1)
    {
            contacted primitive = primitive2;
            distance1 = distance2;
     }
}
move IHIP to the point that is nearest from the
    contacted primitive to HIP;
Vector1 = vector from IHIP to HIP;
Normal1 = normal of the contacted primitive;
if( dot(Vector1, Normal1) > 0)
  collision = FALSE;
else
  collision = TRUE;
}
```

Ray-Based Haptic Interaction Algorithm

The ray-based haptic rendering is a haptic interaction technique we have developed recently (Basdogan et. al, 1997, Srinivasan and Basdogan, 1997). In ray-based rendering, we model the generic probe of the haptic device as a line segment and detect the collisions with 3D objects in the scene. Therefore, the collision detection will be much more complicated than that in point-based case. In the point-based case, the only possible collision situations are *point-point*, *point-line*, and *point-polygon* collisions. However, the collisions in the ray-based case can in addition be *line-point*, *line-line*, *line-polygon* collisions. There can also be multiple contacts composed of a combination of the above cases.

Since the collisions in ray-based are more complicated, it is not easy to develop a set of simple rules that can handle all of the cases. Therefore, we opted to develop a rule-based algorithm that can successfully handle only the collision conditions between a line probe and convex objects. However, this is not a major limitation since the concave polyhedron objects can always be divided into several convex objects. The ray-based collision detection algorithm that we propose follows a *two-stage* approach for detecting the collisions. Analogous to IHIP in point-based interactions, we consider another stylus (we call it the ideal stylus) to follow the trace of the simulated stylus. Similar to the HIP, the simulated stylus can penetrate the objects, but the ideal stylus is constrained to stay on the surface of the object and to be always parallel to the simulated stylus. We assume that there are two springs between the simulated stylus and the ideal stylus - one between the tips and the other between the tails, respectively. This arrangement will help us in computing torque in addition to the forces to be displayed to the users. It should be noted here that the only plausible collision conditions are point-polygon and line-line contacts because we consider only convex objects. The point-point, point-line, line-point, and line-polygon are all unlikely to occur, because at best they occur in a single servo loop.

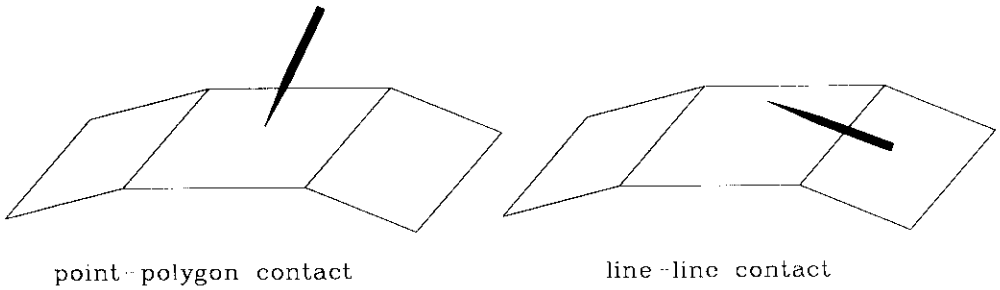


Figure 1. Plausible collision conditions between the simulated stylus and convex object in ray-based interactions.

When implementing this algorithm, the main challenge is to correctly position the ideal stylus. We use two-stage movement technique to move the ideal stylus.

First Stage Movement

In the first stage movement, the first step is to find out whether the contacted primitive is a polygon or a line.

1. If the contacted primitive is a polygon, we move the ideal stylus along the surface. To do this, we first compute the movement (vector **V**) of the simulated stylus. We then calculate its component (**Vp**) along the surface of the contacted polygon. This is the vector that we use to move the ideal stylus (see Figure 2a and 2b).

$\vec{V}p=\vec{V}-(\vec{V}\bullet\vec{N})\vec{N}$ (**N** is the normal of the polygon)

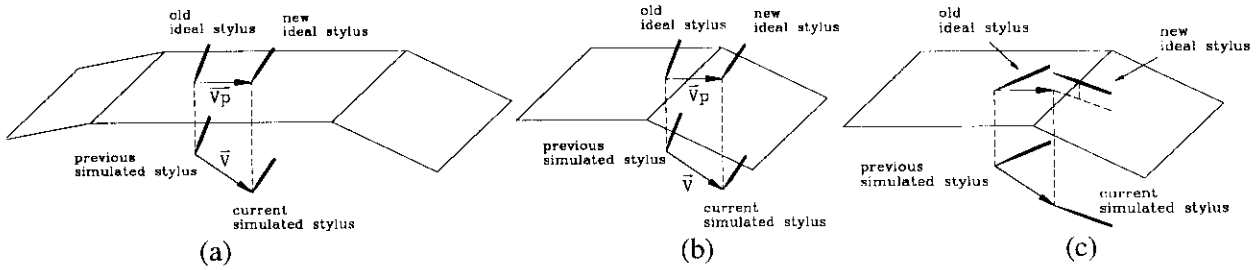


Figure 2. Illustration of the first stage movement in ray-based haptic interactions.

In this movement, if the new ideal stylus doesn't penetrate any line, the movement will be completed in this stage and the contacted primitive will still be set as a polygon. If the ideal stylus penetrates any of the lines, we will move the ideal stylus such that the new position of the ideal stylus will be at the position that has the shortest distance from the line to the current simulated stylus (see Figure 2c). The contacted primitive will then be set as a line.

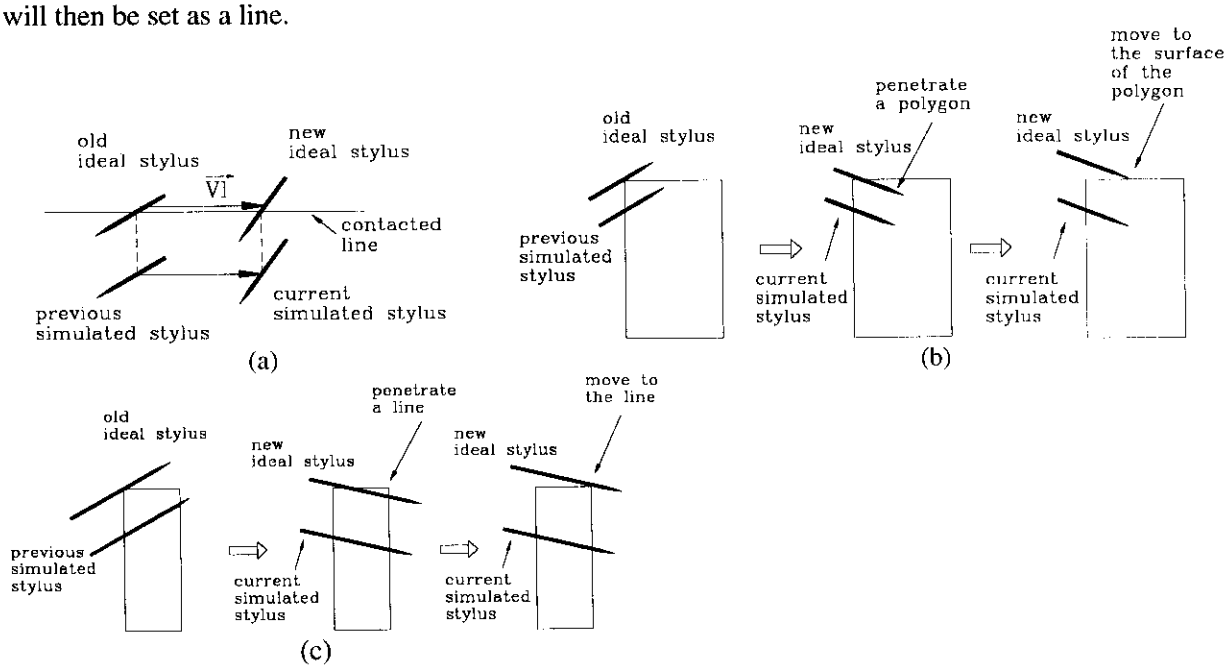


Figure 3. First stage movement (when the contacted primitive is a line) of the ray-based rendering algorithm: (a) the movement of ideal stylus without any constraints, (b and c) the movement of ideal stylus with constraints.

2. If the contacted primitive is a line, we move the ideal stylus along the line. To do this, we first find the point (say, point 1) which is at the contacted line and has the shortest distance from the line to the previous

simulated stylus. Then, we find another point (say, point 2) which is also at the contacted line (or an extension of it) and has the shortest distance from the line to the current simulated stylus. After that, we can get a vector (say, vector Vl) which is from point 1 to 2 and this vector V1 is used to move the contact point of the old ideal stylus to the contact point of the new ideal stylus. The orientation of the ideal stylus, however, is always parallel to the simulated stylus. From the contact point, we could decide the new position of the ideal stylus (see Figure 3a). If the new ideal stylus doesn't penetrate any polygon or line, the movement will be completed at this stage and the contacted primitive will still be a line. If the tip of the ideal stylus penetrates any of the polygons, its tip position will move to the surface of the polygon and the contacted primitive will be set as a polygon (see Figure 3b). If the ideal stylus penetrates another line, the position of the ideal stylus will be moved to this line and the contacted primitive will be the new contacted line (see Figure 3c).

Second Stage Movement

In the second stage movement, we just move the ideal stylus towards the simulated stylus. If there is any line or polygon between the ideal stylus and the simulated stylus, the ideal stylus will be stopped at the first primitive it contacts. The pseudo code for the second stage movement is given below:

```
if (contacted primitive is a line)
{
  if (there is any line between ideal stylus and simulated
        stylus)
    move ideal stylus to the first contacted line,
    contacted = TRUE;
  if (there is any polygon between ideal stylus and
        simulated stylus)
    move ideal stylus to the first contacted polygon,
    contacted = TRUE;
  if (contacted = FALSE)
    move ideal stylus to the simulated stylus
}
else      //(contacted primitive is a polygon)
{
  if (there is any polygon between ideal stylus and
        simulated stylus)
    move ideal stylus to the first contacted polygon,
    contacted = TRUE;
  if (there is any line between ideal stylus and simulated
        stylus)
    move ideal stylus to the first contacted line ,
    contacted = TRUE;
  if (contacted = FALSE)
    move ideal stylus to the simulated stylus
}
```

In this algorithm, similar to the point-based case, we construct a hierarchical data base such that each polyhedron is composed of vertices, lines, and polygons. Each vertex has neighbors that are lines and polygons. Each line has neighbors that are vertices and polygons. And each polygon has neighbors that are vertices and lines. In the hierarchical data base, each line and polygon also has a list of possible contacted neighbors. The possible contacted neighbors are those that can be potentially contacted in the next servo loop. Therefore, when we check the collision, we need to check only the primitives in this list instead of all of the primitives of the object. In this way, we can save on the computation time and make the servo rate independent of the size of the polyhedron that represents the full object.

Acknowledgments

Part of this work was carried out under VETT contract N61339-96-K-0002 funded by ONR and NAWC/TSD.

References

1. Basdogan, C., Ho, C., and Srinivasan, M.A., "A Ray-Based Haptic Rendering Technique for Displaying Shape and Texture of 3D Objects in Virtual Environments ", *ASME Winter Annual Meeting, Dallas, TX 1997. (In press)*

2. Srinivasan, M.A., and Basdogan, C., "Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges", Computers and Graphics, Vol. 21, No.4.,1997.

3. Zilles, C.B., and Salisbury, J.K., "A Constraint-Based God-Object Method for Haptic Display", *IEEE International Conference on Intelligent Robots and System, Human Robot Interaction, and Co-operative Robots.* IROS, pp 146-151, Vol 3, 1995.

A Unified Approach to Haptic Rendering and Collision Detection

Craig Latimer and J. Kenneth Salisbury

Department of Mechanical Engineering
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA

Abstract

This paper presents a unified approach to haptic rendering and real-time collision detection in virtual environments. The foundation for both stems from the geometric support for polygonal objects, which includes topological adjacency information, contact operations, and a space partitioning tree. Haptic rendering uses the space partitioning tree to determine the location of contact and the adjacency information to traverse the surface. Similarly, collision detection checks for overlap between two polygonal objects by comparing their space partitioning trees. The motion of objects in the virtual environment is computed using an impulse-based dynamic simulation.

1.0 Introduction.

Since the conception of virtual environments, sight and sound have played an integral part in developing realistic and convincing simulations. More recently, however, a third sense has been added to virtual environments—the sense of touch. Newly created devices, called haptic interfaces, give the user the ability to interact with virtual environments just as they do in reality—with their hands. Unlike sight and hearing, with touch we can not only sense the environment, we can also affect the environment. When we touch objects, we can gain a sense of texture, stiffness, and shape, while causing these objects to move, rotate, or deform. Although adding touch to virtual environments creates a more convincing and realistic environment, it also requires that the virtual environment be realistically modeled. When we push an object, how quickly the object moves and in what direction the object moves should depend on how the object is being pushed. When objects move in response to the user's touch, their motion cannot be pre-computed, and collisions between objects are possible. For this reason, virtual environments must use a physically based simulation that can model touch interactions and collisions between objects.

Most previous work has focused either on haptic rendering or collision detection but not both simultaneously. Salisbury et al. [5] presented a general haptic framework based on point-object interactions. This work was further developed by Zilles [8] for rendering polygonal objects using a constraint based method. More recently, point-object haptic interaction has been extended to other geometry such as NURBS [7] and implicitly defined algebraic surfaces [6]. A new interaction paradigm, line-object haptic display, has been recently developed by Basadogan et al. [1]. Collision detection, on the other hand, has been widely studied for over the last decade, and the approaches vary greatly between algorithms-- often the best technique is problem dependent. Since we wish to consider large non-convex polyhedra, the collision detection algorithms are similar to those presented by Gottschalk et al.[3].

The outline of this paper is as follows: First, the geometric support for polygonal objects is presented. Next, determining possible contact using the space partitioning tree is presented for both haptic rendering and collision detection. Lastly, contact evaluation including force and impulse generation is presented.

2.0 Data Structures for Polygonal Objects.

Each polygonal object contains a list of its faces, edges, and vertices. The face class contains two data members, a support plane which passes through all of the edges and vertices of the face and a list containing pointers to the edges of the face. These edges are ordered so that they encircle the face in counterclockwise direction. Like the face class, the edge class also has pointers to adjacent features. An edge has pointers to two vertices, a head vertex and a tail vertex, as well as pointers to the faces sharing this edge. An edge has two additional data members, a normalized direction vector which points from the tail vertex to the head vertex and a length representing the distance between the head and tail vertices. The vertex class has only two member variables: a point representing the vertex position and a list containing pointers to adjacent edges.

Each polygonal object also contains a space partitioning tree composed of oriented bounding boxes (OBBs). Each OBB has a 3x3 rotation matrix defining its orientation, a position vector locating of its center of OBB, and a size vector giving the distance from the center to the faces of the box. Gottschalk et al.[3] presents an algorithm to create tight fitting OBBs. The tightest fitting OBB for a given a set of faces is determined using statistical properties of the set. The set is then divided into two subsets based on which side of the box the center point of the faces lie. OBBs are then created for these subsets, and the process repeats recursively creating an OBB tree. The leaf nodes of this tree contain a single pointer to a face.

3.0 Determining Possible Contact

The OBB tree can be used for determining possible contact during both haptic rendering and collision between objects.

For haptic rendering, the user's position is represented by an "interaction point." We want to use the OBB tree to quickly determine either that the interaction point is not touching the object or which face it might be touching. To determine this, we start with the top level box and transform the point into the boxes coordinate system. Then, the coordinate of the point is compared to the size of the box. If the point passes this test, then the children boxes are tested. This process repeats until either no contact or a possible contact face is determined.

For collision detection, the OBB trees for two possible colliding objects are compared to determine possible overlapping faces. Our overall strategy is to start with the top bounding boxes and work our way down the trees. Suppose we have two polygonal objects, A and B, and each has an OBB tree. Initially, the position and orientation of each top level bounding box is defined with respect to some global coordinate system. The first step is to transform our coordinate system such that A's OBB is at the origin and aligned with the coordinate axes. In these new coordinates, we can efficiently check if A's OBB overlaps B's OBB.

Two OBBs will not overlap if there is some plane that separates them. Alternatively, we can determine separation using any line parallel to the plane's normal. If the projection of the two OBBs onto the line forms a disjoint interval then the OBBs do not overlap. For this reason, the normal of the plane is called a separating axis. It is important to realize, however, that if we

project the OBBs onto an arbitrary line and the intervals overlap, it does not necessarily mean the OBBs overlap. Rather, it simply indicates that further tests may be required. To establish that two OBBs overlap we must check planes parallel to the faces and planes parallel to an edge from each object resulting in a total of 15 separating axis tests. Placing one OBB in the other's reference frame and explicitly writing down the equations results in significant simplifications. In all, the 15 separating axes require 166 operations to determine overlap, but usually much fewer are needed to determining disjoint boxes. [3]

If the two OBBs overlap, then we work our way down the trees looking for intersecting faces. The first step is to decide which tree to start descending. If B's OBB is at a lower level than A's OBB or has no children because it is a leaf, then we will descend the children of A's OBB. We do this by placing B's OBB in the coordinate system of the OBB of A's child. On the other hand, if we instead descend to the child of B, then we must put the OBB of B's child into the coordinate system of A's OBB. In this manner, we recursively "stair step" down the trees. The result of this process will be a list of potentially overlapping faces.

4.0 Contact Evaluation

In the previous section, we ascertained which faces the user might be touching or which faces might be overlapping. In this section, we describe how to take these possible contact or collisions and generate forces or impulses. The cornerstone of both algorithms is determining whether a point is on or directly below a face. To check this, we create several test planes passing through each edge and perpendicular to the support plane. Graphically, all the test planes create a region similar to extruding the face. If a point is on or below the face's support plane and above the test planes then the point will indeed be on or directly below the face.

For haptic rendering, we use the algorithm above to test if the interaction point is directly below the faces determined from the OBB tree search. If it is not, then the user is not touching that face. If it is, then the closest point on the face is a projection of the interaction point onto the support plane of the face. The force that the user feels is generated using a penalty method similar to attaching a stiff spring and damper between the interaction point and the closest point on the surface.

For collision detection, we need to compare possible overlapping faces, and if they are overlapping, then we need to determine the collision point and the collision direction. In collisions there are two types of contacts, vertex-face contacts and edge-edge contacts.[2] We will assume that the two faces being considered are FaceA and FaceB. The first step is to determine which side of FaceA the vertices of FaceB are on, and store that information for possible future use. If the vertex is below the support plane of FaceA, then we check if the vertex is directly below FaceA using the algorithm developed above. If the vertex is penetrating the face, then the collision point is the closest point on the face to the vertex, and the collision direction is the faces normal. If all of the vertices of FaceB are above FaceA, then the faces are not touching. An analogous procedure is used to check the vertices of FaceA against FaceB.

The next step is to check for edge-edge contact. We start with an edge of FaceA and compare it to FaceB. If the head and tail vertices of the edge are on opposite sides of FaceB, then we determine where the edge intersects FaceB's support plane. If the intersection point lies on FaceB, as determined by the algorithm above, then we know that edge-edge contact exists and that we must find which edge of FaceB intersects FaceA. Finding the edge on FaceB that

intersects FaceA is done in the same manner. The collision direction is simply a vector orthogonal to both contacting edges.

Typically, several faces interfere for each contact. We must therefore prune our contact list to eliminate these redundancies. When two objects do collide, the response is handled through impulses, instantaneous changes in the velocities of the colliding objects. The impulses are computed analytically based on the velocities and the configuration of the two objects at the time of impact. [4]

5.0 Conclusion

A unified approach allows simultaneous support of haptic rendering and collision detection resulting in richer, more expressive virtual environments. Moreover, it reduces memory requirements and is easily extendible to other haptic rendering paradigms.

Acknowledgments

The material in this paper is based on work supported by ONR Grant no. N00014-97-1-0655, "Multimodal Software Development for VETT" and NAWC/TSD contract no. N61339-96-K-0002, "Development of Haptic Interfaces".

References

[1] C. Basdogan, C. Ho, and M. Srinivasan. "Haptic Display of 3D Rigid Objects in Virtual Environments." Submitted to the *Winter Annaul Meeting of ASME,* Nov. 15-21, 1997.

[2] J. Canny. "Collision Detection for Moving Polyhedra." *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 2. March 1986.

[3] S. Gotshalk, M. Lin, and D. Manocha. "OBB-Tree: A Hierarchical Structure for Interference detection." *Proceedings of ACM Siggrahp'96*, pages 171-180, 1996

[4] B. Mirtich. "Impulse-based Dynamic Simulation of Rigid Body Systems." Ph.D. Thesis, University of California, Berkeley, 1996.

[5] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles. "Haptic Rendering: Programming Touch Interactions with Virtual Objects." *Proceedings of the ACM 1995 Symposium on Interactive 3-D Graphics*, Monterey CA, April 1995.

[6] J. K. Salisbury, and C. Tarr, ``Haptic Rendering of Surfaces Defined by Implicit Functions," *Proceedings of the ASME Sixth Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Nov 15-21, 1997, Dallas, Texas.

[7] T.V. Thompson II, D.E. Johnson, and E. Cohen. "Direct Haptic Rendering Of Sculptured Models" *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, Providence, RI, pp. 167-176

[8] C. Zilles and J. K. Salisbury. "A constraint-based god object method for haptic display." *Proceedings of IROS-95*, Pittsburgh, August 1995.

Adding Motion to Constraint Based Haptic Rendering Systems: Issues & Solutions

Diego Ruspini

Stanford University/Interval Research Corporation

Abstract

Incorporating dynamic motion into a haptic rendering system introduces complexities normally not seen in traditional graphic systems. Haptic systems are capable of taking input from the user to manipulate a virtual environment, as well as applying an output force to simulate contact with that environment. In most graphic systems the major focus is placed on the display of the virtual environment (output) and only limited capabilities are provided for interacting with the environment through a mouse or other device (input). To achieve a realistic impression of motion, haptic systems must run at a very high servo rate. This requirement implies that at least some dynamic capabilities need to be incorporated directly into the haptic rendering process. In this talk we will discuss some of the issues found and the solutions developed to add dynamic motion to the haptic rendering library "HL." In all cases the solutions proposed have attempted to be as general as possible, and applicable to a wide range of applications and fields of interest.

Introduction

To be useful in developing real applications haptic rendering systems must provide the ability to manipulate, interact and modify a virtual environment. Movements that, at 30Hz, visually appear smooth feel discontinuous and jerky when updated at the same rate on a haptic system. To achieve realistic continuous motion the haptic rendering system must incorporate methods to allow motion and force interaction to be specified and executed in the haptic servo loop, at a very high update rates (> 1000Hz).

This presentation will discuss some of the techniques used to incorporate dynamic functionality into the "HL" constraint based haptic rendering library[3]. This library uses a virtual proxy to model all interaction between the user and the environment, including surface contact, friction, shading and texture. This framework is continued when adding motion to the haptic environment. The sensation of motion, mass and inertia are created solely by updating the position of the proxy in the virtual environment. A position control loop is used to attempt to move the user's finger location to the position specified by the proxy.

For many models it is possible to construct a hierarchical linkage that describes the dynamic motion of the model. While forward dynamic solutions, like Featherstone[2] are capable of computing the equations of motion for arbitrary tree-like systems, at present these solutions are too slow to allow for real-time updates at the rates required for haptic simulation. In our implementation a highly simplified set of equations is used to describe the dynamic behavior of the virtual model. This model allows the motion of each joint variable to be computed in constant time. While simplistic this model has been used to model a wide variety of complex objects like windmills, carnival rides and robots. This solution also produces the exact solution for the most commonly found haptic dynamic objects: buttons, sliders and dials.

Other models can not be conveniently decomposed into an articulated hierarchy of rigid bodies. Objects that are deformable or soft move in ways that are difficult to describe with a single general purpose model. In these cases the objects surface must be updated by the application program. Given the often limited bandwidth between the user application and the haptic control loop, it is usually not possible to update the underlying model at a rate sufficient to give the feeling

of a continuous free-form deformation. A method to "morph" between discrete object definitions is proposed which functions by interpolating the proxy's position between the old and new object definitions. This "blending" is often sufficient to restore the apparent continuous nature of the motion seen on the visual display.

Modeling Interaction with Hierarchical Tree Linkages

When modeling a mechanical system, or articulated figure it is often convenient to describe the system as a hierarchical linkage of rigid bodies (links) connected by simple one degree of freedom(d.o.f.) joints. Links may be empty allowing more complex joints to be constructed by specifying several simple joints is series. The position of each joint (rotational, prismatic, screw) can be specified by a joint coordinate Θ_i, $1 \le i \le n$, where n is the number of d.o.f. of the system. The configuration of the entire system can then be specified by the joint space vector $\Theta = [\Theta_1 \quad \dots \quad \Theta_n]$. Some example tree linkages are shown if figure 2.

When a collision occurs between the proxy and an object(s) in the environment, a force is applied to the user to simulate contact with the surface. In a dynamic environment a equal and opposite force f is applied at the contact point(s) which may induce accelerations on the virtual system. In a hierarchical linkage it can be shown [1] that the resulting acceleration is equivalent to that induced by a joint torque vector

$$\Gamma = J_c^T f,$$

where J_c is the Jacobian of the contact point c, such that the velocity v of the contact point is given by $v = J_c \dot{\Theta}$.

More complex mechanical systems can be constructed by introducing an additional level of abstraction. Instead of defining each link as an independent degree of freedom. Each link variable is specified as the function of one or more animation variables *(avars)*. In the general case $\Theta_i = f(q_1, ..q_m)$ for some set of avars q_i $1 \le i \le m$. I similar Jacobian for this space can be constructed, and used to create many closed chain, geared, oscillatory or other more complex systems.

Dynamic Models

Once the effective torque created by a contact is found it can be inserted into the equations of motion of the system to obtained the configuration space accelerations of the system. The equations of motion for the system can in general be described by the equation:

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

where $M(q)$ is the positive definite kinetic energy matrix, $b(q,\dot{q})$ the centrifugal coriolis vector and $g(q)$ the gravity force vector.

In most situations the exact dynamic behavior is not required. In our current implementation the equations of motion of the system have been greatly simplified to allow real-time updates of the animation variables. The centrifugal coriolis vector is a function of the configuration space velocity. For most haptic applications the velocities in the environment are very small (high velocity virtual objects could be dangerous to the user and the haptic device). In most cases this vector can be assumed to be zero with very little perceivable effect on the motion of the linkage. The gravity vector can be easily computed at the same time as the contact forces. At a link with its center of mass at p the configuration space torque on the linkage is given by $\Gamma_{grav} = J_p^T(-m\vec{g})$, where $\vec{g}$ is a vector giving the magnitude and direction of the gravity force, m is the link mass and J_p^T is the Jacobian for the center of mass on the link. The torques of each link can be added with the generalized contact torques to obtain the total torque on the system.

Lastly in our current implementation we have chosen to not permit dynamic coupling between the animation variables. Each variable is treated independently. The set of equations of motion that can be represented by our system can therefore be specified by

$$\ddot{q} = M_{diag}^{-1}(\Gamma - g(q))$$

where M_{diag} is a semi-constant[1] diagonal matrix. Such a system is trivial to compute and the integration can accomplished in closed form. While obviously a tremendous simplification, systems of this type can often closely model the equations of motion of many common mechanical systems. In particular for 1-dof systems like dials, buttons, or sliders (The most common haptic dynamic objects) the equations of motion are exact and no approximation is needed. In many other systems that are intended to be manipulated by humans the mechanism is arranged to minimize the coupling between the d.o.f. of the system (lots of right angles between axes). Finally in many mechanical systems the diagonal elements of the mass matrix greatly dominate the other terms. As an example in most robotic systems the dominant inertia terms come from the motor inertia which because of multi-stage gearing far exceeds the inertial effects caused by the other links in the mechanism.

While space limits the amount that can be discussed it is trivial to add joint limits, spring/biasing forces, friction and other attributes to the individual animation variables. It is also possible to set there effective mass to infinite and have them behave as purely kinematic linkages.

Proxy Blending

Some models can not be conveniently decomposed into an articulated hierarchy of rigid bodies. Objects that are composed of some malleable material like clay, or soft tissue have many more d.o.f. than can be efficiently represented by a mechanical linkage. In these cases the object model may need to be updated by the application program which can exploit specific knowledge about the material being modeled and thereby make interactive movement possible. In many cases it is still not possible for the application program to update the object model at a rate sufficient for haptic rendering (> 1000Hz). In addition, as in the "HL" library, there is often a limited bandwidth between the user application and the haptic control loop.

To bridge this gap between the update rates of the simulation and the haptic rendering processes we can exploit a interesting asymmetry in the bandwidth of human tactile interaction. While humans can sense force vibrations in excess of 500Hz, the maximum human motion output bandwidth is only ~10Hz, and typically much less, on the order of 1 or 2 Hz. In many simulations only the low-frequency motions are of interest for display. In these cases by smoothly transitioning between discrete low-frequency model updates from the simulation process we can give the user a sensation of a continuous free-form deformation.

A method to "morph" between discrete object definitions is proposed which functions by interpolating the proxy's position between the old and new object definitions. An object is haptically rendered as in [3] by using a proxy which attempts to minimize the distance between the proxy and user's position. When an object is to be updated, instead of immediately replacing the object with the new definition, a new proxy is created which is constrained only by the updated environment. The old proxy remains constrained by the old environment specification. Over a blending period, both the old and new proxies are updated as in [3] according to the movements of the user's finger. During this period goal point of the user's finger is smoothly interpolated from the old proxy's to the new proxy's location. When the blending period is over the old proxy and old object definition are deleted and the rendering process continues on the updated environment. This process is illustrated in Figure 1.

---

[1] The elements of the M matrix can be updated by the application program but at a presumably slower rate then that of the haptic servo loop.

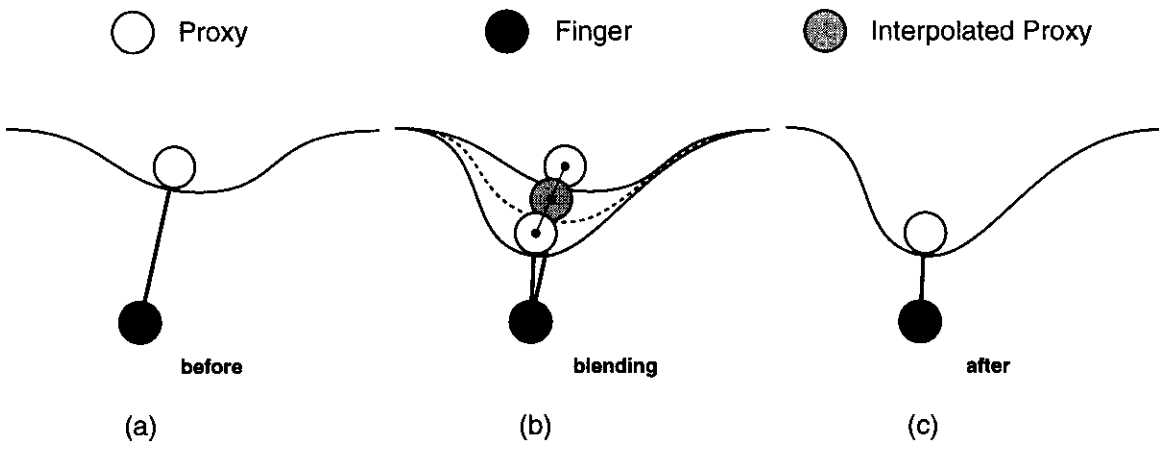


Figure 1: Proxy Blending. During a (b) blend the actual proxy is interpolated from a proxy constrained by the object definition before (a) and after (c) the blending period.

Although results are preliminary it appears that in most cases this "blending" is sufficient to restore the apparent continuous nature of the motion seen on the visual display. This technique can also be used to ensure a gradual change in the output force when objects must appear or change instantaneously in the environment.

Examples

This techniques described have been used to model a variety of virtual environments. A couple of examples are shown if Figure 2. Figure 2(a) shows a two degree of freedom windmill. Both the blades and the cap can be manipulated by the user. Figure 2(b) illustrated a rocket amusement park ride with 6 d.o.f.. Each pod can be individually moved up and down and the entire ride spun by the user.

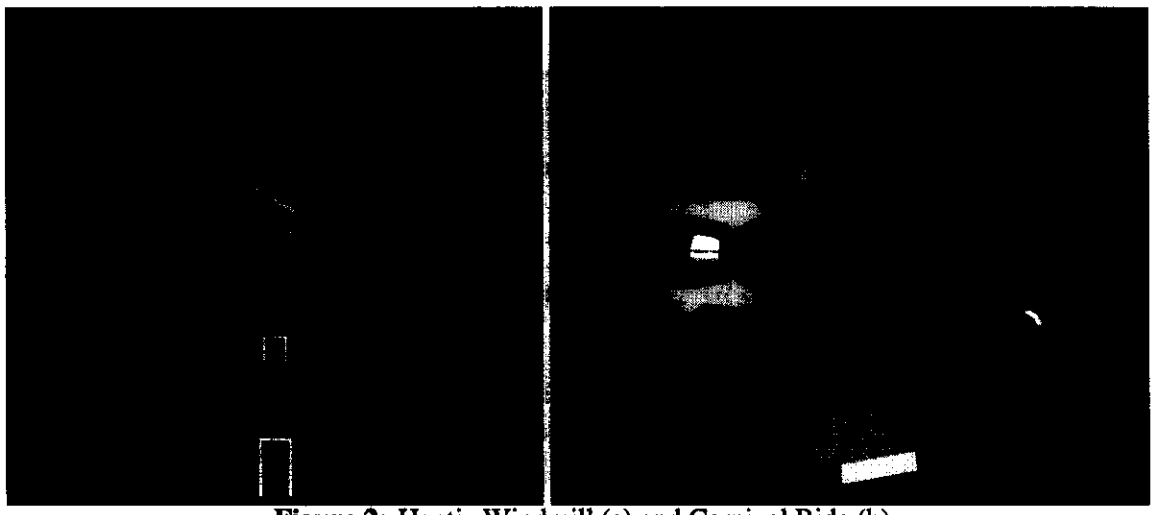


Figure 2: Haptic Windmill (a) and Carnival Ride (b)

References

[1] Craig, J. "Introduction to Robotics Mechanics and Control," *Addison-Wesley Pub. Co,* 1989.
[2] Featherstone, R.. *Robot Dynamics Algorithms.* Kluwer, 1987.
[3] Ruspini, D., Kolarov, K., and Khatib, O. "The Haptic Display of Complex Graphical Environments." SIGGRAPH 97 Proceedings, (August 1997), pp. 345-352.

EXTENDING THE GHOST® SDK SHAPE AND DYNAMIC CLASSES

Theodore Bardasz
SensAble Technologies
tbardasz@sensable.com

Overview

This paper presents an example of how to extend the GHOST SDK shape and dynamics classes. A pyramid and a dial with a user defined force-angular displacement curve is developed. The capability to extend GHOST's haptics classes is critical for application developers who need to model the haptics behaviors of application objects beyond the behaviors provided by GHOST. The ability to easily extend the GHOST SDK classes is critical to its utility, and acceptance, as a haptics renderer.

Background

The GHOST Software Developers Kit is designed to provide developers the ability to haptically render a scene. The goal of the rendering is to allow an end user to feel and interact with an application's objects in an intuitive manner.

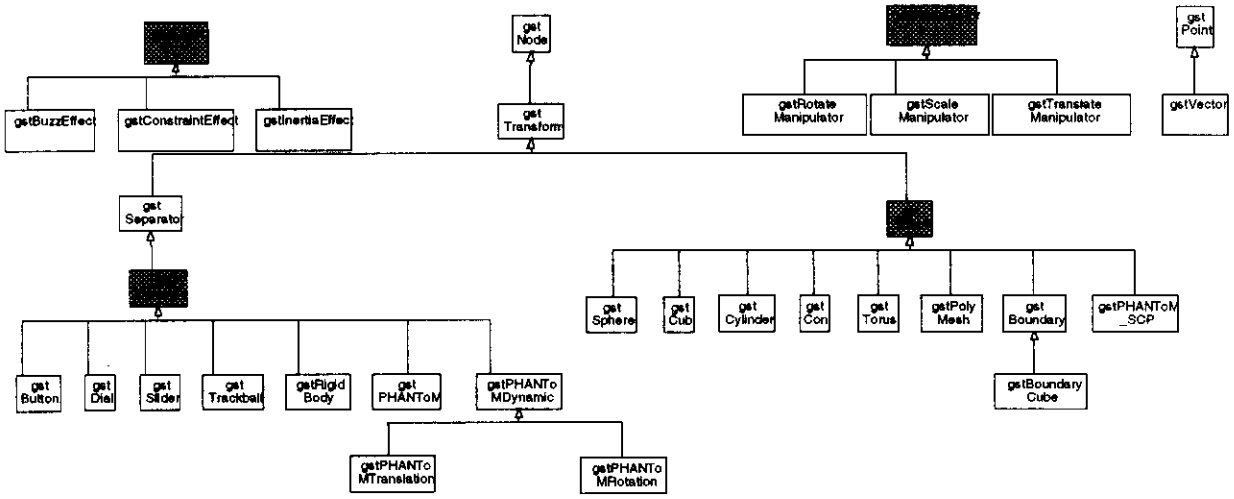
The main two data model components of the SDK are the Haptic Scene Graph and the Haptics classes. A developer haptically enables an application by building a scene graph structure populated with instances of the haptics classes. Two of the most predominant GHOST haptics classes are the Shape and Dynamics classes. The Shape class (gstShape) is used to model physical objects that can be positioned and oriented, but do not move when touched by the PHANToM. Examples of the shape classes are boxes, spheres, cylinders and triangular meshes. The Dynamic classes (gstDynamic) are classes that can be positioned and oriented and do move when touched by the PHANToM. Examples of the dynamics classes are buttons, dials and sliders.

The GHOST SDK provides a set of shape and dynamics objects that the developer can instance directly to populate the scene graph. However, it is likely that there will be cases where a developer would like to model objects that cannot be represented directly by the GHOST SDK classes, either for their complex shape or specific haptics behavior.

It was one of the major design goals of the GHOST SDK to allow the developer to subclass from the GHOST SDK object model to develop application specific haptics classes. The GHOST SDK design provides the developer definitive classes in the object model to subclass from as well as a well defined, finite set of virtual methods to overload for modifying the object's haptic behavior.

Figure 1 depicts the GHOST object model in Universal Modeling Language notation [Booch et al-97]. The classes in the data model from which the developer can begin subclassing are gstShape, gstDynamic, gstEffect and gstManipulator. They are depicted in gray in Figure 1.

Figure 1. The GHOST SDK Class Diagram



Adding a Pyramid and User-defined Dial

In this paper, we present two user-level extensions made to the GHOST SDK classes; a pyramid and a user defined dial. The pyramid is created by subclassing from the gstShape class. The new class, pyramid, provides the capability to define a haptically rendered pyramid by defining the base dimensions, width and depth, and the height. All surface properties are available for assignment, as with the other gstShape classes. When instantiated, the user can feel all five sides of the pyramid. The user defined dial is created from subclassing from the gstDial class. The new class, userDial, allows the developer to define their own force-displacement/velocity-damping characteristics for the dial.

Subclassing gstShape – A pyramid

To create a new subclass of the gstShape class, the two most important methods that need to be overloaded are the *intersection* and *collisionDetect* methods[SensAble–97a]. The *intersection* method determines an intersection point and normal on an object given the start and end point. The *collisionDetect* method is responsible for determining if the PHANToM has penetrated the object, and if so, provides a suggested new PHANToM position. The code for both of these overloaded methods is presented below.

The *pyramid::intersection* method cycles through each of the pyramid's faces to determine an intersection. As part of the constructor for the pyramid class, gstPlanes are defined for each of its faces . The *pointOfSegmentIntersection* method is a method of the gstPlane class [SensAble–97b] and determines if a line intersects the plane, and if so, returns the intersection point. The intersection point returned by *pointOfSegmentIntersection* is on an infinite plane, so the method *pyramid::pointOnFace* is used to determine if the point returned is actually on the trimmed plane of the pyramid. If the point is found to be on the face, the function returns TRUE along with the intersection point and the normal of the face intersected.

```
// Intersection method
// Finds the intersection and normal of the PHANToM with the pyramid
gstBoolean pyramid::intersection(const gstPoint &startPt_WC,
                                 const gstPoint &endPt_WC,
                                 gstPoint &intersectionPt_WC,
                                 gstVector &intersectionPtNormal_WC,
                                 void **userData){

        // Get the start and end position of the line segment against which
        // collision will be detected
        gstPoint P1 = fromWorld(startPt_WC);
        gstPoint P2 = fromWorld(endPt_WC);
        gstPoint intPoint;

        // Check if this line segment intersects any of the planes
        for (int i = 0; i < 5; i++) {
                if (planes[i]->pointOfSegmentIntersection(P1, P2, intPoint)) {
                        if (pointOnFace(i, intPoint)) {
                                intersectionPt_WC = toWorld(intPoint);
                                intersectionPtNormal_WC = toWorld(normals[i]);
                                return TRUE;
                        }
                }
        }
        return FALSE;
}
```

The *pyramid::collisionDetect* method also cycles through each of the pyramid's faces to determine an intersection. Again, *pointOfSegmentIntersection* and *pyramid::pointOnFace* is used to determine a face intersection point and normal. If the point is found to be on the face, the function calculates the projection point of the PHANToM onto the face. It then updates the state of the PHANToM to notify it that it is in contact with an object, and notifies the system of a collision via *addCollision* with the projected PHANToM point.

```
// collisionDetect method
// Returns a suggested SCP
// Given the intersection point, determine the new SCP for the PHANToM
gstBoolean pyramid::collisionDetect(gstPHANToM *phantom){
        gstPoint lastSCP, phantomPos;
        gstPoint newPHANToMPos;

        int inContact;
        gstPoint intPoint;

        inContact = getStateForPHANToM(phantom);

        if(!_touchableByPHANToM || _resetPHANToMContacts) {
                _resetPHANToMContacts = FALSE;
                inContact = FALSE;
                (void) updateStateForPHANToM(phantom, inContact);
                return inContact;
        }

        //Convert phantom data to local coordinate system
        phantom->getLastSCP_WC(lastSCP);
        lastSCP = fromWorld(lastSCP);
        phantom->getPosition_WC(phantomPos);
        phantomPos = fromWorld(phantomPos);

        // Check if the line segment intersects the plane
        // The plane methods gets the intersection point too.

        // Check if this line segment intersects any of the planes
        for (int i = 0; i < 5; i++) {
                if (planes[i]->pointOfSegmentIntersection(lastSCP, phantomPos, intPoint)) {
                        if (pointOnFace(i, intPoint)) {

                                //Project the current PHANToM position onto the face
                                newPHANToMPos = planes[i]->projectPoint(phantomPos);

                                inContact = TRUE;
                                (void) updateStateForPHANToM(phantom, inContact);
                                addCollision(phantom, newPHANToMPos, normals[i]);
                                return inContact;
                        }
                }
        }

        // If we didn't find an intersection, set the inContact state of the
        // PHANToM to false. Otherwise, the PHANToM will stay stuck to the surface.
        inContact = FALSE;
        (void) updateStateForPHANToM(phantom, inContact);
        return inContact;
}
```

There are two other notes of interest that should be made clear when subclassing to provide a new shape. The first is the overloading of the classes type methods. The methods that need to be overload are clearly defined in [SensAble-97a]. Second, it is important to, as accurately as possible, define the bounding sphere of the object. The bounding sphere is used to optimize the number of collision detections called on each of the haptic scene objects. Too small a bounding sphere will result in the PHANToM being inside of the object when a collisionDetect is triggered. Too large a bounding sphere and unnecessary collisionDetects will be called on the object.

Subclassing gstDynamic – A dial with a user defined force/displacement

A common request of the GHOST SDK is to describe sets of controls that may not have linear force/displacement curves. The GHOST SDK uses a linear force-displacement calculation for its gstDial class (actually force-angular displacement). In this example, the gstDial is subclassed to provide the developer the ability to define their own specific function for determining the force-angular displacement calculation. Once implemented, the developer can

use any function, set of functions or table lookups to determine the correct force to deliver to the user for a given angular displacement.

When subclassing a gstDynamic, the virtual method required for overloading is the *gstDynamic::updateDynamics* method. The *updateDynamics* method is used to calculate the new position of the dial given its position in the throw and any user-applied force. The normal procession of the code is to a) acquire the torque applied by the user, b) adjust the torque for spring forces related to the dial, c) adjust the torque for damping, and d) calculate angular acceleration, velocity and position by successive euler integrations.

For the implementation of userDial, subclassing took place from the gstDial. The code for *userDial::collisionDetect* is shown below.

```
// Overloaded updateDynamics method for a user defined dial.
// This method calls a user-defined function to determine adjustments to the torque applied to the dial from external forces
// (e.g. springs, dampers, magnetizm, etc.). The new position of the dial is then determined by euler integrations
void userDial::updateDynamics()
{
        gstPoint localSCP;
        gstVector radius; // vector from center of object to point of contact
        gstVector torque;
        double moment = mass;
        double deltaA;
        double section_tval = 0.0;

        double o_sectionTheta = fmod(orientation, sectionTheta); // orientation within sectionTheta

        // Find out the amount of torque the user is applying to the dial
        double t = reactionTorque[2];

        // If we are going in the counterclockwise direction
        // then the parameterization of the segment is typical, from 0-1 for 0 to sectionTheta
        // Else, if we are going clockwise, then the parameterization reversed, from 0 - 1 for SectionTheta to 0
        if (t >= 0)
                section_tval = 1 - o_sectionTheta/sectionTheta;
        else
                section_tval = o_sectionTheta/sectionTheta;

        // Call the function pointer of the user defined force function specified by force_func
        t += (*force_func)(t, section_tval, angularVel.z(), this);

        angularAccel.setz(t/moment);

        // Euler integration, vel=vel+accel*delta_t
        angularVel += deltaT*angularAccel; // angular velocity


        // Euler integration again, delta_x = vel*delta_t
        deltaA = (angularVel.z())*deltaT;
        orientation -= deltaA; // change orientation of dial

        // keep orientation between 0..2pi
        if (orientation < 0.0) orientation += 2*M_PI;
        else if (orientation >= 2*M_PI) orientation -= 2*M_PI;

        //deltaA *= 5.0;
        rotateDynamic(gstVector(0.0,0.0,1.0),deltaA); // rotate dial

        double tolerance = 0.01;
        currentNotch = (int) ((orientation/sectionTheta) + 0.5); // round to nearest notch
        if (currentNotch == number_notches) currentNotch = 0;


        if (fabs(currentNotch * sectionTheta - orientation) < tolerance
                && currentNotch != notch) {
         gstEvent newEvent = getEvent();
                (void) addEvent(newEvent);
        }
```

```
        gstDynamic::updateDynamics();
}
```

userDial::updateDynamics first determines the amount of torque applied by the user. It then parameterizes the position of the dial in its throw, respecting the direction of rotation. The user-defined force adjustment function is then called to calculate any adjustments that should be made to the torque from external forces. The adjustment is added to the user applied torque and the angular acceleration is then calculated. Successive euler integrations are then performed to determine the angular velocity and displacement. Finally, the dial is rotated accordingly.

A sample of a user-defined force-angular displacement function is given below, *cosine_func*. This function uses a half-cycle cosine curve to determine the resistant torque on the dial given its angular position between dial notches.

```
// User defined function for the determination of resistant force on a dial.
// A half-cycle cosine curve is used to determine the resistant force.  A normal drag equation is used.
double cosine_func(double user_torque, double orient_tval, double ang_veloc, gstDial *dial) {
        double torque_adjust = 0;
        double damp_adjust = 0.0;

        // Calculate the adjustment to the torque based on a cosine curve.
        torque_adjust = cos(orient_tval*M_PI) * dial->getK();

        // Determine the sign of the resistence depending on the direction the user is turning the dial
        if (user_torque >= 0)
                torque_adjust = -torque_adjust;

        // Independent of torque, the damping should always be in the opposite direction
        // of the velocity
        damp_adjust = -dial->getDamping() * ang_veloc;

        return torque_adjust + damp_adjust;
}
```

The only other note when implementing the updateDynamics function for userDial was to make sure to call *gstDynamic::updateDynamics* at the end of the overload *userDial::updateDynamics* method. This call ensures that all new values and states are propagated correctly.

Acknowledgements

I would like to thank Chris Tarr and Josh Handley, both of SensAble, for valuable, timely advice as I was putting together these examples.

Summary

The GHOST SDK was designed to provide for extension via inheritance. The points of inheritance in the GHOST object model are well defined. For each of these inheritance points, a well defined, finite set of methods are presented that allow the developer to provide application specific haptic behavior to the object.

In this paper, two user defined classes were built using GHOST. Both implementations were found to be fairly straightforward with little background in using GHOST. Both examples were completed in about one man-day per example.

It is extremely important for the GHOST SDK to provide a powerful and flexible means for allowing developers to define their own haptically enabled objects. This capability is provided for currently and should be expanded in the future to include both the power to define more radically different haptics object; both in terms of shape and haptics behavior.

References

[Booch et al-97] *Unified Modeling Language User's Guide*, Booch, Jacobson, I., G., Rumbaugh, J., Addison Wesley, 1997

[SensAble-97a] *GHOST Software Developer's Toolkit, Programmer's Guide Version 1.2*, October, 1997.
[SensAble-97b] *GHOST Software Developer's Toolkit, API Reference, Version 1.2*, October, 1997

Controlling Force Feedback Over a Network

Adam Seeger, Jun Chen, Russell M. Taylor II
Department of Computer Science
University of North Carolina, Chapel Hill
seeger@cs.unc.edu

Introduction

The work we discuss in this paper is part of the nanoManipulator project started by Dr. Russell Taylor at UNC Chapel Hill to develop a better interface for scanning probe microscopes. Scanning probe microscopes work by scanning a microscopic tip across a surface. Information about the interaction between the tip and the surface can be used to determine the height, friction, and many other properties of the surface. Our current system allows the user to see a 3D stereoscopic rendering of a surface with height and color mapped to various data sets. The user is also able to feel the shape of the surface using a PHANToM or other force feedback device. The nanoManipulator has allowed researchers in the physics department at UNC to perform qualitatively different experiments than were possible with previous interfaces [Taylor-94].

Recently we have switched to using GHOST to control our PHANToMs. Before GHOST, we used our own force feedback library called ARMlib. This software worked with the Argonne Remote Manipulator and the PHANToM and provided a way to control these devices over a network. Besides providing a network layer, ARMlib provided us with several other important features which we needed to reimplement in switching to GHOST. Although GHOST did not offer all the same features of ARMlib, those that it lacked were added very easily and with GHOST it is easier for us to work on high-level issues in our force feedback system without having to worry about how changes will affect the force update rate.

Our system consists of three separate programs: a microscope server, a graphics engine, and a PHANToM server. The graphics engine is a client of both the microscope server and the PHANToM server. It sends commands to control the microscope and receives data to display. The graphics engine receives position information from the PHANToM server and displays a graphical representation of the PHANToM stylus on top of a graphical representation of the surface. The graphics engine uses the position of the PHANToM on the surface to determine a surface approximation for that point. This surface approximation (or intermediate representation as it is called in [Adachi-95]) is sent back to the PHANToM server about 20 times per second. The PHANToM server then updates the haptic scene to the intermediate representation. The intermediate representation can be as simple as a plane; it is very easy to maintain high force update rates because few computations are required to calculate forces from a plane [Adachi-95]. In our system we allow for up to four planes to be used at one time on the force server with each being updated separately. We may need more than one plane to simulate sharp corners for example. However, we are currently using only a single plane.

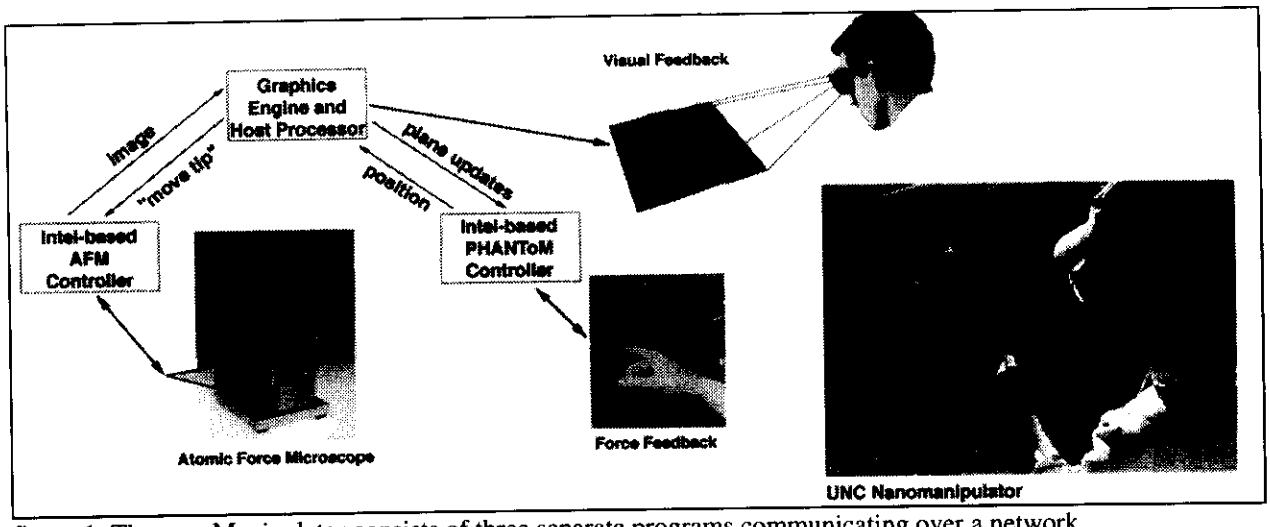


figure 1. The nanoManipulator consists of three separate programs communicating over a network.

Issues we encountered in switching from ARMlib to GHOST

Implementation of a Network Layer

In our department we have four graphics engines and four PHANToMs. While we have displays located next to each PHANToM, the graphics machines are usually in separate rooms. As a result we need to have our PHANToMs communicate with the graphics application over a network. In the past, ARMlib provided this capability in a force server application running on an Intel-based PC which ran the 1000Hz force update loop based on 20Hz plane updates sent from the graphics engine over the network. Another advantage of having a separate force server and graphics engine is that it allows us to switch between different types of graphics machines more easily. For example, we will soon be adding an HP PixelFlow machine as our graphics engine.

In keeping with our philosophy of having peripheral devices controlled remotely over a network, we have developed a library called VRPN (for Virtual Reality Peripheral Network) which provides a unified interface for all of our force feedback, tracking, and other input devices. To provide a network layer for GHOST we extended VRPN by creating a vrpn_PHANToM class which inherits from the base classes vrpn_Tracker, vrpn_ForceDevice, and vrpn_Button. Each of the base classes handles sending and receiving its own network messages. For example, in our force server we have an instance of vrpn_ForceDevice that receives plane update messages from the graphics engine. On the graphics engine side there is an instance of vrpn_ForceDevice_Remote which looks basically the same as vrpn_ForceDevice except that when we tell it that the plane has changed it sends the plane over the network to the force server instead of telling the PHANToM that the haptic scene has changed. On the server side, we have a callback function specific to the vrpn_PHANToM which gets called whenever a plane update is received by the vrpn_ForceDevice. This callback function then calls the GHOST functions to modify the plane in the haptic scene.

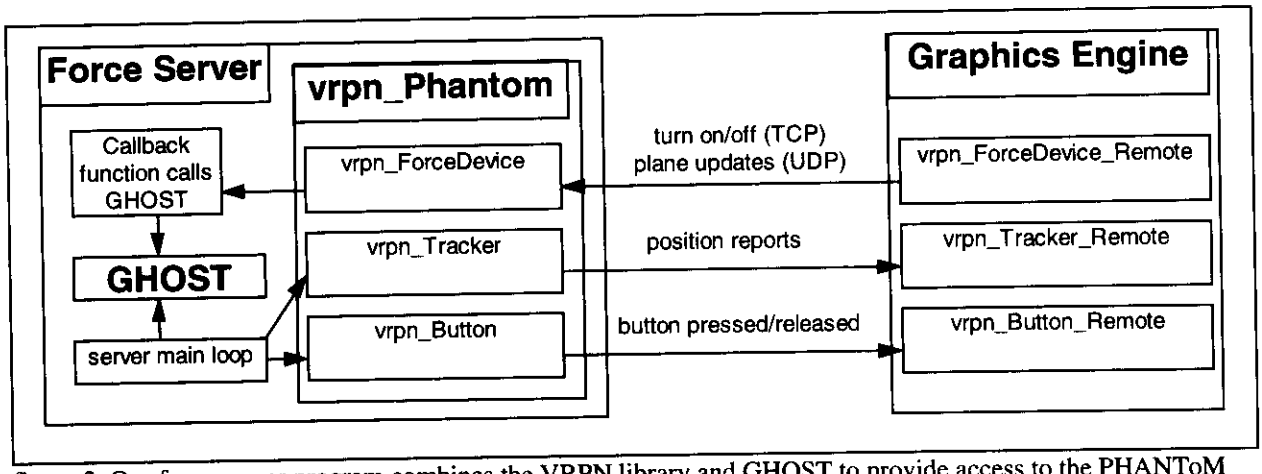


figure 2. Our force server program combines the VRPN library and GHOST to provide access to the PHANToM from a remote graphics application.

One important aspect of how the vrpn_ForceDevice_Remote sends update messages over the network is that it uses different protocols for different types of messages. Messages that tell the force server to start and stop generating forces need to be sent reliably. Messages that simply update the plane approximation should be sent unreliably because they are sent 15-30 times per second and it doesn't make sense to keep sending a message if it gets dropped. We use TCP (reliable) for start/stop messages and UDP (unreliable) for plane update messages. This works quite well over the ethernet in our department at UNC.

Other features we added to our GHOST-based force server

As mentioned earlier, our system computes plane updates at about 20Hz to simulate a more complicated surface stored and displayed by the graphics engine. When the user moves the position of the PHANToM slowly this approximation works very well but as the velocity increases the user starts to feel more discontinuity as the plane changes more between updates. This is especially noticeable when the virtual stylus position runs into the steep side of a bump. To fix this problem, we implemented a recovery procedure that updates the plane more gradually by performing a kind of interpolation between the previous plane and the new plane [Mark-96]. This algorithm takes the new plane and adjusts its height until the stylus is at the same depth in the surface as it was in the last force update loop iteration. Then, with each force update loop iteration it increments the height slightly. The recovery time is set as the number of loop iterations over which the interpolation takes place and can be set remotely on the graphics engine. We found that setting the recovery time to about one fourth of the maximum time between plane updates worked best. This seems reasonable because the time between updates as measured in units of the GHOST force update loop varied mostly between 10 and 40. If the recovery time much longer than it should be and we are colliding with an inclined plane forming the side of a bump then with each plane update, the difference between the perceived depth and the actual depth in the surface grows until the height increment is approximately the same as the difference in height between successive planes and we end up feeling the same discontinuity as without recovery. At most the recovery time should be set to the average time between plane updates.

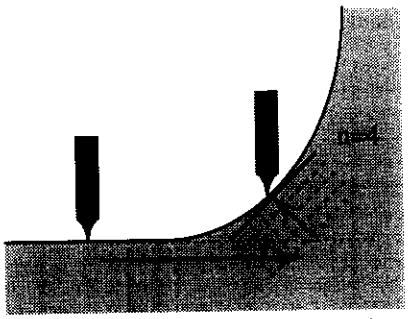


figure 3. Our recovery algorithm gradually updates to a new plane over n force update cycles.

Because GHOST did not include adhesion as a surface force we needed to add this separately [Chen-97]. Although we haven't finished yet, we are implementing adhesion as a gstEffect which generates the appropriate adhesion forces for the current plane. It would be better to implement this as a force like friction and the normal force but unfortunately GHOST does not allow us to modify surface interactions in this way. Another thing we would like to do is calibrate the PHANToM using GHOST. With our nanoWorkbench which is a stereoscopic display with a head tracker and a 'T' model PHANToM we can calibrate the PHANToM precisely enough so that the virtual tip displayed appears to be connected to the end of the actual PHANToM stylus. It is not possible to use the reset position required by GHOST because a calibration rig in this location would be in the way of the display for the nanoWorkbench. With ARMlib we used a calibration mount to the side of the display and the offset of this mount from the neutral position to calibrate the PHANToM. With version 1.1 of GHOST we should be able to do the same thing by putting the appropriate reset angles in the initialization file.

Acknowledgments:

I would like to thank Dr. Russell Taylor for inviting me to join the nanoManipulator project and for taking the time to explain how the nanoManipulator works. I would also like to thank Jun Chen for getting me started on force feedback and introducing me to the PHANToM.

References:

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

[Chen-97] Chen, Jun, Christopher DiMattia, Russell M. Taylor, Mike Falvo, Pichet Thiansathaporn, Richard Superfine, "Sticking to the Point: A Friction and Adhesion Model for Simulated Surfaces," to be presented at Symposium on Haptic Interfaces to VE and TS, Dallas Nov. 15-Nov. 21, 1997.

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

[Taylor-94] Taylor, Russell M., "The Nanomanipulator: A Virtual-Reality Interface to a Scanning Tunneling Microscope," Doctor of Philosophy, UNC Chapel Hill, 1994.

Realistic Haptic Rendering of Flexible Objects

F. De Angelis, M. Bordegoni, G. Frugoli and C. Rizzi

KAEMaRT Group
Dipartimento di Ingegneria Industriale
Università di Parma - Viale delle Scienze
43100 PARMA Italy
e-mail [fda,mb,frugoli,rizzi]@ied.unipr.it

ABSTRACT

The research work described in this paper aims at performing realistic haptic rendering of flexible objects by integrating the Sensable haptic feedback technology with the non-rigid material simulator developed at University of Parma.

Keywords: haptic rendering, non-rigid materials.

1. INTRODUCTION

The research described in this paper is carried out by the KAEMaRT (Knowledge Aided Engineering & Manufacturing and Related Technologies) Group at University of Parma. Of particular interest is the research work carried out on flexible material modeling and their behavior simulation. The research interest on this topic is not purely academic, but is also supported and pushed by some industrial needs related to the automation of industrial processes dealing with the handling and manipulation of flexible products, such as meat, fabrics, wires and cables, etc. The KAEMaRT Group is involved in several projects facing the issue of the simulation of the behavior of non-rigid objects. These projects are funded under the framework of the Brite/Euram Programme supported by the European Union.

It has been developed an environment (SoftWorld) for modeling and simulating flexible materials. The environment allows us to describe flexible objects through a discrete model, known as particle-based model. An object is described by a set of particles distributed in space and related by means of internal forces and constraints. Depending on the type of material to be simulated, different discretization rules and force distribution are used. The environment also allows us to define and simulate some external forces, constraints and obstacles interfering with the objects. In this way, we are able to simulate the behavior of a flexible object when grasped and manipulated.

Presently, the only interaction between the user and the simulated object is purely visual. What would be interesting and crucial for several applications is to provide the user with a real feeling of the object. Providing the simulated object with haptic feedback congruent to the material it is made of, would allow the user to grasp and manipulate it in a way close to reality. Hence we mean to integrate our flexible object simulation software with two PHANToMs.

2. PARTICLE-BASED SIMULATOR

2.1 Usage advantages

The current approach for modeling the objects to be haptically rendered could be defined "primitive-based", which means the scene has to be built by a set of pre-defined primitives, each one having its pre-defined behavior depending on some high-level parameters, like mass, damping, friction, stiffness etc.
Our simulator instead uses a more general approach, non primitive-based, where each object is modeled by its approximation in particles, allowing any object to be modeled. The object properties are low-level defined by the description of forces and constraints among the particles, the mass of each particle, and so on; for this reason non-homogeneous materials (having non-uniform density distribution) can be modeled too.
The interaction between an object and the surrounding environement is always computed by its current, possibly deformed, geometry; the visualization, too, is always congruent with the actual object shape, as it is obtained from the current particles position.

So, the usage of our non-rigid material simulator enables to perform accurate, realistic simulation of complex scenes including free-form objects made by non-homogeneous materials.

2.2 Integration with PHANToMs

The basic idea of this work is to use the non-rigid material simulator to compute realistically both the forces generated by flexible objects against finger pressure and the modified objects shape, to perform an accurate haptic rendering together with a congruent visualization.

The manipulated object is modeled by an approximation of it by particles, and the current PHANToM position is used to impose modifications on its shape where an interference is detected. The simulator computes how the deformation propagates in the object, and the resulting generated forces. At this point it is possible to display the poligonal mesh describing the new object shape and to send the force vector to the PHANToM.

3. REAL-TIME SIMULATION

The problem in implementing this idea is that, as known, haptic rendering requires updating the force vector exerted by each PHANToM every thousandth of a second, while the computational cost of simulating a real object is greater by some orders of magnitude.

In the following section we will give a more precise estimation of the computational power required for such a simulation.

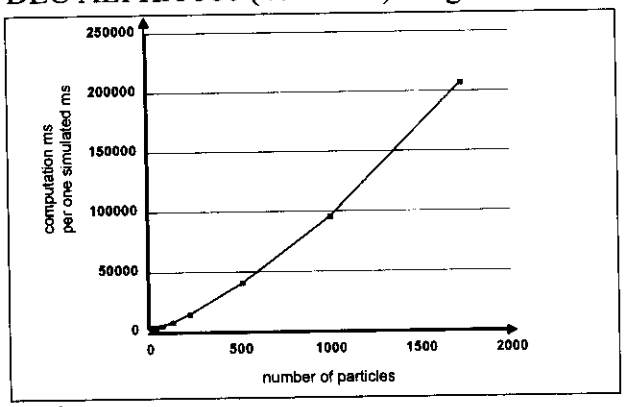
3.1 Hardware requirements

The computational cost of simulating a flexible object by our simulator depends on the number of particles describing the object, but also on other variables, like the constraints among the particles, the type and number of internal forces used and the stiffness of the material, just to mention some. While this implies that it is not appropriate in general to describe the computational cost of the simulator as an O(n) function where n is the number of particles, it is

however possible to get a qualitative idea of that function by performing some measurements on a test case where all variables except n are kept as much as possible constant.

The test case chosen due to its simplicity, is a cube where the particles are uniformily distributed in space (a 3D grid of equidistant particles). Starting from the simpler case, modeled by 8 particles in the cube corners, the number of particles is increased by adding a complete layer of particles on the previous one, to obtain a bigger cube. Laws describing particles interaction are defined only between neighbour particles. Thus while the number of particles increases the uniformity of the model is preserved and the other variables remain constant.

Running such a test on a DEC ALPHA 500 (400 MHz) we got the following results:



There are two main observations:

a) The function is not linear as the computational time increases more than the number of particles. While the validity of this data is restricted to the considered test case, we expect the behavior to be qualitatively the same even using a different test case.
b) The computational cost is dramatically high.

However, the test was performed on a quite accurate model, having a "high" number of forces and constraints defined among the particles, which increases considerably the computational time. A different test case, corresponding to a highly simplified model made of just four particles, did run in 0.4 ms per step, thus allowing real-time simulation and in fact it was possible to haptically render that model on a double PPro-200 MHz. Such a model doesn't correspond to any real object, but the fact the application works is an interesting result as it demonstrates that, leaving the number of particles apart, the real-time simulation with our approach is feasible. Given enough computational power, it will be possible to perform real-time flexible objects haptic rendering with a certain degree of accuracy.

Nevertheless, models of real objects are usually described by hundreds or thousands of particles and the estimated computational time for such values of n, even for the simplified model, is greater than 100 ms, while the time we can spend is just one thousandth of a second.
So we can roughly state that the computational power we need for performing the real-time simulation of real objects is, in the best case, at least a hundred times more than the one we have available now, while the requirements of a highly accurate simulation would be greater by 2 or 3 orders of magnitude than that.

As Personal Computer speed only doubles every 18 months or so, the required computational power won't be available for several years.

4. OFF-LINE SIMULATION

While waiting the required computational power to be available, we are working on some kind of non-real-time integration between the non-rigid material simulator and the haptic application controlling the PHANToM devices.
Some object geometries have the property that, if the object material is homogeneous or at least the material density distribution meets proper criterias, the object behavior is the same or approximately the same when the same deformation is imposed in different points of its surface (an example of such a geometry is the sphere). In other words, such objects show a behavior which is independent from the surface point where the deformation is applied (at least inside certain regions of them).
In these cases, the behavior of the object can be simulated off-line, stored in some form, and used later during the haptic rendering without the need of computing it real-time.

To be suitable to our objectives, the simulated "object behavior" must provide at least the following information:

- the force generated by the object as a reaction to the deformation, to be used for the haptic rendering;
- the geometry of the deformed object surface, to be used for the visual rendering.

Work is in progress to extract this data from particular object geometries showing the above properties, like the sphere, the halfspace (which cannot be a real object, but fits very well the specifications), the box (under the assumption to touch it "enough" far from the edges) and the cylinder (under the assumption to touch it "enough" far from the bases).

A number of approximations and simplifications has to be done to make the data resulting from the simulation usable run time, thus renouncing to many of the non-rigid material simulator features, like history-dependent behavior, friction, damping; nevertheless we mean to obtain a haptic rendering more realistic than the one derived by describing the objects by a primitive-based model, and also to enhance the visual feedback by displaying the deformations taking place in the objects.

5. CONCLUSIONS

The research work described in this paper is ongoing. The real-time simulation of non-rigid objects is appealing but the required computational power is not avalaible yet in everyday computers. In some special cases it is however possible, under a number of assumptions and approximations, to haptically render objects using data precomputed off-line by a non-rigid material simulator, still obtaining results which are more realistic than the ones got modeling the object behavior by a primitive-based approach.

Haptic Rendering of Visco-Elastic and Plastic Surfaces

Chris Tarr and J. Kenneth Salisbury
MIT Artificial Intelligence Laboratory and SensAble Technologies, Inc.
ctarr@ai.mit.edu

Abstract: *Few methods have been developed to haptically interact with deformable three-dimensional surfaces in real time. This body of work develops a way to convert existing surface representations into visco-elastically and/or plastically deformable surfaces capable of interacting with a volume tool commanded by the PHANToM® haptic interface. Using force feedback to the PHANToM interface, the user can take advantage of the force information to sense the compliant properties of the model and guide intended deformations to the model.*

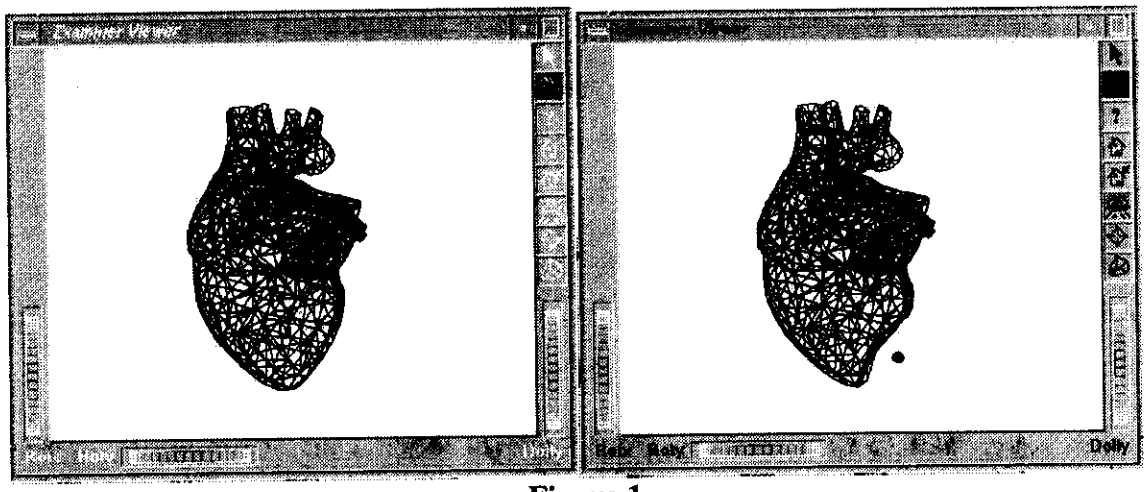


Figure 1.

1 Introduction

The goal of this work has been to develop a system that enables haptic interaction with surfaces exhibiting visco-elastic and plastic surface deformations. It extends and improves Nitish Swarup's Interactive Deformable Media Simulator, IDMS [Swarup-95]. IDMS has been reworked to allow the simulation to handle large surface models and is now able to convert VRML models into for use in the system. This work has led to further innovations that should extend the system to handle plastic deformations that can be used to continually alter the geometry of the model. This new mode of interaction has applications far beyond the originally intended medical simulations. Plastic deformations could add a new robust metaphor to the already large field of model design that includes animation, industrial design and CAD/CAM industries.

1.1 Related Work

Most of the previous results on modeling deformable surfaces has been without the use of force feedback and more often then not, is too complex to perform in real time, hence the lack of force feedback. Several of these include local deformations of solid geometries by Barr [Barr-84], the volumetric sculpting work of Wang and Kaufman [Wang-95], Terzopoulos' [Terzopoulos-87] work with deformations using elasticity theory. Recently, haptic approaches to deforming surfaces have surfaced each with a different approach and goal in mind. Tom Anderson's [Anderson-96] work avoids displaying a surface contact point, commonly referred to as the God object [Zilles-95], by altering the visual model to bend when the PHANToM® enters the object. This is an efficient approach for simulating a homogeneous surface compliance since each point on the surface deforms in the same manner. However, this method is limited to point contact of the PHANToM® with the surface. Rick Avila has also developed a system to deform surfaces [Avila-96], in his case solids, using a volumetric approach. Avila's system deforms the volume through a carving metaphor. This is similar to a plastic surface deformation. Unfortunately, the haptic

model of the tool interaction is based on a stateless potential method. This limits the tool to very simple geometries and can allow the tool to pop through the surfaces. In addition, the intersection and deformation calculations are still too complex to allow the model to be deformed very quickly or with large tools. Finally, Swarup's Interactive Deformable Media Simulator (IDMS), the starting point for my research, uses a finite element approach to model a visco-elastically deformable surface interacting with a spherical tool commanded by the PHANToM® device. I believe this approach was a step in the right direction, but Swarup's system was hard to build useful surfaces with, had unacceptable limits on surface detail, and allowed the tool to slip through the surface when pushed too hard by the tool. I set out to solve these problems and have come across a method to handle some interesting plastic surface deformations as well. In the following sections I will briefly discuss Swarup's IDMS system and then go on to discuss my work simulating deformable surfaces.

1.2 IDMS

IDMS defines a two layer tetrahedral mesh of nodes interconnected by spring/dashpot elements as shown in figure 2. In addition, each surface node is connected to a grounded home position by a spring/dashpot element. A discrete simulation model (DSM) is used to sum the forces acting on the mesh nodes and integrate the state of the mesh at each time step. Intersections with the spherical tool, the master interaction object (MIO), is done through simple point intersections with a sphere. Nodes in the mesh that penetrate into the sphere have forces, proportional to their penetration depths, imparted on them. This compels the penetrating nodes outside of the MIO and the sum of these forces is imparted back onto the PHANToM® interface. Adding these interaction forces into the DSM causes the mesh to deform in a realistic visco-elastic manner given that deformations are small relative to the spatial density of the tetrahedral mesh.

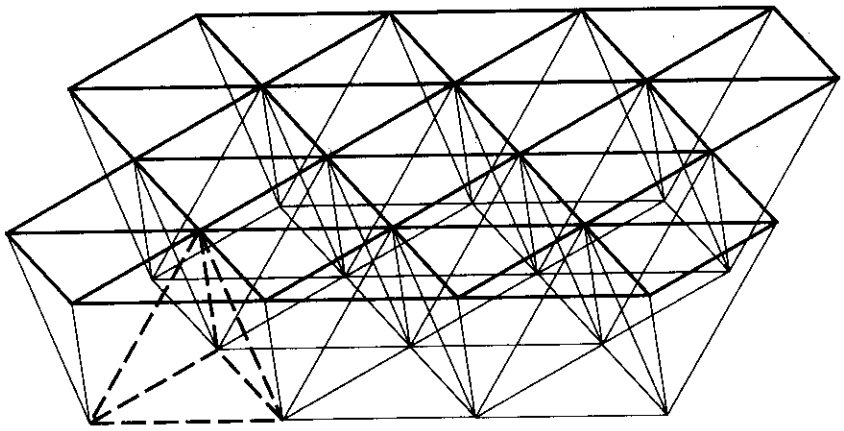


Figure 2.

2 Performance

My first modifications to IDMS were to improve the performance of existing operations. Two of the more time consuming steps of the algorithm were searching for node penetrations with the MIO and updating the state of each node in the mesh during each time step. My first optimization was to use an octree to partition the nodes of the mesh, which limits the number of nodes searched for penetrations with the MIO. This was a suggestion of Swarup's that had not been implemented. With the octree, the penetration calculations were no longer limited by the size of the mesh. The second optimization was to better limit the nodes updated in each time step. Swarup's *haptic windowing* method left nodes stuck in unstable states. I decided to limit the maximum number of nodes updated in any time step to a constant. Only nodes previously updated that still had significant velocity remained on the list to update. New nodes could then be put on the list to update if they had sufficient force exerted on them and only if there was less then the maximum number of nodes allowed on the list to update already. This made the mesh more stable and limited the maximum CPU usage per time step. As a last optimization, I limited calculations of forces on

nodes from neighboring spring/dashpot elements to a slower rate. This eliminated quite a few calculations, since the number of these connections is large with respect to the penetrating nodes in the MIO and forces from the home springs of each node.

3 Surface Model

3.1 Generalized Topology

One of the most limiting factors about IDMS was the 2 layer tetrahedral mesh required to make up the surface. By forcing the nodes to be arranged in this way made it very hard to build usable surfaces and even harder to convert existing surface models to an acceptable format for IDMS. The first problem with this was the two layer structure. Swarup designed the mesh this way so that the second layer would distribute forces realistically according to existing finite element modeling techniques. This second layer was really only useful so long as deformations didn't go much deeper into the surface then the second layer of nodes. Unfortunately, in order to bring the tool size down and have a detailed surface, the spatial density of the surface nodes must be large. My assumption has been that if the spatial density is large then the second layer becomes less useful and can be eliminated. In addition, I have made the assumption that the first layer hexagonal structure can also be violated so long as the spatial density of the nodes remains fairly homogeneous and neighboring element connections are maintained.

3.2 Converting Existing Surface Models

With these new assumptions it became possible now to load in an existing polygonal mesh, use the vertices as nodes, and use the edges as connecting elements for the mesh. The problem then was that most available polygonal surface representations do not optimize the model for homogeneous spatial density of vertices. Instead, the models usually have large polygons with sparse vertices on low curvature sections. These large gaps in the model would allow the MIO to fall through. Also, large variations in the node density caused the surface compliance to change and could cause the gain to exceed the PHANToM® interface's limit. To solve the later problem I would alter the spring/dashpot values according to the local density to normalize the surface compliance for the MIO. But this still didn't fill in the large open spaces in the model and in turn limited the useful size of the MIO.

To resolve this, I've designed a method to remesh the polygonal model optimizing the spatial density of the vertices. Thus, the triangles are divided or removed intelligently to make the density of the vertices roughly constant while maintaining the geometry of the original surface. This allows me to preprocess the model for simulation and adjust the level of the detail according to the size of the MIO.

3.3 Dynamic Remeshing

There was still the problem of slipping through the surface mesh with the MIO when pushing too hard against the surface. This happens when the nodes of the mesh are pushed apart by the MIO creating a gap in the surface large enough for the MIO to pass through. Since the connecting element springs cannot be made infinitely stiff, some method needed to be added to prevent holes in the mesh. The solution was fairly straightforward given my previous remeshing algorithm. The idea is to apply the techniques of dividing and collapsing triangles on the fly. As the mesh is deformed, gaps are filled in with new nodes by dividing the triangles. As nodes are pushed close together, the nodes are collapsed to one node and the appropriate triangles are removed from the mesh.

4 Tool Interaction

Every haptic rendering technique involving a point or points penetrating a volume or surface uses a previous state or stateless method to calculate forces of the tool interacting with a surface or volume. In the previous state method past state information is used to determine the current state of the tool, surface, and resulting forces. Craig Zilles's God object method is a typical example of a previous state method [Zilles-95]. In a stateless method the tool penetrates the surface and the new state of the tool, surface and contact forces are calculated based only on the current state of the system. Note that a stateless method can be implemented for any volume for which, at any interior location, a vector can be given pointing to the

closest surface point proportional in length to the penetration depth. This approach is typically much faster than previous state methods, but can also produce unwanted results on thin sheets and at corners. This method and these problems are typified in Thomas Massie's BS Thesis [Massie-93]. The MIO and surface mesh also use a stateless method, but in the opposite manner. In this case the tool is the penetration volume and the nodes of the mesh can each generate forces based on penetrations into the tool.

4.1 Visco-Elastic Interaction

Swarup's approach was to impart forces on the mesh nodes based on their penetration depths into the sphere. These forces compel the nodes toward the surface of the MIO and, when summed, provide the reaction force sent to the PHANToM® interface. Since only forces were imparted on the nodes, it was actually possible to pass through the surface just by pushing hard enough. Ideally, a infinitely stiff tool would give only the compliance of the surface mesh and would not allow the tool to pass through unless the nodes spread apart. Unfortunately, this also creates an unstable system. As a result, I decided to model the MIO as two spheres, one as an infinitely hard sphere slightly smaller then the original sphere is centered inside the original compliant sphere. Nodes penetrating the outside layer of the sphere receive forces to compel them outside of the surface. When nodes penetrate the smaller inside sphere the nodes are forced to the outside of the small inner sphere immediately and are then compelled outside of the larger sphere. In addition, the resultant force back to the PHANToM® interface is computed from the resulting internal forces of the mesh normal to the tools surface. This actually works pretty well and only allows the sphere to pass through the surface when the nodes spread apart. Thus, when I add the dynamic remeshing, the tool should be prevented from passing through the surface.

4.2 Plastic Interaction

Plastic interactions present the problem of permanently altering the geometry of the surface. There are several ways of doing this based on what has already been done for visco-elastic surfaces and also a new method that does away with the visco-elastic behavior all together. The most straightforward is to freeze the deformed state of a visco-elastic mesh by reseting the nominal positions of all moved mesh nodes. For example, I could press a key or a button when I have deformed the surface to a desired state that I want to keep. Although this isn't a real plastic deformation, it may actually be more desirable in some instances since the user can preview deformations before deciding to keep one. This method is also simple to implement into the previously defined system.

A second approach is to replace the grounded home position of each surface node with a position that moves when the force of the spring/dashpot element exceeds a threshold. The feel desired for the deformation can determine the way that home position moves after the force threshold is exceeded. For example, I may desire a coulomb friction force response to the deforming surface. This would require the nodes to move with this coulomb model. Many other models are possible that could give a desirable effect.

The problem with plastic deformations is that time is needed to both update the dynamic state of the surface mesh and to calculate the permanent deformations. In fact, plastic deformations should be possible without having to perform a DSM of the surface mesh. Simply moving nodes in response to the tool pushing on the surface should suffice if done properly. By adding the dynamic remeshing algorithm, additional surface area, created by the deformation action, can be filled in with more nodes to preserve the spatial density of the mesh. This combination should allow the surface to be deformed without limit.

I am currently developing a hybrid tool interaction metaphor in order to allow a volumetric tool to deform the surface as described above. These ideas are still being worked out, therefore I will only give a rough outline of this method and hope to show successful results in the near future. The most important concept of this method is the use of a combination of stateless and past state algorithms to support the tool interaction. I intend to use a combination of a stateless method to quickly find the points of contact on the tool and also a God object method to keep the tool on the surface while making contact. This combination of the speed of a stateless method and the past state information of the God object method should create realistic surface interaction with an arbitrary tool shape. Instead of attempting to move the tool in one step to a goal location, I am limiting the maximum step size that the tool can move between intersection calculations to eliminate the problems of stateless methods. Each servoloop, as many steps will be taken as

allowed by time to move the tool to the goal location. This allows the tool to move at the maximum speed allowed by the CPU and scene complexity. For example, in a sparse area the tool should move unconstrained, but in a complex area or with a large tool the simulation will slow the tool down to keep up with the calculations. This will hopefully ensure that the stateless method will guarantee that penetrations by any surface nodes are resolved to the correct point on the surface and maintain stability at all times. The tool can now be considered a God object since the tool is no longer coincident with the PHANToM® position at all times. Using the traditional spring/dashpot element connecting the PHANToM® interface position and the God object, I can then calculate forces to be sent to the PHANToM® device. Finally, since the tool can be kept on the surface at all times, deformations should be straightforward. If the reaction force to the PHANToM® device exceeds a threshold then the nodes on the surface preventing the tool from penetrating can be moved toward the PHANToM® interface position by some prescribed amount. One should note that this should allow not only plastic deformations, but also rigid interaction with an arbitrary volume tool. Both of these are interactions that have not been exploited as of yet.

5 Results

Currently, I have implemented the spatial partitioning, improved haptic windowing, the visco-elastic tool interactions, and a VRML converter. Using these optimizations on a heart model of around 2000 nodes, I was able to simulate in excess of a couple hundred nodes on a dual pentium 200Mhz Intergraph machine. This is shown in figure 1. Although the mesh was not optimally meshed, the results were very pleasing. The next step is to implement the remeshing algorithm. This is now in the final stages and is showing promising results already. I hope to show these results when presenting this paper. Afterwards, the plastic deformations will become possible. Most of this work has been done in my spare time over the past 2 years and has been put on extended hold on several occasions. Fortunately, the remaining work should be done by the end of the Fall '97 semester upon completion of my Bachelor's thesis at MIT.

6 Acknowledgements

Thanks to Nitish Swarup for his significant work that led to my own discoveries, Ken Salisbury for his constant support, insightful ideas and undying friendship, and to SensAble for a job that allows me to help shape the world of haptics.

7 References

Tom Anderson, A virtual universe utilizing haptic display, PHANToM® Users Group, September 1996.

Ricardo S. Avila and Lisa M. Sobierajski, A haptic interaction method for volume visualization, IEEE Visualization, pp.197-204, 1996.

Alan H. Barr. Global and local deformations of solid primitives. *Computer Graphics*, 18(3), July 1984.

Thomas Massie. Design of a three degree of freedom force-reflecting haptic interface. Bachelor's thesis, Department of Electrical Engineering and Computer Science, MIT, 1993.

Nitish Swarup, Haptic interaction with deformable objects using real-time dynamic simulation. Master's thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, September 1995.

D. Terzopoulos et al. Elastically deformable models. *Computer Graphics, Preceedings of SIGGRAPH*, 21(4):205-214, July 1987.

Sidney W. Wang and Arie E. Kaufman, "Volume Sculpting," Proceedings of the 1995 Symposium on Interactive 3D Graphics, pp.151-156, April 1995.

Craig Zilles. Haptic rendering with the toolhandle haptic interface. Master's thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, May 1995.

Hot and Cold Running VR: adding thermal stimuli to the haptic experience

Mark P. Ottensmeyer and J. Kenneth Salisbury
Haptics Group, Artificial Intelligence Laboratory, MIT, Cambridge, MA
canuck@ai.mit.edu

Abstract:

A device that can present thermal stimuli to a virtual environment (VE) user has been developed for use with the PHANToM Haptic Interface. Using a thermo-electric heat pump and a water cooled heat sink, it can achieve heating and cooling rates of +11°C/s and -4.5°C/s, respectively, and can display temperatures over the entire comfort range of human sensation (5 to 45°C). The ThermoStylus can display the thermal characteristics of a VE, or represent any other scalar data by substitution (e.g. pressure, concentration, etc.). Applications include improved medical simulation, increased VE realism and improved physical simulations (e.g. thermo-fluid systems with flow and thermal variations). This stylus-style interface is both familiar to PHANToM users, and intrinsically safe. Water cooling prevents build-up of heat in the heat sink, and should a failure occur, the user can easily release the stylus. Current performance and future developments, including the determination of real thermal transients, are discussed.

Introduction:

In presenting haptic feedback to VE and teleoperator users, some areas have been well researched, including vibratory displays [Kontarinis and Howe–95, Caldwell et al.–96], tactile arrays [Kontarinis and Howe–93], and force feedback such as that generated by the PHANToM [Salisbury and Srinivasan–97]. One haptic mode that has not been extensively examined is displaying thermal information to VE users. Thermal stimuli provide information about the material being touched (via thermal conductivity), warnings against prolonged contact with very hot or cold objects, and can be used to present other scalar information through mapping to temperature. Potential applications of augmenting haptic feedback with thermal stimuli include: more realistic medical simulations (e.g. warmth due to fever); increased VE realism, as thermal sensation is a natural component of touch; or physical simulations such as thermo-fluid systems where flow and thermal variations can be presented haptically.

While thermal stimulators have been used to study human perception of temperature, very few groups have used them for providing thermal feedback in either VEs or teleoperation. C/M Research [Zerkus et al.–93] has developed a system intended for use with VE displays, which can control eight thermal stimulators (thermodes). [Caldwell et al.–95] have developed an exo-skeleton and tactile feedback glove to control a teleoperator arm. The arm has a thermal sensor, and the glove feeds the detected temperature to the back surface of the user's index finger. One other notable example is that of [Dionisio–97], in which a virtual space is presented visually, but is augmented with air heater/blowers for gross thermal feedback, and a thermo-electric heat pump to display thermal information to the user's hand.

To date, there is no comparable device for use with the PHANToM, so one was developed in the AI Lab at MIT to explore its utility in providing multi-modal haptic feedback.

The neuronal basis for the sensation of thermal stimuli is less well understood than that for tactile stimuli. In the case of mechano-receptors, a number of structures have been identified whose behavior corresponds with different regimes of tactile stimulation (e.g. pressure or vibration). In the thermal domain, it is recognized from experiment that there are two different sets of thermo-receptors--one set for detecting warmth, another for detecting cold. It has been suggested, based on response times, that the cold receptors are located approximately 0.15mm underneath the skin's surface, while the warm receptors are deeper, at 0.3mm, but no specific nerves have been unequivocally identified. There is also debate as to whether the thermo-receptors respond to changing temperatures (i.e. dT/dt), or to the temperature directly. The comfortable range of temperature sensation for humans is typically between 15°C and 48°C [Caldwell et al.–96]. Within the range 31°C to 36°C, in which the sensation of temperature fades as the skin adapts to the temperature, the JND (just noticeable difference) is rate dependent (e.g. when heating at 0.1°C/s, the JND is less than 0.5°C while at +0.02°C/s, the JND is a larger 2°C). Outside of this range, persistent heat or cold are perceived [Sherrick and Cholewiak–86].

In terms of stimulating the thermo-receptors, all forms of heat transfer have been used by various researchers, including: radiation (IR and microwave), liquid and air convection [Sherrick and Cholewiak–86, Dionisio–97], and various modes of conduction. An early attempt to standardize methods for determining responses to thermal stimuli is the Minnesota Thermal Disks test [Dyck et al.–74]. In this test, disks of various materials are placed on the skin with equal force, to determine if a subject could distinguish between them. More amenable for use in VEs has been electrical heating (though cooling is a problem), thermodes with temperature-controlled water flowing through them, and thermo-electric heat pumps which use the Peltier effect [Sherrick and Cholewiak–86]. This last technique uses doped semi-conductor "dice" in series electrically, and in parallel thermally, to create a heat flow which is a function of the current passed through the device (compare with thermocouples, which create a voltage difference dependent on temperature difference). Peltier cells have been used by [Dionisio–97], [Zerkus et al.–93] and [Caldwell et al.–96] in their displays, and have been used in the device that will be described in the remainder of this paper.

ThermoStylus design:

The heart of the stylus-based thermal display is a sandwich of a water-cooled heat sink, the Peltier cell, and an aluminum "touch plate" with a temperature sensor (see figure 1). The user's finger is placed on top of the touch plate which, because of the high thermal conductivity of aluminum, is at a nearly uniform temperature. The temperature sensor is a commercially available RTD (resistive temperature device), whose resistance increases linearly with temperature. The Peltier cell pumps heat into or out of the touch plate depending on the direction of the current passed through it. The heat pumped by the cell is given by:

$$Q_{pump}=S \cdot T \cdot I$$

where S is the Seebeck coefficient (based on choice of semi-conductor and device geometry), T, the absolute temperature in Kelvins and I, the current. In addition to the heat pumped, the device also generates heat by the Joule effect ($Q=I^2R$) within itself, and there is a leakage heat flow due to the temperature difference across the device of the form:

$$Q_{leak}=k \cdot A \cdot dT / l$$

where k is the thermal conductivity of the cell, A is the cross sectional area and l is its thickness.

When the device is in finger-cooling mode, heat is drawn through and generated by the device, and must be rejected to a heat sink. This role is played by a water-cooled heat sink. In the current version of the device, an aquarium water pump is used to pump 130mL/min of ice-water through the sink, resulting in a thermal resistivity of approximately 0.78°C/W. In an earlier version, a CPU heat sink and cooling fan were used, but it was found that when in cooling mode for long periods, the sink could not dissipate heat quickly enough. Because of this, the sink would heat up, and when the current to the Peltier cell was shut off, this stored heat would flow back through the Peltier cell and could burn the user. With the water cooled sink (with a separate power source), the maximum temperature estimated for the sink under normal use is 22°C, well within the comfort range for human users. Ice water has been used for a coolant for convenience and also for safety–the ice water is not so cold that a user is in immediate jeopardy when the heat pump is shut off.

The prototype thermal display was a desktop version, to verify that the device and the control software functioned properly. The current version is in the form of a 7/8" stylus (see figure 2), designed so that the user places his or her index finger on the touch plate on the "top" surface of the stylus. It can be used by right or left handed users equally well. A safety feature inherent in the stylus design is that in the unlikely event of an over or under-temperature command, the user can simply release the stylus to prevent injury–something that would be more difficult with a thimble or glove interface.

Performance:

The ThermoStylus is limited in the maximum rate of heating and cooling by the maximum current that can be borne by the device. In heating mode, with a maximum of 8A, the device will increase temperature at a rate of approximately 11°C/s. In cooling mode, the maximum current is -4A, and the cooling rate is approximately -4.5°C/s. The rate of temperature change versus current is non-linear because of the quadratic Joule heating term–it aids in heating mode, but hinders the cooling action. The current-limit asymmetry is also a function of the quadratic term; if the current is increased past -4A, the increment in additional heat generated is larger than the increment in the heat pumped through the device, so the net

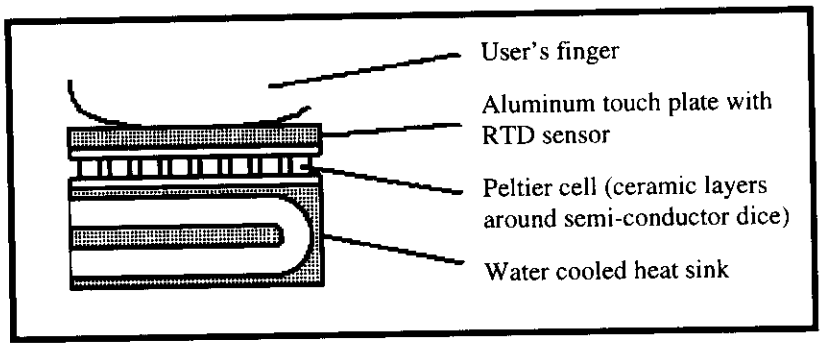


Figure 1: Sandwich of user's finger, touch plate, Peltier cell and water cooled heat sink

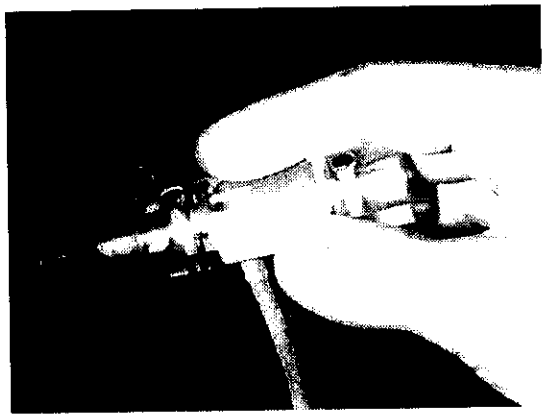


Figure 2: Holding onto the prototype ThermoStylus

heat pumped away from the finger decreases. The maximum cooling current could be increased if a colder coolant were employed, since then the leakage heat would be larger, aiding the cooling action of the pump (though presenting more of a danger of damage from cold).

To achieve rapid response with no steady state temperature error, a modified proportional plus integral controller is used. When the proportional gain alone is sufficient to reach the current limits, maximum current is applied, and the integral term is set to zero. Once the temperature error is small enough, the integral term is allowed to become active and brings the error to zero. Because of the rate limit on temperature change, the system bandwidth is dependent on the commanded amplitude, but is roughly 0.1 Hz.

To demonstrate the use of the device, a series of simple demonstration programs has been written for the ThermoStylus. Among these are simulations of: a human patient with a fever, whose forehead is maintained at 40°C (104°F); a thermo-fluid simulation which allows the user to feel the temperature in a flow field with viscous heating, and at the same time feel the drag force on a small probe in the flow; and a simulation of the interior of the Sun, including temperatures scaled from 5700 to 15 million K down to 15 to 45°C, and forces due to gravity acting on a 0.5g mass. The multi-modal haptic feedback adds to the realism or amount of information presented to the user in each case. In addition, temperature can be used as a substitute for other data; pressure, concentration, radioactivity or any other scalar field could be presented thermally if mapped appropriately to temperature.

Comments, Conclusions and Future Work:

To date, a thermal display for the PHANToM has been constructed, and has been shown to function fairly well. However, the device described by [Caldwell et al.–96] has achieved higher rates of heating and cooling, so further development is warranted to improve the capabilities of the ThermoStylus.

A number of simple demonstration programs have been written for the device, demonstrating the utility of thermal information in a haptic display. It was noted by some users that, because of the location of the heat sink, the other fingers being used to hold the device detect the cold of the sink. This is a distraction from the stimulus on the index finger (from the touch plate). This too will be addressed in the next version of the device.

In terms of creating convincing thermal feedback from VEs, the next major step is to study the thermal transients that occur when a finger comes in contact with a real surface. At the skin's surface, according to the description of semi-infinite bodies coming in contact, the change in temperature is instantaneous. However, this description applies for only a very short time, and is complicated below the skin's surface. To find estimates for the real transients, experiments will be conducted in the near future, in which fast temperature sensors attached to the skin will be brought into contact with materials of varying conductivity and heat capacity (e.g. wood, ice, aluminum). The transients can be recorded, and will serve as a target for the performance of the device. To put such a target in physical terms, one can imagine touching an object while wearing thin latex surgical gloves. The thinner the glove, the closer the perceived transient would be to touching an object barehanded. Currently, the display behaves as if the glove is approximately 0.4mm thick. Real surgeon's gloves are approximately 0.15mm thick, a reasonable goal for the next version of the ThermoStylus.

Acknowledgments:

This work is supported under the "Look and Feel: Haptic Interaction for Biomedicine" funded by DARPA Contract DAMD17-94-C-4123. Principal Investigator: Dr. Kenneth Salisbury

References:

[Caldwell et al.–95] Caldwell, DG, Andersen, U, Bowler, CJ, Wardle, AJ, 1995. "A high power/weight dexterous manipulator using 'Sensory Glove' based motion control and tactile feedback," Transactions of the Institute of Measurement and Control, Vol. 17, no. 5, Inst. Of Measurement and Control, London, England, pp. 234-241.

[Caldwell et al.–96] Caldwell, DG, Lawther, S, Wardle, A, 1996. "Tactile Perception and its Application to the Design of Multi-modal Cutaneous Feedback Systems," Proceedings of IEEE International Conference on Robotics and Automation, IEEE, New York, pp. 3215-3221.

[Dionisio–97] Dionisio, J, 1997. "Virtual Hell: A Trip Through the Flames," IEEE Computer Graphics and Applications, Vol. 17, no. 3, IEEE, New York, pp. 11-14.

[Dyck et al.–74] Dyck, PJ, Curtis, DJ, Bushek, W, Offord, K, 1974. "Description of 'Minnesota Thermal Disks' and normal values of cutaneous thermal discrimination in man," Neurology, Vol. 24, no. 4, Lancet Publications, New York, pp325-330.

[Kontarinis and Howe–93] Kontarinis, D, Howe, R, 1993. "Tactile Display of Contact Shape in Dexterous Telemanipulation," Proceedings of ASME WAM, DSC-Vol. 49, ASME, New York, pp. 81-88.

[Kontarinis and Howe–95] Kontarinis, D, Howe, R, 1995. "Tactile Display of Vibratory Information in Teleoperation and Virtual Environments," Presence–Teleoperators and Virtual Environments, Vol. 4, No. 4, MIT Press, Cambridge, MA, pp. 387-402.

[Salisbury and Srinivasan–97] Salisbury, JK, Srinivasan, MA, 1997. "PHANToM-based haptic interaction with virtual objects," IEEE Computer Graphics and Applications, Fall 1997, IEEE, New York (in press).

[Sherrick and Cholewiak–86] Sherrick, CE, Cholewiak, RW, 1986. "Cutaneous Sensitivity," In Handbook of Perception and Human Performance, Eds. Boff, KR, Kaufman, L, Thomas, JP, John Wiley and Sons, New York, pp12.28-12.37.

[Zerkus et al.–93] Zerkus, M, Becker, B, Ward, J, Halvorsen, L, 1993. "Temperature Sensing in Virtual Reality and Telerobotics," Virtual Reality Systems, Vol. 1, no. 2., pp.88-90.

Texture Sensing and Simulation Using the PHANToM: Towards Remote Sensing of Soil Properties.

Donald F. Green and J. Kenneth Salisbury

Department of Mechanical Engineering
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA

dfg@ai.mit.edu

Abstract

A method for sensing and simulating rigid, planar textured surfaces is presented. Test surfaces were a range of various grit value pieces of sandpaper along with a sheet of acetate. Static friction coefficients and surface height deviations are sensed by directly stroking the surface with the PHANToM fitted with a pointed probe tip. The resulting static coefficients were found to have Gaussian distribution. A very computationally fast, haptically convincing simulation of the textured surface is then generated from the mean and standard deviation of sensed coefficients and the sample vector of height values. The resulting simulation demonstrates that a haptically accurate representation of a real textured surface can be created from a very compact set of data. This method does not attempt to reproduce the textured surface in an exact sense, but determines a small set of properties which will allow an accurate-feeling reproduction of surface.

Introduction

This paper presents an approach to simulating textured surfaces by directly sensing properties such as static friction coefficients and high frequency/low amplitude surface deviations. The PHANToM is used as both a data acquisition platform and a display device. In an automated three step process, the PHANToM strokes a surface saving force input and position data, analyzes the data collected and generates a simulation of the textured surface. Currently, surfaces are assumed to be rigid and planar, with randomly distributed geometric textural properties. Specifically, we use various grit value pieces of sandpaper ranging from 50 Grit down to smooth pieces of acetate as the set of test textures.

Remote texture sensing could be used to build simulations of soil and rocks encountered during planetary explorations. A rover outfitted with a manipulator having capabilities like the PHANToM could conceivably acquire the data necessary to construct a realistic virtual environment of an interesting area that planetary geologists back on earth could interact with. The texture sensing process is relatively quick and uncomplicated and the resulting numerical representation of the texture is compact.

The mechanical properties of the PHANToM make it an excellent platform to begin research on remote sensing of soil and rock properties for the same reasons that it makes a good haptic display device: namely, it's high stiffness and low friction characteristics coupled with very high spatial resolution capabilities. These factors allow the PHANToM to perform very accurate position control while permitting reasonably accurate feed-forward force control and even force sensing, as we shall see. The PHANToM's nominal spatial resolution of about 0.03mm is fine enough to represent textures made up of sand particles as small as 0.06 mm but not silts or clay, which are defined in soil mechanics literature to range from .06mm down to less than 0.002mm [Lambe p.40].

Adding micro-geometry and tribological factors to the haptic display of virtual objects will increase the accuracy and perceptual credibility for many kinds of haptic simulations. Therefore, improvements in methods for haptic texture rendering should be of interest to a wide range of researchers within the field.

Related Work

In her seminal work on haptic texture generation, Margaret Minsky presents a method of simulating 3 dimensional textures with only 2 actuated degrees of freedom [Minsky p.48-51]. This work demonstrated how lateral forces added to the normal force changes the users perception of surface geometry. At last year's PHANToM Users Group Workshop, Chen and Taylor presented a stick-slip friction implementation for synthesizing textures using the PHANToM [Chen p75-76]. They modeled stick-slip friction as the interaction of a compliant probe catching and breaking away from small indentations uniformly distributed in a virtual surface. Siira and Pai discussed generating realistic feeling texture by varying lateral forces on a 2 degree of freedom haptic display [Siira]. Using randomly generated surface height deviations with Gaussian distribution, normal forces are generated proportional to virtual endpoint penetration beneath the surface, and lateral forces are generated proportional to the normal.

Physical Model

The interaction between the probe and a surface is modeled using the basic static friction force equation from adhesion theory [Mitchell p.306]. The two important physical constants are the static friction coefficient and the compliance of the probe. Figure 1 illustrates the physical model we use to inform our approach to analyzing the data taken during the sensing stage, and also the simulation of the sensed texture.

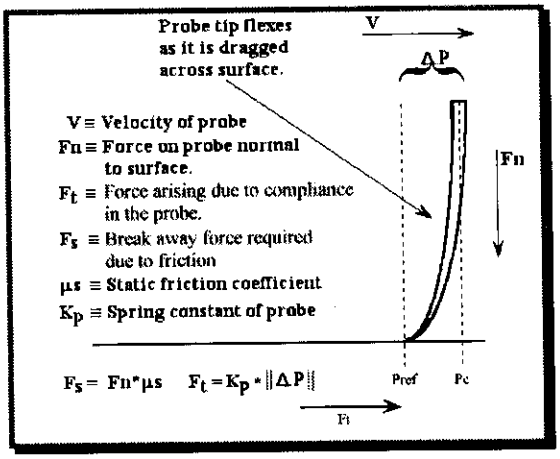


Figure 1.

It has been observed that stick-slip friction is commonly exhibited in point contact mechanical interactions involving minerals [Mitchell p.310]. This provides a theoretical justification for choosing this model for a probe stroking a surface, such as sandpaper. However, it should also extend to the surfaces of various kinds of rocks as well, if we disregard gross geometric perturbations of the surface.

Implementation

The first step involves taking a sample of the surface of interest. A hybrid control scheme is used which commands the PHANToM to follow a given trajectory (simply a line parallel to x-axis in PHANToM space) at a constant velocity, under P or PD control. In the process a constant normal force is exerted as the PHANToM tracks the commanded trajectory. The resulting lateral forces and the z position of the endpoint are recorded. Figure 2 shows a photograph of the sampling setup.

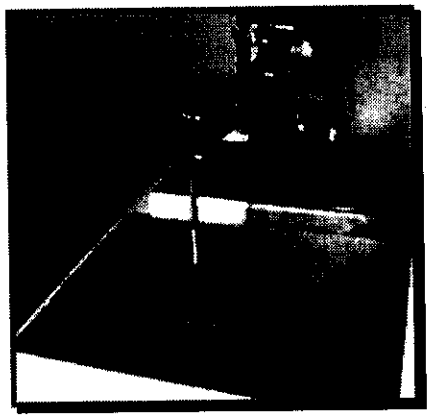
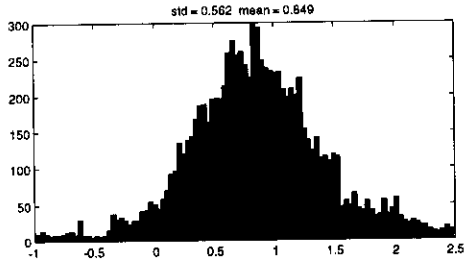
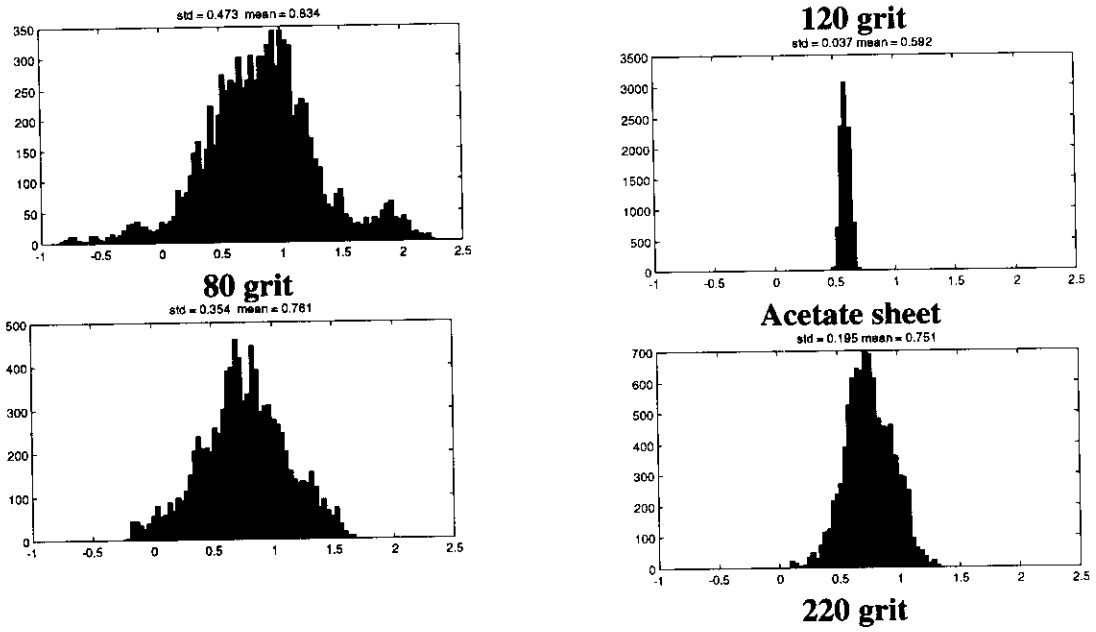


Figure 2

Each element of the force vector is then divided by the applied normal force and the result is interpreted as a vector of static friction coefficients. When plotted in a histogram, the sensed static friction values fall into a Gaussian distribution. Examples of histogram plots of sampled surfaces appear below. The vertical axis indicates the number of coefficients whose value fell into the bin represented by the bar. The bins were chosen as 100 evenly spaced centers in the range spanned by the 50 grit values (the broadest range). This has the effect of horizontally compressing and vertically stretching histograms of data sets with smaller standard deviations (decreasing resolution). When plotted with 100 evenly spaced centers across their own range of values, the shape of the other histograms more closely resemble that of the 50 grit.



50 grit



The mean and standard deviation of the computed static μ values are then used during the simulation stage, along with the vector of z - position values, which are normalized to have a mean value of zero. The vector of z values and the mean and standard deviation of the computed static friction coefficients are then used to drive the surface simulation.

The following is a description of the computations made during each iteration of the simulation to compute the output force exerted on the PHANToM:

> We first compute a perturbation for the base surface height for the current x,y position, Pc, by using ‖ Pc ‖ as an index into the vector of z values from the sensing process. The area each discrete height represents is computed from the ratio of the sample rate to the commanded velocity of the sample stroke. This has the effect of creating concentric circles of common height about the origin of the PHANToM space, but we have found the resolution to be too high for a user to perceive this.
>
> The next step is to check if the endpoint of the PHANToM has penetrated the virtual surface, and if not, no force is output. If it has, then the normal force, Fn, is computed to be proportional to the depth of penetration and the x,y point of contact, Pref, is recorded. A Static μ value, μs, is then computed for this contact point by a Gaussian random number generator with the characteristic mean, α, and standard deviation, σ, computed from the sampled data. The break away friction force required, Fs, is then defined to be Fs = μs * Fn. Tangential output force magnitude, Ft, is then computed to be Ft = Kp * ‖ Pc - Pref. ‖, where Kp is the stiffness of the probe and Pc is the current x,y position of the end effector. The direction that the tangential force is to be applied can easily be computed from Pc - Pref as well. When Ft ≥ Fs, a new μs is generated, the reference point Pref is updated to be the current position Pc and the process is repeated.

Results

This process produces very realistic feeling simulations of the textured surfaces. It does not reproduce an exact geometric or tribological mapping of the surface and we believe it would be impractical to attempt to do so using this mechanical sensing process. In any case, the phase information that an exact representation provides is perceptually unimportant when representing of the kinds of textured surfaces considered here. Perhaps this insensitivity to phase provides a hint towards quantifying the point at which the geometric properties of a surface pass from being described as shape to being described as texture.

Because of the underlying assumptions about the geometric properties of the surfaces being sampled, this process gives poor results when objects with regular texture patterns are tested. The resulting simulation lacks any directional dependence and when a surface with such a texture in sampled (such as a metal file) the resulting simulation feels like sandpaper. Spectral methods of analyzing data vectors taken from these regularly patterned surfaces should provide better results.

Future work

Future efforts will be directed towards extending these methods to homogeneous soils. We will explore sensing the surface compliance and viscosity or other factors of particulate systems. It may be possible to test the compressibility or shear limits of soils using the PHANToM as well. We are also interested in integrating this texture sensing process with methods for acquiring the overall shape of objects.

References

1.) Chen Jun; Taylor, R M. "Nanomanipulator Force Feedback Research." Proceedings of the First PHANToM Users Group Workshop, December 1996. MIT A.I. Technical Report No 1596.

2.) Minsky, Margaret D. R. "Computational Haptics: The Sandpaper System of Synthesizing Texture for a Force Feedback Display." Ph.D. thesis, Media Arts and Sciences, Massachusetts Institute of Technology, Cambridge, 1995.

3.) Mitchell, J.K. "Fundamentals of Soil Behavior" New York, John Wiley & Sons, Inc, 1976.

4.) Lambe, T.W; Whitman, R.V. "Soil Mechanics" New York, John Wiley & Sons, Inc. 1969.

5.) Sirra, J.; Pai, D.K. "Haptic Texturing - A Stochastic Approach." Proceedings of the 1996 International Conference on Robotics and Automation. Minneapolis, April 1996.

Simulation of contact sounds for haptic interaction *

Dinesh K. Pai and Kees van den Doel
Department of Computer Science
University of British Columbia
Vancouver, Canada
{pai|kvdoel}@cs.ubc.ca

1 Introduction

Sounds provide important perceptual cues for contact events such as impact and sliding. We believe real-time synthesis of realistic sounds will complement and greatly enhance haptic interfaces (however, see also [Led79, DBS97]). We will first describe physically-based simulation of the sound produced by contacting objects in virtual environments. The computed sounds depend on the material of the body, its shape, and the location of the impact. Finally we describe implemented software for interaction with simulated objects with auditory feedback, using a "Sonic Probe".

2 Sound Modeling and Simulation

When an object is struck, the energy of impact causes deformations to propagate through the body, causing its outer surfaces to vibrate and emit sound waves. The resulting sound field propagates through and also interacts with the environment before reaching the inner ear where it is sensed. Real sounds therefore provide an important "image" of various physical attributes of the object, its environment and the impact event, including the force (or energy) of the impact, the material composition of the object, the shape and size, the place of impact on the object, and finally the location and environment of the object.

*Supported in part by grants from IRIS NCE, NSERC, and BC ASI

In [vdDP96a, vdDP96b] we developed an algorithm for synthesizing such impact sounds, based on the physics of vibration. Related work on sound modeling and synthesis is also described in [vdDP96a]. We outline our basic approach here. We have recently extended this approach to synthesize continuous contact sounds, such as scraping sounds, in real time.

The vibration of an object is described by a function $\mu(\boldsymbol{x}, t)$, which represents the deviation from equilibrium of the surface, defined on some region $\mathcal{S}$, which defines the shape of the object. We assume that μ obeys a wave equation of the form

$$(A - \frac{1}{c^2}\frac{\partial^2}{\partial t^2})\mu(\boldsymbol{x}, t) = F(\boldsymbol{x}, t) \qquad (1)$$

with c a constant (related to the speed of sound in the material), and A is a self-adjoint differential operator under the boundary conditions on $\partial\mathcal{S}$.

In some cases this type of equation is well known, and can be solved by analytic methods for simple geometries with homogeneous materials. In that case, we can rely on pure simulation, as follows:

1. First, compute the vibration modes of the object. The solution to Equation 1 can be written as

$$\mu(\boldsymbol{x}, t) = \sum_{n=1}^{\infty}(a_n \sin(\omega_n c t) + b_n \cos(\omega_n c t))\Psi_n(\boldsymbol{x}), \qquad (2)$$

where a_n and b_n are arbitrary real numbers, to be determined by the initial conditions.

The ω_n are related to the eigenvalues of the operator A under the appropriate boundary

conditions, and the functions $\Psi_n(\boldsymbol{x})$ are the corresponding eigenfunctions, i.e.,

$$(A + \omega_n^2)\Psi_n(\boldsymbol{x}) = 0. \quad (3)$$

2. Next we compute the vibrations resulting from an impact at some point $\boldsymbol{p}$, when the body is initially at rest. We model the impact as causing an initial velocity

$$v_0(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{p}), \quad (4)$$

with $\delta(\boldsymbol{x})$ the k-dimensional Dirac delta function. This allows us to obtain the amplitudes of the vibration modes as a function of the impact location. The resulting sounds reflect the dependence on impact location.

3. The sound field around a vibrating body is very complex. Under reasonable assumptions [vdDP96a] and a simple model of the material damping (following [WR88, KK95]), we can compute the sound-wave $p_S(t)$ to be given by

$$p_S(t) = e^{-t/\tau_0} \sum_{i=1}^{n_f} a_i^S e^{-t f_i \pi \tan\phi} \sin(2\pi f_i t), \quad (5)$$

with the amplitudes a_i^S are computed based on the contact location, and

$$f_n = \frac{\omega_n c}{2\pi},$$

with the ω_n determined by Equation 3.

The model parameters for most real objects can not be obtained very easily if the object has complex geometry and material properties. In such cases, we assume a form of the sound response similar to that given by Equation 5 (but allowing more general damping), and estimate the parameters from measured sounds. See also [KK95]. This is more practical for modeling existing objects.

3 Interaction with Sonic Objects

To support interaction with a haptic device, the parameters for sound synthesis (in Equation 5), are mapped on to the respective contact locations on the geometric models of the objects. Based on the user's interaction, the sound parameters are extracted and the sound is rendered.

We have implemented a Sonic Explorer program, which allows us to create a 3D virtual environment with sonic objects. It is written in C++, using the OpenInventor graphics library, and SGI's Audio Library. A Windows95 version is planned.

Currently, our haptic device consists of a "sound probe" – a specially rigged pen and tablet. We do not have any force feedback associated with it, but it should be straightforward to replace the tablet with a force feedback device such as the PHANToM. In our current prototype, the input stream (force) is provided by a contact mike embedded in the probe, with which the user taps and scrapes objects with real-time generation of sounds.

The sounds under any kind of interaction can be obtained by convolving the impulse response with the interaction force. For this we have constructed an algorithm which runs in real time, and whose complexity is linear in n, the number of frequencies, or partials, in the sound model. The interaction is modeled as a live input data stream. The quality of the sound depends on the number of partials we can synthesize in real time, and can be dynamically adapted to the speed of the hardware. This means that there will be no breaks in the sound when other tasks require CPU time, but rather a smooth degradation in quality.

Figure 1 shows a 3D vase with sound parameters mapped on to the geometric model. Users can hit or scrape the object with the sound probe, and produce realistic sounds. A demonstration of the software will be provided at the workshop.

References

[DBS97] D. E. DiFranco, G. L. Beauregard, and M. A. Srinivasan. The effect of auditory cues on the haptic perception of stiffness in virtual environments. In *Sixth Annual Symposium on Haptic Interfaces for Vir-*

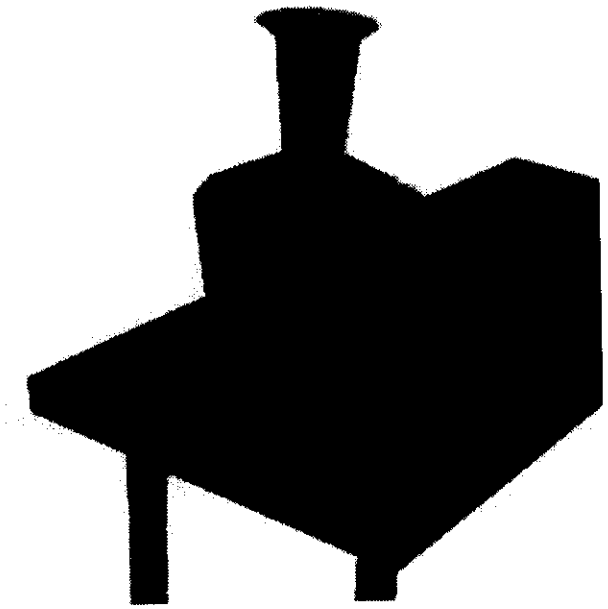


Figure 1: A 3D sonic object

tual Environments and Teleoperator Systems, page (To appear). ASME, 1997.

[KK95] Eric Krotkov and Roberta Klatzky. Robotic perception of material: Experiments with shape-invariant acoustic measures of material type. In *Preprints of the Fourth International Symposium on Experimental Robotics, ISER '95*, Stanford, California, 1995.

[Led79] S. J. Lederman. Auditory texture perception. *Perception*, 8:93–103, 1979.

[vdDP96a] K. van den Doel and D. K. Pai. The sounds of physical shapes. CS Technical Report 96-03, Univ. of British Columbia, February 1996. Submitted to Presence.

[vdDP96b] K. van den Doel and D. K. Pai. Synthesis of shape dependent sounds with physical modeling. In *International Conference on Auditory Display (ICAD 96)*, Xerox PARC, Palo Alto, November 1996.

[WR88] Richard P. Wildes and Whitman A. Richards. Recovering material properties from sound. In Whitman Richards, editor, *Natural Computation*, Cambridge, Massachusetts, 1988. The MIT Press.

FLIGHT
A 3D Human-Computer Interface and Application Development Environment

Tom Anderson
Sandia National Labs
email: tganders@u.washington.edu

Abstract

This paper discusses the FLIGHT application development environment, a three dimensional human-computer interface (HCI) software and Application Programming Interface (API). The development of HCI's is one of the few areas in computers that has not grown significantly compared with the enormous increases that have occurred in computing power. For over two decades, HCI's have only increased marginally, often relying solely on the ubiquitous mouse. With the now dominating presence of 3D environments, considerable amounts of effort have been spent in finding ways to use 2D input devices in these environments, which innately involves a level of non-intuitiveness and which is somewhat surprising given the ineffectiveness the mouse often has in 3D environments. 2D inputs and the lack of feedback are becoming the bottlenecks in productivity, especially with the increasing power of computers, enormous amounts of computational information, and the increasing prevalence of stereo, haptics, and 3D sound. With the advances in computational ability and human-computer interaction there is an increasing demand for software that no longer relies on ineffective 2D strategies, but is designed specifically for 3-D inputs and feedback. FLIGHT is a software that is based on the principle that as the human-computer interface is strengthened through the use of more intuitive inputs and more effective feedback, the computer itself will be far more valuable.

1. Introduction

An interesting phenomenon has developed in the computer world in which computers themselves have undergone orders of magnitude increases in performance, while inputs and outputs have remained relatively constant since the advent of the monitor, mouse, and keyboard. Upon first inspection this might seem natural, as this is the current mode of thought in computers, but in considering the purpose of computers, it is an odd progression. Computers exist to solve problems or to speed tasks, to give insight, for entertainment, or in any other way to help a person. Although high speed calculations are a part of all that a computer might do, it is not an end unto itself. It is odd that when a typical customer purchases a computer, relatively little thought, money, or advertising, is given towards the monitor, and much less to the mouse and keyboard, when nearly all interactions occur through them.

As computers become even more powerful and the amount of information that they process increases enormously, the lopsidedness of the HCI will become more of an issue. Three-dimensional environments will no longer be able to be controlled with a mouse, and feedback will no longer be able to be limited to simply a monitor. The bottleneck will be the interface between the computer and the user, and therefore a new mode of thought is necessary. A larger portion of the computer budget will necessarily be committed to interfaces and better outputs such as six degree of freedom (DOF) input devices, force feedback, tactile feedback, 3D sound, and stereo graphics or holography. This shift in thought will also require new software strategies to accommodate the enriched hardware environment. FLIGHT is a software that was designed to accommodate such a change in modality and focuses on improving one of the largest frontiers to come in computers -- advancing the HCI.

FLIGHT is a 3D application development software that can be used by both end users and programmers. It can be used to develop and interact with a wide variety of multi-dimensional environments including animation, computer-aided design (CAD), virtual reality, medical applications, scientific visualization, educational environments, games, an alternative HCI for the hearing or visually impaired, or nearly any other multi-dimensional environment, especially in which a user wishes to interact in ways that are impossible with standard 2D interfaces and feedback.

For an end user, FLIGHT extends the human-computer interface away from 2D into an entirely three-dimensional domain. A user interacts through a craft metaphor rather than through a desktop metaphor. Control modules, 3D versions of the common windows metaphor, allow access to the system in ways that are difficult or even impossible with only a mouse and a 2D interaction scheme. The system is designed to be intuitive, in that it tries to mimic real life interactions and feedback as much as possible to keep them familiar. For example, in real life and in FLIGHT one actually pushes a button to use it, rather than "pointing and clicking", and can hear and feel the button click as it is pushed. Force Feedback and 3D sound allow a much higher human-computer bandwidth than can be accomplished with graphics alone, and many of the interactions with the system can take place entirely without graphics. In addition, a user has access to 3D tools and also to interaction techniques that make easy transitions from the world to the control panel and navigation.

For a programmer, FLIGHT is a software shell based on OpenGL and C that can be used to incorporate graphics, haptics, and 3D sound in an established framework. FLIGHT utilizes "the next generation mouse," i.e. a three or more degree of freedom device, for input to the system in a way that lends itself towards device independence. The FLIGHT API consists of a library with calls for creating a 3D windowing system, for object creation and manipulation, for navigation, for tool interactions, and for general system and world interactions that take advantage of the added human senses.

2. The FLIGHT User Interface

The user interface for FLIGHT consists of a craft analogy with a 3D control panel and windowing system, including techniques for interacting with the system itself and with the application or world through a 3D cursor. Included in the user interface are navigation techniques, 3D tools, multiple human sensory channels of input and feedback, application specific interaction techniques, and a standard set of 3D windows, called control modules. The control modules have associated control objects such as buttons, sliders, knobs, etc. FLIGHT differs from more traditional graphical user interfaces (GUI's) in that it focuses more on haptics and 3D sound, and it is based entirely on a three-dimensional strategy. However, there is a smooth transition from typical 2D interfaces in that FLIGHT extends analogies that users are already familiar with while also extending the interface in ways that are more like real life and are often difficult with a more standard GUI. For Example, the control panel and control modules are extensions of a desktop/windows metaphor, yet the newer aspects that are created with the third dimension, such as buttons that one can actually feel and push in, more closely resemble real life. The reason that this is important is the assumption that as computers more closely resemble real life and interactions that people use every day, the interface will be more intuitive and useful. This strategy keeps a user grounded yet still allows for extensions that are impossible in real life. An example is a three-dimensional slider that is not attached and therefore hangs in the air, but still has resistance to moving. People know how to use one dimensional sliders, and so one that is 3D is not a large leap, yet it can be more useful in controlling aspects of an application given the added degrees of freedom.

2.1 The FLIGHT Control Panel and Control Modules

FLIGHT consists of two types of interaction modes -- interaction with the control panel and interaction with the world. These two modes of operation can be easily traversed by cursor movements with respect to the current frame of reference. To change to world interactions, the cursor

moves through the 'windshield' of the craft which then places the cursor in the world. To return to the control panel, the cursor must be pulled backwards through an invisible wall. This type of interaction technique allows easy transitions in addition to a fuller range of movement in each mode of operation.

The control panel consists of a bottom and front panel that have buttons to interact with the system. In general, the front panel contains buttons for navigation, while the bottom panel contains buttons for accessing the control module system.

The control modules are separated into six areas: Tools, Preferences, Craft, Information, Operations, and Application Specific control modules. The tool control modules allow interactions through the use of 3D tools. The preferences and information control modules give access to the overall system except for craft interactions which are contained in the craft control modules. The application control modules are specific to and defined by the application itself.

Control Modules have a variety of standard control objects that can be used to manipulate the system or the world. These objects often have advantages over their 2D counterparts because of the added degrees of freedom. For example, a 3D slider can be used to directly control an object's x, y, and z positions simultaneously and intuitively. Preferences in many situations can be accessed by pushing a button to the left which is natural given the button's 3D nature. Another important point to consider is that with the 3D nature of FLIGHT there are possibilities for new types of control objects as well. One type of object might be a multiple sided geometric surface in which each side controls a different action and in which the layout of the object is key to its use. Other possible objects might be 3D miniature and simplified representations of real objects in the computer world that could be used to control the real objects directly, for example a puppet used to control an animation. While the 3D user-interface has already shown benefits, the full extent of the advantages still remain to be explored.

2.2 FLIGHT Navigation

The control panel also gives access to the different types of navigation, defined here as all aspects involved with changing the user's viewpoint, which are based on different craft metaphors. Each craft is specific to a different type of situation. The crafts include a hovercraft, a plane craft, a six DOF craft, several types of teleportation, a circular craft, a sphere craft, and a terrain craft.

Although the specifics of all the types of navigation are extensive, the craft names can be used to imply their uses and methods of movements to some extent. The sphere craft, teleportation methods, and terrain craft are used to control the craft's position and orientation directly while the other crafts control them through velocities. The crafts are often limited in some ways to make them more intuitive, for example the hovercraft and terrain craft can only move on a three-dimensional plane, while the plane craft moves more like an airplane. The circular craft and sphere craft are always centered on a particular point in the world that can be changed at any time, which makes them good general crafts for overall inspection of an area or object. All of the crafts also use feedback to make their operations more intuitive. For example, the six DOF craft is controlled with a control box that the cursor pushes into for movements. Each side of the box controls a translation in its respective direction and controls the rate of movement as well. Therefore, as the cursor pushes into the left wall of the control box, the craft moves left. The harder the cursor pushes against the wall, the faster the craft moves. This force feedback allows a user to precisely control the craft's velocity especially in a non-linear control situation.

The three dimensional cursor also greatly enhances the abilities of the navigation. It can be used in the world interaction mode to determine specific teleportation points and can set the center points for the sphere craft and circular craft quickly, as they must be placed in a 3D space. In addition, while in the world interaction mode a user can quickly change between interaction and navigation by double clicking the input switch. This presents a useful method for quick interactions.

Finally, the control modules themselves aid in the navigation process. They can be used in controlling how the different crafts move, and can even give information to aid in the process of navigation. There are, for example, control modules that give numerical values for position and

orientation and control modules that give visual representations of the craft and the world, i.e. through a representational map and orientation graphic object.

2.3 FLIGHT Tools

The tool control modules allow interactions with the system, world, and specific application through tools such as clip plane tools, measurement tools, or object creation and manipulation tools. The tools are particularly effective given the 3D nature of the cursor. For example, a clip plane can be easily placed by the 3D cursor in the world interaction mode, where this is a particularly difficult problem with a mouse. A measurement tool can, for example, move through a 3D space to the exact position where a measurement is desired, even if the area is obscured. Other types of feedback can also enhance the tools' effectiveness as a user does not need to rely solely on the graphics for a tool's use.

3. The FLIGHT API

The FLIGHT API is the set of commands that allow a programmer to create a 3D computer environment through the use of the FLIGHT library. A FLIGHT application is created within a modular software structure that is designed towards rapid application development and hardware independence. FLIGHT in its core self is not a renderer, although that is one possible use for the system. Instead, a programmer would use it more as a shell and structure to create a specific application. It is based on OpenGL and C, so that any graphics in an application are created through an OpenGL based renderer in conjunction with the FLIGHT graphics.

FLIGHT has a multi-threaded structure in which the first thread controls the graphics and related control, and the second, faster thread controls much of the user interaction. One strong purpose of a second thread which runs at a faster rate is to accommodate the needs of the haptics aspects, as haptics requires a faster cycle rate than does graphics.

The base of all the interactions with the system occurs in a globally shared memory area, which is key to the system's design. The shared memory block allows an interface layer with which platform specific or device specific hardware can interact. This also allows a interface layer that is grouped into several key areas that allows intuitive programming especially with respect to the human sensory channels. In addition, the API includes a large number of commands for interacting and adding to the system. Key among these commands are those for navigation, for object interactions, and for control modules.

The navigation aspects of the API consist of a simple set of commands that allow complex movements, because a user can control multiple degrees of freedom simultaneously given the types of input and feedback FLIGHT uses. By interfacing through the shared memory block, the navigation commands allow a user's input to affect the craft's movements with few lines of code. All of the crafts, described in section 2.2, were created with the navigation commands, which allowed an effective way to find the most useful modes of navigation in each craft's respective situation. Although there are many crafts that already fit most situations, the navigation aspects of the API can be used in an application specific way to extend the craft's movements to fit the situation even more precisely.

One of the strongest aspects of the FLIGHT API is in the creation of the control modules. The control module aspects of the API allow a simple and convenient way to create control modules and their respective control objects. One large benefit is that with a single command, a control module or one of its objects can be created in both the haptic, auditory, and visual domain. When a command to create an object is called, its information is put into the shared memory block which is then accessed by the graphics, haptics, and sound portions of the code to present it to a user. The specifics of the individual objects can then be altered with additional commands to customize them to their task. Other types of world objects created with the API have the same characteristics, in that with a single command, they can be created with default graphical, haptic, and audio characteristics. This saves a

considerable amount of programming time, because objects do not have to be created once in each of the multiple sensory domains.

4. Example Applications

FLIGHT has been used at Sandia National Laboratories in several scientific applications. It has been used on several Silicon Graphics (SGI) platforms with both a PHANToM haptic interface, from SensAble Technologies, and a spaceball as inputs and an Acoustetron, from CRE inc., for the 3D sound. At Sandia's recent "Revolution in Engineering" presentation, several FLIGHT applications were bundled into an overall demonstration application called "Crash and Burn". This application provided the visualization tool for simulations run on Sandia's teraflop supercomputer, currently the world's fastest computer. The applications included three fire dynamics simulations, a structural impact simulation of an airplane wing crashing through a post, and a heat dynamics demo for a weapon. In addition, FLIGHT was used at Supercomputing 1997, which demonstrated another bundled application. The demos presented included a haptic texture editing demo, a motion dynamics demo, intersensory discrepancy effects with geometric objects, a billiards game, a 3D sound demonstration, and a fire dynamics simulation. FLIGHT has also been used in an experiment conducted at Sandia, in conjunction with the human factors department, in which FLIGHT was compared with the SGI Cosmo Suite through navigation and manipulation tasks. FLIGHT is now being used at the Human Interface Technology (HIT) lab at the University of Washington by the author for graduate research. Given FLIGHT's strengths in interaction and navigation, CAD and animation seem good prospects for future applications and are being explored.

5. Conclusions

FLIGHT was designed to solve several problems that were encountered while trying to create and interact with 3D computer environments, particularly the long development times for creating applications and the added difficulties of input and feedback. Its focus has been to use the human senses and advanced inputs in ways that allow intuitive interactions. Although the mouse is an overwhelmingly standard input for computers, it is nearing its end. Computers are nearing a point where 3D will be the new interaction structure, which will require a new mode of thought. As the next great hurdle in computers nears, advancing the HCI, software with design strategies such as FLIGHT's will be crucial in advancements towards breaking the HCI barrier.

6. Acknowledgments

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

7. References

[1] J. Preece, *Human-Computer Interaction*, Addison-Wesley Publishing, New York, NY, 1994.

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

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

[4] T. Anderson, "A Virtual Universe Utilizing Haptic Display", PHANToM User's Group Workshop, MIT AI Lab, Cambridge, MA, 1996.

[5] W. Barfield, T. Furness III, *Virtual Environments and Advanced Interface Design*, Oxford University Press, New York, 1995.

LEGOLAND: A Multi-Sensory Environment for Virtual Prototyping

Peter Young, Tom Chen, David Anderson, Jiang Yu
Department of Electrical Engineering
Colorado State University
Fort Collins, CO 80523

Shojiro Nagata
NHK Science & Technical Research Labs
Tokyo, Japan

October 1, 1997

Abstract

This paper briefly describes the LEGOLAND project, which is a virtual prototyping environment. The environment provides virtual LEGO blocks which can be manipulated to assemble virtual LEGO models. This environment is rendered in real time on a multi-sensory computing platform, with integrated 3D-visual, audio, and haptic components.

1 Multi-Sensory Computing

The LEGOLAND project was developed to study the issues involved with virtual prototyping in a true multi-sensory environment. The platform for this study is the the Stereoscopic Haptic Acoustic Real-Time Computer (SHARC), currently being developed at Colorado State University (CSU) for the study of multi-sensory computing. The central processing unit consists of an SGI R10000 workstation, which is interfaced to an autostereoscopic 3D display system using a 14" LCD panel with lenticular lens. The operation of the lens is illustrated in figure 1. An interlaced pattern of left-eye and right-eye images is displayed on the LCD screen. The lens splits these images along radial directions so the the viewer's left and right eyes intercept different images, providing the stereoscopic 3D effect. Note that in this setup no goggles are needed by the viewer, which is an important consideration for many applications.

The setup also includes a 3D acoustic head-tracking subsystem, which provides the computer with the viewer's position. This information is used to update the display in real-time, so that the viewing angle is adjusted accordingly. Thus the viewer can "look around" a 3D object without having to manually rotate it. This is important, not only for efficiency, but also to provide the viewer with the correct visual cues, so that the environment appears realistic. The head position information can also be used to optimize the alignment of the 3D autostereoscopic display.

The system also includes a PHANToM haptic interface. This force-feedback robotic system is used as a "display" to provide haptic/tactile information to the user. At the same time it is used

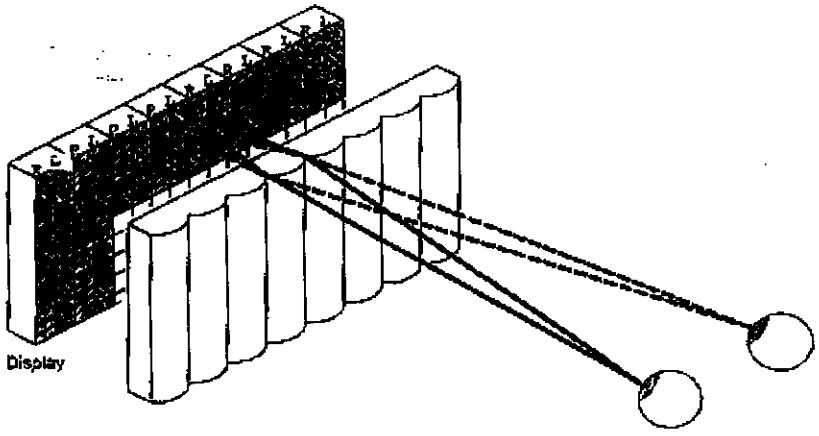


Figure 1: Lenticular Lens Operation

as a very efficient input device, since it can acquire position in six degrees of freedom (position and rotation), and hence it provides us with a 3D mouse to efficiently navigate and manipulate the 3D environment in a natural way. A photograph of the overall system is shown in figure 2.

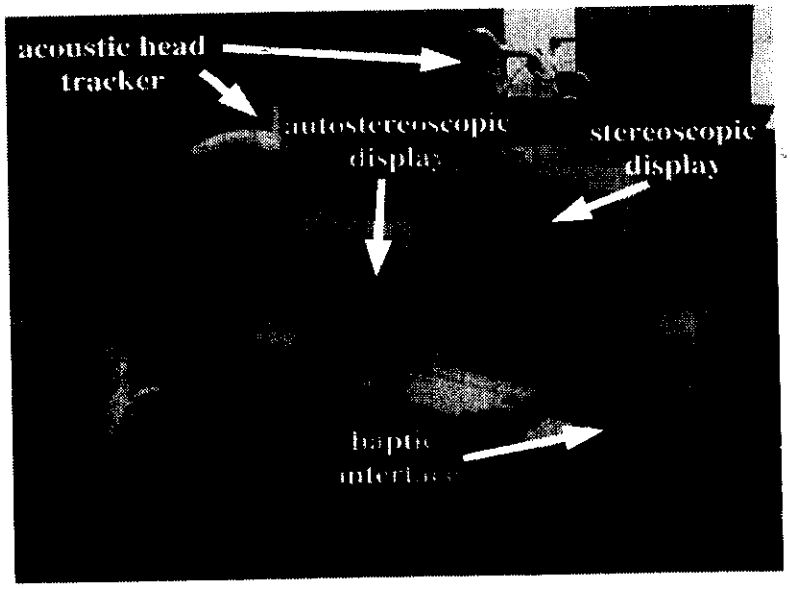


Figure 2: The SHARC System

This environment provides a unique multi-sensory computing platform, with interactive capabilities among the different sensory components. The facility for multi-sensory computing enables us to provide a high degree of realism for virtual or augmented reality environments. It also affords us a high bandwidth interface between human and computer, as illustrated in figure 3.

It is well recognized that the development of complex physical skills, associated with dextrous manipulation, relies on a person's ability to simultaneously process a number of different sensory

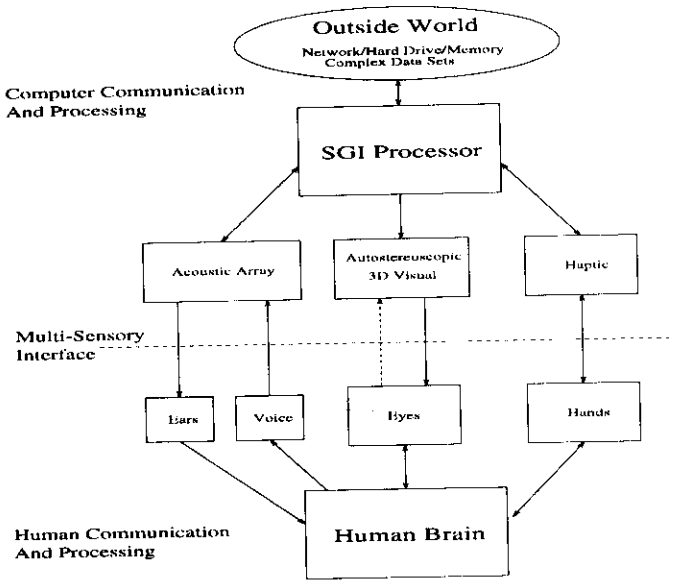


Figure 3: Multi-Sensory Human-Computer Interface

cues. For instance the tactile component, enabling one to assess the elasticity of tissue, is vital to a surgeon performing an operation. Hence, for many applications, it becomes important to render visual (preferably 3D), audio, and haptic/tactile information. The SHARC system affords us this possibility, but with it come a number of issues that need to be addressed for a viable application, including interfacing, speed, and bandwidth.

2 The LEGOLAND Environment

In order to study these issues, and demonstrate the viability of this setup for virtual prototyping, we developed the LEGOLAND environment. This environment provides one with a multi-sensory rendering of a set of LEGO blocks, using the SHARC system. The blocks can be viewed in 3D, with head tracking and viewing angle adjustment implemented in real time. One may use either a stereoscopic display (with goggles) or an autostereoscopic display. The surface geometry of the blocks (or assembled models) can be examined by touch, via the haptic interface, providing a very realistic rendering of these objects. The blocks can be manipulated using the haptic interface as a 3D mouse to move through the three dimensional scene. The user can pick up and manipulate blocks with a full six degrees of freedom in position and rotation. The PHANToM provides a very natural interface for this type of interaction, which is much more efficient than the standard keyboard and mouse. The LEGO pieces can be joined together (or pulled apart) to assemble virtual LEGO models, and a screenshot of some assembled pieces is shown in figure 4.

The dynamic properties of the blocks are programmed in to the virtual environment, so that the blocks have weight and inertia. Moreover real-time collision detection between "solid" objects is fully implemented as the user grasps and moves objects. These collisions result in the appropriate dynamic reactions, which include visual and haptic/tactile components. Similarly the act of snapping a LEGO block into place is accompanied by a realistic multi-sensory dynamic simu-

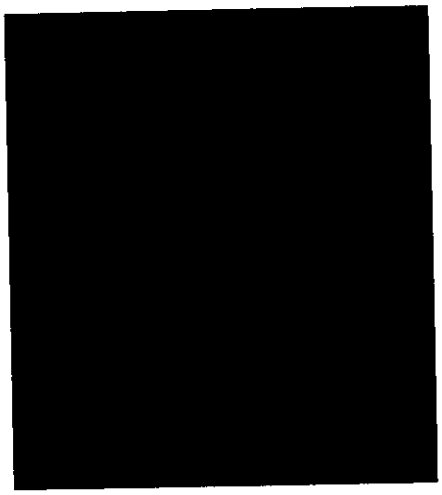


Figure 4: A Simple LEGOLAND assembly

lation. Although LEGOLAND is only a fairly simple multi-sensory environment, it incorporates many of the properties and challenges of the more complex virtual prototyping setups that will be required for real-world engineering problems. We believe this is one of the first multi-sensory virtual prototyping systems in existence.

3 Future Directions

Despite its simplicity, the computational requirements of the LEGOLAND environment are already high. The visual/haptic rendering and real-time collision detection are based on polygonal models for efficiency. Even so, with about 15-20 pieces, the environment can already use 70% of the CPU time. The visual rendering typically occupies less than 5% of this, so that the CPU spends the bulk of its time dealing with haptic rendering. This illustrates the inefficiency of current techniques for haptic rendering, especially for dynamic simulations (with collisions). We are currently investigating both hardware and software architectures for high-speed haptic rendering, and its integration with the other components of a multi-sensory computing environment.

Finally, we note that there is growing interest in this type of approach for tele-operation of robotic systems, such as tele-surgery. In this regard, we are currently developing an Internet implementation of the LEGOLAND environment, so that a user with the appropriate hardware configuration can access and interact with LEGOLAND through our Web page.

Adding Force Feedback to Interactive Visual Simulations using Oxygen™

Peter Larsson
Prosolvia Clarus AB
peterl@clarus.se

ABSTRACT

Prosolvia Clarus AB is a Swedish software company providing tools for creating Interactive Visual Simulations for industrial use. One of Prosolvia Clarus' products is Oxygen™, a GUI based tool for creation and real-time manipulation of Virtual Reality environments. Prosolvia Clarus is currently co-operating with Sensable Technologies, Inc. to add support for the PHANToM Force Feedback device to Oxygen™. This will make it possible to interactively create simulations in which you not only can see objects but also touch or even grasp them.

INTRODUCTION

Prosolvia Clarus is focused on providing industrial solutions using Virtual Reality Techniques. The company offers solutions throughout the complete life-cycle of a product; design support, virtual prototyping, product visualisation, etc. Prosolvia Clarus also offers a wide range of other products such as medical simulators and simulators for ground based vehicles.

Force Feedback is a technique to allow force interaction within virtual environments. Virtual objects can be touched or even grasped and lifted. Properties like surface friction and stiffness can be modelled or full rigid body behaviour simulated.
Prosolvia Clarus AB is the distributor of the PHANToM™ Force Feedback device [1] in Scandinavia and Germany. We are also currently co-operating with Sensable Technologies, Inc. to add Force Feedback support to some of our products. With this technology Interactive Visual Simulations will be made even more realistic.

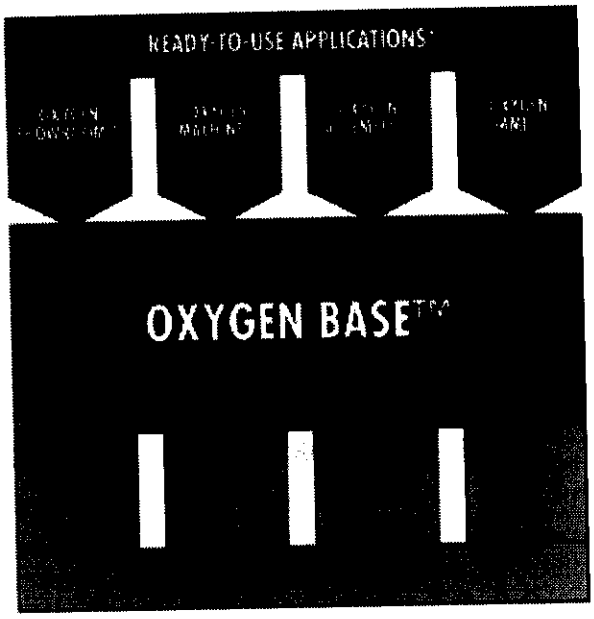
OXYGEN™

Oxygen™ [2] is a multi-platform, interactive tool that allows you to create and manipulate Interactive Visual Simulations even at run-time. CAD-data or other graphic models can be imported and brought to life using the Oxygen™ Graphical User Interface.
Oxygen has a modular design that can be divided into three major parts:

Ready-To-Use Applications: Applications built on top of Oxygen™. E.g. Oxygen Showroom™ for product visualisation, and Oxygen Assembly™ for virtual assembly and testing.

Oxygen Base: Provides basic functionality for importing models and creating Interactive Visual simulations.

Technology Plug-ins: Adds support for CAD data import, Peripherals, etc. Soon there will also be a support for Force Feedback.



Oxygen™ is built on top of Open GL Optimizer™ and supports a node-based approach for creating Interactive Visual Simulations. A set of nodes defining a wide range of behaviours (Interpolators, sensors, etc.) and supporting many kinds of peripherals (Head Mounted Displays, Trackersystems, CyberGloves, Stereo-Sound, etc.) may be added to the simulation. Each node has a number of fields containing information about it's current state. By connecting the fields of different nodes to each other, thus transferring information between them, desirable behaviours can be obtained.

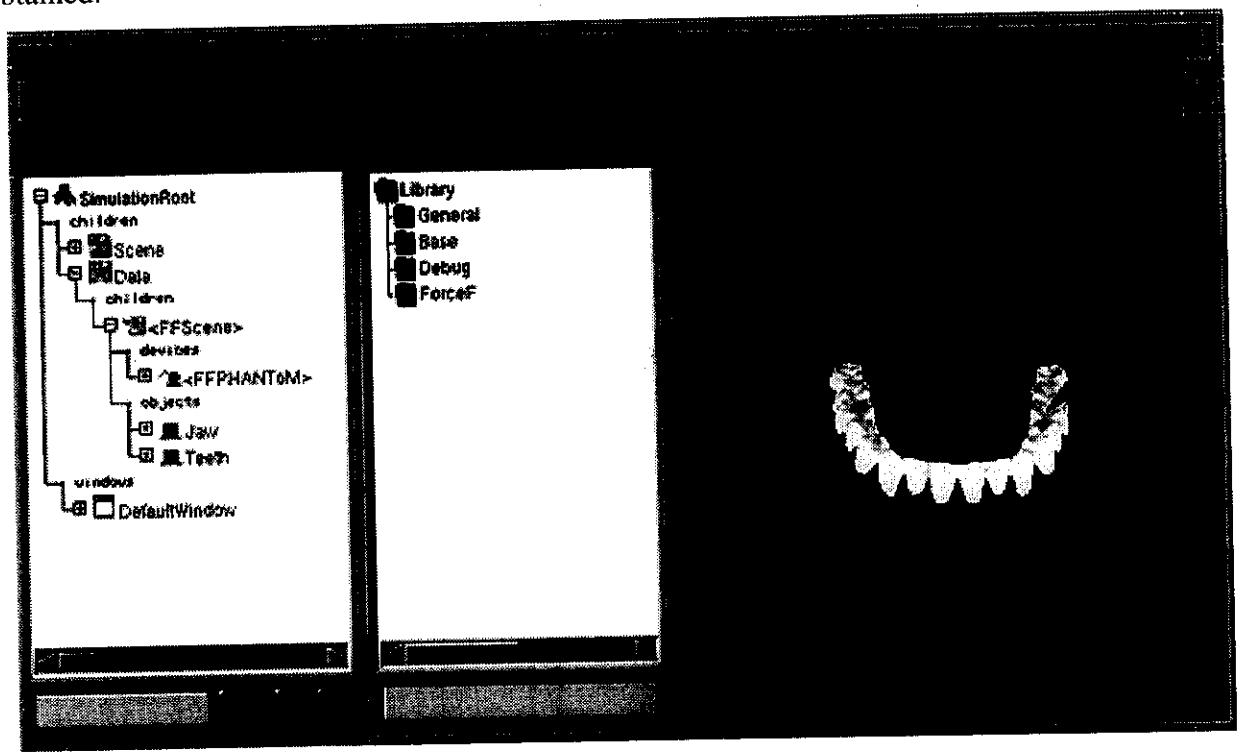


Figure 1: Parts of the Oxygen GUI.

OXYGEN FORCE FEEDBACK™

Oxygen Force Feedback™ [4] is a technology plug-in to Oxygen™ that allows force feedback to be added to simulations using the same node-based approach as described above. Graphic objects in a scene can easily be given haptic properties, and parameters such as surface stiffness, friction, etc., can be set at run-time in the Oxygen GUI. Oxygen Force Feedback™ supports the PHAN ToM™ device from Sensable Technologies, Inc. and is built on top of their GHoST™ SDK [3], which provides a high-level API to PHANToM™.

To add Force Feedback to a simulation, a Force Feedback scene is created in parallel to the graphic scene. In the Force Feedback scene nodes can be added to support one or more PHANToM™ devices and haptic objects. Instead of adding transforms and geometries directly to the Force Feed back scene, there are references to objects, or groups of objects, in the graphic scene. Using this approach, haptics can be added to an existing simulation without recreating it, also allowing haptic and non-haptic objects to be used simultaneously.

COMMENTS AND FUTURE WORK

Since forces have to be updated at a high and stable rate (roughly 1000 Hz) we have chosen to separate the graphic rendering from the haptic servo loop, using separate processors for the tasks. In the first version of Oxygen Force Feedback™, Oxygen™ is run on a SGI machine, while the haptics loop is run on a PC, with the processes communicating via a TCP/IP connection. Future versions will support other configurations (e.g. both processes running on the same platform: SGI, PC,...).

Our future plans include adding Force Feedback support to some of our existing products, for example Oxygen Assembly™, and to create new ready-to-use applications using Force Feedback. New features can easily be added by creating new node-types in the Force Feedback module. Support for deformable objects and real-time collision detection between dynamic objects are examples of possible extensions.

REFERENSES

[1] Massie, T. H. "Initial Haptic Exploration with the Phantom: Virtual Touch Through Point Interaction", MSc Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology. 1996.

[2] Prosolvia Clarus AB, Sweden. Product Definition Oxygen Base. Document Number PDF-5148-1. 1997.

[3] Sensable Technologies, Inc. Cambridge, MA. "GHoST Software Developer's Toolkit, API Reference", Version 1.1, 1997.

[4] Prosolvia Clarus AB, Sweden. Product Definition Oxygen Force Feedback. Document Number PDF-4021-1. 1997.

THE HAPTIC WORKBENCH APPLIED TO PETROLEUM 3D SEISMIC INTERPRETATION

K V Nesbitt[1], B J Orenstein[1], R J Gallimore[1], J P McLaughlin[2]

[1] **BHP Research - Newcastle Laboratories**
PO Box 188 Wallsend NSW 2287 Australia
Email: gallimore.randall.rj@bhp.com.au

[2] **CSIRO Mathematical and Information Sciences**
GPO Box 664 Canberra ACT 2601 Australia
Email: John.McLaughlin@cmis.csiro.au

1.0 INTRODUCTION

This paper is the result of work performed for the Advanced Computing Applications Project (ACAP) within the BHP Research Advanced Systems Corporate Program. A key role of ACAP is to evaluate emerging and embryonic computing technologies through the implementation of application test beds identified to have a high business impact.

One of the computing technologies selected for its potential to have a significant business impact is Virtual Environments (VE).

An initial survey was undertaken to identify relevant VE component technologies and to determine potential centres of excellence with whom ACAP could collaborate to progress the technology evaluation and test bed development. After an evaluation of these centres it was decided to develop two main test beds.

The first test bed is being developed with "Collaborative Workbench" technology in conjunction with the Virtual Environment Technology Laboratory (VETL) at the University of Houston, Texas. This project will evaluate the potential of VE to support collaborative geoscientific decision making. The second test bed is being developed with "Haptic Workbench" technology in collaboration with the Commonwealth Scientific and Industrial Research Organisation (CSIRO) Mathematical and Information Sciences, Canberra. This test bed will evaluate the potential of displaying and interpreting multi-dimensional geoscientific data sets with the visual, touch and auditory senses.

This paper details the hardware, software and multi-sensory user-interface capabilities of the Haptic Workbench platform. It describes some initial demonstrations developed with CSIRO for assisting interpretation of 3D seismic data sets in the petroleum exploration domain. In depth discussion of the technical issues related to haptic rendering of the data is presented in a related paper [7].

2.0 THE HAPTIC WORKBENCH

The primary goal of VE is to significantly increase the communication bandwidth between humans and computers. One of the key drivers for VE is the development of multi-sensory interfaces which support natural human interaction.

Haptic interfaces enable manual interaction with computer systems, receiving user movements and feeding back tactile information. This allows for the touching, feeling and manipulation of objects in a virtual environment and can provide a sense of immersion not possible with visual and auditory feedback alone. The haptic sense is used for both exploration and handling tasks. Exploration tasks are mainly sensory in nature and involve the extraction of object properties such as shape, mass and texture. Sensory information is both tactile (sense of contact) and kinaesthetic (sense of position and motion). Other receptors signal skin temperature, mechanical and thermal pain as well as chemogenic pain and itch. Handling tasks are dominated by user motor actions such as grasping and object manipulation. These actions may include precision and power tasks. Most haptic actions involve a synergy of sensory explorations and motor manipulation.

The Haptic Workbench [Figures 1, 2] is a single-user, multi-sensory interaction platform. It provides support for auditory, haptic and stereographic visualisation. It was developed at CSIRO Mathematical and Information Sciences, Canberra and incorporates the PHANToM™ force feedback device [5] from Sensable Technologies [3] Inc. with a "Virtual Workbench" [4] stereographic display

platform, developed at the Institute of System Science (ISS) of the National University of Singapore [6]. Given that this conference is centred on PHANToM™ technology, we have assumed that its capabilities are already understood by the audience and a detailed technical description is obtainable elsewhere if required. However, the "Virtual Workbench" is described below.

3.0 THE VIRTUAL WORKBENCH

The "Virtual Workbench" was developed by and is commercially available from ISS. It uses standard Liquid Crystal Display (LCD) shutter glass technology to view images generated on a normal monitor. Left and right eye images are displayed sequentially to the screen and synchronised with the glasses which "shutter" vision to alternate eyes. Because the alternating images are offset in perspective this produces a stereographic effect. LCD glasses are lightweight and non-intrusive and as the images are rendered on a monitor, they provide a relatively high resolution display.

The workbench suspends a computer monitor face down on a fibreglass frame, projecting the image onto a mirror. Using the binocular parallax generated with the LCD glasses, a 3D image is suspended in a virtual space which corresponds to the user's normal hand position when seated at a desk.

In the Haptic Workbench force feedback can be used to indicate contact with a surface and hand location and orientation are calculated directly from the mechanical linkages of the PHANToM™. However, in the "Virtual Workbench" interaction with the 3D space is only provided through visual feedback of the position and orientation of the hands (which are determined by an electromagnetic Polhemus™ [6] tracker and stylus) in the 3D display. The virtual world is projected into a space which is coincident with the user's hands. This provides a reach-in, high-resolution, 3D environment which brings the displayed model into the user's natural work volume. It allows two-handed dexterous manipulation but is less intuitive without the haptic feedback.

Most applications being developed for the workbench are from the medical domain. However, many parallels exist between the medical and petroleum exploration domains since both involve large multi-attributed volumetric data sets and modelling of structure.

A C++/OpenGL software toolkit called BrixMed™ is available to enable the development of applications. This library does provide built-in support for left and right handed tracking with a Polhemus device as well as the necessary transformations to invert the images being projected onto the mirror. This inversion is necessary so that the reflected image appears correctly oriented to the user.

4.0 DEMONSTRATIONS

The demonstrations were developed to provide input for a major decision point in the development of test bed applications with the Haptic Workbench. Without positive feedback from BHP Petroleum geoscientists (including geophysicists, geologists, petrophysicists and reservoir engineers) further work on this test bed would not be undertaken.

The strategy was to develop representative applications for seismic interpretation in the petroleum exploration domain; demonstrate these to the geoscientists and subsequently capture their feedback.

Early experience with virtual environments indicated that users have difficulty understanding verbal or written descriptions of the new models of interaction they provide. To elicit useful feedback it was considered appropriate for users to experience the new interaction styles in a relevant application area.

The demonstrations were ordered so as to gradually introduce the concepts of haptics and multi-modal interaction to the geoscientists. In particular three key enabling features of haptic force feedback were demonstrated:

1. The intuitive nature of spatial location in 3D models when combining touch and visual modes. The ability to touch surfaces, manipulate and navigate through models is a natural way for users to interact with objects in 3D space.
2. The ability to mimic naturally occurring physical force models such as spring dynamics and vibrations.
3. The ability to map complementary data attributes to the haptic mode from multi-attributed data sets. The aim is to prevent overloading of a single sensory mode and

present greater information detail by using synergy between visual, haptic and auditory senses.

The "Orientation" demonstrations were designed to introduce simple haptic concepts. In the "Spheres" demonstration the ability to feel solid spheres and touch 3D surfaces is a compelling introduction to the realism created by combining haptic and visual feedback. The "Springs" demonstration allows the user to manipulate a model of four spheres connected by springs and experience the dynamic behaviour of this physical model.

The "Single Plane" demonstration [Figure 3] presents a vertical slice through a 3D petroleum seismic survey. A standard red-blue amplitude colour map shows typical horizon structures. Changes in amplitude are mapped to force, with large gradients requiring greater user effort to move between them. This demonstration was designed to assist the task of tracking along a single horizon, as the user is restrained to specific values in the data. The task of picking points of interest in the data is mimicked by clicking the button on the PHANToM™ stylus to "drop spheres" which act as markers. While the plane is visible in stereo the main aim is to introduce the concept of using force feedback for horizon following.

The "Triplanar" demonstration also uses force feedback for horizon following. The stereoscopic display now provides a three plane view of the seismic data with orthogonal crossline, inline and timeslice planes. It allows two different 3D data sets to coexist in the volume. The user can show raw amplitude on each plane and choose to view semblance processed values on the timeslice. The planes can be positioned in the volume by pushing them to the desired location. The entire volume can be selected with the left-hand Polhemus tool and rotated. The user can also perform 3D picking by "dropping spheres" using the button on the PHANToM™ stylus with the right hand.

The "Seeding Isosurfaces" demonstration [Figure 4] uses a similar triplanar display but introduces the concept of generating isosurfaces. These surfaces correspond to volumes within the 3D data with like amplitudes. In this case semblance processed timeslice data is used to automatically generate faulting structures after the user selects a single point on the fault. Once the structure is picked, the user can then feel the fault as a discontinuity or force barrier when tracking along an intersecting horizon.

5.0 FEEDBACK FROM GEOSCIENTISTS

As a result of positive feedback from BHP Petroleum staff the following application areas were identified as having the potential to be impacted by Haptic Workbench technology:

1. Evaluating hydrocarbon migration paths in 3D. Analysis of migration paths is an inherently 3D problem and involves understanding complex vector flow models within the interpreted geological structures of a prospect area. Cost of exploration wells can range from $10-40+ million. This application would provide a confirmation step before exploration drilling proceeded. There is a natural mapping of flow vectors to a force vector model. It could incorporate a 3D lithology model for visual feedback and is relevant in all geographic locations of exploration, especially where complex geology is seen such as deep water in the Gulf of Mexico.
2. An integrated 3D seismic interpretation and reservoir modeling tool would be an extension of the initial demonstration work. The intent would be to integrate down stream reservoir models back into seismic data to improve quality of interpretation. This could involve integrating velocity components, lithology, porosity, density, stacking velocity, flow characteristics and reservoir characterisations. There are other multi-attributed and derived data sets which originate from gravity, magnetic, electromagnetic and seismic surveys with which the quality of interpretations could be improved by utilising the multi-sensory support of the Haptic Workbench.
3. Support for the petrophysical task of examining bore hole and core measures could also be provided. A large amount of data is collected through borehole measures or obtained from examining cores. This includes core scans such as CAT, X-ray and sidewall scans such as acoustic. Well-log data is collected on porosity, mineralogy and fracturing. Borehole imaging is also performed. These data sets are often oriented cylinders of data but are examined in 2D. There are a number of extra data

attributes which could be explored haptically, overlaid on an image of the 3D core. Interpretation comments could be captured and tied to locations like 3D post-it notes. Again this application could be integrated with reservoir and seismic models to improve overall reservoir characterisation.

4. Further downstream in the exploration value chain are applications such as pipeline routing. Haptics could be used to help investigate the best route to shore for an undersea pipeline. The sub-sea surface may be soft and lead to sagging or actively shedding which can also damage pipelines. Shallow sub-surface gas also has potential to damage pipelines during blow-outs. By mapping dip-vector data to vibration, routers would be discouraged from laying pipes in unstable regions while they concentrated on minimising overall pipeline length.

Other suggested applications for haptic feedback in production engineering involve developing general CAD support, for example object collision detection for model design and evaluation.

Additional feedback on the Haptic Workbench was also obtained from BHP Minerals. Some applications in this domain include: discriminating values in aeromagnetic images; improved dynamic range of data display by using higher resolution haptics to "visualise" exploration data, and DC leveling in geophysical line data.

6.0 OUTCOMES

The major outcomes are as follows:

1. Develop a number of demonstrations of haptic workbench technology selected from the candidate domains discussed in section 5.0.
2. Demonstrate the technology to a broad range of BHP staff to determine its relevance and potential impact outside of the petroleum and minerals exploration domains.

7.0 ACKNOWLEDGMENTS

This work was supported by the Advanced Computing Applications Project (ACAP) at BHP Research, Newcastle Laboratories, Australia. We wish to acknowledge the invaluable efforts of our collaborators at CSIRO Mathematical and Information Sciences, Canberra and in particular John McLaughlin who implemented the prototypes. We thank Nick Hoffman and Terry Pritchard from BHP Petroleum for their suggestions and encouragement. We also thank Peter Littlejohn (Corporate General Manager, Corporate Information Systems, BHP) and Gus Ferguson (Chief Technologist, Corporate Information Systems, BHP) for their support as program and project champions, respectively.

8.0 REFERENCES

[1] Bishop, G. et al. "Research Direction in Virtual Environments.", Computer Graphics, Vol 26, No3, pp153-177, August 1992.

[2] Durlach, NI, Mavor, AS. Virtual Reality. Scientific and Technological Challenges. National Academy Press, Washington, DC. 1996.

[3] Sensable Technologies Inc. http://www.sensable.com/

[4] Poston, T and Serra, L, "The Virtual Workbench: Dextrous VR," Proceedings ACM VRST'94, 111-122 (1994).

[5] Massie, T and Salisbury, JK. "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," Proceedings of the ASME Winter Annual Meeting 295-302 (1994).

[6] Polhemus Inc.: http://www.polhemus.com/

[7] McLaughlin, JP and Orenstein, BJ, "Haptic Rendering of 3D Seismic Data," The Second PHANToM User's Group Workshop Proceedings (1997).

Figure 1: The Haptic Workbench. Looking through stereo glasses into a mirror reflecting a computer screen display, the user perceives a virtual object in a work space within hand's reach. One reaches in, holds a tool physically where it appears virtually, and uses it to manipulate a virtual object, in a natural hand-eye coordinated way as one would when handling real objects.

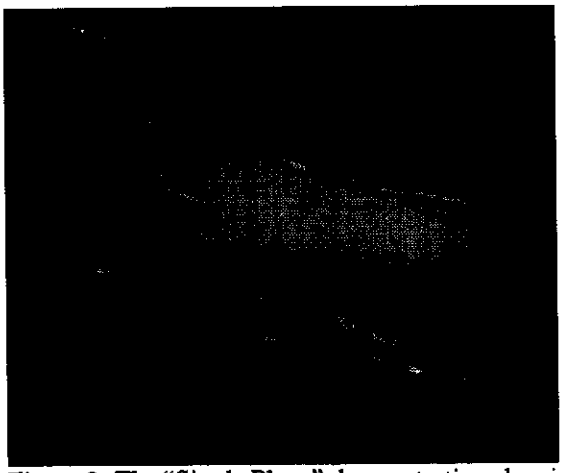


Figure 3: The "Single Plane" demonstration showing a vertical slice of amplitude data from a 3D petroleum seismic survey. Changes in amplitude are mapped to force allowing the user to track along horizons. Points of interest are marked by dropping a sphere. As this is repeated through slices in the volume the spheres can be linked to create a surface. This surface would correspond to a horizon of interest to the interpreter.

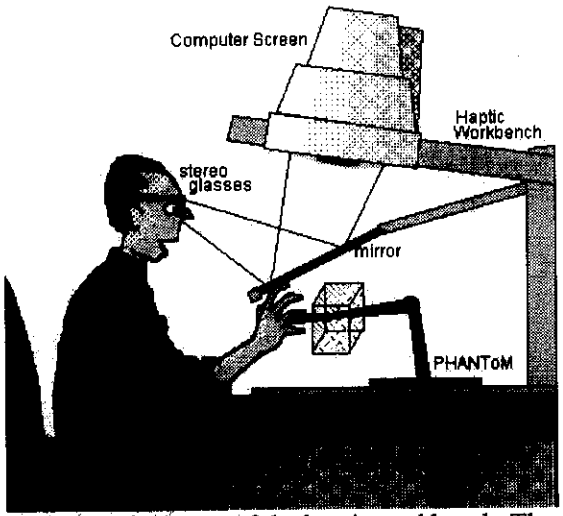


Figure 2: The layout of the haptic workbench. The stereo image is created using shutter glasses. A left and right image is alternatively displayed by the monitor and reflected from an angled mirror to create the illusion of an object in the space below the mirror. The haptic device which is positioned to coincide with this same space can be used to feel and manipulate the 3D objects displayed.

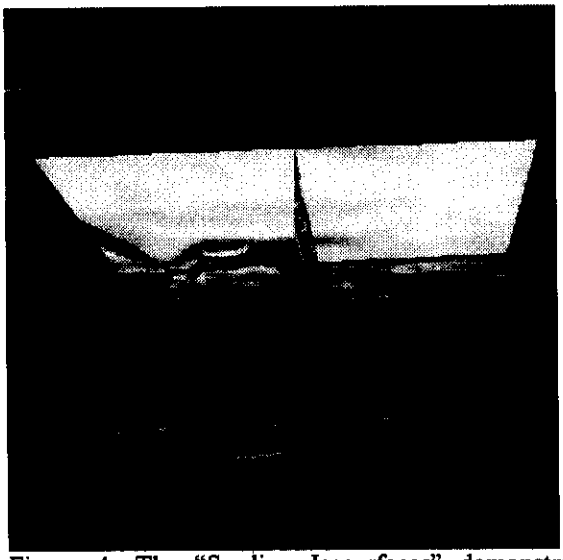


Figure 4: The "Seeding Isosurfaces" demonstration uses a triplanar display and shows an automatically picked fault surface. These surfaces are generated from the semblance processed timeslice. Using the PHANToM™ the user can move along a horizon in one of the vertical planes. Where the horizon intersects a fault surface the user feels a force barrier.

Haptic Rendering of 3D Seismic Data

J P McLaughlin[1] B J Orenstein[2]

[1]CSIRO Mathematical and Information Sciences

John.McLaughlin@cmis.csiro.au

[2]BHP Research

gallimore.randall.rj@bhp.com.au

Abstract

Understanding the terabytes of data produced from offshore seismic surveys is an expensive bottleneck for the petroleum industry. The state of the art currently involves viewing the data in section and as a volume at various levels of interactivity. This paper introduces a method of haptic interaction that overcomes the interactivity constraints of visual methods and provides new opportunities for insight. Forces are calculated directly from the seismic volume data that allow the user to feel features that may be too subtle for visual identification, or occluded by other visual data. A prototype application for the Haptic Workbench [4] allowing haptic and visual interaction with multi-attributed seismic data is described.

1. Introduction

The purpose of this project was to investigate the relevance of haptic devices to the activities of the petroleum industry. Three dimensional seismic data was identified as a likely target for useful haptic applications since the task of interpreting such datasets is currently very time-consuming and hence costly. A 3D visual/haptic multi-attributed seismic volume viewing prototype was developed for the purposes of a scenario demonstration to practising geoscientists from BHP Australia. It investigates several forms of haptic interaction both with the seismic data itself, and with the visual cross-section planes.

The platform used for this project was the Haptic Workbench [4] (figure 1) a hardware system under development at CSIRO Mathematical and Information Sciences in Canberra, Australia. It consists primarily of the Virtual Workbench [2], the PHANToM force feedback device [1], a Silicon Graphics workstation and software libraries developed to support the integrated hardware.

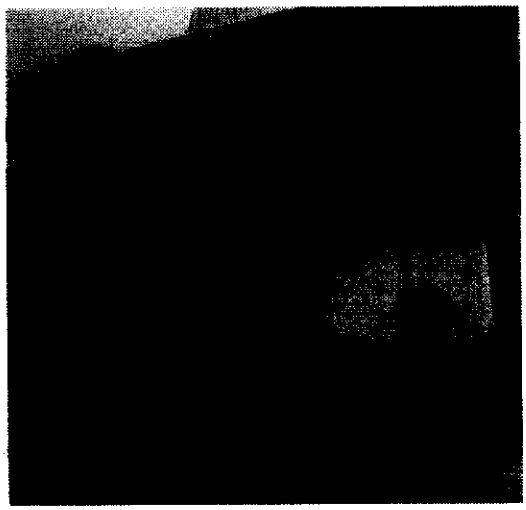


FIGURE 1. JPM at the haptic workbench.

The most significant feature of the Haptic Workbench is the co-location of the haptic and visual models. In figure 1, the user is sculpting a "ball of clay" that appears to be floating in space at exactly the same place that he feels it to be with his hand. Co-location allows exploitation of human proprioception [5], the sense of the position and orientation of one's body and limbs. The user operates on the model (sculpts for example) with their dominant hand (left for JPM) and can re-orient and posi-

tion the model with their non-dominant hand using a (non-haptic) Polhemus magnetic tracking device.

2. Seismic Data

Offshore seismic data is collected in large-scale operations using survey vessels trolling hydrophones (figure 2) . The hydro-

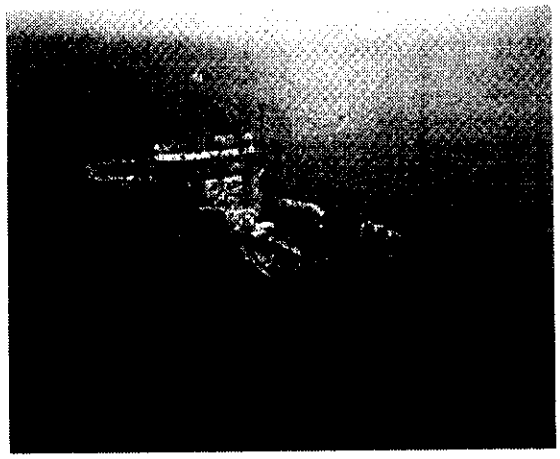


FIGURE 2. A seismic survey vessel.

phones record time-traces of the responses from the seismic explosions, thus building up a grid of scalar voltage values. This data is then cast onto a regular grid using GPS fixes of the hydrophone positions. The result is a sequence of 2D data files containing floating-point values at each point on the grid. For use within this project, the files were quantised into the range 0-255 after viewing the histogram and choosing an appropriate minimum and maximum bound. This data was stored as a sequence of 8-bit SGI image files. These will be referred to as amplitude data.

Also provided in the same file format was a derivative dataset referred to as semblance [6]. These data provide a measure of the local continuity of the amplitude data and are used primarily for identifying faults (slippages) in the sediment which are critical to the interpretation process.

3. Haptic Volume Rendering

Three dimensional seismic datasets differ from volume datasets previously used in haptic rendering (volume based [3] or otherwise) in that they do not define solid shapes. Rather they suggest cloud-like structures with ill-defined boundaries. Also, consideration must be given to methods for rendering multi-attributed volumes simultaneously. As in [3], a gradient-descent method was used to generate forces from a single attribute. We define a *voxel* to be the cubic element of the volume possessing a single value of a single attribute (amplitude or semblance) and *grid points* to be the infinitesimal corners of the voxels.

We first associate with each grid point a gradient vector approximating the direction and magnitude of the attribute's first derivatives with respect to x, y and z. This is calculated from the attribute values in the immediately adjacent voxels (figure 3). This array of gradient vectors is pre-computed at start-up time to reduce computation during the haptic servo loop.

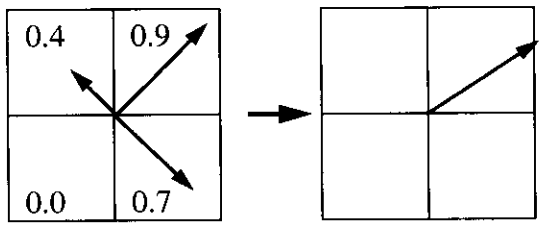


FIGURE 3. 2D analogy of gradient calculation. The 3D case follows trivially. (a) Diagonal component vectors associated with each of the neighbouring voxel values. (b) The sum of these components representing the gradient vector for this grid point.

During the servo loop we first identify the voxel that contains the PHANToM position. We then derive a gradient vector through tri-linear interpolation of the 8 corner grid points. This gradient vector is then scaled appropriately to give a force in Newtons.

4. Multi-Attributed Haptic Volume Rendering

Having calculated a force $\boldsymbol{A}$ for the amplitude data and $\boldsymbol{S}$ for the semblance data, there are several ways these forces could be combined to provide a unique mode of interaction with the seismic volume. The most obvious is a scaled sum

$$F = aA + sS$$

Another is the vector cross-product.

$$F = cA \times S$$

It is not clear whether the cross product would make any sense to the user for this pair of attributes, therefore the scaled sum was used. In order to choose appropriate a and s for this equation it is important to understand the meaning of the two forces $\boldsymbol{A}$ and $\boldsymbol{S}$.

The amplitude force $\boldsymbol{A}$ guides the PHANToM along seismic *reflectors* which are large horizontal regions of minimum amplitude and are of particular interest to the interpreter. They are visually represented as red regions in the volume. The user feels the pen being constrained to the reflector to a degree proportional to the strength of that reflector. If the user pushes hard enough in a vertical direction, the pen will pop out (and possibly into another reflector).

The semblance force $\boldsymbol{S}$ attracts the PHANToM to regions of low continuity (indicative of faulting) and is a fairly sparse force field (it is negligible most of the time). When combined with amplitude forces a repulsive, rather than attractive, force was appropriate. This provides an intuitive feeling of barriers or walls where faulting is evident in the data. The equation used for the combined force mode is

It is, of course, also possible to just feel the amplitude or semblance forces in isolation.

5. Other Haptic Interactions

The seismic data can be rendered visually in full-volume mode with control over transparency. This mode allows viewing the red reflectors as horizontal clouds in space. However this technique causes a low frame-rate which is particularly undesirable in that it lags behind the haptic rendering which runs at 1kHz. An alternative is to render the volume with 3 orthogonal cross-sections, known as inline (the direction the survey vessel travels in), crossline (orthogonal to inline) and time (vertical). The inline and crossline sections bear amplitude data and the time-slice section bear semblance data. This is quite acceptable since geoscientists are already very familiar with viewing these types of section with conventional seismic data software.

An intuitive method of changing the positions of the cross-section planes is required. We present 3 different methods as follows.

1. Clicking and dragging to set the intersection point of the 3 cross-sections. This method is simple, however it requires the user to move all 3 planes at once when they usually only wanted to move one. Also, we wish to reserve the clicker function for interpretive tasks (see below).
2. Constrained movement. The PHANToM is constrained to move within the plane of the cross-section. If the user pushes above a threshold force in an orthogonal direction, the plane will start moving in that direction (static friction). This method allows the PHANToM to move around within the plane clicking to perform an interpretation and pushing in and out to re-position the plane.
3. Plane pushing. The PHANToM can move unrestricted within the volume but when pushing up against a cross-section plane, it feels static resistance.

If pushed hard enough the plane will move away from the PHANToM (static friction). To pull the plane back towards the user, the PHANToM has to be twisted around slightly to push the plane back from the other side.

Method 3 is the most effective for users since it allows free movement within the volume and is more intuitive since it closely resembles real-world positioning tasks. However all 3 methods are available to the user as options.

6. Interpretation

A simple interpretation capability is added to the prototype to complete the demonstration scenario. The task is to "drop spheres" in space to represent points, then create triangles joining these spheres to form a surface. The following modes allow the user to do so without switching modes or using more than the PHANToM's single clicker.

- Clicking close to an existing node initiates dragging of that node. A small inertial effect is added to dragging to provide stability.
- Otherwise clicking and dragging initiates triangle creation by highlighting up to 3 nodes that the PHANToM passes close to during the drag. When the clicker is released a triangle is created between the highlighted nodes. This allows the user to "click and wave" at the 3 nodes of the desired triangle, then release to create the triangle.
- Otherwise clicking and releasing produces a node (small sphere).

Thus the user can push the cross-section planes, feel and view the seismic data either in amplitude, semblance or combined modes, re-orient the volume (with their non-dominant hand) and create an interpretation surface all at the same time and without switching modes.

Finally a marching cubes isosurface construction capability is added. The procedure is to click on a seed point from which a constant-value surface is then grown, often forming a closed envelope. This feature is useful for quickly sketching semblance features such as faults and can be used as an annotation while the semblance data is not visible.

7. Conclusion and Future Work

While feeling semblance data (faults) as barriers it is possible to differentiate between ambiguous visual features. For example, some breaks in the visual red reflectors are due to aliasing artifacts, and some are real faults. By running the PHANToM through this area one can immediately differentiate without cluttering the visual field. Unlike graphic models, haptic models are rendered locally (at the pen position) which potentially allows a much higher resolution than in graphic models. The combined force mode was found to be extremely effective in conveying more than one seismic attribute at once.

Allowing the cross-sectioning planes to be pushed around with a static coefficient of friction was found to be very intuitive. Particularly so in comparison to other packages that require numeric entry or picking and dragging with non-haptic input devices.

Work is continuing through BHP Research on application of the Haptic Workbench to related tasks in the petroleum industry.

8. Acknowledgements

The authors would like to thank Kevin Smith, Duncan Stevenson (CSIRO), Keith Nesbitt, Randall Gallimore and Lawrence Leung (BHP Research) for useful discus-

sions with respect to this paper and the work in general. In particular Don Bone for his insights into force combination (e.g. feeling faults as barriers), the constrained plane mode and for reviewing this paper.

This work was conducted at CSIRO Mathematical and Information Sciences in Canberra, Australia within the Virtual Environments program of the Advanced Computational Systems CRC [7] under funding from and in collaboration with the Advanced Computing Applications Project within the BHP Research Advanced Systems Corporate Program.

9. References

1. T Massie and J K Salisbury, "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," *Proceedings of the ASME Winter Annual Meeting 295-302* (1994).
2. T Poston and L Serra, "The Virtual Workbench: Dextrous VR," *Proceedings ACM VRST'94, 111-122* (1994).
3. Ricardo S. Avila and Lisa M. Sobierajski, "A Haptic Interaction Method for Volume Visualization," *The First PHANToM User's Group Workshop Proceedings* (1996).
4. K N Nesbitt, B J Orenstein, R J Gallimore and J P McLaughlin, "The Haptic Workbench Applied to Petroleum 3D Seismic Interpretation," *The Second PHANToM User's Group Workshop Proceedings* (1997).
5. M R Mine, F P Brooks, C H Sequin, "Moving Objects In Space: Exploiting Proprioception in Virtual-Environment Interaction," *SIGGRAPH 97 Conference Proceedings* (1997)
6. R E Sheriff, "Encyclopedic Dictionary of Exploration Geophysics, Third Edition," *Society of Exploration Geophysicists*, pp264.
7. The Advanced Computational Systems CRC Virtual Environments Program `http://acsys.anu.edu.au/Programs/VE.shtml`

A NOVEL VIRTUAL REALITY SURGICAL TRAINER WITH FORCE FEEDBACK: SURGEON VS. MEDICAL STUDENT PERFORMANCE

O'Toole, R.[1,2], Playter, R.[1], Blank, W.[1], Cornelius, N.[1], Roberts, W.[1], and Raibert, M.[1]

1. Boston Dynamics Inc., Cambridge, MA, USA.
2. Harvard Medical School, Harvard-MIT Division of Health Sciences and Technology, Boston, MA, USA.

INTRODUCTION: We have developed an interactive virtual reality (VR) surgical trainer with force-feedback to simulate the anastomosis of vessels (See Figure 1). The VR surgical trainer assesses skill level during a simulated surgical procedure. The virtual surgical performance is evaluated based upon measures of damage to the tissue, needle technique, and other dynamic parameters. An initial experiment has compared the performance of surgeons and medical students on a simulated surgical task. We found that on average the surgeons performed better than the medical students.

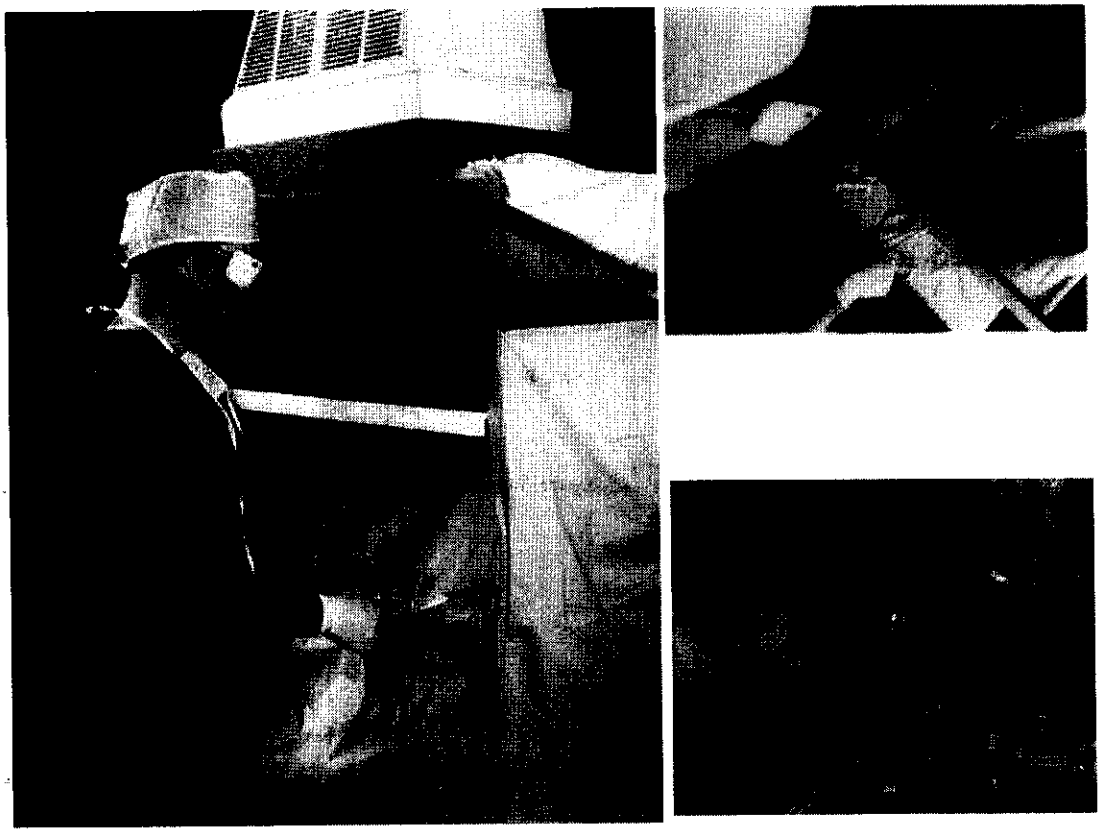


Figure 1. The virtual reality surgical trainer lets you see, touch, and feel simulated organs using 3D computer graphics and advanced force feedback technology. Using a needle holder and forceps attached to force feedback devices, the user can grasp, poke, pluck, and suture flexible tube organs. The user can feel the vessels when touched and the vessels move realistically in response to touch. The surgical station uses a mirror arrangement to place the 3D image of the patient in the correct position relative to the user. This system allows the user to control a virtual needle and thread to perform a simulated end-to-end anastomosis.

CLINICAL RELEVANCE: Virtual reality simulations hold the potential to improve surgical training in a manner analogous to its impact on the training of pilots. The advantages of VR for surgical training include a reduction in operating room time required for training, the quantification of trainees' performances, and an ability to provide experiences independent of a hospital's limited patient population. An interactive virtual reality surgical trainer could improve the training of surgeons and also play a role in the accreditation process.

METHODS: We conducted an experiment to compare the performance of medical students and vascular surgeons using the VR surgical trainer. Both groups were given the task of repeatedly inserting a curved needle through a simulated vessel. Twelve Harvard Medical School students and nine vascular surgeons from five Boston area hospitals participated in the study. We tracked eight parameters during the study: 1) tissue damage, 2) accuracy, 3) peak force applied to the tissue, 4) time to complete the task, 5) surface damage to the tissue, 6) total tool movement, 7) average angular error from an ideal needle path, and 8) an overall error score. The users were given visual feedback regarding their performance after each trial of the experiment.

RESULTS: The surgeons tended to outperform the medical students by having less tissue damage (See for example Figure 2). The surgeons' average performance scores were better than those of the medical students for all of the measured parameters (See Figure 3). These differences were statistically significant ($p<0.05$) for: damage to the tissue, time to complete the task, excessive tool motions, proper angle of the needle, and overall error score.

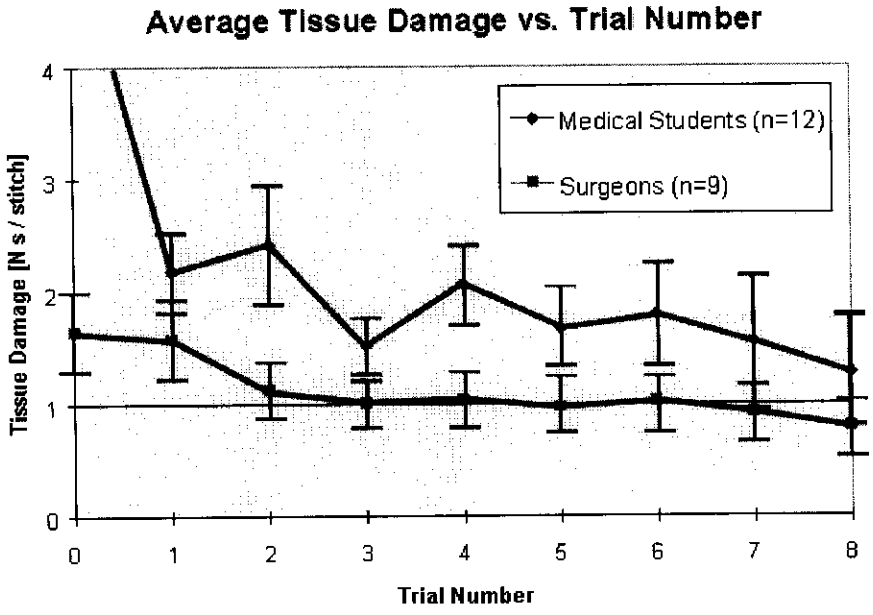


Figure 2. The average tissue damage values are plotted versus the trial number in the experiment. Tissue damage is defined as the sum of forces that exceed a tissue damage threshold.

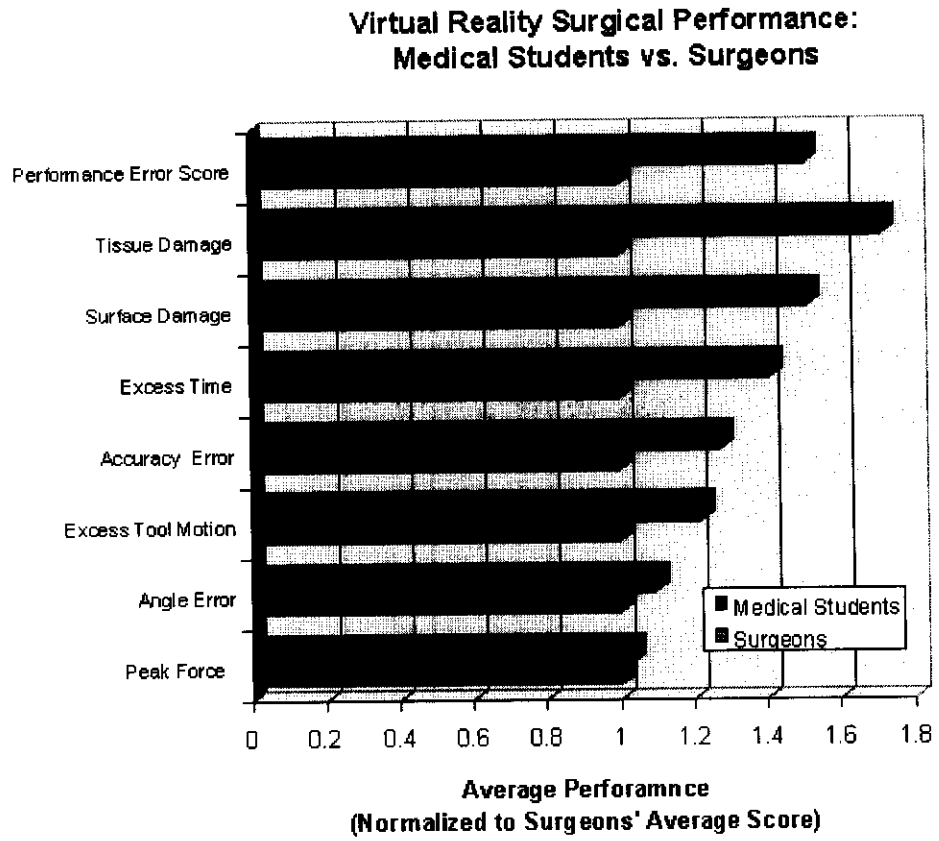


Figure 3. The bar chart displays the average medical student and surgeon performances for the simulated surgical task. The values have been normalized to the average surgeon values. The surgeon values were better than the medical student values for all eight of the metrics. Five of the differences (performance error score, tissue damage, excess time, excess tool motion, and angle error) were statistically significant ($p<0.05$).

CONCLUSIONS: We have developed a VR trainer that tests a user's technique in placing a needle through a simulated vessel. An initial experiment has compared the simulated surgical performance of medical students and vascular surgeons. On average, the surgeons outperformed the medical students in the 8 parameters that were measured (statistically significant for 5 of the parameters) . These results suggest that the VR surgical trainer may be useful in quantifying surgical skill.

REFERENCES: [1] Playter, R., and Raibert, M., A Virtual Surgery Simulator Using Advanced Haptic Feedback, Journal of Minimally Invasive Therapy, To Appear, 1997. [2] Satava, R. M., Medical Virtual Reality, The Current Status of the Future. S. Weghorst, H. Siegurg and K. Morgan Eds., Health Care in the Information Age. IOS Press, Amsterdam, pp. 542-451996. [3] Higgins, G., *et al.*, New Simulation Technologies for Surgical Training and Certification: Current Status and Future Projections, Presence, 6 (2), pp. 160 - 172, 1997. [4] Satava, R. and Jones, S., Virtual Environments for Medical Training and Education, Presence, 6 (2), pp. 139 - 146, 1997.

Some Calibration Information for a PHANToM 1.5 A.

Karl Reinig, Ryan Tracy, Helen Gilmore, Tom Mahalik

University of Colorado Center for Human Simulation

Abstract

At the University of Colorado's Center for Human Simulation (CHS) we have been building fundamental tools for interacting both graphically and haptically with anatomical models derived from the Visible Human (and similar anatomic) data sets. We are also beginning a study of knee alignment that will require flexible and accurate 3-D measurements. This paper tells of a recent effort to turn our PHANToM 1.5 A into an accurate 3-D measuring device.

Introduction

The PHANToM has been an integral part of the various surgical simulator prototypes that we have built over the last two years. These prototypes include needle insertion, surgical cutting, dental, and ophthalmic simulators. In each, the feel of interacting with the virtual tissue seems at least as important as the visual cues. The high resolution of the PHANToM, both in force and position, make it ideal for creating the subtle differences in tactile response required of the simulators.

Our mission has recently expanded to include measuring relevant landmarks (as they move) of pathologic joints in an attempt to help orient the surgeon in the operating room. There is a lot of preliminary work to be done and a device that can digitize with sub-millimeter accuracy across a workspace the size of the PHANToM 1.5 would be very handy.

SensAble Technologies I/O library makes it easy to receive 3-D positional information from the PHANToM. However, the accuracy of the position reported by our PHANToM through its I/O library is not as good as a device with such high resolution and repeatability could be. We found that measurement along a single axis could be off by more than 3/4 inch across the workspace. This is not generally a problem in haptics since accuracy is rarely an issue.

We decided to see if we could generate a correction grid for the PHANToM that would give us sub-millimeter accuracy across its workspace. This would then be used to give us an accurate 3-D digitizer, as well as improve the accuracy of our simulations.

Assumptions

Without delving into the encoders or the transforms that take their measurements to 3-D space, we make the following assumptions: 1) Locally, the encoders and transforms can resolve to far less than .1 mm. 2) The current errors of the PHANToM are repeatable to below .1 mm. 3) As long as the PHANToM is reset in the same neutral position, its motors may be swapped, cables replaced, etc., without changing the repeatability of the errors to within .1 mm. In fact, it would seem that the correction grid that we are attempting to create could be used on any other PHANToM 1.5 A. Our .1 mm assumption is rather arbitrary. It simply means that we assume the error is less than other errors which will be introduced during our measurements.

Experimental Setup

The ideal setup would compare a known accurate 3-D measuring system with measurements from the PHANToM at uniform grid coordinates spanning the PHANToM workspace. The result would be a grid which can be sampled and interpolated directly by the PHANToM. Unfortunately we do not have access to an accurate 3-D measuring device. Instead, we placed the PHANToM next to a 2-D numerically controlled table. For the third dimension we clamped a straight edge (ruler) perpendicular to the plane of the two controlled axes. The task then became to drive the PHANToM tip to a grid coordinate and move the numerically controlled table until the tip touched the upper right edge of the ruler. Measurement of two axes then came directly from the table's digital readout. The third dimension was measured by eyeballing the tip with the ruler.

We wrote a simple program which expands 3-D integer coordinates into PHANToM coordinates spread 100 mm apart. Our first attempt to use 50 mm increments (approximately 300 measurements) resulted in blurry eyes and unpredictably unreliable readings. The PHANToM was tied to the coordinates by a simple spring model. Each new input coordinate was "phased in" over 20000 cycles (approximately 2 seconds). The haptic was essentially transformed into a traditional position seeking robot. The PHANToM tip could then be driven to each grid coordinate within its workspace.

To be used as a measuring device it is important that the PHANToM be initialized in as close to the same orientation as possible. Our machine shop built an aluminum "arm" which reaches from the base of the PHANToM to its gimbal when the gimbal is in the neutral position. Pushing the gimbal against an interior three-sided corner completely fixes its state.

After initializing the system, we removed the gimbal and replaced it with a measuring tip whose length was adjusted to place the tip at the normal intersection of the gimbal encoders. This resulted in a small area of contact between the PHANToM and the ruler and removed the gimbal from the measurements.

Driving the PHANToM beyond its work envelope is an inefficient and short-lived method of heating a room. Our software allowed us to move the PHANToM tip by hand when nearing a grid point on the edge of the workspace. This allowed us to be assured that the point was reachable before making it the set point for the PHANToM tip.

Sources of Error

The setup described above injects at least the following errors into the correction grid: Inaccuracy of the NC table, ruler reading resolution, the difference between the ordered and actual haptic position, and initialization of the neutral position. We assume the accuracy of the NC table to be considerably better than one tenth millimeter. We have no way of checking this directly but the "etched glass" system maintains a reading accurate to 5/10,000 of an inch regardless of how it got there. It is the same table used to create the Visible Human data. The ruler resolves to 1/64th inch. Measurements were made to the nearest tick introducing at least 1/128th inch error or roughly .2 mm. If the person counting the tick marks were to be off by one, the error would jump to .6 mm. The PHANToM was nearly neutrally weighted throughout its workspace. We monitored the set point verses the reported PHANToM position. The components orthogonal to gravity were always within .1 millimeter of their set point. The other component (the y component) never varied more than .16 millimeters from its set point.

The Correction Grid

We took measurements for 49 grid points spread uniformly (separated in each dimension by 100 mm in uncorrected PHANToM Coordinates) throughout the PHANToM's workspace. To utilize the measurements we simply truncated the PHANToM's perceived location to determine a starting grid coordinate. The fractions truncated were used to create a convex hull of weights for the 8 surrounding grid points. These weights were multiplied by the measured values associated with the grid locations to return a corrected PHANToM position. This sort of algorithm is very fast and not likely to effect any of our haptics algorithms.

Expanding the Correction Grid

Because of the low resolution of our grid a fair amount of the PHANToM's work volume was left outside the grid. In order to take measurements throughout the entire workspace we augmented the grid points. This was done in a primitive way. We transformed the measurements associated with each grid point from absolute coordinates to correction values, i.e., the vector which should be added to the PHANToM's perceived location to make it agree with the measured value. Then working from the center out we identified grid points which had not been assigned a measure and assigned to them the average of the corrections of however many neighbors it had with assigned values. This completely ignored any trends building along adjacent grid points, a likely place to improve the system in the future.

Results

The correction grid created here may be downloaded from our web site, http://www.uchsc.edu/sm/chs/haptics. The following table shows the truncated grid location, the corrected PHANToM position, and the position as measued by our "ad hoc" yet presumed accurate measuring system for a few points. All measurements were in inches. The 1st, 3rd, and last row came from grid locations beyond the range of the measured grid. The worst case appears to be the X axis of row 1, where the "Actual" and corrected PHANToM Measurements differ by 0.1668 inches.

Truncated Grid			Corrected PHANToM Meas			"Actual" location		
1	2	1	3.2898	4.1273	7.671	3.123	4.136	7.641
4	2	2	12.0297	4.126	9.4948	11.986	4.136	9.5156
6	2	0	22.4348	4.1166	2.6968	22.497	4.137	2.7344
5	2	0	18.872	4.134	3.828	18.842	4.137	3.8125
5	2	0	18.743	5.752	3.623	18.716	5.754	3.5940
4	3	0	15.7513	8.5060	4.5182	15.736	8.504	4.484
2	3	1	6.497	9.667	8.334	6.457	9.709	8.234

Conclusions

Our simple attempt at calibrating our model 1.5 A PHANToM resulted in apparent accuracy of approximately .004 inches for each axis when the tip is within the calibration grid. Our attempt to extrapolate beyond the grid using a simple averaging scheme was not as productive as we'd hoped. A case was found at the extreme of the PHANToM's workspace in which one axis had an error of .1668 inches.

Nothing in our experiment indicates that we pushed the resolution or repeatability of the PHANToM. Our intent is to continue to improve the effectiveness of our correction grid until we achieve sub millimeter accuracy throughout its workspace. This can probably be achieved by a combination of measuring more points near the edge of the workspace and creating a more intelligent extrapolation of the grid.

DEVELOPING A PC-BASED SPINE BIOPSY SIMULATOR

Kevin Cleary* , Corinna Lathan, R. Craig Platenberg***,**
Kevin Gary**, Fred Wade*, Laura Traynor****

**Imaging Sciences and Information Systems (ISIS), Radiology Department,*
Georgetown University Medical Center, 2115 Wisconsin Avenue, Washington, DC, 20007.
Email:cleary@isis.imac.georgetown.edu

***Biomedical Engineering, The Catholic University of America. Email: lathan@pluto.ee.cua.edu*

****MRI Center, Radiology Department, Georgetown University Medical Center*

***** Computer Science and Engineering Department, Arizona State University*

1 INTRODUCTION

Surgical simulation is a rapidly expanding field that uses computer graphics to simulate surgical procedures. Surgical simulators can be used for medical education and training, surgical planning, and scientific analysis including the design of new surgical procedures. Surgical simulators have the potential to revolutionize medical training in much the same manner that flight simulators revolutionized aeronautical training. This paper presents a spine biopsy simulator under development at Georgetown University Medical Center for training purposes. The paper describes the spine biopsy procedure, presents the simulator system architecture, discusses some software development issues, and details the training protocol.

This simulator project has been undertaken in cooperation with the radiology department and an interventional radiologist. This project was chosen as a good starter application for investigating surgical simulation, and represents an area of demonstrated clinical need. A related simulator effort is at the University of Colorado [Reinig 1996], where a needle insertion simulator was constructed to help train anesthesiologists to do celiac plexus blocks. Other surgical simulator efforts incorporating force feedback include Delp [1997], Gibson [1997], Playter [1996], and Preminger [1996]. The key characteristics of these systems and several others are reviewed in Cleary [in press].

A brief description of the procedure is as follows. In CT-directed biopsy, the patient lies prone on the CT table, and an initial CT scan is done. The doctor then selects the best slice to reach the lesion, and the entry point on the patient is marked. The skin is anesthetized over the entry site, and the needle is placed part way in. Another scan is done to confirm the needle position, and the procedure continues by inserting the needle further and rescanning as needed to verify needle position. When the lesion is reached, a core sample is removed and sent for pathologic analysis.

2 SYSTEM ARCHITECTURE

A block diagram of the system architecture is shown in Figure 1 and includes the following components: computer, computer software, image database, graphics card and display monitor, and physical interface.

The image database is responsible for storing and managing the images used in the simulator. All images are stored in the DICOM file format. DICOM stands for Digital Imaging and Communications in Medicine, and it allows images to be exchanged between different medical

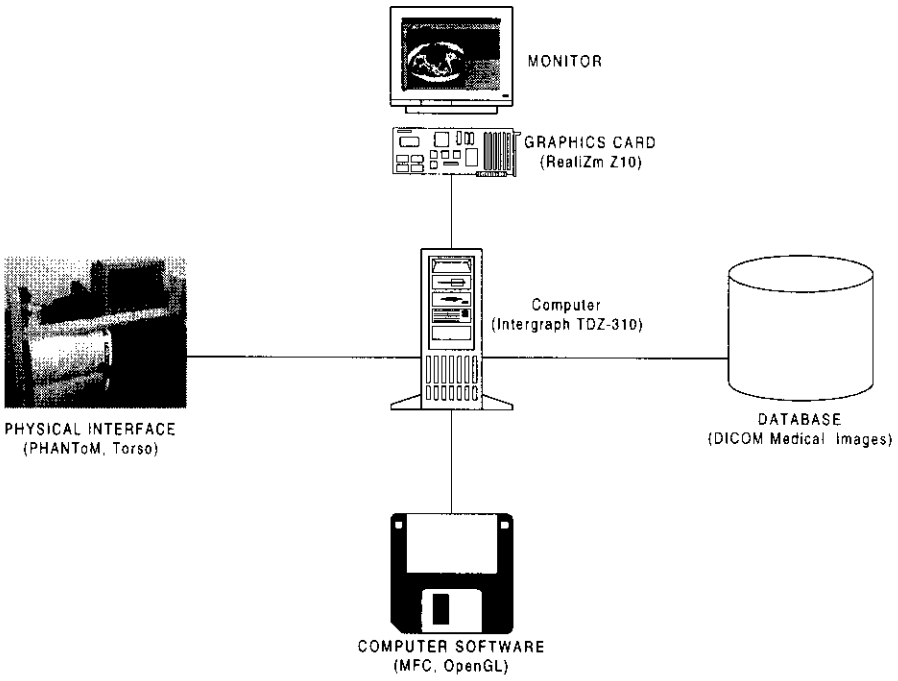


Figure 1 *Spine Biopsy Simulator Components*

imaging modalities such as computed radiography (CR), computed tomography (CT), and magnetic resonance imaging (MRI).

The computer is the brains of the simulator and handles tasks such as interfacing with the operating system, servoing the force reflecting joystick, and refreshing the graphics display. The system is an Integraph 310 unit with a RealiZm Z10 PCI graphics board running Windows NT 4.0. This system is a perfect candidate for a personal computer implementation since the computing requirements are not that demanding and the graphics are two-dimensional. The RealiZm Z10 is OpenGL based, has 12 megabytes of frame-buffer memory, and supports resolutions up to 1 Mpixels.

The physical interface consists of a dummy human torso and a PHANToM force reflecting joystick from SensAble Technologies. A photo of the physical interface is shown in Figure 2.

3 SOFTWARE DEVELOPMENT

The software is being developed using Microsoft Visual C++ 5.0 on Windows NT and Windows 95 computers. The following software libraries are used:

- Microsoft Foundation Classes (MFC) for the user interface
- OpenGL for rendering the images and graphics constructs
- BasicIO library for interfacing with the PHANToM

One goal of the software development effort was to make the software as portable as possible. In particular, we might want to migrate the system to a Silicon Graphics Unix-based environment at some point, particularly if we require complex 3-D graphics. Therefore, we initially looked only at software libraries that were supported on both Windows '95 and Silicon Graphics platforms,

Figure 2 *Physical Interface: Torso and Joystick*

such as OpenInventor[1]. However, OpenInventor provides only limited user interface elements, so a decision was made to use MFC for the user interface. A properly designed user interface is critical to the ease of use of the system, and MFC eases the coding task. If the software needs to be migrated to the Silicon Graphics in the future, the user interface elements will need to be rewritten, most likely using the X-Window/Motif libraries. However, since OpenGL was used for the rendering and graphics constructs, this code will be portable to the Silicon Graphics.

The BasicIO library from SensAble Technologies is used to interface with the PHANToM. PHANToM related data is encapsulated in a PhantomWrapper class, which wraps some of the BasicIO C function calls in C++ methods. These methods include routines to initialize, start, and stop the PHANToM device. Each of these routines are invoked in response to some user interface action. A real-time log has also been developed, in which PHANToM positions and forces for each servo loop iteration are stored in memory for some amount of time (limited by the available memory of course). Then, when the PHANToM servo loop is stopped, this log can be written to disk for analysis. We have found this log essential for investigating how the PHANToM is actually responding during our simulation.

A screen capture of the user interface in its current state of development is shown in Figure 3. The user can page up / page down through the CT slices. The vertical toolbar at the left of the screen will be used to select the biopsy location and activate the joystick.

[1] Http://www.sgi.com/Technology/Inventor

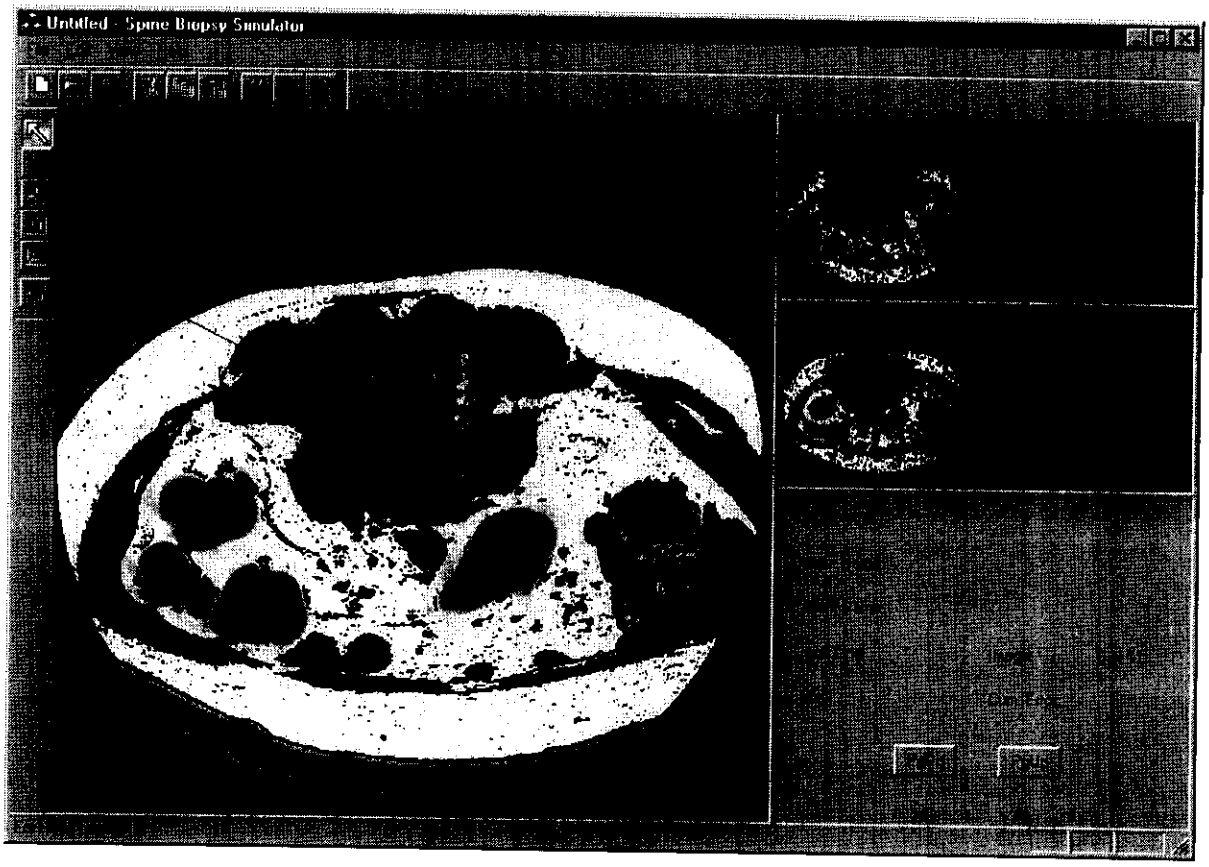


Figure 3 *User Interface*

4 TRAINING PROTOCOL

The training protocol is designed to mimic the existing procedure as closely as possible. The proposed procedure is as follows:

1. Subject starts training run by selecting a CT case from those available.
2. Subject pages through the CT slices and selects the best slice for accessing the biopsy location. Feedback is given as to the correctness of the choice.
3. On the selected slice, subject chooses a skin entry point and biopsy location point. The computer draws a line between these two points, and gives feedback on the suitability of these choices.
4. Until this point in time, the subject has been working on the computer. The force reflecting joystick is now turned on, and the subject moves to the torso for the remainder of the procedure.
5. Subject inserts the needle and feels appropriate forces. The motion is also tracked on the computer screen.
6. When the procedure is complete, subject is evaluated on how close the final position was to the target point.

5 SYSTEM STATUS AND FUTURE PLANS

The initial version of the system software will be completed shortly. We have already begun working with a radiology fellow to evaluate the system and the realism of the force feedback. At the moment we are using simple algorithms to provide the force feedback, but we anticipate this

will be an area of substantial development in the future. The next step will be to perform training tasks and evaluate the training effectiveness of the system. In the long range, we hope to extend the technology developed here to investigate real-time image guided biopsy, including the use of three-dimensional reconstructions in biopsy procedures.

ACKNOWLEDGEMENT

The authors would like to acknowledge the clinical input from Matthew Freedman of the ISIS Center. Thanks are also due to Chin-Shoou Lin of the MRI Center for assistance in obtaining the medical images. The authors appreciate the excellent customer support provided by SensAble Technologies and software discussions with Walter Aviles and Chris Tarr of SensAble. This work was funded by U.S. Army grant DAMD17-96-2-6004. The content of this manuscript does not necessarily reflect the position or policy of the U.S. Government.

REFERENCES

Cleary, K., C. Lathan, R.C. Platenberg (in press). "Surgical Simulation: Research Review and PC-Based Spine Biopsy Simulator", to be presented at the 2nd Workshop on Medical Robotics (wwwipr.ira.uka.de/~iarp/) and published in the proceedings, Nov. 10-12, 1997, Heidelberg, Germany.

Delp, S., P. Loan, C. Basdogan, et al. (1997). "Surgical Simulation: An Emerging Technology for Training in Emergency Medicine", *Presence*, Vol. 6, No. 2, pp. 147-159.

Gibson, S., J. Samosky, A. Mor, et al. (1997). "Simulating Arthroscopic Knee Surgery Using Volumetric Object Representations, Real-Time Volume Rendering and Haptic Feedback", *CVRMed-MRCAS*, pp. 369-378, Springer-Verlag.

Playter, R., B. Blank, N. Cornelius, et al. (1996). "Integrated Haptics Applications: Surgical Anastomosis and Aircraft Maintenance", *Preprints of The First PHANToM User's Group Workshop*, Sept. 27-30, Cambridge, MA.

Preminger, G., R. Babayan, G. Merril, et al. (1996). "Virtual Reality Surgical Simulation in Endoscopic Urologic Surgery", *Medicine Meets Virtual Reality 4*, pp. 157-163, IOS Press.

Reinig, K. (1996). "Haptic Interaction with the Visible Human", *Preprints of the First PHANToM User's Group Workshop*, Sept 27-30, Cambridge, MA.

The Phantasticon

The PHANToM for Blind People

Carl Sjöström, Ph D student
CERTEC, Center for Rehabilitation Engineering Research at Lund University,
Calle.Sjostrom@certec.lth.se

Abstract

This paper deals with CERTECs work to use haptics to provide a new way of using computers for visually impaired people and people with physical disabilities. We have chosen to call the concept "The Phantasticon".

CERTEC has developed a set of programs for the Phantasticon. The programs are working units themselves, as well as a basis for further development. We have developed a program for making mathematics easier to understand, a tactile based battleship game and a painting program with associated colors and structures for visually impaired people.

The work in progress is to gain more experience about haptic user interfaces for blind people and to make the Windows environment accessible via haptics.

Introduction

Computers are becoming everyday technology for more and more people. Computers have opened up many doors for disabled people, for example it is now rather easy for a blind person to access written text. Any text in a computer can be read either with a one row Braille-display or a speech synthesizer. This is done in real time and is of course much more flexible and less space consuming than books with Braille-text on paper. There is a big problem though: Since the introduction of graphical user interfaces computers are getting easier and easier to use for sighted people, but GUIs has the opposite effect for non-sighted people. The many fast accessible objects on a Windows desktop become a big mess for a user who can't see them. And if you don't know what is on the screen it is almost impossible to control the computer.

This is where the haptic interfaces can be a good help. With the PHANToM or a similar device it is possible to *feel* things represented in a computer. CERTEC has developed a set of programs that demonstrate different ways for a blind user to control a virtual reality with finger movements and to get feedback via the sense of touch. One of the big tasks for the future is to make the Microsoft Windows environment completely accessible through haptics. If you can feel the START-button in the lower left corner etc. it is not only possible to control the environment, but it is also possible to start speaking about how to do things since both sighted and non sighted users have a common ground to start from.

CERTEC and the Phantasticon

CERTEC is working with the meeting between human needs and technical possibilities. Normally we start with the human needs and develop technical solutions from that aspect. Sometimes though it is motivated to start from the other side.

In this case we have used a technical solution which has been developed for other purposes

and modified it to correspond to the needs of a disabled person. We have turned the PHANToM into the Phantasticon.

When the PHANToM and the CERTECs software are used to meet the needs of disabled persons it becomes a complete system. This system is what we call the Phantasticon (Fantomaten® in Swedish).

Software development for the Phantasticon

CERTEC is continuously developing programs for the Phantasticon. Our first steps with the PHANToM have resulted in these programs:

Paint with your fingers

A program with which the user can paint computer pictures with a finger. One can choose a color from a palette and paint with it on the screen. The harder you push with your finger, the thicker becomes the line. Each color has an individual structure. When you are painting you can feel the structure that is being painted. You can also feel the structure of the whole picture by changing program mode with a simple click on the space key.

Mathematical curves and surfaces

People who try to explain mathematics for blind persons often notice that mathematics is a partially visual subject. With the help of the Phantasticon also blind persons can learn to understand equations as curves and surfaces. CERTEC has developed a program that makes it possible to feel any mathematical curve or surface with the PHANToM.

"Submarines"

"Submarines" is a PHANToM variant of the well-known battleship game. The player can feel 10x10 squares in a coordinate system. In the game your finger is a helicopter which is hunting submarines with depth charge bombs. If you put your finger on the "water surface" you can feel the smooth waves moving up and down. The surface feels different after you have dropped a bomb, and it also feels different if a submarine has been sinked.

This computer game uses the PHANToM, the screen and the keyboard for the interaction with the user. It also uses sound effects as most games do nowadays. "Submarines" is one of the first computer games that can be played by a deaf blind person.

Work in progress

The big efforts at this moment are laid on developing a general user interface that is easily accessible for blind people. As a test bench for haptic interface objects and at the same time a haptic computer game we have developed *Haptic Memory*. The task for the user is to find pairs of sounds that are played when the user pushes different buttons. The Memory program is a good base to find out how different parts of a haptic interface should be designed to work as good as possible for low vision users.

The Haptic Memory has also been expanded into "the Memory House". The Memory House is CERTECs "first steps towards a Haptic Operating System". The Memory House is a bigger haptic memory with five floors and five buttons on each floor. With this program we can gain some experience about how blind persons can use haptics to build inner pictures of complex

environments. That knowledge is an important cornerstone when we start building the complete haptic windows system or other haptic programs for visually disabled people.

"The Memory House" has been tested with four children and five adults. All of them are blind. A reference group of 21 children in the age of 12 has tested a program with the same function and layout, but with a graphical user interface. Since the haptic interface has the same layout as the graphical interface and both programs work exactly the same except for the way of interacting with the user it is possible to compare the results of the blind users with the results of the sighted users. All of the blind testers had a little more than one hour of experience with the phantom before the test started.

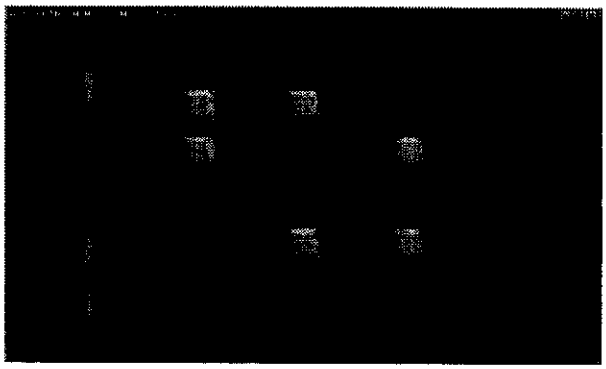


Figure 1. The Memory House

The tests show that a majority of the blind users could complete the task using about as many button pushes as the sighted users, but the blind users generally needed more time. The differences in the results where bigger in the group with blind users and two of them did not finish the task at all.

The results imply that it is meaningful to keep trying to make graphical user interfaces accessible for blind people using haptic technology. Most of the blind users showed big confidence when using the haptic interface even with the rather limited experience they had and the differences in time will probably be lower after more training.

"The Memory House" has been programmed in C++ using Sensable´s GHOST toolkit.

"Touch Windows"

As mentioned in the introduction there is a big problem for non-sighted users that computer nowadays mostly have graphical users interfaces. However, Windows and other graphical user interfaces are widespread and accepted, so almost all new programs are made for those environments. The good thing is that if one can make Windows accessible for non-sighted users then almost automatically the Windows programs are accessible as well. That is why we have started the "Touch Windows" project.

Our idea is that haptics with the PHANToM can provide the overall information and that sound, speech synthesis and Braille-displays can provide the details of the interface. In this way it is possible to use the same or at least similar interface metaphors for both sighted and non-sighted users. People can work together and help each other and it doesn't make a big difference whether you see what's on the screen or if you feel it, you get the same inner picture anyway.

Conclusions

One of the key elements in the process of learning is variation. This implies that an already existing knowledge is a vital precondition for new knowledge. For a blind person, vital parts of the experiences of the world around them are touch based from the very beginning. Thus, it is a most exhilarating possibility to use the PHANToM for widening blind people's touch-based

experiences and learning outside an armlenght's distance. So far, we have not come across any unclimbable obstacles.

URLs for reference

http://www.certec.lth.se/index_e.html-ssi
http://www.certec.lth.se/research/projects/fantomat/phantasticon.html
CERTECs homepage and our page about the Phantasticon

Contact persons

Carl Sjöström
Bodil Jönsson Ph D

CERTEC, Lund University
Box 118
S-221 00 LUND
Sweden

Tel: +46-46-222 46 95
Fax: +46-46-222 44 31

Second PHANToM User's Group Workshop
October 19-22, 1997

Attendee List

NAME:	COMPANY	EMAIL ADDRESS::
Abrahams-Gessel, Daniel	Dartmouth College, Computer Science Dept.	gessel@cs.dartmouth.edu
Anderson, Tom	Sandia National Laboratories	tganders@u.washington.edu
Aulet, Bill	SensAble Technologies, Inc.	aulet@sensable.com
Aviles, Walter	SensAble Technologies, Inc.	aviles@sensable.com
Bardasz, Ted	SensAble Technologies, Inc.	tbardasz@sensable.com
Basdogan, Cagatay	M.I.T. Research Lab. of Electronics	basdogan@mit.edu
Ben-Ur, Ela	M.I.T., Mechanical Engineering Dept.	elabenur@mit.edu
Boatwright, Bill	SensAble Technologies, Inc.	bill@sensable.com
Brederson, J. Dean	University of Utah, Dept. of Computer Science	jdb@wiggy.cs.utah.edu
Burgin, Jon	Texas Tech University, Computer Science Dept.	jburgin@cs.ttu.edu
Cleary, Kevin	Georgetown University, Medical Center Dept. of Radiology	cleary@isis.imac.georgetown.edu
Cooke, Anne	Fidelity Ventures	Anne.Cooke@fmr.com
Davidson, Scott	NAWC/TSD	scott@rdvax.ntsc.navy.mil
De, Suvranu	M.I.T., Mechanical Engineering Dept. and Research Lab. of Electronics	suvranu@mit.edu
De Angelis, Franco	University di Parma, Viale delle Scienze	fda@ied.eng.unipr.it
Ernst, Marc	Max-Planck-Institut fuer biologische Kyberntick	marc.ernst@tuebingen.mpg.de

Feldberg, Ian	Johns Hopkins University, Computer Science Dept.	ief@aplcomm.jhuapl.edu
Gallimore, Randall	Newcastle Laboratories	gallimore.rancall.rj@bhp.com.au
Green, Donald	M.I.T., Mechanical Engineering Dept.	dfg@ai.mit.edu
Green, Greta	SensAble Technologies, Inc.	gretag@sensable.com
Goldstein, Rich	SensAble Technologies, Inc.	richg@sensable.com
Gorman, Paul	Pennsylvania State University M.S. Hershey Medical Center	pgorman@prs.hmc.psu.edu
Grabowski, Nikolai	University of Delaware Electrical Engineering Dept.	grabowsk@ee.udel.edu
Gray, Steve	DRK Advertising	sgray@gray.com
Halle, Mike	M.I.T. Media Laboratory	halle@media.sensable.com
Ho, Chih-Hao	M.I.T. Mechanical Engineering Dept. and Research Lab. of Electronics	chihhao@mit.edu
Harada, Masayuki	Mitsubishi Electronic Corp. Information Technology and Research & Development Center	haradam@pif.isl.melco.co.jp
Hou, Alexandra	M.I.T. Mechanical Engineering Dept. and Research Lab. of Electronics	hou@mit.edu
Jansson, Gunnar	Uppsala University, Dept. of Psychology	Gunnar.Jansson@psyk.uu.se
Kasabian, David	DRK Advertising	davek@drk.com
Kasabian, Pam	SensAble Technologies, Inc.	pamk@sensable.com
Kobayashi, Hiromi	Nissho Electronics Corp.	kbhiromi@kk.iij4u.or.jp
Kolarov, Krasimir	Interval Research Corp.	kolarov@interval.com
Larsson, Peter	Prosolvia Clarus AB	peterl@clarus.se

Lathan, Corinna	The Catholic University of America, Dept. of Mechanical Engineering	lathan@pluto.ee.cua.edu
Latimer, Craig	M.I.T. Mechanical Engineering Dept.	lat@ai.mit.edu
Maclean, Karon	Interval Research Corp.	maclean@interval.com
Massie, Thomas	SensAble Technologies, Inc.	thomas@sensable.com
Miller, Timothy	Brown University,	tsm@cs.brown.edu
McLaughlin, John	CSIRO	John.McLaughlin@cmis.csiro.au
Minsky, Margaret		marg@media.mit.edu
Morgan, Fritz	Mitsubishi Electric ITA	morgan@merl.com
Mufti, Yasser	Rochester University, Dept. of Computer Science	yasserm@cs.rochester.edu
Mukai, Nobuhiko	Mitsubishi Electric Corp. Information Technology and Research & Development Center	mukai@isl.melco.co.jp
Nagano, Yasutada	Mitsubishi Electric Corp. Information Technology and Research & Development Center	naganoy@pif.isl.melco.co.jp
Nakajima, Yoshikazu	Mitsubishi Electric Corp. Information Technology and Research & Development Center	y-nakaji@pif.isl.melco.co.jp
Ottensmeyer, Mark	M.I.T. Mechanical Engineering Dept.	canuck@ai.mit.edu
Pai, Dinesh	University of British Columbia Dept. of Computer Science	pai@cs.ubc.ca
Playter, Rob	Boston Dynamics, Inc.	playter@bdi.com
Plesniak, Wendy	M.I.T. Media Laboratory	wjp@media.mit.edu
Reinig, Karl	University of Colorado Center For Human Simulation	karl@pixsun.uchsc.edu
Ranta, John	SensAble Technologies, Inc.	jranta@sensable.com

Ruspini, Diego	Stanford University, Dept. of Computer Science	ruspini@robotics.Stanford.edu
Salisbury, Kenneth	M.I.T. Mechanical Engineering Dept. and Artificial Intelligence Lab.	jks@ai.mit.edu
Seeger, Adam	University of North Carolina, Computer Science Dept.	seeger@cs.unc.edu
Shaw, Rob	Interval Research Corp.	shaw@interval.com
Sjöström, Calle	Lund University, CERTEC	Calle.Sjostrom@certec.lth.se
Srinivasan, Mandayam	M.I.T. Mechanical Engineering Dept. and Research Lab. of Electronics	srini@mit.edu
Tarr, Chris	SensAble Technologies, Inc.	ctarr@sensable.com
Temkin, Bharti	Texas Tech University, Dept. of Computer Science	temkin@ttu.edu
Verplank, Bill	Internval Research Corp.	verplank@interval.com
Wendlandt, Jeff	Mitsubishi Electric ITA	wents@merl.com
Wu, Wan-Chen	M.I.T. Mechanical Engineering Dept. and Research Lab. of Electronics	wcwu@mit.edu
Young, Peter	Colorado State University, Dept. of Electrical Engineering	pmy@engr.colostate.edu