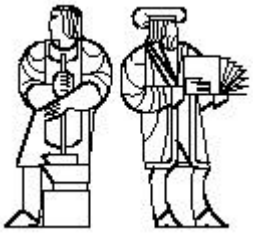
The Artificial Intelligence Laboratory and
The Research Laboratory of Electronics
at the
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Proceedings of the
Fourth
PHANTOM Users Group Workshop

Edited by:
Dr. J. Kenneth Salisbury and Dr. Mandayam A. Srinivasan

A.I. Technical Report No. TBD
R.L.E. Technical Report No. TBD

October 1999



Massachusetts Institute of Technology

Proceedings of the

Fourth

PHANTOM Users Group Workshop

October 9 – 12, 1999
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 Fourth PHANTOM Users Group Workshop (PUG99)
October 9-12, 1999

Program and Schedule

Saturday October 9 - ARRIVAL AND SETUP

Sunday October 10 - TUTORIALS AND DEMONSTRATIONS

Monday October 11 - SYMPOSIUM ON SCIENTIFIC ISSUES

□□□

Tuesday October 12 - SYMPOSIUM ON APPLICATION ISSUES

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Using Fast Local Modelling to Buffer Haptic Data

Remis BALANIUK
Universidade Católica de Brasilia - Brasil
SHARP/GRAVIR INRIA Rhône Alpes - France
Email : Remis.Balaniuk@inrialpes.fr

Abstract

It is well known that realistic haptic rendering depends on high update rates. Nevertheless, there are many applications for which haptic interfaces could be useful where haptic data cannot be delivered at these rates. In VR and real-time dynamic simulation of deformable objects, for instance, update rates are limited by the computation time. In distributed systems, as telerobotics or network-based simulators, communication rates are unstable and limited by the network features. In this paper we show how this problem can be minimised using a "buffer model" between the haptic interface and the data source (the system being interfaced). This buffer is intended to be able to deliver data to the haptic device at the required rate and its role is to maximise the information contained in the data flow. The buffer is implemented as a local numerical model of the data source. The buffer model must be simple, generic and easily adaptable. The data flow will be used to continuously update a number of parameters of this model in order to render it locally as close as possible of the data source. We will show also how a buffer model can be useful to easily implement a proxy-based calculation of the haptic forces. We will finish showing how we used this approach to interface a PHANToM with a deformable objects simulator.

1. Introduction

We are interested in simplifying the conception of a haptic interface in an existing Virtual Reality (VR) system.

In a haptic interface the force feedback represents the contact forces between the haptic probe and the objects of a virtual scene. This physically based scene will typically be simulated in real-time by a VR software component. The instantaneous contact forces depend directly on the instantaneous scene state, and the scene state can be affected by the user actions. Consequently, the haptic systems are usually organised in a way that for each haptic refresh cycle we have also a virtual scene update. In this configuration, the haptic rate depends entirely on the scene simulator computational delay.

The estimation of haptic forces can also present some difficulties. In physical simulators, surface contacts are usually modelled using penalty or impulse-based methods. Impulse based methods compute state changes, not forces. Penalty based methods compute repulsive forces and can work well to model simple obstacles, such planes or spheres, but we can find a number of difficulties when trying to extend these models to display more complex environments. A better suited framework to estimate contact forces for generic haptic display are the constraint based methods [zilles-94] [ruspini-97], but most of the existing physical simulators do not use it.

In this paper we propose an approach to conceive haptic interfaces where the haptic rendering does not depend directly on the physical simulation rates nor on the methods used by the simulator.

The haptic interface is conceived as an external module of a VR system, connected to this system by a *buffer model*. This buffer model is a generic numerical model, that will locally emulate the haptic interaction between the user and the virtual environment simulated by the VR system. We do not care about the methods used to simulate this environment.

The buffer model will act as a very simple local physical model, and the haptic loop will interact with this local model instead of interacting with the virtual environment. The simplicity of the buffer model makes easy and fast the collisions detection and the estimation of contact forces.

A buffer manager process will communicate with the VR system to inform the user motions and to obtain relevant data to update the buffer model.

The buffer model will be updated at a rate proportional to the VR computational delay, but this will not compromise the haptic loop rate.

2. Conceiving an haptic interface based on a buffer model

The central idea in our approach is that the haptic interface locally emulates the virtual environment, continuously adapting its own simple physical model.

Inside the virtual environment, the user is represented by a virtual object, the *probe*. The user is controlling the probe and the probe is interacting with the other virtual objects.

Inside the haptic interface we use a constraint-based framework. The user is represented by a *virtual proxy*, as proposed in [ruspini-97]. In our approach, the proxy is a point moving in the configuration space of the haptic device.

The proxy interacts with one constraint non deformable surface, corresponding to one configuration space obstacle (c-obstacle). This surface represents the nearest portion of the nearest object in the virtual environment with respect to the probe. The position and the shape of this surface will be defined by a set of parameters, continuously adapted to fit the closest c-obstacle form.

The haptic interface is defined by two processes: the model manager and the haptic loop.

The model manager interacts with the simulator, informing the user motions and obtaining the minimum distance between the probe and its closest neighbour in the virtual environment. Using this distance and its derivatives, the manager will adapt the constraint surface to locally fit these values. The simulator must to inform also the local stiffness of the neighbour.

The haptic loop follows the constraint based rendering framework.

2.1. The model manager

The model manager role is to inform to the VR simulator the user motions and to continuously update the constraint surface.

We assume that the VR simulator is prepared to accept this interaction. It means that the simulator can handle our demand within its real-time loop.

The mapping between the probe in the virtual environment and the proxy will be determined by the haptic device features. In an ideal framework, the haptic device would offer force and torque feedback, and the proxy would move in a 6-D configuration space. In this case, the proxy motions could be expressed in full probe motions (6 dof : translation and rotation). In the case of a haptic device with no torque feedback, the proxy would be a point in a 3-D configuration space, and its motions would be expressed only by translations of the probe in the virtual environment (3 dof). The probe would not rotate.

After the probe position update, the VR simulator will find its nearest neighbour, estimate the minimum distance between these two objects and estimate the partial derivatives of this distance with respect to the probe degrees of freedom.

The VR simulator will return the estimated values to the model manager and will update its dynamic state (motions and deformations).

The model manager uses the obtained distance d and its partial derivatives d' to update the buffer model. Let us consider the distance d as being defined by an unknown mapping $f: \Re^n \rightarrow \Re$:

$$d = f(x) \qquad\qquad d' = \left[\frac{\partial f}{\partial x_1} \mathrm{K} \frac{\partial f}{\partial x_n} \right]$$

The buffer model can be defined as a function $\phi: \Re^{n+r} \rightarrow \Re$, which can be written as :

$$\phi(x) \equiv \phi(u,x)$$

for certain $u=[u_1 \ldots u_r]^T \in \Re^r$ .We will refer to $u_1 \ldots u_r$ as the *parameters* and to $x_1 \ldots x_n$ as the *variables* . The variables correspond to the configuration space dimensions.

We define also the partial derivatives for ϕ:

$$dx_i = \frac{\partial \phi(u,x)}{\partial x_i} (i = 1, \mathrm{K}\ n)$$

The fitting algorithm to adapt the buffer model with respect to f is based on the following first order approximations :

$$d\phi = \Delta\phi(u,x) \approx \sum_{i=1}^{r} \Delta u_i \frac{\partial \phi(u,x)}{\partial u_i}$$

$$ddx_i = \Delta dx_i \approx \sum_{j=1}^{r} \Delta u_j \frac{\partial dx_i}{\partial u_j} (i = 1, \mathrm{K}\ , n)$$

We define $du=[\Delta u_1 \ldots \Delta u_r]$ as the vector of corrections to be applied to the model parameters, $dx=[dx_1 \ldots dx_n]^T$ as the vector of the partial derivatives of ϕ, $ddx=[ddx_1 \ldots ddx_n]$ as the vector of differences over the partial derivatives that we obtain applying du to u, $d\phi$ as the difference on ϕ we obtain applying du to u and dxu as the matrix of partial derivatives of dx with respect to u:

$$dxu = \begin{bmatrix} \frac{\partial dx_1}{\partial u_1} & \Lambda & \frac{\partial dx_1}{\partial u_r} \\ \mathrm{M} & \mathrm{M} & \mathrm{M} \\ \frac{\partial dx_n}{\partial u_1} & \Lambda & \frac{\partial dx_n}{\partial u_r} \end{bmatrix}$$

Writing in a matrix form :

$$\begin{bmatrix} d\phi \\ ddx \end{bmatrix} = \begin{bmatrix} dx \\ dxu \end{bmatrix} . du = J . du$$

We consider the problem of fitting ϕ to locally emulate f as the problem of finding the du corrections to be applied to the model parameters u so that the model estimation errors $(d\phi, ddx)$ are minimised. The model estimation errors are assumed to be :

$$d\phi = d - \phi(u,x) \qquad ddx = d' - dx$$

We use the pseudo-inverse of J ($J^\dagger = (J^T J)^{-1} J^T$) to estimate the parameters corrections (du):

$$du = J^\dagger . \begin{bmatrix} d\phi \\ ddx \end{bmatrix}$$

Using iteratively this approach the model manager will continuously keep $\phi(u,x) \approx f(x)$ nearby the probe position.

2.2. The haptic loop

The haptic loop estimates the contact forces using the constraint surface defined by the buffer model. This estimation is based on a very simple physical simulation, with a point interacting with a non deformable surface determined by the points x respecting $\phi(x)=0$. The proxy location is defined by two values: the haptic device effective position and a virtual location, subject to the constraint surface. The effective position must to be informed to characterise the collisions in the virtual environment, but the distance we need to update the buffer model must be estimated

with respect to the proxy position, which is always in the free space, to assure a correct distance estimation. The distances will always be positive (or null) and no collisions will be missed.

To implement this framework, at each time step of the haptic loop we obtain the haptic device effective position and we change the proxy position to reduce its distance to the device effective position subject to the constraint surface.

Generically, this problem can be written as :

Minimise $D(x) = \frac{1}{2}(x-p)^T (x-p)$ *s.t.*

$\phi(x) \geq 0$

where x is the proxy position and p the device effective position.

Before moving the proxy we must to know if it will collide with the constraint surface. The collision detection between the proxy and the surface can be simplified when the ϕ function defines a large surface[1]. Using ϕ we simply compute the distance between the device position and the constraint surface. If the distance is positive or zero the device is in the free space and there will not be collision. The proxy can be moved directly to the device position. If the distance is negative the device is inside the obstacle. The proxy is moved until it makes contact with the constraint surface in its linear path towards the device position. At this point it is no longer possible to move directly to the device position, but it may be still possible to reduce the distance between the proxy and device positions, keeping the surface contact. The minimisation problem becomes :

Minimise $D(x) = \frac{1}{2}(x-p)^T (x-p)$ *s.t.* (1)

$\phi(x) = 0$

Zilles and Salisbury [zilles-94] and Ruspini *et al*. [ruspini-97] solve this problem using constraining planes and Lagrange multipliers. We can use the same approach if we introduce a first order approximation for ϕ, corresponding to a constraint plane tangent to the constraint surface and going through the proxy position ($x=xp$) :

$$\phi'(x) = dxp^T.(x - xp) \qquad dxp = \left[\frac{\partial\phi(xp)}{\partial x_1} \text{K} \frac{\partial\phi(xp)}{\partial x_n}\right]$$

Introducing the λ multiplier, the Lagrangian for our minimisation problem becomes :

$$F(x,\lambda) = \frac{1}{2}(x-p)^T.(x-xp) + \lambda\phi'(x) \qquad (2)$$

The new location of the proxy is found by minimising F, setting its $n+1$ partial derivatives to 0. We obtain :

$$\frac{\partial F}{\partial x} = (x-p) + \lambda\frac{\partial\phi(xp)}{\partial x} = 0 \qquad \frac{\partial F}{\partial \lambda} = \phi'(x) = 0$$

In a matrix form we have :

$$\begin{bmatrix} I & dxp \\ dxp^T & 0 \end{bmatrix}\begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} p \\ dxp^T.xp \end{bmatrix}$$

As proposed in [ruspini-99], we can combine these equations and find a solution which is independent from x : $dxp^T.dxp.\lambda = dxp^T.(p - xp)$. We can solve this equation for λ and use the result to obtain $x = p - dxp.\lambda$, which will be the proxy position.

Because we used the first order approximation ϕ', the obtained x position is also an approximation of the exact position that minimise (1). We can iteratively apply the minimisation of (2) in order to approach the exact solution.

---

[1] A large surface can assure that no collisions in the configuration space will be missed. If we use a more complex surface we will must to use a more sophisticated collision detection algorithm also

In the constraint based framework, once the proxy location is determined, the contact forces estimation becomes easy. Simple impedance control techniques can be used to estimate the forces, using the given object stiffness. The general form of the forces estimation can be given by the Hooke's law :

$$f = k\,(x - p)$$

where k is proportional to the local stiffness, x the proxy position and p the device position.

3. Applying the approach to a deformable objects environment

We applied this approach to interface a dynamic simulator of deformable objects to a desktop 3 dof PHANToM manipulator. The buffer model was implemented in two forms. As a sphere :

$$\phi(u,x) = \sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + (x_3 - u_3)^2} - u_4$$

where (u_1, u_2, u_3) determine the sphere center and u_4 the sphere radius, and as a plane :

$$\phi(u,x) = \frac{x_1.u_1 + x_2.u_2 + x_3.u_3}{u_1^2 + u_2^2 + u_3^2} + u_4$$

The vector $x = [x_1\ \ x_2\ \ x_3]$ are the probe Cartesian co-ordinates [2] and u = [u_1 u_2 u_3 u_4] are the parameters identified by the model manager.

The obtained haptic rendering is realistic and stable.

Refers to [aulignac-99] to see how we used this approach to implement a haptic virtual exam of the human thigh.

4. Summary and Conclusions

We presented an approach to implement haptic interfaces based on a buffer model. The haptic interface is considered as an external module of a VR system. Using this modularised structure the haptic update rate does not depend on the scene simulation rate. The collision detection and the contact forces estimation are simplified, and do not depend on the simulation methods used inside the VR system.

The main advantage of using our approach is to simplify the haptic interface conception, and to permit to existing VR systems to offer a haptic interface without changing its algorithms.

Even applications having high and/or unstable computational delays, as deformable objects simulators of and network-based VR systems, can implement a haptic interface using this approach.

Bibliography

[aulignac-99] D. d'Aulignac, R. Balaniuk. Providing reliable force-feedback for a virtual, echographic exam of the human thigh. In *Proc. of the Fourth Phantom User Group Workshop –PUG99*, Boston (US), October 1999.

[ruspini-97] D. Ruspini, K. Kolarov and O. Khatib. Haptic interaction in virtual environments. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Grenoble (FR), September 1997.

[zilles-94] C.B. Zilles and J.K. Salisbury. A constraint-based god-object method for haptic display. *In Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, vol. 1, pages 149-150, Chicago ,IL (US),1994.

[ruspini-99] D. Ruspini, O. Khatib. *Haptic Display for Human Interactions with Virtual Dynamic Environments*, unpublished.

[2] We used the GHOST programming interface (ControlForce::calculateForceFieldForce) to directly apply forces over the Cartesian axis.

Algorithms for Network-Based Force Feedback

John P. Wilson
Creare Incorporated
jpw@creare.com

Dr. Robert J. Kline-Schoder
Creare Incorporated
rjk@creare.com

Dr. Marc A. Kenton
Creare Incorporated
mak@creare.com

Dr. Neville Hogan
Massachusetts Institute of Technology
neville@mit.edu

1.0 Introduction

Network-based force feedback is an important tool in many distributed robotic systems, including master/slave systems and haptic displays connected to networked virtual reality (VR) simulations. Standard distributed implementations in applications such as these require that force and kinematic data be transferred over a communication link between computers. Delays may arise from the finite time required for the transmission and processing of data. Since delays add phase lag to the signal, this lag limits the effective bandwidth of the closed loop and may result in an unstable telerobotic or haptic display system. Such instability poses a significant safety threat because of the relatively large force-generating capability of the hardware.

Currently, force reflecting systems either reduce the total loop gain (which severely limits performance) or they rely on adding large amounts of damping, with no consideration of formal stability or performance requirements. These approaches can account for a fixed, known time delay, however, they cannot optimize the performance in the presence of random and unpredictable time delays. Consequently, the design and fabrication of telerobots and network-based haptic displays requires the development of control algorithms and software which guarantee stability and improve the response characteristics in the presence of unstructured and unpredictable time delays.

To address this issue, Creare Inc., in a Phase II SBIR project for the U.S. Army STRICOM, is developing a Haptic Display Toolkit which will allow users to easily incorporate haptic devices into network-based force feedback applications. Two techniques for stabilizing force-feedback in the presence of random time delays are implemented in this toolkit: Passive Transmission Line Modeling and Haptic Dead Reckoning. Passive Transmission Line Modeling allows the stable distribution of network-based force feedback. Haptic Dead Reckoning allows a simple, local environment model to service the real-time needs of the haptic display while being periodically updated by a more complex and accurate network environment model. Both of these techniques are described below.

2.0 Passive Transmission Line Modeling

Time delays are a natural property of transmitting power in a continuous medium between two points. An electrical transmission line is an example of a system which includes time delays and yet is passive. Using wave

scattering theory [Anderson-89], it can be shown that passive communication can be attained by replacing the standard communication protocol with the mathematical model of a passive transmission line. Essentially, stability is maintained by modeling the communication link as a collection of passive objects: i.e. masses and springs. Therefore, the key to ensuring stability of telerobots and haptic displays in the presence of arbitrarily long time delays is to construct a communications procedure that mimics a passive transmission line. This configuration can also be considered lossless, because the transmission line model does not dissipate energy. Thus, this communication link model will result in lossless, wave-like behavior and reflections will take place at boundaries which have mismatched impedance. Impedance control [Hogan-88] can be used to match all of the elements in the system in order to avoid these reflections. Therefore, this method of transmitting data implements a lossless, passive communication link.

A two-port lossless transmission line has the input-output relationships shown below [Anderson-89]:

$$fm(t) = fe(t-T) + b\,(vm(t) - ve(t-T))$$

$$ve(t) = vm(t-T) + b^{-1}\,(fm(t-T) - fe(t))$$

where fm is the force applied to the haptic display, fe is the force calculated by the environment, vm is the velocity of the haptic display, ve is the haptic display velocity as represented in the environment, T is the communication link time delay, and b is a physical constant that characterizes the transmission line. Taking the z-transform of this expression, assuming that the environment impedance is controlled to be b (that is, fe(z) = b ve(z)), and then solving for fm(z) in terms of the other input quantities yields:

$$fm(z) = b\ vm(z).$$

This expression shows that the input impedance of the transmission line is simply the physical constant, b, for all time delays. The constant, b, is calculated from the effective stiffness and mass of the transmission line and has the units of damping. Therefore, the transmission line appears to be a simple damper for <u>any</u> time delay. Performing the same calculation when transmitting kinematic variables (i.e. ve(t) = vm(t-T) and fm(t) = fe(t-T)) results in the following expression:

$$fm(z) = b\ z^{-2T}\ vm(z)$$

which shows a strong dependence of the impedance on the time delay, T. This means that by implementing the transmission line input/output relationships, the <u>destabilizing</u> effect of time delays in force reflecting systems (i.e. the dependence of the communication link impedance on the time delay) is eliminated by implementing the equivalent of a lossless transmission line.

The parameter, b, in the equations described above, is the characteristic impedance of the transmission line. For a transmission line implemented with a flexible rod with length L, cross-sectional area A, density ρ, and Young's modulus E, the net mass, m, and the net stiffness, k, of the rod can be shown to be:

$$m = \rho AL$$

$$k = \frac{EA}{L}.$$

The characteristic impedance, b, of the rod can be shown to be:

$$b = \sqrt{k\,m}\,.$$

The transmission delay time may be interpreted as the length of time it takes for a stress wave to travel the length of the rod, i.e. the length of the rod divided by the wave speed in the continuous medium. The wave speed, c, is well known to be:

$$c = \sqrt{\frac{E}{\rho}}\,.$$

Thus,

$$T = \frac{L}{c} = \sqrt{(\rho AL)\left(\frac{L}{EA}\right)} = \sqrt{\frac{m}{k}}\,.$$

Combining this with the above equations results in the following expressions for m and k:

$$m = bT$$

$$k = \frac{b}{T}\,.$$

In our applications, the delay time is determined by features of the system beyond our control (i.e. the physical communication link). Thus, for a constant b, the effective mass and stiffness of the rod vary as the time delay varies. Short delay times result in a rod that feels light and stiff and long delay times result in a rod that feels heavy and compliant. This demonstrates how the performance of the system varies as the time delay varies, however, the stability of the system is still guaranteed.

3.0 Haptic Dead Reckoning

Dead Reckoning is a technique employed by distributed military simulations where a local entity (an aircraft, for instance) keeps track of the position and orientation of each surrounding entity of interest (all other aircraft in range, for instance) through the use of a local model. Updated information from the actual surrounding entities is used to update the local entity's model. This technique is used in order to limit the rate of networked data; the local entity "fills in the gaps" between data samples using the local model.

Similarly, we have defined a procedure and implemented supporting code which enables stable network-based force feedback by employing two models of an environment system: (1) a simplified version of the environment to meet the high rate local needs of the haptic display, i.e. a model that can be run at 1000 Hz; and (2) the full system representation which resides in an environment server. Whereas the military dead reckoning model calculates updated position and orientation, our local "haptic dead reckoning" model calculates updated force to be applied to the haptic device. Kinematic information is passed from the local haptic system to the environment server and the environment server in turn passes updated model parameters back to the local system which are plugged into the local haptic dead reckoning model. For example, consider an environment whose lowest order mode is represented by a stiffness term and a damping term. The full environment server receives updated kinematic information from the local system and uses whatever technique is desired to calculate the new values of stiffness and damping. These updated parameters are then passed to the local model. Force is only calculated by the local model, which is called at a 1000 Hz rate to meet the needs of the attached haptic display.

The heart of the dead reckoning technique is the procedure employed by the environment server to develop a simplified model for use on the local system. This process relies on linearizing the full system model and then simplifying this linearized model. Linearization and simplification are justifiable because the resulting model need only be accurate over a short "update" time interval for calculating a small number of key outputs caused by changes in a specified set of inputs. For example, the process has been used to develop a very simple model that

can accurately calculate the effect on force of changes in haptic display position for time intervals greater than 100 milliseconds.

The simplification process consists of the following steps:

1. **Linearization of the system model** around the current operating point. For a system of N first-order ordinary differential equations (ODEs), the result of this calculation is the NxN Jacobian matrix of the rates-of-change of the states in the full system model.

2. **Calculation of the eigenvalues and eigenvectors** of the Jacobian matrix.

3. **Diagonalization of the Jacobian** using the eigenvectors.

4. **Model-order reduction** is then performed on the diagonalized system of ODEs.

5. **Update the simplified model.** The preceding steps result in the calculation of a set of coefficients that define a simplified, linear set of ODEs in time. These coefficients are transmitted to the local system for use over the following update interval.

The development of the simplified model has proven to be highly effective for cases in which the linearization process is justified. The only situations in which problems have been encountered are those in which the character of the system changes dramatically during an update interval, e.g. due to a collision. Where necessary, such problems can be addressed by various expedients. For example, the linearization procedure can include an explicit procedure for detecting and dealing with fundamental changes in the model. One way to accomplish this is to develop a separate linearized system model that is accurate for the post-collision state of the system and making both models available to the local haptic processor.

4.0 Demonstration Program

A distributed application which employs the PHANToM haptic display and our Haptic Display Toolkit is provided to demonstrate this technology. The user is presented with a dialog box containing a number of options when the program starts. In this dialog, the user can choose Passive Transmission Line Modeling or Haptic Dead Reckoning. In addition, the user can choose an arbitrary amount of communication delay or alternatively choose to use the default amount of delay present in the network connection between the haptic and environment portions of the application. This program uses the freely available ACE communications package (ADAPTIVE Communication Environment developed at University of California, Irvine; see http://www.cs.wustl.edu/~schmidt/ACE.html). Users are invited to try this demonstration at PUG99 or contact Creare Inc. for more information. Alternatively, a modified version of this application that demonstrates Passive Transmission Line Modeling is available via the SensAble Technologies Web site (http://www.sensable.com/products/Ghost_Demos/netfeel.htm).

5.0 Conclusions

Creare Inc. is developing a number of network-based force feedback applications, including master/slave telerobot systems, distributed driving simulations, and HLA (High Level Architecture) applications. We have demonstrated two methods for achieving stable network-based force feedback: Passive Transmission Line Modeling and Haptic Dead Reckoning. We have shown the advantages and disadvantages of each method and are in the process of developing a commercially-available Haptic Display Toolkit based on these technologies.

Acknowledgements

This research has been supported by the U.S. Army Simulation, Training, and Instrumentation Command (STRICOM) Small Business Innovative Research (SBIR) Program contract numbers M67004-97-C-0041 (Phase I) and M67004-98-C-0030 (Phase II).

References

[Anderson-89] Anderson, R. and Spong, M.; *Bilateral Control of Teleoperators with Time Delay*; IEEE Transactions on Automatic Control, 1989, V34, N5, pp. 494-501.

[Hogan-88] Hogan, N.; *On the Stability of Manipulators Performing Contact Tasks*; IEEE Journal of Robotics and Automation, 1988, V4, N6, pp. 677-686.

A Haptic Constraints Class Library

Matthew Hutchins and Chris Gunn
CSIRO Mathematical and Information Sciences[1]
GPO Box 664, Canberra, ACT 2601, Australia.
Matthew.Hutchins@cmis.csiro.au

Abstract

Constraints are an important part of many new interaction techniques. They can be used to simplify input, reduce degrees of freedom, and eliminate error conditions. Constraints can be part of low-level interactions or be derived from high-level application-specific rules. There is a natural synergy between the use of haptics and the use of constraints in interfaces. Force feedback can be used to control the interaction so that the user is either encouraged or obliged to "do the right thing". This paper describes the initial design of a C++ class library that encapsulates some of the essential ingredients for building applications that use haptic constraints. It describes our early experiences with the library, and our plans for its future development.

Keywords: haptics, constraints.

1. Introduction

Three dimensional multi-sensory virtual environments provide unparalleled amounts of freedom for users to explore data and solve difficult design problems. But in a user interface, sometimes too much freedom can be a bad thing. Unnecessary degrees of freedom can lead to confusion, ambiguity, irrelevant decisions, or errors. An obvious example is trying to solve a 2D problem in 3D, or a 1D problem in 2D. Consider the scrollbar in Figure 1. If the "knob" could be moved anywhere on the page, the interface would be unusable. The sensible application of *constraints* is a fundamental principle for good interaction design [1]. Constraints can be used as part of low-level interaction techniques (like the motion of the scrollbar), or can be used to enforce high-level design rules specified by an application expert.

Previous papers have described our Haptic Workbench configuration [6] and the motivation for using haptics to enforce design constraints [2]. There is a natural synergy between the use of haptics and the use of constraints in interfaces. Force feedback can be used to control the interaction so that the user is either encouraged or obliged to "do the right thing". The haptic display of seismic data described in [4] created the effect of haptic constraints using special-purpose rendering techniques. This paper addresses some of the specific design details of a more general software library we are developing to enable programmers to easily construct applications that use haptically rendered constraints.

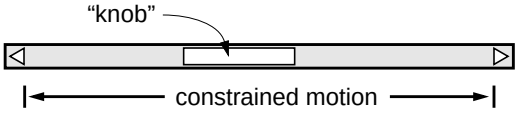


Figure 1: A scrollbar as an example of constrained motion.

2. Design

The Haptic Constraints Library is a C++ class library that extends the Magma multi-sensory scenegraph API from ReachIn Technologies [10], which in turn uses the GHOST API [11]. A class hierarchy diagram for the library is shown in Figure 2. A *scenegraph* is a hierarchical representation of a collection of objects in a scene. To render the scene, the scenegraph is repeatedly traversed, starting from the root. Traversals accumulate graphic and haptic rendering instructions from certain nodes in the graph (the "drawable" nodes). The haptic rendering instructions are passed to a realtime process that is responsible for controlling the PHANToM at the high refresh rate required. The haptic instructions can be potential or actual surface contacts, or more abstract instructions called *force operators* that define an output force as a function of the device position and other state information, such as time. Any drawable node in the scenegraph can generate graphic or haptic rendering instructions. The starting point for our Haptic Constraints Library was the "`PenPusher`" class — an abstract base class for nodes that just do haptic rendering with force operators. This could be seen as analogous to the GHOST "`gstEffect`" or "`gstForceField`" classes.

1. This work was carried out within the Cooperative Research Centre for Advanced Computational Systems established under the Australian Government's Cooperative Research Centres Program [8,9].

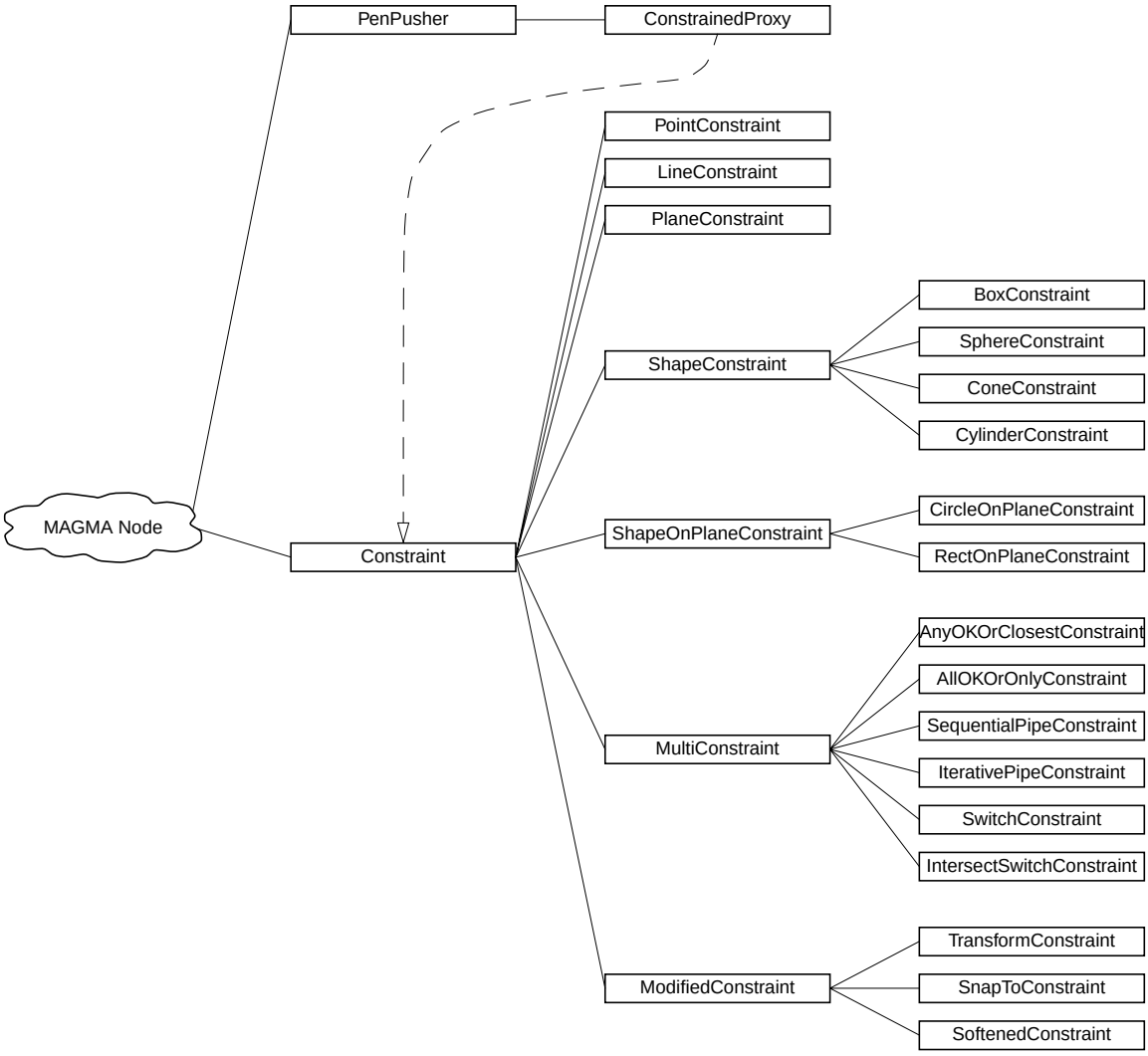


Figure 2: Class hierarchy for the Haptic Constraints Library.

Our first experiments with constraints were therefore based on a pure force model, and we implemented examples such as attraction and repulsion from points and line segments. We encountered several problems with this approach, particularly with getting stable force behaviour. For example, using a "gravity" model for attraction to a point (force inversely proportional to distance) creates an unstable region near the point (which is a singularity). A "spring" model (force proportional to distance) is better, but can still be unstable with large forces. With respect to implementing constraints with forces, we note that:

- a constraint must be violated before any force can be generated;
- the PHANToM can only generate a finite amount of force, so the constraint can always be violated by an arbitrary amount by a user who pushes hard enough;
- using large forces to overcome these problems magnifies any instabilities in the constraint geometry, and can be uncomfortable for the user.

As an alternative approach, we implemented a *constrained proxy* node. This is essentially the same solution that is often used for haptic surface rendering [7]. Instead of using the actual PHANToM device position as the user's interaction point, the application uses the position of a proxy point that is subject to an algorithmically cal-

culated constraint. The proxy will attempt to follow the tool position, but will not violate the constraint. Spring-like forces are used to attract the tool to the proxy position. This architecture ensures that the constraint is satisfied, but can use reasonably small forces (or even no force) as a form of feedback to the user. Naturally, graphical feedback can also be used to show the position of the constrained proxy. Instead of using a linear spring model, we designed a force curve that allows the user to smoothly pull away from the proxy position if desired (Figure 3). This curve prevents large forces from being generated if an instability in the constraint causes the proxy to be separated from the current tool position. The system also "disengages" forces at startup or if the proxy position jumps a large distance. In the disengaged mode, only the low background force will be applied. Full forces are re-engaged when the tool is brought close to the proxy.

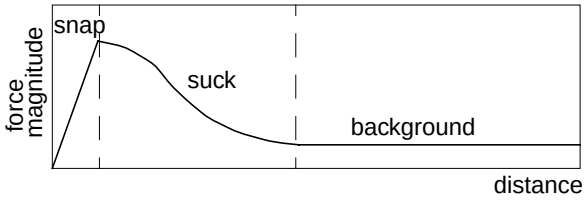


Figure 3: Force curve for the constrained proxy.

As shown in Figure 2, the constrained proxy is implemented as a scenegraph node, a subclass of the abstract `PenPusher` class. The `ConstrainedProxy` node has a child node, which is an instance of a subclass of the abstract base class "`Constraint`". This base class specifies the function interface that is used to define all constraints. The function interface is defined in terms of a *request* point, typically the physical position of the haptic tool. The simplest interface would be a predicate: does this request satisfy the constraint or not? We add to this the notion of a *projection*: if the request does not satisfy the constraint, what is a "good" replacement point? Sometimes this will be the closest satisfying point, but projections are not always that straightforward. We also allow for the projection to fail, meaning the request does not satisfy the constraint, but no reasonable replacement can be found. For some constraints, detecting whether the constraint has been violated and/or computing a projection requires knowledge of the current point. For example, a constraint that acts like a surface is concerned with preventing transitions from one side to the other, and so must know the current point and the request point to detect violations. An example of the use of the current point in computing projections is discussed below. In summary, a constraint is defined by a function that looks at a current point and a request point, and returns a value indicating one of: the request satisfies the constraint; the request does not satisfy the constraint, with a projection; the request does not satisfy the constraint, but projection has failed. The `Constraint` base class also provides some functionality common to all constraints. This includes enabling and disabling of constraints, and handling some issues to do with constraint coordinate systems.

Our first crop of constraints are based on familiar 3D geometric primitives — point, line, plane, a set of solid shapes (box, sphere, cone, cylinder), and some planar shapes (circle, rectangle), shown in the hierarchy in Figure 2. The line constraint includes constraining onto a line, ray or segment, the plane constraint includes constraining above, below or onto the plane, and the solids include constraining inside, outside or onto the surface. Each of these constraints poses three problems: how to parameterise or describe the constraint, how to detect whether a request satisfies or violates the constraint, and how to project a violating request to a satisfying point. The parameterisation problem is familiar from other kinds of 3D modelling, and we use similar solutions. For example, a plane is described by an origin and a normal, a sphere is described by an origin and a radius, and a circle on a plane by origin, normal and radius. Detecting whether a request satisfies a constraint can usually be done with a few vector calculations, e.g. for a sphere, compute the length of the vector from the origin to the request and compare it with the radius. As mentioned previously, projections can require more work. Figure 4 shows inward and outward projections for a box. The inward projection is to the closest point on the surface. The outward projection doesn't necessarily use the closest point — the projection is toward the side of the box that the current point is on, if possible. This stops the constrained proxy from jumping from one side of the box to the other when the PHANToM is moved around inside the box.

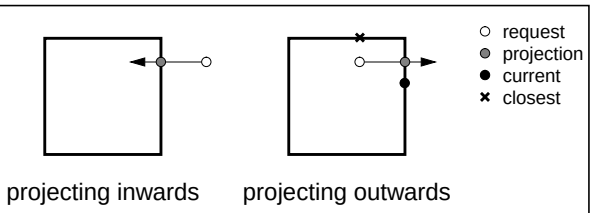


Figure 4: Inward and outward projections for a box.

Figure 5 shows some of the projection options for a cone. Each has different perceptual properties from the user's point of view. The radial and perpendicular projections, for example, have a discontinuity at the centre of the cone, which the axial projection does not have. For large and small angle cones, the radial and axial projections (respectively) can be a large distance from the request point. The choice of the "best" projection is probably application dependent. The current version uses the radial projection, because it is consistent with the radial projection used by the sphere and cylinder.

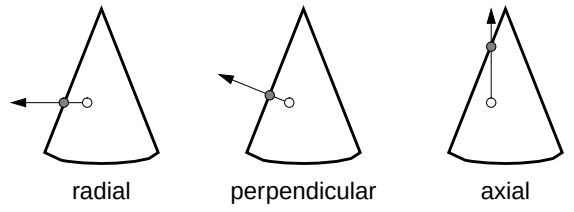


Figure 5: Potential projection methods for a cone.

An application using the Haptic Constraints Library will often need to achieve specific effects by combining or modifying the basic constraints. The "`MultiConstraint`" and "`ModifiedConstraint`" classes shown in Figure 2 allow this to happen. There is no single approach to combining constraints that will suit all situations. `MultiConstraint` is an abstract base class that supports different approaches to combining the results from a set of "child" constraints into a single result. Some of the specific instances shown in Figure 2 include:

- `AnyOKOrClosestConstraint`: if the request satisfies any child then it is OK, otherwise the closest of the children's constrained points is used.
- `SequentialPipeConstraint`: the request is fed through a chain of child constraints, with the constrained result of one becoming the request for the next.
- `IterativePipeConstraint`: works like the `SequentialPipeConstraint`, but feeds the result of the final child back to the first child, and continues until all children agree on a constrained point, or a time limit is reached.
- `IntersectSwitchConstraint`: a single child is active at a time, with switching between active children occurring at intersection points.

The `IterativePipeConstraint` is the only algorithm we have implemented so far to search for a point to simultaneously satisfy multiple constraints, but this could be an interesting area for future work.

Subclasses of `ModifiedConstraint` modify the input and output of a single child constraint. Examples include:

- `TransformConstraint`: like a Transform node in a scenegraph, this defines a matrix coordinate transformation through translation, rotation and scaling.
- `SnapToConstraint`: activates the child when the request is close to the constrained point, but ignores the constraint if the request is far away.

3. Experience

One of the features of the Magma API is its support for rapid prototyping of demonstrations using a modified version of VRML '97 [12]. This VRML support extends to user-defined nodes, and so we are able to demonstrate aspects of the Haptic Constraints Library using simple scenes that include a selection of the available constraints. We have also developed demonstrations targeting particular interaction techniques, such as constrained rubberbanding of objects.

Our initial work on haptic rendering of constraints was motivated by design problems where experts could provide rules for good and bad design which could be turned into interaction constraints. One of these problems was the planning of mine access shafts that had to adhere to strict rules concerning, for example, the gradient of the shaft and the radius of curvature of turns, because of operating limits of the machinery in use. The design of the Haptic Constraints Library allowed a demonstration to be constructed component-wise. For example, a gradient constraint without a specific direction can be implemented with a constraint onto the surface of a cone. A combined gradient and direction constraint can be implemented with a ray. A combined gradient and radius of curvature for a left or right turn is achieved by combining two planar circle constraints. We have developed a prototype system that shows the potential of this style of interaction.

We have used the constraints library to implement a program demonstrating the use of haptic assistance to a user sketching in a 3D workspace. The program was based on the idea of predicting the type of curve the user was drawing (straight line, circle or spiral) from the initial part of a stroke, and then constraining to that curve for the remainder of the stroke. The constraints worked well, but the prediction mechanism was not effective, and was replaced with a simple mode selection (more details will

be published in [3]). A different approach to using haptic constraints in sketching is described by Snibbe *et. al.* [5].

We have also begun experimenting with using haptic constraints as a method of enhancing visual presentations of data. One example is constraining the interaction to a data surface (defined as a toroid) that is also presented visually. This work is still in its early stages, and it remains to be seen whether this method is more effective than traditional surface rendering.

Our experience has been that the Haptic Constraints Library provides a good platform for developing new interaction techniques and interesting haptic-enabled applications. We are now looking for an appropriate application niche in the Australian industrial context where we can most effectively demonstrate the power of these techniques.

4. Future work

We are currently working on a new design for the Haptic Constraints Library to address several issues that have arisen as a result of the initial design. One of these is the problem of realtime rendering of constraints. In the current system, constraints are calculated in the graphics-rate traversal of the scenegraph, and a force operator implementing the spring force to the constrained point is passed to the realtime haptic renderer. At best, this introduces an undesirable roughness and drag to the constrained interaction, but in some cases, particularly at constraint boundaries, it can create large oscillations. Hence one of our goals for the new design is to improve this aspect of haptic rendering. Our current plan is to continue to have the constraints accurately calculated in the scenegraph traversal, but to provide better haptic approximations of the constraint for realtime updates. A significant design goal is to be able to use the same code for the accurate calculation and the approximation where it is appropriate to do so. This will reduce the amount of programming required to develop new constraints.

Another issue to be addressed by the new design is increased flexibility in what types of things can be constrained. The current design only allows for a single interaction point to be constrained. That is, constraints are fixed, and the interaction point moves around them. If we invert this concept, we could attach a constraint to the haptic tool and use it to affect points or objects in the workspace. Another option is to constrain the motion of dynamic objects that are pushed or pulled by the haptic tool.

We are also aiming for increased flexibility and re-usability of constraint detection and projection components, so that they can be used as general tools for implementing interaction techniques. Naturally, we would also like to provide computational optimisations, such as concatenating transformation matrices. Another goal is to reduce the dependence on any particular haptic API, so that the library can be more easily ported to different software and hardware configurations.

References

[1] Doug A. Bowman and Larry F. Hodges. User interface constraints for immersive virtual environment applications. Graphics, Visualization, and Usability Center Technical Report GIT-GVU-95-26. 1995.

[2] Chris Gunn and Paul Marando. Haptic constraints: guiding the user. In proceedings of SimTecT '99, Melbourne, Australia, March 1999. Pages 261–264.

[3] Matthew Hutchins and Chris Gunn. Sketching in space with haptic assistance. Paper submitted to the First Australasian User Interface Conference, to be held in Canberra, Australia, January 2000.

[4] J.P. McLaughlin and B.J. Orenstein. Haptic rendering of 3D seismic data. In proceedings of the Second PHANToM Users Group Workshop, 1997.

[5] Scott Snibbe, Sean Anderson and Bill Verplank. Springs and constraints for 3D drawing. In proceedings of the Third PHANToM Users Group Workshop, 1998.

[6] Duncan R. Stevenson, Kevin A. Smith, John P. McLaughlin, Chris J. Gunn, J. Paul Veldkamp, and Mark J. Dixon. Haptic Workbench: A multi-sensory virtual environment. In *Stereoscopic Displays and Virtual Reality Systems VI*, proceedings of SPIE Vol. 3639, 1999. Pages 356–366.

[7] C.B. Zilles and J.K. Salisbury. A constraint-based god-object method for haptic display. In proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems, August 1995. Pages 146–151.

Web Cites

[8] ACSys: The Cooperative Research Centre for Advanced Computational Systems.
http://acsys.anu.edu.au

[9] CSIRO Mathematical and Information Sciences.
http://www.cmis.csiro.au

[10] ReachIn Technologies.
http://www.reachin.se

[11] SensAble Technologies.
http://www.sensable.com

[12] Web 3D Consortium.
http://www.vrml.org

Haptic Interaction with Three-Dimensional Bitmapped Virtual Environments

Jered Floyd
MIT Artificial Intelligence Laboratory
(jered@mit.edu)

Abstract

This paper briefly describes the design and implementation of a system integrating a force-reflecting haptic interface, the PHANToM, with a high-powered computer designed to efficiently compute cellular automata, the CAM-8. The goal was to build a system to allow users to interact with real-time three-dimensional bitmapped simulations running on the CAM. Such simulations are far more computationally complex than can currently be computed on the PHANToM host. This system can also provide an intuitive user interface for manipulating data within CAM simulations.

Background

Computational systems exist that allow large-scale 3D cellular automata systems to be rapidly computed. In general, these systems are used to simulate physical environments with very limited user interaction. Using the PHANToM, we can develop methods to allow intuitive human interaction with these large-scale bitmapped simulations.

Most computer generated virtual environments are constructed from polygonal models of objects. This limits user interaction with such systems to rules that may not accurately model a physical system. With advances in technology it has become feasable to work with three-dimensional bitmapped environments, computed by cellular automata simulations. Systems exist that can iterate a set of rules over a large multi-dimensional space, primarily for examining physical simulations such as lattice-gas or hydrodynamic models. Interactive user interfaces for real-time manipulation of these existing systems have been lacking. Force-feedback *haptic* touch interfaces provide interesting possibilities.

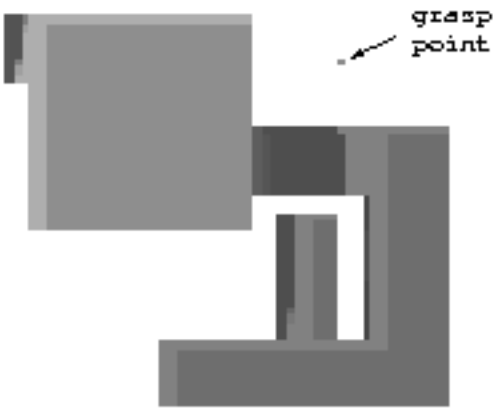
Previous Work

A substantial amount of work has been done in the past on the technologies underlying this research. Physical modeling with discrete spatial-lattice *cellular automata* models has been studied[margolus-99]. Haptic interaction with polygon-based computer simulations has been extensively explored, however, haptic interaction with volumetric simulations has not been as well studied.

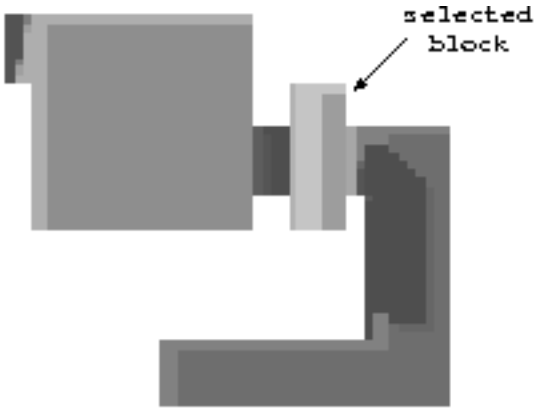
Approach

Work in the past year has included the design and implementation of a basic system combining a PHANToM haptic interface[massie-94] and the CAM-8 cellular automata machine[margolus-96] to provide a three-dimensional haptic user interface for real-time simulations running on the CAM. This was a successful demonstration of what can be done with current technology. Given the high (1 kHz) response rate required for haptic interaction, there were initially problems due to the communication and computation delays inherent in the system. These were overcome with a novel method for coping with latency that in effect trades some degree of spatial accuracy of interaction for timely and sharp haptic responses[floyd-99].

This is an acceptable trade-off. If the haptic response is inaccurate, this is eminently noticeable to the user. Objects feel soft or unwieldy, and in the worst cases violent oscillations may occur. The result of



Static 3-D Pieces (with grasp point)



Static 3-D Pieces (with selected block)

spatial inaccuracy, however, is far less noticeable to the user of a haptic interface as long as other forms of feedback match the haptic feedback. As the user is operating the haptic interface in free space, they have very little in the way of a frame of spatial reference beyond their own proprioception. For small coordinate offsets, this effect is not easily perceptible to the user.

Latency Issues

As the PHANToM and the CAM-8 connect to different host computers, the two machines need to exchange information for the system to work. I first examined the possibility of standard closed loop operation at a lower response rate. Given Ethernet communications and the speed of the CAM-8, a loop rate of 100 Hz would not be an unfeasable goal. To determine whether or not this was a viable direction to continue working towards, the simple "Hello, sphere" program was reimplemented to operate over the network.

Testing revealed serious problems with this model of operation. The environment presented to the user is of very low fidelity. The stiffness of items in the enviroment is determined by the stiffness of the haptic interface and the stiffness of the closed servo loop, of which the latter is the limiting factor with the PHANToM [massie-93]. Additionally, the large delays in force updates could often lead to violent instabilities in the system.

The same problems this system encountered with communication and computational latency have previously been seen when using force feedback devices for teleoperation. The computational delays are essentially no different from additional communication delays, so the whole system can be likened to time-delayed teleoperation in a virtual environment. Some existing systems have been designed to cope with delays at least as large as 1 s [niemeyer-96], which is far greater than the expected system latency. A number of different options were explored, but none of them were found to be suitable, as most made unacceptable assumptions such as a passive remote environment.

Time vs. Space Tradeoff

To meet system design goals, it was necessary to formulate a new method for coping with latency in virtual teleoperation. For the kind of high-fidelity force response desired, it is clear that the closed haptic response loop should not extend beyond the haptic interface host, and should be designed to operate as quickly as possible based on a local model of the remote virtual environment. By relaxing environmental restrictions further, we are able to find an acceptable solution to the latency problem.

All of the conventional methods explored make a fundamental assumption about the nature of the system: There is a direct spatial correspondence between the local and remote environments. That is to

say, if there is a wall five inches from the neutral starting position in the remote environment, then the user will perceive it five inches from the neutral starting position with the haptic interface. Fixed objects in the remote environment are also fixed in a frame of reference about the haptic interface.

This assumption is understandable when teleoperating in a remote real environment. The slave robot physically exists in that remote environment, and when it collides with an object it encounters natural restoring forces that should be communicated back to the master. This is not, however, the case when operating in a virtual environment. The slave robot exists only inside the computer, and does not necessarily encounter forces unless we choose to compute them, and these forces can be computed without affecting the state of the environment. It is free to penetrate solid objects and violate the physics of the virtual universe if we so desire. None of the other similar systems encountered took advantage of these properties.

This understanding allows us to make a simple trade-off of temporal versus spatial accuracy. We are able to provide a high-fidelity haptic response at the exact time the user penetrates an object in the virtual environment by relaxing the spatial correspondence between the haptic interface workspace and the virtual environment.

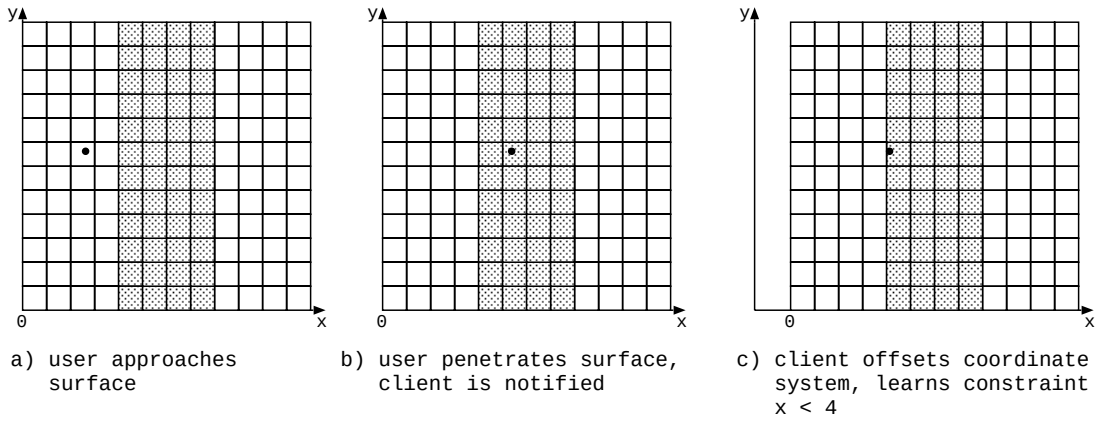


Figure 1: Time / Space Tradeoff

A simple example: Consider that the user is moving unimpeded through free space, and there exists a solid surface some distance in front of the user (Figure 1a). As the user moves towards the surface, at some point they will penetrate the surface given the current frame of reference of the haptic interface (Figure 1b). Some delay length later, the computation server will realize that the user has violated physical constraints of the environment. In a traditional teleoperation system, it would then compute a force to deliver to the user which by the time it was delivered would be incorrect, as the user may have moved significantly during the time it took to compute and communicate this force. Instead, the server tells the haptic client that the user has penetrated a surface in the environment, and where that collision occured. The client uses this information to offset the coordinate system the user is operating in so that instead of having significantly penetrated the surface the user is merely just within it, computes an appropriate force response, and caches the constraint implicit in the existance of that surface so that forces to impede further progress in that direction are computed on the client alone (Figure 1c).

In other words, when a user collides with an object in a remote virtual environment, instead of immediately encountering a restoring force they frictionlessly push the coordinate system (and all objects therein) away from them for the duration of one round-trip delay. After this delay, the haptic client has learned of the environmental constraints and can provide force feedback as appropriate.

Conclusions

The currently available hardware puts this project just within the bounds of possibility, where a proof-of-concept system was feasible and more complex systems will be possible soon. Designing a general purpose interface to bitmapped simulation is not easy; due to the properties of different simulations it often seems necessary to tailor the haptic interface to the task at hand. Additionally, the current method for coping with the large latencies found in the implemented system significantly restrict the realm of simulations that can be easily interacted with via the haptic interface.

Parallel computers such as the CAM-8 are engines that allow real-time presentation and manipulation of large amounts of data, such as a three-dimensional bitmapped virtual environment, however they suffer from the fact that there are few effective ways for a user to directly interact with a simulation. This research begins to provide interfaces for automata-based simulations that are easily understood and consistent with the way a user would expect a simulation to behave, based on their real-life haptic experience.

Such interaction may be of use for controlling and analyzing scientific and engineering simulations, for interacting with 3D bitmapped medical data, and in general for many of the applications for which direct haptic interaction has been studied and proposed in connection with non-bitmapped virtual reality simulations.

Future Work

There are a number of directions in which future work on this research could proceed. Currently implemented examples do not make particularly efficient use of the available computational power and should be expanded to explore more dynamic applications in a variety of fields. The power provided by a next-generation parallel computer would greatly expand the scope of simulations accessible to this research. Expanding the implemented system to include a user interface beyond single point haptic interaction, by the use of multiple PHANToMs or other more complex interfaces, would also be interesting. Further, some of the methods devised for coping with communication and computation delays should also be applicable to conventional force-reflecting teleoperation.

Acknowledgements

Support for this research was provided in part by DARPA under contracts DABT63-95-C-0130 and N00014-96-1-1228.

References

[floyd-99] Jered Floyd. Haptic Interaction with Three-Dimensional Bitmapped Virtual Environments. *M.Eng Thesis*, MIT, Department of Electrical Engineering and Computer Science, May 1999.

[margolus-96] Norman Margolus. CAM-8: a computer architecture based on cellular automata. In Lawriczak and Kapral, editors, *Pattern Formation and Lattice-Gas Automata*. Addison-Wesley, 1996.

[margolus-99] Norman Margolus. Crystalline computation. In Hey, editor, *Feynman and Computation*. Perseus Books, 1999.

[massie-93] Thomas H. Massie. Design of a Three Degree of Freedom Force-Reflecting Haptic Interface. *SB Thesis*, MIT, Department of Electrical Engineering and Computer Science, May 1993.

[massie-94] Thomas Massie and J, Kenneth Salisbury. The PHANToM haptic interface: A device for probing virtual objects. In *ASME International Mechanical Engineering Exposition and Congress*. Chicago, November 1994.

[niemeyer-96] Gunter Niemeyer. Using Wave Variables in Time Delayed For Reflecting Teleoperation. *PhD Thesis*, MIT, Department of Aeronautics and Astronautics, Sep 1996.

The use of CORBA in haptic systems - a requirement for future applications

Christoph Giess, Harald Evers, Hans-Peter Meinzer

Deutsches Krebsforschungszentrum
Abteilung Medizinische und Biologische Informatik
Im Neuenheimer Feld 280
D-69120 Heidelberg, Germany

Ch.Giess@DKFZ-Heidelberg.de

Abstract: *Currently, haptic rendering finds its way from research laboratories to industrial applications where it has to be integrated in existing, often distributed and heterogeneous systems. CORBA has become the industrial standard for such tasks. A surgical planning system, developed by our group, will be used to demonstrate possible solutions for the integration of haptic volume rendering in a legacy system. We will present the advantages of CORBA over traditional network programming. Based on this application we derived a concept that enables the testing of the user interface of haptic applications without changing the application. Hence, it shows the flexibility and extensibility of our approach.*

Index Terms: CORBA, distributed systems, haptic volume rendering

1 Introduction

Distributing a haptic application across multiple computers in a network is a commonly used method to:

- fulfill the time constraint calculating the forces
- take advantage of specialized graphic hardware and
- integrate legacy systems.

All previously described distributed haptic applications (e.g. [Seeger 97], [Fritz 96], [Plesniak 96], [Vedula 96]) used an proprietary communication infrastructure to connect the haptic rendering with the visualization processes. The development of that infrastructure always includes the definition of protocol and message formats, an implementation of addressing and initialization, the marshalling and unmarshalling of data in the appropriate formats, a demultiplexing of requests, error detecting and recovering strategies [Gullekson 99]. This time consuming and expensive task is only one prerequisite for a haptic application. The implementation effort is seldom justifiable.

Additionally, the „simple“ message passing between processes does not fit into the higher abstraction level of an object oriented programming model as offered by GHOST [SensAble 99]. With that, neither inheritance nor polymorphism can be adequately addressed.

All these drawbacks are already known from industrial applications. In this field, CORBA has been established over the last years and has become the standard middleware for distributed systems. In this paper, it is shown how the functionality of CORBA can be used in haptic applications.

2 Haptic Volume Rendering in a Surgical Planning Tool

A surgical planning tool for different tasks in heart, liver and cranio-facial surgery is one project our group is currently working on. This tool is designed to be executed as a plugin of the teleradiology system CHILI® [Evers 99]. It consists of two logically disjunct parts: the user interface and the image processing algorithms. All image processing related functionality is implemented as an independent server which allows multiple clients to share its resources. The first implementation of this server was realized with a proprietary communication library and used

a RPC based communication model [Mayer 99].

2.1 Redesign with CORBA as middleware

When we decided to integrate a PHANToM as an additionally input device in the surgical planning tool, it turned out that the haptic rendering has to be done inside the server since it maintains an exclusive copy of the images. These images are used as model for the force calculation. The size of the data makes an replication inappropriate and would require an expensive data exchange when performing image processing algorithms. The data were visualized with volume rendering which was also choosen for the haptic rendering. An overview of the entire system is shown in figure 1.

The use of an own communication library has several drawbacks as mentioned in the introduction. Replacing it with CORBA was one decision when redesigning the server. The first step was to select a CORBA implementation which fulfills the requirements. In our selection we considered the following implementations: Inprise Visibroker 3.1, Iona Orbix 2.3, TAO 0.2.46, MICO 2.2.1 and OOC ORBacus 3.1. We have chosen ORBacus [OOC 99] for our project because of the availability of source code, the support for all our platforms with their native compilers, the single and multithreaded version, the excellent support from OOC and the free license for non-commercial use. The new implementation of the image processing server was tested on Solaris SPARC/Intel, Linux, Irix and Windows NT whereby the haptic rendering is currently supported only on the last two platforms.

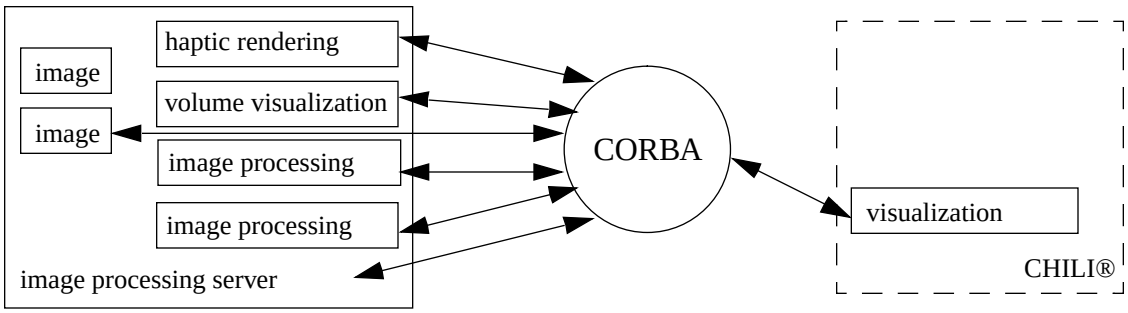


Fig. 1: Overview of the structure of the surgical planning tool

2.2 Results of the redesign

The use of CORBA has simplified the implementation of the image processing server. The previous implementation consist of 3100 lines of code (communication library and image processing algorithms not included). For the new version only 2550 lines of code were needed, although the number of accessible image processing functions increased from 1 to 80 and the haptic rendering was added. A clearer structure will make maintenance and extensions easier.

Performance issues play an important role in haptic applications. The time needed to call a method with 16 double values as parameters and returning a long value (transmission of rotation matrix and acknowledge receive) is between 0.5 and 2.2 msec. The times were measured on a ordinary loaded 100BASE-T ethernet network. They are depending mostly on the performance of the used hardware.

The use of the image processing server was much simplified after using CORBA. The developers of the graphical user interface do not need to deal any further with network programming. They can access the whole functionality of image processing and haptic rendering as if they were implemented in the same process. Additionally, all concepts of object-oriented programming can be used on client side with fully integration of the server.

With this implementation, it is possible that multiple clients can retrieve all information from the haptic rendering process using an standardized interface. Such a feature can be helpful in surgical training or for testing a new user interface. Each client has to pull all state transitions of the server process. Depending on the number of clients, this can lead to an remarkable load on the server where the requests have to be processed. This may consume large amounts of computing power which will not be available for the haptic rendering and eventually stopping the process. A solution to this problem is the event model presented in the next section.

3 Event Model

For some tasks, e.g. psychophysical experiments or testing the usability of an application, it is necessary to log all of the user‘s actions for an further evaluation or to observe the user solving a problem. For that, there are two approaches using the previously described implementation.

The first approach would require a modification of the implementation of the haptic rendering process. The server is responsible to write all interesting information to disk. Testing several alternatives require a change in all implementations which is clearly an expensive solution.

The second solution implements a logging process which polls the server permanently for state transitions. With that, no changes in the server are necessary but an avoidable load of the network occurs.

3.1 Implementing the Event Model

To avoid these drawbacks, the Event Service, one of the Common Object Services specified by the OMG can be used. The central concept of the Event Service is the event channel which decouples the communication between objects. It defines two roles for objects: supplier and consumer role. The supplier produces event data and the consumer processes it [OMG 99a]. In our scenario, the server is the supplier of events. The user interface and the logging process are consumers.

However, implementing this concept with the current GHOST version (2.1) seems to be complicated. The generation of events may depend only on actions in the haptic rendering loop. But GHOST requires an explicit call of `gstScene::updateGraphics()` or `gstScene::updateEvents()` in the application loop. Only these methods invoke the appropriate callbacks [SensAble 99, p. 2-26] to transmit data to the event channel. A possible solution might be the introduction of a „helper“-object that will permanently call one of the methods (see fig. 2).

Beside GHOST, other libraries exist for haptic rendering. The Magma framework [ReachIn 99a] is based on an internal field network. The field network can be set to perform calculations on incoming and outgoing values as well as passing messages and data. This extremely powerful feature allows a seamless incorporation of the event channel [ReachIn 99b] in the application.

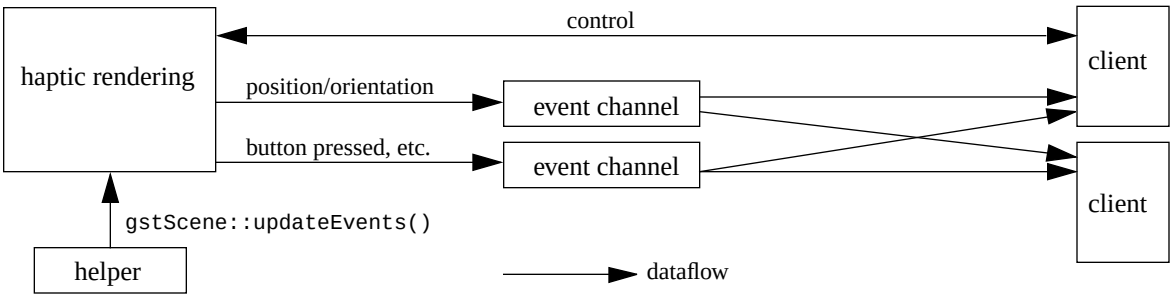


Fig. 2: The use of an event channel to transmit state changes from the haptic rendering process

3.2 Testing a haptic application

Based on the event channel concept, the configuration of a system for testing the usability of a haptic application is shown in fig. 3. The haptic application has not to be changed for this purpose.

The testing is done through additional modules which were connected to the haptic application via the event channel or may access it directly through its CORBA interface. Beside a logging facility, any module may be connected with the system. An observer of the experiment may have different views to the proband‘s application (Graphic) and may change parameters of the system during runtime (Controller). Such interferences may also be automated. A different behaviour can be simulated with an appropriate model (Model). The simple reuse of the test modules is possible because of their generality.

This abstraction is highly generic and its use is not restricted for the specified case.

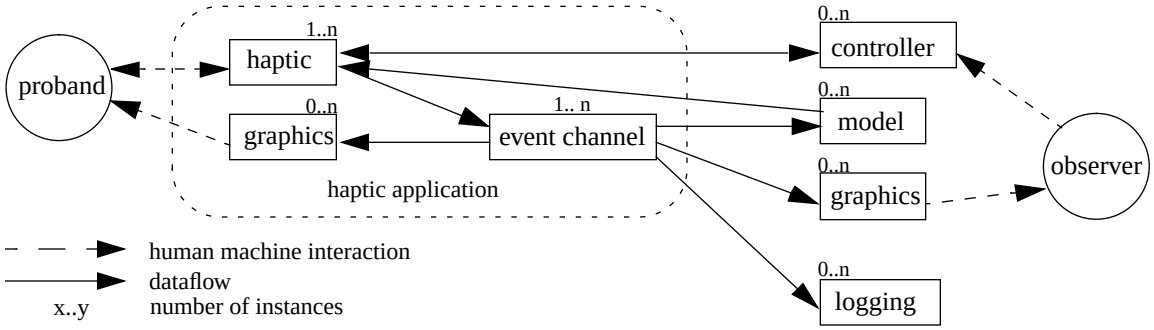


Fig. 3: Modules in an environment for testing haptic applications

3.3 Notification Service

A further enhancement of this concept can be accomplished by replacing the event channel from the Event Service with the extended version of the Notification Service [OMG 99b] when first implementations will be available. This will allow a more precise adaptation of the event flow by event filtering and the setting of quality of service levels.

4 Acknowledgements

This research was performed at the Division of Medical and Biological Informatics at the Deutsches Krebsforschungszentrum (German Cancer Research Center). The work was funded by the Deutsche Forschungsgemeinschaft (SFB 414: Information Technology in Medicine - Computer and Sensor Supported Surgery).

5 References

[Evers 99] Evers H, Mayer A, Engelmann U, Schröter A, Baur U, Wolsiffer K, Meinzer HP: „Extending a teleradiology system by tools for visualization and volumetric analysis through a plug-in mechanism“, Int. Journal of Medical Informatics 53, 2-3 (1999) 265-275.

[Fritz 96] Fritz JP, Barner KE: „Haptic Scientific Visualisation“, Proceedings of the First PHANToM Users Group Workshop, Dedham, MA, 1996.

[Gullekson 99] Gullekson G: „ORBs For Distributed Embedded Software“, ObjecTime Limited, Kanata, Ontario, Canada, 1999.

[Mayer 99] Mayer A, Meinzer HP: „High performance medical image processing in client/server-environments.“, Computer Methods and Programs in Biomedicine 58(1999), 207-217.

[OOC 99] „ORBacus For C++ and Java Version 3.2“, Object Oriented Conceptes, Inc., Billerica, MA, 1999.

[OMG 98a] „CORBAservices: Common Object Services Specification“, Object Management Group TC Document 98-07-05, Framingham, MA, 1998.

[OMG 98b] „Notification Service (Joint Revised Submission)“, Object Management Group TC Document telecom/98-06-15, Framingham, MA, 1998.

[Plesniak 96] Plesniak W, Underkoffler J: „SPI Haptics Library“, Proceedings of the First PHANToM Users Group Workshop, Dedham, MA, 1996.

[ReachIn 99a] „Magma User Guide“, ReachIn Technologies AB, Stockholm, 1999.

[ReachIn 99b] Sienkowski P: personal communication, ReachIn Technologies AB, Stockholm, 1999.

[Seeger 97] Seeger A, Chen J, Taylor RM: „“Controlling Force Feedback Over a Network“, Proceedings of the Second PHANToM Users Group Workshop, Dedham, MA, 1997.

[SensAble 99] „GHOST SDK Programmer‘s Guide“, SensAble Technologies, Inc., Cambridge, MA, 1999.

[Vedula 96] Vedula S, Baraff, D: „Force Feedback in Interactive Dynamic Simulation“, Proceedings of the First PHANToM Users Group Workshop, Dedham, MA, 1996.

A VR Three-dimensional Pottery Design System Using PHANToM Haptic Devices

Shingo Hirose and Kazuo Mori

Mechanical Engineering Laboratory, AIST, MITI
1-2 Namiki, Tsukuba, Ibaraki 305-8564, Japan
hirose@mel.go.jp

Richard M. Y. Lee and Yutaka Kanou
3D Incorporated
Queen's Tower C, 17F, 2-3-5 Minatomirai, Nishi-ku, Yokohama 220-6217, Japan

Abstract

We explore a novel design system for fabricating various three-dimensional structures by arbitrarily transforming the clay object within a virtual reality environment. The setup consists of a personal computer, PHANToM haptic devices, and a rapid prototyping system.

The virtual object is deformed plastically, with the necessary calculations computed in real time, while haptic rendering is provided concurrently with the PHANToMs. This gives the designer the ability to push and to mark as if we touch clay directly by fingers. Finally, the deformed object can be realized using a rapid prototyping (RP) system.

I. 1. Introduction

Within glassware and pottery industries, it has become more important for the customer to be able to visualize the final design before the actual manufacture. Also, customers want to get the originally designed object that he and she can satisfy. With this in mind, we are developing a virtual reality (VR)-based design system, inside which the user can manipulate clay in a VR environment any way he and she so desires.

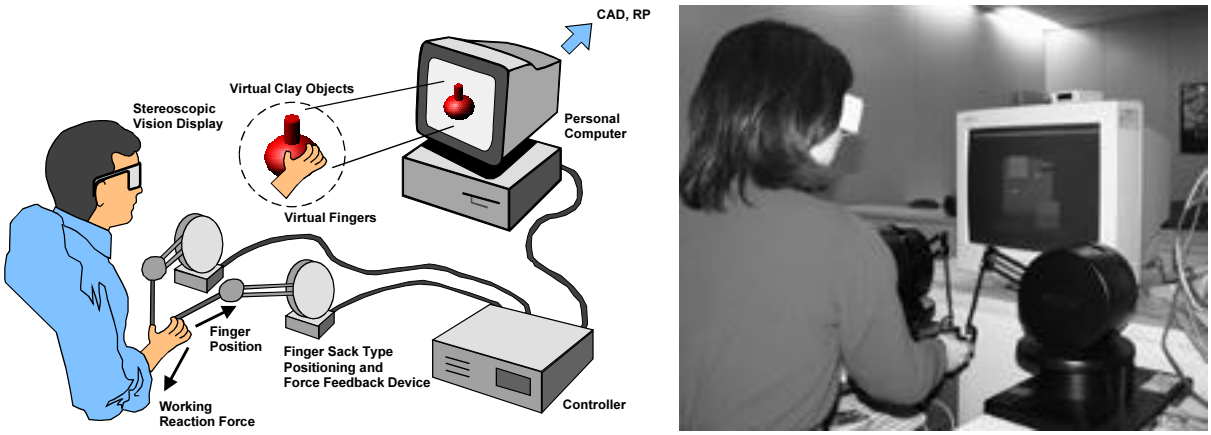


Figure 1 Schematic illustration and photograph of the system.

Recently, studies of haptic interaction with an object surface have matured rapidly and have attracted much attention within the PHANToM community [C. Tarr-97, K. Hirota-98]. This is because the PHANToM provides a high spatial resolution and faithful force fidelity for users. In this study, we choose to work the real-time plastic deformation on a modern Intel PC, which now has a performance comparable to a UNIX workstation.

2. Implementation

The design system we are developing consists of a personal computer (PC), a stereo display and two PHANToM haptic devices as shown in Fig. 1. A rapid prototyping (RP) system is also connected to this system via a local network. The PC is of a high-end type (Pentium 300 MHz ×2, Windows NT) with a graphics accelerator card. The PHANToMs read in the finger positions, and the reaction force is calculated via the software development kit GHOST.

3. Results and Discussion

We simulated the action of pushing and hence deforming a three-dimensional (3D) virtual object. As shown in Fig. 2, the object surface is made up of a mesh of triangular polygons, where vertex nodes are connected via polygon edges. The deformation algorithm entails a mass-spring-dashpot model, whereby each vertex node has a mass, and a parallel arrangement of a spring and a damper is inserted onto each polygon edge. Such a model is usually named the Voigt model [D. K. Felbeck-96]. As for a thin object, corresponding vertex nodes on opposing surfaces are also connected in addition. The stiffness of the object could be altered by changing the spring constant.

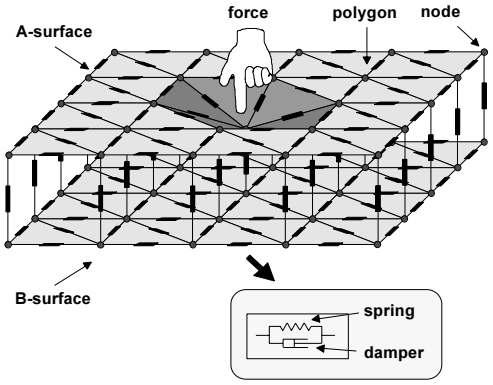


Figure 2 Three-dimensional plastic deformation using mass-spring-dashpot model.

We prepared a cup-shaped cylinder in the virtual environment as an initial object (Fig. 3). The white ball represents the position of the finger. When the finger touched the inside or the outside surface of the cylinder, deformation was carried out in real time such that what the user touched would always be the continuously deforming cylinder, as we would expect in the real physical world. It should be noted that the deformation was not in an absolute plastic mode but in a quasi-plastic mode. The volume of the clay object became slightly smaller while deforming. This resembles the behavior in actual clay pottery making where the density of clay object is increased. We therefore think that this deformation model is suitable for deforming clay objects.

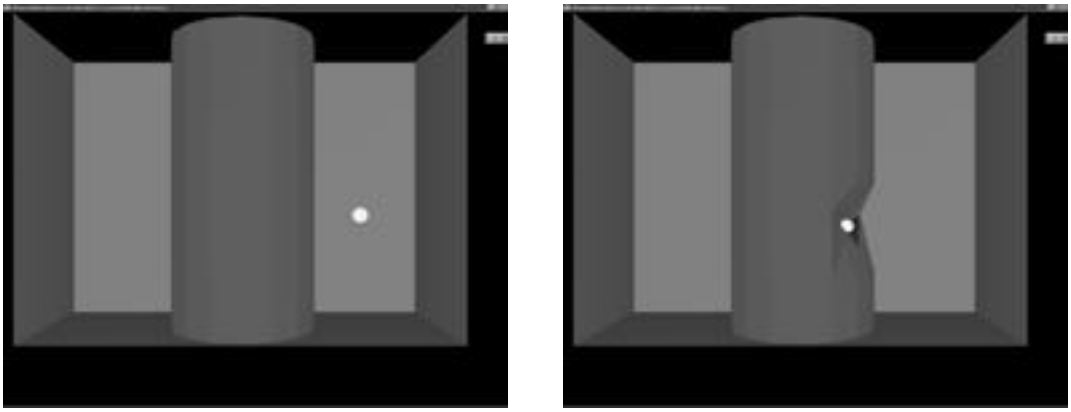


Figure 3 The virtual clay cylinder before (right) and after deformation (left).

Finally, the polygon mesh of the object can be used in CAD and rapid prototyping (RP) systems. We realized the object designed using a RP system, by saving the shape of the object in a STL (STereoLithography) file and sending it to a RP system via the local network. Figure 4 shows the shape of deformed object (left) and the realized product (right).

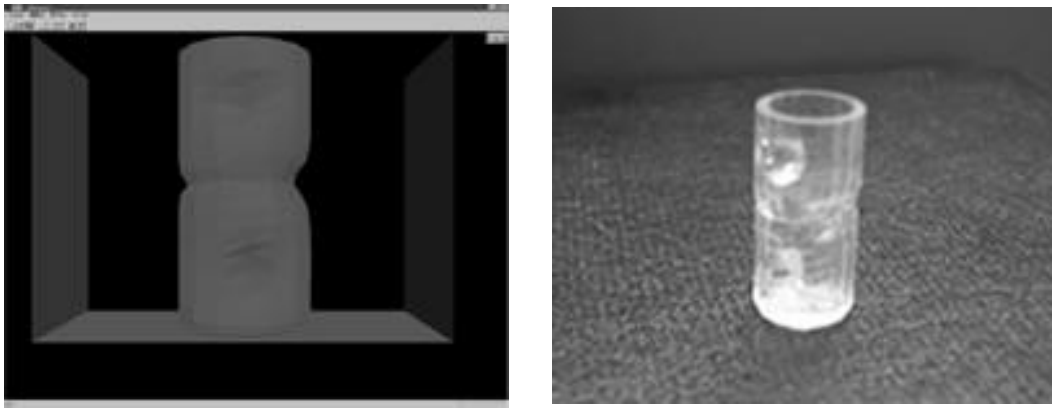


Figure 4 The shape of deformed object (left) and its RP result (right).

4. Summary

A virtual clay object is molded using the plastic deformation model proposed in this study. Plastic deformation of the object is computed in real time so that the haptic rendering process can be executed in parallel on the latest deformed contour. This enables the designer to receive the same tactile sensation as that of touching real clay by finger. Finally, a real object of the desired deformed shape is directly realized using a RP system.

Acknowledgement

The author (S. H.) would like to thank Dr. T. Tanikawa at MEL for helping me learn how to use the RP system.

References

[C. Tarr-97] C. Tarr and J. K. Salisbury, "*Proceedings of the second PHANToM Users Group Workshop'* 1997.

[K. Hirota-98] K. Hirota and T. Kaneko, Japanese Society of Information Processing, 39 (1998) 3261, in Japanese.

[D. K. Felbeck-96] D. K. Felbeck and A. G. Atkins, "*Strength and Fracture of Engineering Solids*", (Prentice Hall, NJ 1996) p. 256.

Mobility Training using a Haptic Interface: Initial Plans

Frances L. Van Scoy, Vic Baker, Chaim Gingold, Eric Martino
West Virginia Virtual Environments Laboratory
and
Department of Computer Science and Electrical Engineering
West Virginia University
fvanscoy@wvu.edu, vic@csee.wvu.edu, gingold@csee.wvu.edu, designmartino@yahoo.com

Darrin Burton
West Virginia Assistive Technology System
West Virginia University
dburton3@wvu.edu

Abstract

We describe the design and progress towards implementation of a haptic-based interface for mobility training. This system will provide the user with information about the site to be visited, such as names of businesses, types and locations of doors, and types of traffic control devices.

Statement of Problem

For a blind person or someone with low vision, the thought of travelling alone to a new city can be daunting. We have designed and have begun implementation of a prototype for using the PHANToM and sound to present a model of a portion of a city to give pre-trip. We have chosen a one-block section of downtown Morgantown, West Virginia, for our prototype. We chose this particular block because the county court house is in that block, set back from the street about 75 feet from other buildings.

Information to be Presented to User

We are using a digital camera to photograph facades of buildings and the Canoma software package to construct 3-D models of the buildings. The user interface will use the regular keys of the keyboard used by the left hand of a touch typist, the PHANToM (operated by the right hand), and sound. The user will use the PHANToM to explore the street by pushing the stylus along the front of buildings. At each store front the user can be told the name and/or address of the business and optionally additional information about the business. At each door, information will be available on how the door opens, on the left or right, and by pushing or pulling. Optionally the user may enter the length of pace and have steps counted aloud while moving along the street. At each intersection information on the kind of traffic control device will also be provided.

User Interface Design

This section describes the design of our system, which is in the process of being built.

Our user interface uses both the senses of touch and sound, and, optionally for debugging and demonstrations, sight. The user controls the system via the PHANToM (using the right hand) and the left side of the computer keyboard (using the left hand). The system responds via tactile feedback using the PHANToM and pre-recorded messages.

The user traverses the model by moving the tip of the PHANToM along the modeled sidewalk. Artificial ridges in the sidewalk indicate the boundary between two buildings and between the sidewalk and a cross street. Such ridges also are added to the facades of buildings to indicate the presence of doors.

Using the left hand on the computer keyboard the user has easy access to 15 letter keys. These keys control various options of the system.

When the appropriate options are selected the user hears:

> when in front of a building, the name of the current building, a list of the various businesses in that building (important in the case of buildings with offices and shops on floors above the ground floor)
>
> when touching the door of a building, whether the door is hinged on the left or right edge, and whether it opens inward or outward
>
> the distance from the more recently visited end of the current block to the current location, in feet and, optionally, in number of steps
>
> when at an intersection, the nature of the traffic control device and, where relevant, information on how to activate it

The system also will display visibly our model of the block and a 2-D map of the downtown area with a "you are here" indication. Toggling between these two views and rotation of the 3-D model are controlled by the number keys on the left side of the keyboard.

Building the 3-d Models

We began the project by taking a series of photographs of the High Street business district. Because of the narrowness of the street and the height of the buildings, we were unable to capture the entire façade of an individual building in one photograph. Instead we typically took six photographs of each building to cover the entire front.

Because each of these photos was taken at a different angle or from a different location, we then used Photoshop to combine the series of images of a single building into one image of the entire façade (and context) of that building.

Next we used Canoma to construct a 3-d model of the exterior of each building. We exported these models from Canoma in VRML and then imported the VRML building models into 3D Studio Max. Using 3D Studio Max we created the terrain, sometimes flat and sometimes sloping, and placed each building model onto the terrain. From 3D Studio Max we then exported a single VRML model of the portion of the downtown business district of interest to us.

Experience with the PHANToM

We acquired two PHANToM systems in summer 1999. One of these is connected to the Onyx computer which powers our ImmersaDesk. The other is attached to a Pentium-based system running Windows NT. In these first few months of using the PHANToMs West Virginia Virtual Environments Laboratory members have done several small projects.

(1) In June 1999 we built two VRML objects, one of a teapot and one of a maze, and asked users to identify these models by touch, using the PHANToM.
(2) We have attempted to port a VRML model of a DNA molecule to the PHANToM system using Haptic Viewer, but this software often fails to load the model or crashes when presented with a fairly large or complex VRML model.
(3) One of us has built a model of a small atom using calls to OpenGL and has incorporated sight, sound, and haptics into the model. The model presents to the user information about the probability density function for an electron.

In related work, another lab member this summer wrote a small C program consisting primarily of calls to the World Tool Kit library with the Immersive Display Option which allows us to display and interact with VRML models on our ImmersaDesk. We use this program to display and manipulate the DNA molecule mentioned above.

Building on these experiences we are in the early stage of writing a World Tool Kit (without IDO) + GHOST application which will display our VRML model of High Street visually on the computer monitor and allow us to interact with the model via the computer keyboard and the PHANToM. This application will incorporate the user interface described earlier in this paper.

Future Direction

We continue work on implementation of the system described here.

This fall we expect to buy a Cyrax 3-D scanner for the laboratory. This system should greatly simplify the task of constructing VRML (or other format) models of a street scene.

Acknowledgements

This work is underway at the West Virginia Virtual Environments Laboratory, which is funded by the EPSCoR programs of the National Science Foundation and the state of West Virginia and by the Department of Computer Science and Electrical Engineering of West Virginia University.

The teapot and maze models were built by Chaim Gingold and demonstrated by Vic Baker and Alexi Leontopoulos at the West Virginia Information Technology conference in June 1999. The DNA model was constructed by Pete Gannett, professor of pharmacy at WVU. The atomic model was built by Chaim Gingold and Erica Harvey, associate professor of chemistry at Fairmont State College. Alexi Leontopoulos wrote the VRML display and interaction program for the ImmersaDesk. Young Oh took the photographs for the downtown High Street model described in this paper, and he, Radhika Mukkamala, and Erik Martino built the VRML models.

Structurally-Derived Sounds in a Haptic Rendering System

Nikolai A. Grabowski and Kenneth E. Barner
Department of Electrical and Computer Engineering
University of Delaware
Newark, DE 19716
E-mail: grabowsk@eecis.udel.edu, barner@eecis.udel.edu

1 Introduction

Objects being rendered by the PHANToM can often be thought of as structures that, in the real world, would vibrate when struck to produce sounds. We have implemented a system that triggers impact sounds when the PHANToM strikes the objects it is rendering. To create the sounds, we use a modal analysis framework put forth by Van den Doel and Pai [1]. The sound is determined by the structure's natural frequencies, mode shapes, and location of impact. For certain objects of simple shape, an equation of motion can be written and solved analytically. However, for irregular shapes, we find the vibration using the Finite Element Method. We have embedded this rendering system in MATLAB for Windows using ActiveX technology. By connecting the PHANToM with MATLAB, we provide an environment in which the user can quickly and easily create haptic renderings of data.

2 Related Work

When an object moves, it causes the air around it to move. The ear senses this air movement and we perceive the sensation as sound. Van den Doel and Pai [1] developed a method for synthesizing impact sounds using the equation of motion,

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

Here, $\mu(\boldsymbol{x}, t)$ represents the deviation from equilibrium of a surface defined on some region $\mathcal{S}$, A is a self-adjoint differential operator under the boundary conditions on $\partial\mathcal{S}$, and c is a constant to be determined by the material being considered. If the external force, $F(\boldsymbol{x}, t)$ is set to zero, the solution to (1) can be written as

$$\mu(\boldsymbol{x}, t) = \sum_{i=1}^{\infty}(a_i \sin(\omega_i c t) + b_i \cos(\omega_i c t))\Psi_i(\boldsymbol{x}), \tag{2}$$

where the coefficients a_i and b_i are determined by the inital conditions. The ω_i's derive from the eigenvalues of the operator A under the specified boundary conditions, and the $\Psi_i(\boldsymbol{x})$'s are the corresponding eigenfunctions (modes). To model the response from impact at a point $\boldsymbol{p}$, the initial conditions can be set as $\mu(\boldsymbol{x}, 0) = 0$ and $\mu'(\boldsymbol{x}, 0) = \delta(\boldsymbol{x} - \boldsymbol{p})$. Assuming the eigenfunctions are normalized, Van den Doel and Pai argue that a reasonable model of the resulting sound wave can be computed as a sum of sine waves with frequencies, ω_i and amplitudes given by

$$a_i^S = \text{constant} \times \frac{\Psi_i(\boldsymbol{p})}{\omega_i}, \tag{3}$$

where the constant takes into account factors such as energy of impact and distance of the listener from the object. The key here is that the relative amplitudes of the sine waves vary according to the values of the eigenfunctions at the impact location. Therefore, the sound will change depending on where the user strikes the object. Finally, the sine waves should be scaled by decaying exponentials to represent damping.

3 The Finite Element Method

When the object being modeled is irregularly shaped, we may not be able to find it's natural frequencies and eigenfunctions analytically. To approximate the solution, the Finite Element Method (FEM) subdivides the object into a number of simpler elements [2]. If the load-displacement relations for a single element are derived in matrix form, it is possible to combine the equations of all the elements into a single system of equations.

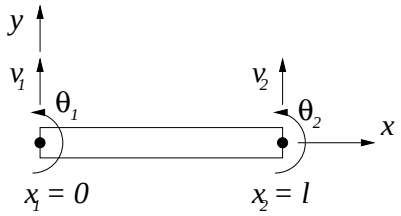


Figure 1: Beam Element

An example of a commonly used element is the beam element shown in Fig. 1. The endpoints of the beam are the nodes and the transverse displacements, v_1,v_2, and the slopes, θ_1,θ_2, are the nodal variables. Between the endpoints, the transverse displacement, v, is found by polynomial interpolation as $v(x) = c_0 + c_1x + c_2x^2 + c_3x^3$. If the constants, c_i, are expressed in terms of the nodal variables, $v(x)$ can be written as

$$v(x) = H_1(x)v_1 + H_2(x)\theta_1 + H_3(x)v_2 + H_4(x)\theta_2 \tag{4}$$

where the H_i are known as shape functions. A matrix equation of motion can be written for the element in terms of the nodal variables as

$$\mathbf{M}^e\ddot{\mathbf{d}}^e + \mathbf{K}^e\mathbf{d}^e = \mathbf{F}^e \tag{5}$$

where $\mathbf{d}^e = [v_1\ \theta_1\ v_2\ \theta_2]^T$ is the displacement vector for the element, and $\ddot{\mathbf{d}}^e$ is its second derivative with respect to time. $\mathbf{M}^e$ is the element mass matrix, $\mathbf{K}^e$ is the element stiffness matrix, and $\mathbf{F}^e$ is the vector of forces and moments at the nodes. Utilizing Galerkin's weighted residual method to develop the finite element formulation [3], the mass and stiffness matrices are found as

$$\mathbf{M}^e = \int_0^l \rho[H]^T[H]dx, \text{ where } [H] = [H_1\ H_2\ H_3\ H_4] \tag{6}$$

$$\mathbf{K}^e = \int_0^l [B]^T EI[B]dx, \text{ where } [B] = [H_1''\ H_2''\ H_3''\ H_4'']. \tag{7}$$

In (6) and (7), ρ, E, and I are the mass density, Young's Modulus, and moment of inertia, respectively, and l is the length of the beam element. The double prime in (7) denotes the second derivative with respect to x.

Once the equations are established for the individual elements of the structure, they can be assembled into the global system of equations as illustrated below:

$$\begin{Bmatrix} F_1^e \\ F_2^e \\ F_3^e \end{Bmatrix} = \begin{bmatrix} M_1^e & & \\ & M_2^e & \\ & & M_3^e \end{bmatrix} \begin{Bmatrix} \ddot{v}_1 \\ \ddot{\Theta}_1 \\ \ddot{v}_2 \\ \ddot{\Theta}_2 \\ \ddot{v}_3 \\ \ddot{\Theta}_3 \\ \ddot{v}_4 \\ \ddot{\Theta}_4 \end{Bmatrix} + \begin{bmatrix} K_1^e & & \\ & K_2^e & \\ & & K_3^e \end{bmatrix} \begin{Bmatrix} v_1 \\ \Theta_1 \\ v_2 \\ \Theta_2 \\ v_3 \\ \Theta_3 \\ v_4 \\ \Theta_4 \end{Bmatrix}. \tag{8}$$

Wherever the element matrices overlap, the coefficients are added. These overlapping positions correspond to nodes that are shared by different elements. The same rule applies for the overlapping external load vectors. Note that, to reduce notational complexity, we have ignored any coordinate transformations that may be needed between the local and global coordinate systems. The assembled equations can then be written compactly as

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{F} \tag{9}$$

where $\mathbf{d}$ is a vector containing all the nodal variables. If we set $\mathbf{F} = \{0\}$ we can solve for the free vibrations to obtain

$$\mathbf{d}(t) = \sum_{i=1}^{n} a_i \sin(\omega_i t) + b_i \cos(\omega_i t))\boldsymbol{\phi}_i \tag{10}$$

where the $\boldsymbol{\phi}_i$'s are the eigenvectors, and the ω_i's are the natural frequencies. The eigenvectors contain the mode shapes in terms of the discrete nodal variables. If we wish to find a mode shape along any given element, we can substitute the appropriate nodal values from the eigenvector into (4) to obtain the interpolated mode shape. Note that coordinate transformations from global to local coordinates may be necessary. Since we now have the interpolated mode shapes and the set of natural frequencies, we have all the required information to calculate the sound amplitudes from (3).

4 Implementation

To perform the Finite Element computations, we use MATLAB equipped with the CALFEM toolbox [4]. Upon obtaining the eigenfunctions and eigenfrequencies, various points along the surface are chosen for generating a matrix of impact sounds. In order to trigger the sounds using the PHANToM, a Microsoft Visual C++ program using the GHOST development toolkit is created. The GHOST program accepts a matrix of heights to be rendered as a haptic trimesh. In addition, the program accepts the matrix of impact sound data associated with locations on the trimesh. Using the Windows multimedia timer, the PHANToM event callback mechanism is updated as frequently as possible ($\approx$ 1–20 ms). When the PHANToM touches the trimesh, the appropriate sound is determined based on impact location and played using the Windows audio functions.

To connect between MATLAB and the C++ program, we make use of ActiveX technology. This technology allows a program to be compiled as a Windows control that can be embedded in another ActiveX-supporting application such as MATLAB. Using ActiveX, data can be transferred between MATLAB and the C++ program. In addition, the control appears in a MATLAB figure window as if part of the MATLAB environment. Thus, the user can generate a matrix of data in MATLAB and then bring up a haptic/graphic/sound rendering with a single command.

5 Results

Here we present two structures modeled with the FEM. The first is a simply supported beam which has boundary conditions of zero displacement at the ends but allows changes in the slope. The beam's material properties are set according to measured values for steel which are $\rho = 7800$ kg/m and $E = 20 \times 10^{10}$ N/m [5]. The beam has a thickness of $h = 0.5$ cm. The length of the beam was varied until it's lowest frequency was 880 Hz which corresponds to the note A5. This is one of the lowest notes on a glockenspiel. The length of the beam turned out to be 11.42 cm which seems reasonable. For the FEM model, we use 41 frame elements. A frame element is the same as the beam element discussed earlier, but it has an added nodal variable which accounts for displacements along the x-axis of the element. The mode shapes and time/frequency plots for impact sounds are shown in Figs. 2 and 3. Notice that the first mode is maximum

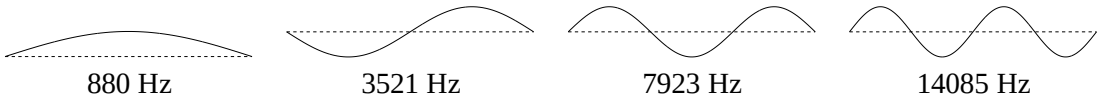
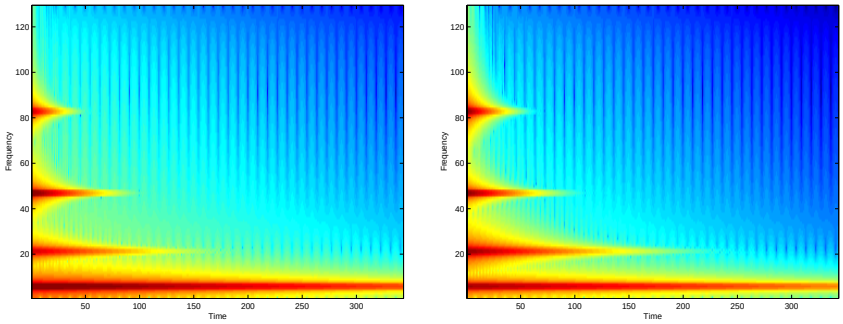


Figure 2: Mode shapes (solid) and original shape (dashed) for beam



(a) Strike at $x/l = 5.9/11.4$ (b) Strike at $x/l = 0.7/11.4$

Figure 3: Tim/Frequency plots for beam

at the center and decreases with increased closeness to the edge. Comparing Fig. 3 (a) and (b), we see the differences in relative strength of the frequency components depending on impact location. Striking near the center excites the low frequency more than striking near the ege. The opposite is true for some of the high frequencies.

The second structure is also made of frame elements but is bent into an irregular shape. The material properties, thickness, width, horizontal span, and boundary conditions are the same as those for the first example. Figure 4 shows the mode shapes and natural frequencies. We observe that the frequencies are not

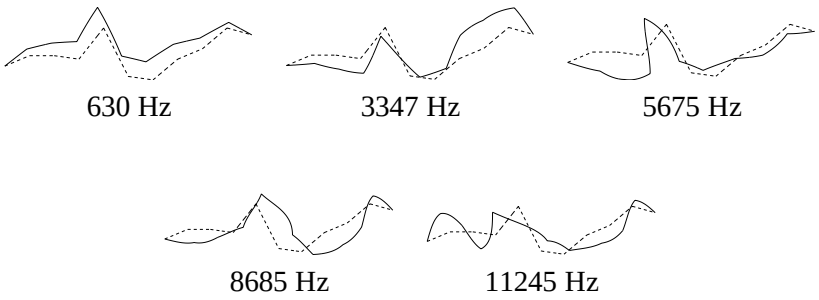
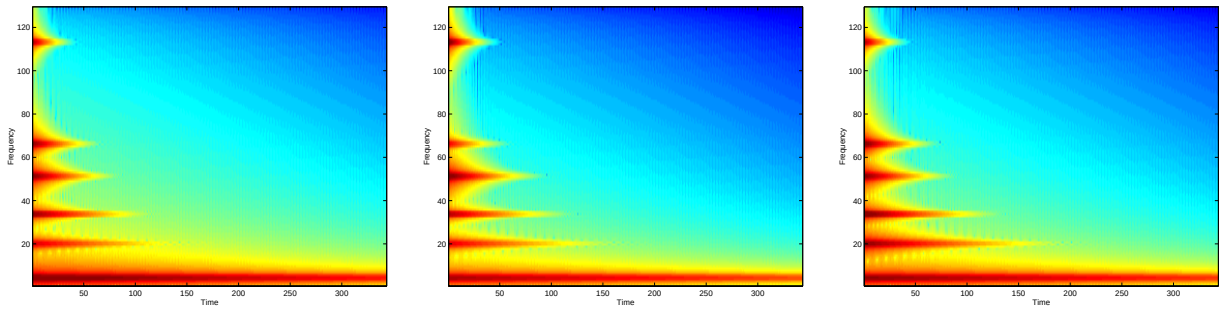


Figure 4: Mode shapes (solid) and original shape (dashed) for an irregular frame structure

as far apart as those of the flat beam. As with the previous example, the fundamental mode is stronger near the center and weaker near the edge. However, it does not decrease evenly. Rather, it is strong in the center, very weak slightly left of center, strong again as you move further to the left, and weak at the very edge. This is different from the monotonic decrease in strength from center to edge of the flat beam. Figure 5



(a) Strike at $x/l = 5.9/11.4$ (b) Strike at $x/l = 5.2/11.4$ (c) Strike at $x/l = 1.4/11.4$

Figure 5: Time/Frequency plots for impact with irregular frame

shows time/frequency plots for the irregular structure.

6 Conclusions and Future Work

A haptic and sound rendering program has been created which can be embedded in the MATLAB environment. Aural feedback in the form of physically-modeled impact sounds has been successfully integrated with the PHANToM haptic interface. The sounds provide the user with information about the location of impact and the structural features of the object being touched. How much information these sounds convey and how useful they are for virtual environments are important subjects for future work. The structural sounds may help to increase information bandwidth when the PHANToM is used for scientific visualization applications [6].

Currently, only frame finite elements have been implemented. These are only useful for narrow objects. To create sounds for the haptic trimesh surfaces, the plate finite element should be implemented. Furthermore, our everyday sound-producing interactions are not limited to impacts but include contact sounds such as scraping. Thus, the physical model should be generalized to an arbitrary excitation, and the implementation should make more use of the dynamics involved in the haptic interaction.

References

[1] K. van den Doel and D. K. Pai, "The sounds of physical shapes," *Presence*, vol. 7, no. 4, pp. 382–395, 1998.

[2] O. C. Zienkiewvicz, *The Finite Element Method*. McGraw-Hill, 1977.

[3] Y. W. Kwon and H. Bang, *The Finite Element Method using MATLAB*. CRC Press, 1997.

[4] Division of Structural Mechanics, Lund University, *CALFEM: A Finite Element Toolbox to MATLAB Version 3.2*, 1996.

[5] P. M. Morse, *Theoretical Acoustics*. McGraw-Hill, 1948.

[6] J. P. Fritz and K. E. Barner, "Design of a haptic data visualization system for people with visual impairments," *IEEE Transactions on Rehabilitation Engineering*, vol. 7, Sept. 1999.

Evaluating Haptic Interfaces
In Terms of Interaction Techniques

Arthur E. Kirkpatrick

University of Oregon

(ted@cs.uoregon.edu)

Sarah A. Douglas

University of Oregon

(douglas@cs.uoregon.edu)

1. Introduction

One of the most important attributes of a haptic device such as the PHANToM is its usability. However a haptic device by itself has neither good nor bad usability. Rather, usability is determined by the relationship between a task and an interface, which includes the haptic device as one hardware component of an integrated hardware and software system. It is the interface that is usable (or not) for a given task. How might we evaluate the usability of a haptic interface?

We argue that we must begin to incorporate more complete and integrated levels of human perception into haptic interface assessment. Previous work on haptic devices has used psychophysical protocols (for example Tan, 1997; von der Heyde & Häger-Ross, 1998), which focus on the device alone or on the device together with a highly stylized interface. While these methods exercise important aspects of device usability, particularly its resolution along a given haptic dimension, we believe that other crucial aspects of usability can only be understood using protocols that test the *interaction techniques* of an interface. Interaction techniques are the most basic operations that still exercise the interface as a whole: the input devices, the graphics output, and the haptics output.

We begin the paper by considering the task of haptic object recognition, which is best described in terms of hand gestures. Hand gestures are supported or inhibited by the interaction techniques of an interface, and so we consider interaction techniques next. We develop this into a framework for evaluating a haptic interface in terms of its interaction techniques. Finally, we apply that framework to make performance predictions for a common PHANToM interface.

2. Factors determining speed of haptic object recognition

Usability is evaluated with respect to a specific task. What tasks might we use for evaluating haptic interfaces? Psychophysical protocols emphasize the task of measuring intensity along some perceptual dimension, such as roughness or sphere size. However, this says nothing about such important tasks as object recognition. In daily life, this can be performed exclusively with haptics (locating an object in a dark room), or in combination with vision. Object recognition also requires the apprehension of the attributes of an object, an important haptic task in its own right. Understanding the factors determining the speed of haptic object

recognition in the physical world will give us a basis for predicting the usability of interaction techniques for object recognition (and object attribute apprehension) in virtual environments.

The factors determining the speed of haptic object recognition have been extensively studied. There are three key factors: the number of different attributes apprehended simultaneously, the number of fingers used to apprehend those attributes, and the gestures used to apprehend the attributes. Klatzky, Loomis, Lederman, Wake, and Fujita (1993) studied the first two factors. With respect to the number of fingers used, they found that it took seven times longer for blindfolded participants to recognize an object using just a single, gloved finger than using unrestricted exploration with their hands. They concluded that the participants using a single finger took so much longer because they had to construct a model of the object's shape over time (temporal integration), whereas using both hands they could directly perceive overall object shape in a single gesture (spatial integration).

Klatzky et al. (1993) also found that access to fewer object attributes reduced speed of recognition: participants took 1.5 to 2 times as long to recognize an object while wearing gloves versus when their bare fingertips could directly touch the object. This demonstrates the importance of material properties (including thermal cues and texture) in object recognition. Factor analysis showed that material properties were most important when the participants could only touch the object with a single finger.

Another research program (summarized in Lederman & Klatzky, 1996) demonstrated that the patterns of hand movement used by participants in object recognition experiments were also crucial determinants of the speed and accuracy of objects recognition. Lederman and Klatzky found that participants used specific, stereotyped hand movements, each associated with the apprehension of one or more haptic attributes, such as hardness, texture, or weight. They named these movements *exploratory procedures* (EP). They found two classes of such procedures: *optimal* procedures, for very accurate apprehension of a single attribute, and *broad* procedures, for coarse but rapid apprehension of several attributes. They concluded that rapid object recognition is a two-stage process, beginning with a broad EP to quickly categorize the object, followed by an optimal EP to precisely identify the object. We will consider the implications of this for the usability of the PHANToM in section 4.

3. Evaluating haptic interfaces using interaction techniques

In the physical realm, gestures are the medium through which exploratory procedures are performed. Gestures have an indirect relationship to EPs. They enable one or more EPs to be performed but do not always constitute an EP. For example, someone may grasp a ball not to perform an enclosure EP but simply to reorient it in a new direction. Some gestures, such as grasp-and-lift, combine multiple compatible EPs.

In haptic interfaces, interaction techniques are the medium for EPs. An interaction technique (IT) is a procedure for accomplishing a fundamental task in a user interface. Examples of interaction techniques include:

- dragging a file icon to move it from one folder to another
- selecting a command from a menu using a mouse
- selecting a command by using a command key
- scrolling down by dragging the scrollbar

- scrolling down by pressing the “page down” key
- determining an object’s shape by running the PHANToM along its contours

Note that interfaces may offer several interaction techniques to accomplish the same task. An interaction technique is unitary—all its component steps are required to accomplish the task—and elementary—it has no subsidiary steps that can be used on their own. Interaction techniques potentially use the entire interface: all input and output devices may be used in a coordinated manner.

In contrast to the physical environment, the designed nature of interaction techniques requires that the interface designer directly account for the relationship between ITs and EPs. The user of a haptic interface is restricted to whatever ITs the system designer included. If the repertoire of ITs does not support a specific EP, the EP simply cannot be performed. For example, with a single point force device such as the PHANToM, there is no way to contact an object at multiple points, so the enclosure EP (wrapping the hand around an object to apprehend its approximate shape) must be replaced by the contour following EP (running the PHANToM cursor around the contours of the object). If the attribute optimally apprehended by that EP can instead only be apprehended by other EPs less specific to that attribute, the effective resolution of that attribute is reduced. Continuing the example, since the contour following EP uses temporal integration rather than spatial integration and so is far slower than the enclosure EP, shape recognition will be less effective. The restriction is more stringent if the missing EP is the only one capable of apprehending an attribute: the attribute cannot be rendered in any meaningful way by the system at all.

The enabling role of interaction techniques shifts our attention from the device to the interface. Although at the psychophysical level the ability of a haptic device to display a given property is an inherent result of its mechanical and electrical structure, at the cognitive level the ability of the device to render a given attribute is a result of the design of the entire interface. We cannot speak of whether a device can render a given attribute, but can only speak of whether an interface can render that attribute—because the ITs of the interface enable the EPs, and it is the EPs that allow an attribute to be apprehended or not. Furthermore, the speed of the interaction technique determines the speed of the EP.

This suggests an approach for analyzing usability of haptic interfaces. We argue that usability must be analyzed at a higher level than simply hardware display capabilities. Usability is affected by the relationships between the cognitive aspects of haptic perception, the various senses and motor systems, and the task the user is performing. We cannot discuss the usability of a haptic device alone, but can only discuss it in terms of that device working within a given configuration of graphics hardware and software, input devices, haptic algorithms, and interaction techniques. The fundamental principle of this research is that the usability of a haptic interface is determined by how well its constituent parts work together. Usability is a property of the interface as a whole, not an individual hardware or software component.

Interaction techniques are the lowest level of interaction that tests the whole interface. They are shared within whole classes of applications, so usability predictions for ITs are widely applicable. Finally, there is strong *a priori* reason to believe they affect usability: The interaction techniques supported by an interface can facilitate or hinder the use of each EP, in turn making the corresponding haptic property more or less perceptible.

4. Example evaluation: Haptic interfaces in point force environments

The above framework can be applied to any specific haptic interface, or even to a single hardware configuration that might be used for any number of different haptic interfaces. As an example, we analyze several interaction techniques for a hardware configuration that is expected to be widely used in commercial environments: the PHANToM with a monoscopic (i.e., non-stereoscopic) graphics display. This style of interface is used for the FreeForm haptic sculpting program, as well as in many other 3-D modeling and CAD programs.

One of the most important tasks for these applications is determining the shape of an object. Three ITs are available for accomplishing this task: a haptics-only IT, running the PHANToM along the contours of the object without looking at the object at all (either because the object isn't being displayed or because the user is looking elsewhere); a graphics-only IT, looking at the object, perhaps reorienting its position, but not touching it; and an IT combining graphics and haptics, both looking at the object and running the PHANToM along its contours. We consider in turn the likely performance of each of these ITs.

We can make some predictions about the performance of the haptics-only IT based upon the studies of object recognition described in Section 2. The object recognition task used in those studies includes object shape recognition as an important subcomponent. The Klatzky et al. (1993) study demonstrated the importance of using multiple fingers to spatially integrate shape information. The PHANToM, a point force device, requires the user to perform temporal integration of shape information, a much slower process. Second, Lederman and Klatzky (1996) showed that rapid object recognition requires the ability to perform broad EPs, procedures that perform rapid, coarse discriminations of several attributes simultaneously. Broad EPs invariably involve spatial integration, which the PHANToM cannot provide. These results strongly suggest that the haptics-only IT will be a slow and unreliable means of determining object shape.

The low predicted performance of the PHANToM for haptic object recognition suggests that vision will be essential in haptic interfaces using the PHANToM. But will it be sufficient by itself? In the physical world, vision is superlative for apprehending object geometry, and sighted participants rely on it exclusively, ignoring haptics for that task. However, vision may be less effective in the virtual environments we are considering here, because important depth cues are missing. There is no stereoscopic display, no interobject shadowing, no texture on the object, and no environmental reflections. In such an environment, while vision may well be ultimately effective in apprehending shape, it will be probably be considerably slower than for a physical object of corresponding complexity.

Finally, we consider the combined graphics and haptics IT. We suspect that the addition of haptic contour following will at least partly compensate for the reduced depth cues and produce an IT that is faster than vision alone—and far faster than haptics alone. As admittedly anecdotal evidence for this claim, we point out that users of FreeForm, which provides both haptic and visual display of the object being created, apparently have a better feel (pun intended!) for the shape of their objects than users of graphics-only modeling programs.

We emphasize that the above predictions cannot be made on the basis of psychophysical arguments. Simple considerations of the psychophysics of vision and haptics would tell us nothing about the unavailability of broad exploratory procedures, and little about the interrelationship between the two senses. Interface research has tended to treat the senses as additive, as though adding haptics to an interface has no effect on the performance of vision. This is obviously untrue. In our example, the whole interface will probably be more effective

than the sum of its constituent senses. An important contribution of evaluating an interface through its interaction techniques is to explicitly consider intersensory interactions.

5. Acknowledgements

We gratefully acknowledge Intel Corporation for the donation of the PHANToM and Pentium computer used in this research.

6. References

Klatzky, R. L., Loomis, J. M., Lederman, S. J., Wake, H., & Fujita, N. (1993). Haptic identification of objects and their depictions. Perception and Psychophysics, 54, 170-178.

Lederman, S. J., & Klatzky, R. L. (1996). Action for perception: Manual exploratory movements for haptically processing objects and their features. In A. M. Wing, P. Haggard, & J. R. Flanagan (Eds.), Hand and brain: The neurophysiology and psychology of hand movements, (pp. 431-446). New York: Academic Press.

Tan, H. Z. (1997). Identification of sphere size using the PHANToM: Towards a set of building blocks for rendering haptic environment. In Proceedings of ASME Dynamic Systems and Control Division, DSC-Vol. 61, (pp. 197-203). New York: ASME.

von der Heyde, M., & Häger-Ross, C. (1998). Psychophysical experiments in a complex virtual environment. In Proceedings of Third Phantom User's Group Workshop. Cambridge, MA: MIT AI Lab (http://www.ai.mit.edu/conferences/pug98).

FGB: A Graphical and Haptic User Interface For Creating Graphical, Haptic User Interfaces

Tom Anderson
Sandia National Laboratories
tgander@sandia.gov

Arthurine Breckenridge
Sandia National Laboratories
arbreck@sandia.gov

George Davidson
Sandia National Laboratories
gsdavid@sandia.gov

Abstract

The emerging field of haptics represents a fundamental change in human-computer interaction (HCI), and presents solutions to problems that are difficult or impossible to solve with a two-dimensional, mouse-based interface. To take advantage of the potential of haptics, however, innovative interaction techniques and programming environments are needed. This paper describes FGB (FLIGHT GHUI Builder), a programming tool that can be used to create an application specific graphical and haptic user interface (GHUI). FGB is itself a graphical and haptic user interface with which a programmer can intuitively create and manipulate components of a GHUI in real time in a graphical environment through the use of a haptic device. The programmer can create a GHUI without writing any programming code. After a user interface is created, FGB writes the appropriate programming code to a file, using the FLIGHT API, to recreate what the programmer created in the FGB interface. FGB saves programming time and increases productivity because a programmer can see the end result as it is created, and FGB does much of the programming itself. Interestingly, as FGB was created, it was used to help build itself. The further FGB was in its development, the more easily and quickly it could be used to create additional functionality and improve its own design. As a finished product, FGB can be used to recreate itself in much less time than it originally required, and with much less programming. This paper describes FGB's GHUI components, the techniques used in the interface, how the output code is created, where programming additions and modifications should be placed, and how it can be compared to and integrated with existing API's such as MFC and Visual C++, OpenGL, and GHOST.

I. Introduction

A significant bottleneck in productivity for the use of 3D computer generated environments is the relatively poor interface between the user and the computer. While 3D computer graphics and 3D applications have become much more complicated, interfaces to these 3D environments have remained consistent for decades. To unlock the potential of today's computers, advanced Human-Computer Interface (HCI) techniques are needed. There have been several recent thresholds that allow a new, fundamental shift in the ways that people interact with computers. Graphics cards, for example, have reached prices and rendering capabilities where personal computers can effectively utilize 3D graphics. One of the most important thresholds, however, is the ability to use our sense of touch in computer environments, which will be a critical element of future interfaces.

As with any computer technology, both software and hardware must be developed in parallel to make the technology effective. The dramatic improvements in haptics hardware are now being followed by more advanced research in haptics software. This paper describes FGB, a programming tool that can be used to accelerate haptics software development and research. FGB, a graphical and haptic interface itself, can help a programmer quickly create an application specific interface that utilizes 3D graphics, haptics, and 3D (HRTF) sound. It contains a variety of components such as buttons, sliders, text boxes, etc. that can be used to create an interface. These objects, and the

3D windows that contain them, called control modules, are easily manipulated in the FGB GHUI without any need for programming. A programmer can create an interface in an intuitive way by creating and modifying components using techniques that mimic real life. Objects can be moved directly with all six degrees of freedom, and the programmer can immediately see the end result of the GHUI as it is created. FGB then outputs appropriate code that can be compiled to create the specified GHUI.

A developer working in FGB, for example, can click and drag the cursor to create a 3D window containing objects needed in an application. If a developer desires a button, it can be created and named, and can then be directly positioned and sized within the 3D window on any wall, or even in the air. The button can be set to call a callback when it is pushed in any direction. Any modifications are handled through the FGB GHUI, which has all of the options available for any selected object. These options can be modified directly with the cursor or through text input boxes. The system therefore represents an efficient tool that can be used to quickly create a user interface that a programmer can use as a front end to haptic applications.

The code that is output uses the FLIGHT Application Programming Interface (API). FLIGHT is a development environment that was created for rapid haptic application development [2]. It is both a user interface and a programming interface. FLIGHT uses a craft metaphor in which a user can navigate the craft through a virtual space, utilize 3D tools, and interact with a 3D control panel and user interface. FGB is a tool that aids in programming within the FLIGHT development environment.

II. GHUI Components

There are several components currently available that can be used in creating a GHUI. These include control modules, buttons, sliders, button panels, text, text input boxes, force callbacks, graphics callbacks, and removal callbacks. The objects and their descriptions are given in Table 1. All of the objects given in Table 1 can be sized, scaled, positioned, and otherwise modified to fit the needs of the application while still maintaining a consistency in developed interfaces.

Control Modules	Control Modules are windows that are analogous to the common 2D windows metaphor, except that they are three-dimensional. They can be moved and scaled at any time by a user. They contain the objects listed below, in any configuration, and are the base element to create a FLIGHT GHUI. The control module walls can be made to disappear while they still contain objects or callbacks, so any of the objects below can appear to be a standalone object. Control Modules can be moved to any domain in an application. For example, they can be used within the control panel (in the craft metaphor) or near the application specific data.
Buttons	Buttons are components that are similar to their real-life counterparts. When they are pushed, a user can feel them click. They can be attached to walls, can be set to move in any direction, and have a callback associated with each movement direction that is enabled.
Sliders	Sliders are also analogous to their real-life counterparts, except that they are more flexible. They can be pushed along any axis and therefore can control three-dimensional variables directly (for example, a 3D slider can control the RGB components of color). Sliders can be set to control any variable and can be automatically adjusted by the system in real time if that variable changes elsewhere in the code.
Button Panels	As their name suggests, button panels are two-dimensional panels of buttons. There is a callback associated with a button panel that receives the number of a button if it is activated and another one that receives the number of a button if it is deactivated. The buttons can be activated and/or deactivated depending on the following button panel modes: CM_BP_TOUCH: While a button is pressed, it is on. Otherwise it is off. CM_BP_TOGGLE: When a button is pressed, it toggles on or off. CM_BP_ ONE: Exactly one button is always active in the button panel. CM_BP_MAX_ONE: At most one button is on, but the active button can be turned off.
Text	This is a text object in the control module. Text (and text input boxes below) can be sized, scaled, colored, and positioned.
Text Input Boxes	This is a text input object in the control module. These objects can be set to return ints, floats, or character strings. They are two-dimensional, however, they can be positioned anywhere in 3D space. To activate them, a user touches the text input box and a flashing cursor appears

	indicating that it is active. The tab key can be used to move between text input boxes.
Force Callbacks	Force callbacks are called during each cycle of the haptics programming loop within FLIGHT. They can be made to be called only when the cursor is in the vicinity of the control module if desired. A control module can have multiple force callbacks in any order.
Graphics Callbacks	Graphics callbacks are called during each cycle of the graphics loop within FLIGHT. They can be made to be called for the control panel or world interaction modes in FLIGHT. A control module can have multiple graphics callbacks in any order.
Removal Callbacks	A removal callback is called when a control module is deleted.

Table 1: GHUI Components.

Objects can be created in any or all domains (i.e. an object can be seen but not felt, or it can exist in the graphics, haptics, and sound domains). All of the properties for the objects can be changed while the application is running. For example, if a button is only applicable in certain situations, it can be programmed to appear or disappear appropriately. There are a number of other components that will be available in the next release of FLIGHT and FGB. Several of these are toolbars, wizard components, and tabs for different uses within a single control module.

III. FGB Techniques

When a programmer starts FGB, the base control module appears. It can be used to create a new control module, modify an existing one, and add components into a control module. When a control module is selected or created, its attributes can be modified directly with the cursor or through text input boxes. A powerful feature of FGB is that it can modify any of FLIGHT's existing control modules. If, for example, there is a control module already built that is similar to one that is desired, it can be highlighted and modified, and then the new programming code for it can be written to a file.

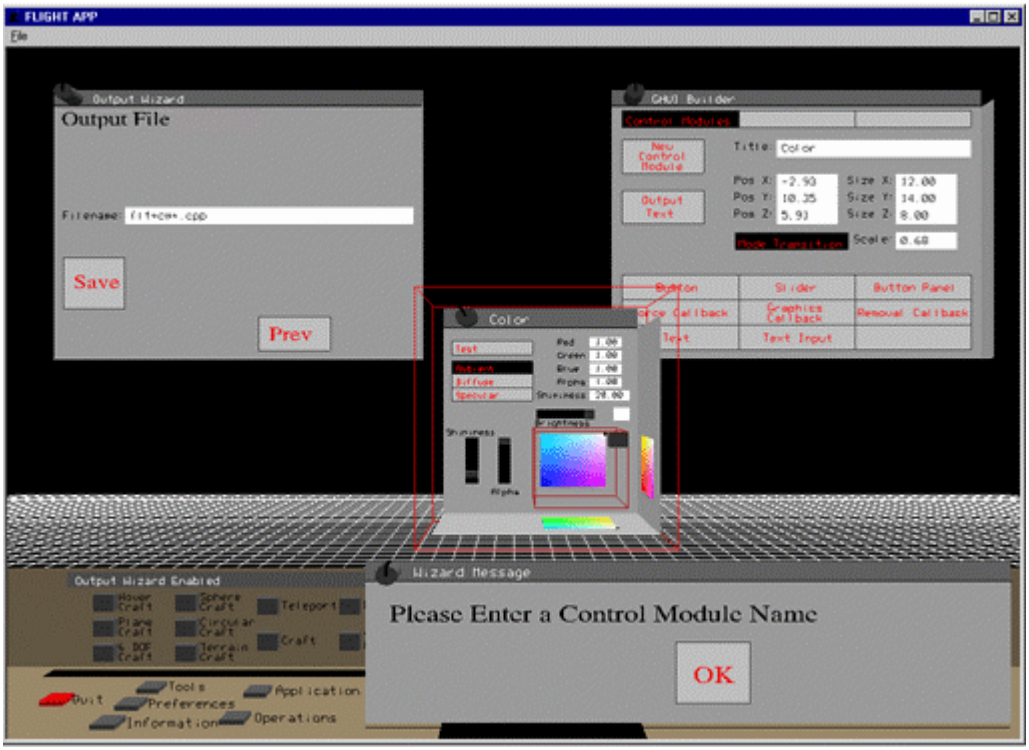


Figure 1: This figure shows FGB in use. A user has created a 3D window and is using the output wizard to output the programming code. The wizard is prompting the user for additional information needed to create the output file.

Through the base control module in FGB, any number of objects can be added to or deleted from a control module. When an object is added or selected, another of FGB's control modules appears to adjust the parameters.

Objects can also be positioned, sized, or scaled directly with the cursor, or can be adjusted through text input boxes. Whenever an object is highlighted, all of its available options are given in the FGB Objects Control Module.

After a control module is created and its objects are in place, a user can generate the code through the output wizard as shown in Figure 1. The output wizard leads a user through an intuitive process of gathering any additional information that is needed to output the code. For example, the output wizard makes suggestions for variable names that the control module code will use, and allows the user the option of changing any of those names or variables. The final step in the output wizard saves the programming code for the active control module. Before the code is saved, the wizard makes sure all the necessary information has been entered and that there are no discrepancies. If there are any errors, an error message appears and the user is taken directly to the step in the wizard where the error occurred. A control module's programming code is output to a modular C++ file, which is ready to be compiled without any further modifications. The file contains the callbacks that were created for any of the GHUI's components, where additional programming code can be added to describe the components' functionality.

After an application specific GHUI is created, it can be incorporated into other applications or API's through FLIGHT's programming interface. FLIGHT has three primary functions that are used to incorporate it into other applications: an initialization callback, a graphics callback, and a haptics callback. The initialization callback sets up memory for process communications and initializes the system. The graphics callback is called each cycle of an OpenGL based graphics loop. At the lowest level, the haptics callback simply receives a device's position and outputs a force based on that position. Because of this simple programming interface for integrating FLIGHT into existing applications, FGB represents a tool that can be used to create interfaces for a variety of applications. For example, FLIGHT was easily integrated into the GHOST Application Programming Interface. FGB can therefore be used as a tool to create haptic interfaces for GHOST based applications.

IV. Conclusions

There are a number of two dimensional graphical interface builders such as Microsoft's Visual C++ software. These tools are comparable to FGB in some ways, in that they represent an intuitive way to create interfaces that can be viewed as they are created. There are a number of differences, however. Primarily, FGB was built around haptics. Haptics is a tightly coupled component in FGB, which is an important aspect of any haptics software as it is often difficult to simply integrate haptics into applications that were not specifically designed for the need of a high speed (approximately 1000 Hz) interaction loop. Additionally, FGB is entirely three-dimensional and allows direct 3D manipulation. This is essential to both creating and using 3D interfaces where the mouse is often ineffective. A 3D interface is useful to prevent the need for alternating between using the mouse and using the haptic device when working with a 3D application.

As an initial note on the successful use of FGB, it was used to help create itself as it was built. Intuitively, it was expected that an interface such as FGB would be as useful for 3D, haptics based applications as comparable 2D graphical interface builders are for 2D interfaces. It was unexpected, however, that it would be as useful as it was in its own creation. As added functionality was incorporated into FGB, it was easier to create other new control modules that added further functionality. As a finished product, FGB could recreate itself in less than a quarter of the time and effort.

Acknowledgments

The authors would like to thank Brian Pino, Kendall Mauldin, Kevin Oishi, and Dan Zimmerer for their help in the work that lead to this paper. The inclusion of any external products described in this paper does not imply any endorsement by Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the US Department of Energy under Contract DE-AC04-94AL85000.

References

[1] J. Preece, *Human-Computer Interaction*, Addison-Wesley Publishing, New York, NY 1994.
[2] T. Anderson, "FLIGHT Users and Programmers Manual", Sandia National Laboratories, 1999.
[3] W. Barfield, T. Furness III, *Virtual Environments and Advanced Interface Design*, Oxford University Press, New York, 1995.

Haptic Vector Fields for Animation Motion Control

Bruce Randall Donald and Frederick Henle*
Dartmouth Computer Science Department

Abstract

We are developing paradigms and algorithms for browsing and editing families of animations using a haptic force-feedback device called a Phantom. These techniques may be generalized to navigation of any high degree-of-freedom system from a lower degree-of-freedom control space, with applications to telerobotics and simulation of virtual humans. We believe that modeling the animation configuration space coupled with the highly interactive nature of the Phantom provides us with useful and intuitive means of control.

We have implemented our ideas in a system for the manipulation of animation motion capture data; in particular, anthropomorphic figures with 57 degrees of freedom are controlled by the user in real time. We treat trajectories, which encode animation, as first-class objects; haptic manipulation of these trajectories results in change to the animation. We have several haptic editing modes in which these trajectories are either haptically deformed or performed by the user with expressive control subject to dynamic haptic constraints. The initial trajectories are given by sample animations (for example, motion capture data) but may be authored by other means.

1 Introduction

In our system, a parametric family of animations is encoded by a *bundle* of trajectories. This bundle in turn defines a time-varying, higher-order vector field (HOVF) on a configuration space for the animation. A haptic input device (a Phantom) provides a low-dimensional parameterization of the resulting dynamical system, and the haptic force feedback permits browsing and editing of the space of animations, by allowing the user to experience the vector field as physical forces.

In order to encode a family of animations in this manner, a number of representational problems must be solved. The mathematical and computational underpinnings of this work devolve to the theory of vector fields and dynamical systems, developed in robotics and control theory. However, their use in the context of animation authoring is novel and requires some extension.

2 How Can Haptic Vector Fields Control Animations?

The state of an animation is encoded as a point in its configuration space. A continuous animation or animation segment is encoded as a continuous trajectory in the configuration space. Since the animation and the trajectory are equivalent, we may alter the trajectory and derive a new animation from the altered trajectory. However, it is difficult to work in such a high-dimensional configuration space directly, so we provide a mapping from a lower-dimensional control space to the configuration space, and manipulate trajectories in the control space.

The control space is defined by the degrees of freedom of the Phantom. The user interacts with the Phantom by manipulating a pen-like appendage (called a "stylus"). It has six degrees of freedom, three for the position of the tip of the stylus and three for its orientation. There is also a switch on the stylus which may be used like a mouse button, e.g. to click and drag.

Thus, a trajectory in the control space is represented visually (in a two-dimensional projection on the computer monitor) and haptically (through the Phantom) as a continuous path in three dimensions. We have provided several techniques for editing existing trajectories, and as this is done the user can see the effect on the animation in real time.

2.1 Mathematical Model

We call the configuration space D, with a point in D representing one "frame" of the animation.[1] For example, in this paper we take D to represent the

*{brd,henle}@cs.dartmouth.edu

[1] In practice, the time domain will be discretized or sampled. We follow [5] in our terminology for sampling: "An animation

set of possible joint angles [13] for an articulated human figure. A control map is established so that the Phantom's degrees of freedom control the animation. This is done by constructing a mapping $h : C \longrightarrow D$ where C is the control space representing the six input degrees of freedom of the Phantom (in our case, $C = SE(3)$, the Special Euclidean group of rigid body motions in 3D). We take as input a smooth trajectory[2] $\varphi_1 : I \longrightarrow C$. Here φ_1 represents an entire animation "clip," because the mapping $h \circ \varphi_1$ defines an animation "frame" for each point t in I. Note that φ_1 trivially defines a vector field along its image $\varphi_1(I)$, namely the field of tangent velocity vectors $(\varphi_1(t), \dot{\varphi}_1(t))$; see Fig. 1.

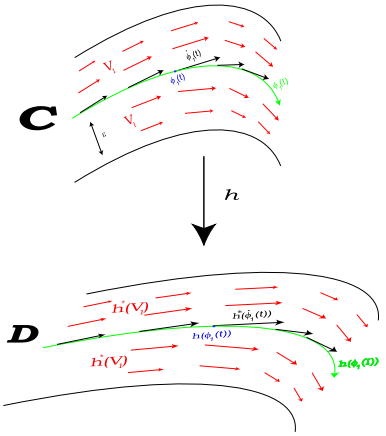


Figure 1: The haptic vector field V_1 is defined on haptic control space C. The haptic map h maps from C to the animation space D.

2.2 Vector Fields for Haptics

Follow mode: We have developed several haptic modes. Let us begin by describing one of the simplest ones. We take as input a default trajectory, corresponding to a default animation. We ask the user to perform a similar trajectory by tracing the default one, and we use a haptic vector field to make this task easier–in fact, when the user releases the Phantom or merely relaxes her grip it will trace the default trajectory autonomously. Thus, any deviations from the default trajectory are the result of an expressive act of will on the part of the user, and not simply an inability to trace with a steady hand. In order to accomplish this, we need to embed a vector field in the control space and express the field as Phantom forces.

We define a small tube of radius ε about the image of φ_1, and extend the vector field to this tube in the following manner. The field has a radial and a tangential component. The radial component Y_1 points towards the center of the tube, where $\varphi_1(I)$ lies. The tangential component X_1 near $\varphi_1(t)$ lies parallel to $\dot{\varphi}_1(t)$. Both components decrease in magnitude with distance from the tube center. In fact, the radial component also vanishes at the trajectory, so that we avoid a force discontinuity. The sum $V_1 = X_1 + Y_1$ of the radial and tangential components defines a dynamical system on C that may be viewed as a "river," pulling configurations into and along a central attracting flow defined by the animation. This vector field defines not only the flow of the animation, but also a force function, parameterized by the position in C of the Phantom; this field is experienced by the user as haptic forces. Finally, when h is injective, the vector field on C may be "pushed forward" using h^*, the derivative (or *Jacobian*) of h, to the configuration space D. See Fig. 1.

Now, the vector field in the ε-tube about $\varphi_1(I)$ defines a dynamical system on the haptic control space C, linked via the haptic control map h to the animation configuration space D. To play back an animation, the Phantom is positioned in space and allowed to travel along with the vector field. Mathematically, the resulting trajectory is obtained by ordinary integration of the vector field from a starting configuration. During this traversal, the haptic control map h defines an animation "frame" for every configuration in the resulting trajectory; sequential display of these frames results in an animation. Hence as the Phantom moves in the vector field, an animation plays (Fig. 4).

Stretchy Tubes mode: During playback, the user-supplied forces define another vector field, U. During interactive modification, the new family of animations can be represented by the sum of V_1 and the user-supplied force field U. We can record the combined vector field $U + V_1$ as a stored representation for the new animation system. See Fig. 2.

We have experimented with a few different techniques for direct manipulation of such systems, using haptic browsing and force fields. For example, suppose we are given a set of trajectories $\varphi_1, \varphi_2, \ldots$ defin-

system should have a sampling rate that is decoupled from the nominal 'frame rate' of the final product. We will speak of 'frames' at the sample rate without intending any loss of generality."

[2]Here I represents time, parameterized to the unit interval $[0, 1]$. In general, of course, animations could take different amounts of time. For cyclic animations (e.g. walking, running, hopping), time is viewed as circular (parameterized by the unit circle $\mathbb{S}^1$) and cyclic animations are represented by mappings $\mathbb{S}^1 \longrightarrow C$

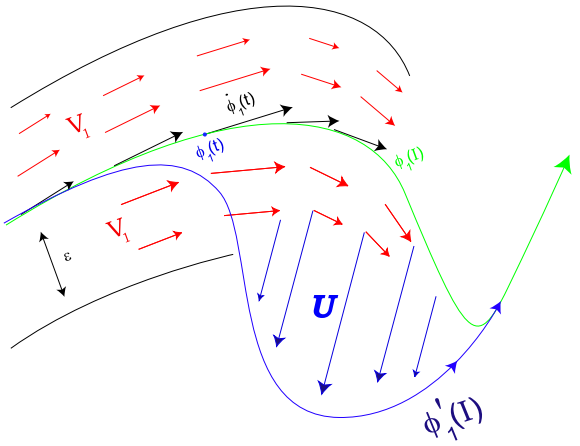


Figure 2: A sample animation is encoded as a trajectory φ_1, which induces a vector field V_1 about its image in C (see Fig. 1). During playback in FOLLOW mode, the Phantom's manipulandum by default follows the flow of V_1, therefore tracing out φ_1. Here, the user alters the trajectory by exerting physical forces (the force field U) on the Phantom. This results in an edited trajectory φ_1', which is an integral curve of the edited dynamical system $V_1 + U$. φ_1' represents a new path for the Phantom; given a haptic map $h : C \longrightarrow D$, $h \circ \varphi_1'$ encodes the edited animation.

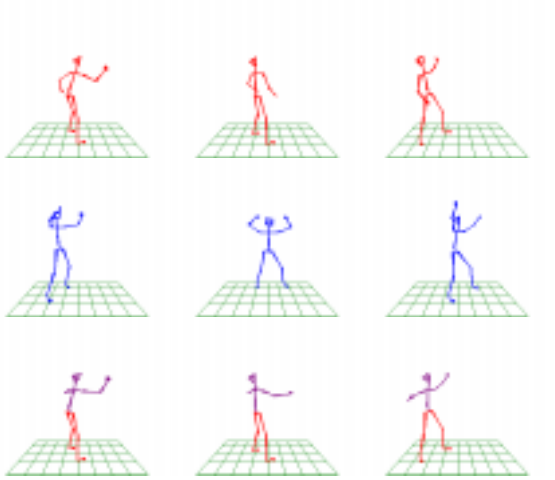


Figure 3: Our system takes as input several motion capture files. Here one is red, and depicts an angry figure making a few derisive one-armed gestures. The second is blue, and depicts a figure cheering with both arms. Next, we define a composite figure whose lower half is taken entirely from the red example but whose upper half ranges between red and blue examples according to the parameter of interpolation. To see detailed figures and animations for this paper, visit `http://www.cs.dartmouth.edu/~henle/PUG99/`

ing example animations. It is possible to build virtual tubes around the images of these trajectories in haptic control space, and to directly manipulate the tubes. These tubes may be treated as a set of springy fibers in a virtual 3-D space. We can manifest these tubes both visually and haptically as virtual objects. The Phantom can then be used to push, pull, or manipulate a folded trajectory, and thereby change the animation. During the direct manipulation, the tube haptically appears rubbery and resistant to motion ("stretchy"). See Fig. 4. For example, the manipulandum can virtually approach a trajectory tube, grab it, stretch it, and move it to a new position. Simultaneously, the user views the corresponding animation playing, while the point $\varphi_i(t)$ in configuration space (representing the animation) is seen to move along the virtual tube. Deformation of the tube changes the trajectory from φ_i to φ_i' and therefore the animation changes from $h \circ \varphi_i$ to $h \circ \varphi_i'$.

3 Previous Work

Few techniques use haptics to browse and edit the dynamical system of an animation through direct manipulation. The encoding and editing of such systems as palpable vector fields appears to be novel. Previous research falls into a few broad categories. Fundamental work in haptics and force-feedback [15, 4, 6, 20] has allowed devices such as the Phantom to be integrated with computer graphics. Most of this work is targeted for scientific visualization, or for the combined visual-haptic display of complex virtual-reality environments. Control systems and abstractions in this work have been important in building our haptic system. Vector fields have been widely used in robot control [11, 12, 17, 16, 3], and these mathematical foundations were influential in our system design. Nonholonomic control and HOVFs were developed in the context of control for non-linear geometric dynamics, and have a wide range of applications [1, 13, 2, 9, 14]. There have been a number of elegant papers on processing motion data [5, 21] multi-target motion interpolation [18], real-time control of virtual humans [10], retargeting of motion [7], motion transitions [19], and constraint-based motion adaptation [8]. Inspired by this work, we employ very simple forms of interpolation and motion processing in order to demonstrate the power of haptic vector fields for animation motion control. We believe that in the future, sophisticated motion processing, interpolation, and retargeting algorithms will be integrated with haptics for direct manipulation of trajectory bundles, and for haptic browsing of an animation's dynamical systems using vector force fields. Our paper represents a first step towards realizing that goal.

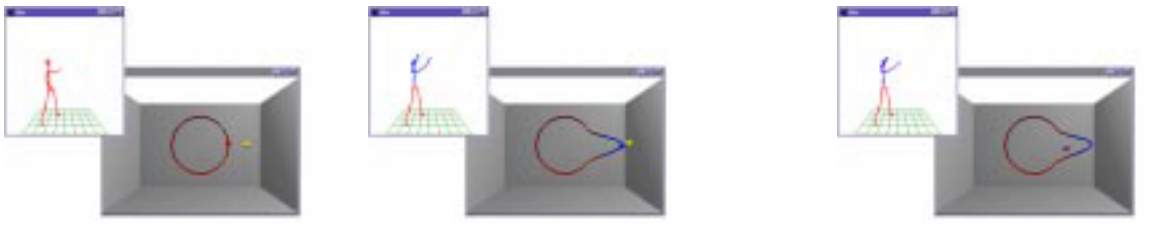


Figure 4: The STRETCHY TUBES and FOLLOW paradigms. The animation is in approximately the same frame in each screenshot. Note the green arrow indicating the force vector from the Phantom cursor in the stretch and follow pictures. **Left:** before stretch. **Center:** after stretch. **Right:** FOLLOW phase using result of stretch. The Left and Center frames show STRETCHY TUBES phase of editing the trajectory, using the inputs in Fig. 3.

References

[1] J. Baillieul and R. Brockett. *Robotics*, volume 41 of *Symposia in Applied Mathematics*. American Mathematical Society Press, Providence, RI, 1990.

[2] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning the the presence of obstacles. *Algorithmica*, 10(2):121–155, August 1993. Special Issue on Computational Robotics.

[3] K.-F. Böhringer, B. R. Donald, and N. C. MacDonald. Programmable Vector Fields for Distributed Manipulation, with Applications to MEMS Actuator Arrays and Vibratory Parts Feeders. *Int. Journal of Robotics Research,* vol. 18(2), Feb., (1999) pp. 168–200.

[4] F. P. Brooks, Jr., M. Ouh-Young, J. J. Batter, and P. J. Kilpatrick. Project GROPE — haptic displays for scientific visualization. *Computer Graphics*, 24(4):177–185, Aug. 1990.

[5] A. Bruderlin and L. Williams. Motion signal processing. *Computer Graphics*, 29(Annual Conference Series):97–104, 1995.

[6] T. A. Galyean and J. F. Hughes. Sculpting: An interactive volumetric modeling technique. *Computer Graphics*, 25(4):267–274, July 1991.

[7] M. Gleicher. Retargeting motion to new characters. *Proc. SIGGRAPH (Computer Graphics)*, Aug. 1998.

[8] M. Gleicher and P. Litwinowicz. Constraint-based motion adaptation. *Jour. Visualization and Comp. Graphics*, 9:65–94, 1998.

[9] K. Goldberg, J.C.-Latombe, D. Halperin, and R. Wilson, editors. *The Algorithmic Foundations of Robotics: First Workshop*. A. K. Peters, Boston, MA, 1995.

[10] P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Amaury, and D. Thalmann. Real-time animation of realistic virtual humans. *IEEE Computer Graphics and Applications*, pages 42–56, Sept. 1998.

[11] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research*, 5(1):90–99, Spring 1986.

[12] D. E. Koditschek and E. Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 1988.

[13] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Press, 1991.

[14] J.-C. Latombe. Robot algorithms. In T. Kanade and R. Paul, editors, *Robotics Research: The Sixth International Symposium*, 1993.

[15] M. Minsky, M. Ouh-young, O. Steele, F. P. Brooks, Jr., and M. Behensky. Feeling and seeing: Issues in force display. *Computer Graphics*, 24(2):235–243, Mar. 1990.

[16] J. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonoomous robots. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 431–459. A. K. Peters, Wellesley, MA, 1995.

[17] E. Rimon and D. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5), October 1992.

[18] C. Rose, M. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–30, Sept. 1998.

[19] C. F. Rose, B. Guenter, B. Bodenheimer, Cohen, and M. F. Efficient generation of motion transitions using spacetime constraints. *Computer Graphics*, 30(Annual Conference Series):147–154, 1996.

[20] D. C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. *Computer Graphics*, 31(3A):345–352, Aug. 1997.

[21] A. Witkin and Z. Popovi'c. Motion warping. *Computer Graphics*, 29(Annual Conference Series):105–108, 1995.

Twice the Fun: Two Phantoms as a High-Performance Telemanipulation System

Robert D. Howe
Division of Engineering and Applied Sciences
Harvard University
howe@deas.harvard.edu

Thomas Debus and Pierre Dupont
Aerospace and Mechanical Engineering
Boston University
debus@bu.edu, pierre@bu.edu

We have connected two Phantoms as a telerobotic system for use in research on remote manipulation. In this abstract we describe the hardware and software configuration of the system, which provides exceptionally good force feedback without the use of a force sensor. To quantify this capability, we present experimental measurements of forces at the remote robot using only position information and carefully calibrated gains.

System Description

Both Phantoms are the Classic 1.5 model. The operator moves the master Phantom with the stylus to control the motion of the other Phantom, which acts as a remote robotic manipulator. A simple, lightweight pneumatic gripper has been added to the remote Phantom to enable grasping of objects. The gripper is attached directly to the Phantom linkages in place of the gimbal, creating in a 3 degree-of-freedom robot manipulator.

Motion of the slave robot and force feedback to the operator are both generated by a symmetrical position controller. This control method avoids the need for a force sensor on the remote manipulator. The position of the master is used as the position command to the remote robot, which servos to this position using a conventional proportional-derivative controller

$$F_i^{remote} = K_{pi}(X_i^{master} - X_i^{remote}) + K_{vi}(\dot{X}_i^{master} - \dot{X}_i^{remote})$$

where X_i and F_i are the i^{th} components of the tip position and force ($i = x, y, z$), and K_{pi} and K_{vi} are the proportional and derivative gain components. The position of the remote robot is likewise used as the master position command, using the same control law with master and remote terms interchanged. The resulting force applied to the operator by the master is the same as the above expression with reversed sign.

The feedback gains were tuned to balance force fidelity with stability, resulting in values of $K_p = [0.5, 1, 0.5]$ and $K_v = [0.0001, 0.001, 0.0005]$. The system was programmed using the BASIC I/O subroutine library. The usual Phantom control loop runs at approximately 1 kHz, which proved inadequate for teleoperation purposes. We thus increase this rate to approximately 10 kHz by modifying the servo loop callback routine.

Force Measurement Experiment

Although only position and velocity signals are used in the controller, the system provides surprisingly high-fidelity force feedback to the operator. Subjectively, the system produces some of the best force feedback of any telemanipulation system. To quantify this performance, we measured the system's ability to measure forces at the remote robot gripper.

Two blocks with masses of 160 and then 80 grams were grasped in the remote robot gripper, lifted a few inches above the table, and replaced (Figure 1). During the experiment the robot positions and velocities, and the forces computed by the controller using the equation above, were recorded 50 Hz. As with forces measured directly by a force sensor, the recorded signals contained significant noise, so these signals were low-pass filtered using a 4th order Butterworth filter. Analysis of the power spectral density of the signals suggested that a cut off frequency between 3 and 10 Hz would preserve the essential force information.

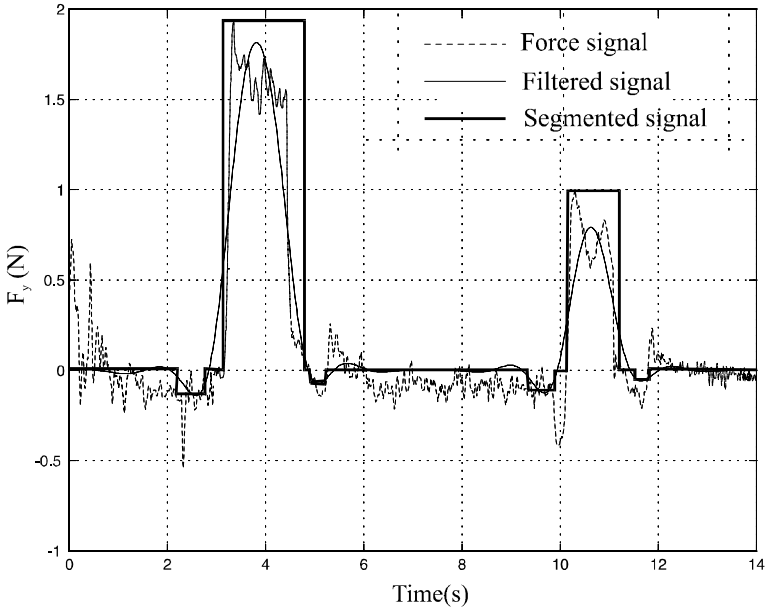


Figure 1. Force signals from lifting 160 and 80 g blocks.

Each block mass was estimated during the time the signal exceeded a 0.2 N threshold as

$$mass = \frac{\sum_{t=t_0}^{t_1} \frac{F_y(t)}{g}}{(t_1 - t_o + 1)}$$

where F_y is the vertical component of the force signal, (t_0,t_1) is the time interval when the signal exceeded the threshold, and g is the acceleration of gravity. The resulting estimates are summarized in Table 1, which shows that the measured forces are accurate to within a few percent.

Actual Mass	Estimated Mass	Error
160 g	154.2 g	3.6%
80 g	75.6 g	5.5%

Table 1. Mass estimate results.

Applications

This system is used in a project aimed at automatically identifying the properties of objects as they are manipulated in a remote environment (Debus, Dupont, and Howe 1999). Such a system can assist operators by providing quantitative information about the properties of remote objects; at present, we have developed algorithms for estimating weight, size, shape, stiffness, and friction. Applications include assisting in the disposal of unexploded ordinance and remediation of hazardous waste dumps. In addition, the information collected in real teleoperated manipulation tasks can be used to generate a virtual environment model of the task, which may then be used for operator training and critical procedure rehearsal (Dupont, Schulteis, Millman, and Howe 1999).

References

P. E. Dupont, T. M. Schulteis, and R. D. Howe , "Experimental Identification of Kinematic Constraints," Proc. IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 20 - 25, 1997, pp. 2677-82.

T. Debus P. Dupont, and R.D. Howe, 1999. "Automatic Property Identification via Parameterized Constraints," Proc. IEEE International Conference on Robotics and Automation, Detroit, May 1999.

P. Dupont, T. Schulteis, P. Millman and R. Howe, 1999. "Automatic Identification of Environment Haptic Properties," *Presence* 8(4):392-409, August.

Contact Determination for Real-time Haptic Interaction in 3D Modeling, Editing and Painting*

Ming C. Lin Arthur Gregory Stephen Ehmann Stephan Gottschalk Russ Taylor
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175
{lin,gregory,ehmann,stefan,taylorr}@cs.unc.edu

1 Introduction

Collision detection and distance computation are important for a number of engineering applications including dynamic simulation, tolerance verification, object manipulation, motion planning and control. Numerous algorithms and techniques have been proposed. (See a recent survey [LG98].) In order to meet the stringent requirement of haptic rendering, new algorithms and specialized system implementation need to be developed to substain KHz haptic update rates on complex models. This requires improving the state of the art in contact determination by at least an order of magnitude.

In this paper, we present a general and extensible algorithmic framework for fast and accurate contact determination for haptic display of complex geometric models. Our ultimate goal is to support a wide range of force feedback devices. Given a model, we pre-compute a hybrid hierarchical representation, utilizing both spatial partitioning and bounding volume hierarchy. At run time, we use hybrid hierarchical representations and exploit frame-to-frame coherence for fast proximity queries. We further discuss technical issues involved and propose approaches to improve the overall system performance. An initial prototype system has been implemented and interfaced with a 3-dof PHANToM arm and its haptic toolkit, GHOST, and applied to a number of models. As compared to the commercial implementation, we are able to achieve up to 20 times speedup in our experiments and sustain update rates over 1000Hz on a 400MHz Pentium II.

Based on our prototype implementation "H-Collide", we develop an intuitive 3D interface for interactively editing and painting a 3D polygonal mesh using a 3-dof PHANToM. An artist or a designer can use this system to create a multi-resolution polygonal mesh, further refine it by multi-resolution modeling techniques or enhance its look by painting colors and textures on it. The system allows the user to naturally create complex forms by a sense of touch and to freely interact with the design space without specification of rigorous mathematics.

2 Our Approaches

In this section, we briefly examine the requirements of general haptic rendering systems and then highlight our approaches to the problem of contact determination for haptic display.

*Supported in part by NSF grants EIA-9806027, DMI-9900157, and IIS-9821067 and Intel.

2.1 System Requirements

Any contact determination algorithm or system for haptic rendering should be able to address the following requirements for smooth interaction in virtual prototyping and engineering design: (1) scalable performance on large complex model, (2) unstructured models that may deform over time due to forces or manipulation, (3) multiple contacts and close proximity, (4) extensibility and generality.

These requirements imply that the algorithms need to have the following characteristics: (1) the runtime should be independent of the model complexity, (2) it should work well on dynamic scenes, in addition to static environments, (3) it should be able to handle contacts rapidly and *accurately* to generate realistic forces for haptic interaction, (4) the data structure and the algorithmic framework should be appicable to a wide range of haptic devices and applications.

2.2 An Extensible Design Framework

Based on the problem characteristics and the system requirements, we propose a general design framework, which specializes many earlier algorithms for haptic interaction in virtual prototyping of mechanical and structural design. This framework utilizes:

- **Spatial Decomposition:** It decomposes the workspace into regions (e.g. coarse-grain uniform grid cells, adaptive grids, etc.), implemented as a hash table to efficiently deal with large storage requirements. At runtime, the algorithm quickly finds the region(s) containing the volume swept out by the probe or the bounding volumes of the moving objects, and thereby the "region(s) of potential contacts".
- **Adaptive, Embedded Bounding Volume Hierarchy:** For each region containing some primitives of the objects in the simulated environment, we pre-compute a corresponding bounding volume hierarchy (based on OBB's, SSV's [LGLM99], K-DoP's, Spherical Shells, or others) for that region and store the pointer to the associated bounding volume hierarchy using a hash table for performing constant-time proximity queries. Each hierarchy is an embedded node of the bounding volume hierarchy of the entire model. At run-time, most of the computation time is spent in finding collisions between a bounding volume and the path swept out by the tip of the probe or between a pair of bounding volumes. To optimize this query, we have developed specialized and fast overlap tests that take very few arithmetic operations. This embedded hierarchical representation is adaptively modified for deformations due to external forces or haptic manipulation.
- **Temporal and Spatial Coherence:** The algorithm exploits temporal and spatial coherence by caching the contact geometry from the previous step to perform incremental computations.

After pre-processing, the on-line computation consists of three phases. In the first phase, it identifies "the region(s) of potential contacts" by determining which region(s) are touched by the probe path or the bounding volumes of the objects, using the precomputed look-up table. This allows the algorithm to quickly eliminate many levels of tree traversal by zooming in directly to the portions of subtrees that correspond to the regions of close proximity. In the second phase, it traverses down the bounding volume hierarchies using associated nodes of the region(s) of potential contacts, to rapidly determine if collisions have occurred using the specialized fast overlap test. In the third phase, if the contacts occur, it computes the (projected) surface contact point(s). If contacts occurred in the previous frame, we exploit temporal and spatial coherence by caching the previous pair of contact witnesses to initialize the queries and computation.

2.3 Other Optimizations

In addition to the overall framework, we also need to conduct further investigations to extend and generalize this design framework for more challenging scenarios, including handling surface-surface contacts at KHz rate. There are several algorithmic issues that remain to be addressed: (a) computation of optimal hierarchies, (b) intelligent choice of bounding volumes, (c) specialized overlap tests for spline surface primitives, and (d) more efficient handling of deformable models. Due to the space limitation, we will not go into details about each issue.

3 Implementation and Results

We are currently working on extending the design framework with a 6-DoF haptic device. However, we have implemented many of the algorithms described and developed a prototype system using a 3-DoF force feedback arm. In this section, we briefly describe the initial results we have achieved to indicate the potential of the proposed approaches.

3.1 Prototype Implementation Using a 3-DoF PHANToM Arm

Using on the design framework described in Section 2.2, we have implemented a preliminary version of the algorithms described earlier. For comparison, we have implemented adaptive grids, our hybrid approach and an algorithm using only OBBTrees and the specialized overlap test. We have applied them to a wide range of models of varying sizes (from 5,000 polygons to over 80,000 polygons as shown at http://www.cs.unc.edu/~geom/HCollide/model.pdf). Their performance varies based on the models, the configuration of the probe relative to the model, and machine configuration (e.g. cache and memory size). Our hybrid approach results in a factor of 2-20 speed improvement as compared to a native *GHOST* method.

3.2 Further Enhancement

In addition to the early prototyping system based on our design framework, H-Collide, we also investigated some of the technical issues addressed in Section 2.

Hierarchy Construction: We have implemented a combinational hierarchy construction scheme that uses both the top-down splitting and "tiling" of the polygons. In our implementation, we observed a significant speed up (more than two order of magnitude) when using hierarchies of spheres. However, we did not observe similar performance gain for OBBTrees.

Adaptive, Hybrid Hierarchies: We have implemented a software framework for performing contact determination based on hybrid hierarchies consisting of a family of *swept sphere volumes* [LGLM99]. The desired BV types are specified either at run time or computed statically offline. We observe some modest performance gain only in some cases and have not been able to reach any conclusion regarding the appropriate selection mechanism.

Specialized Overlap Tests: We also have implemented a specialized overlap test between two OBB's with SIMD instruction sets. We were able to obtain an average speed-up factor of 2-3. We plan to implement a specialized overlap test between two higher-order bounding volumes for splines/NURBS models. We believe that a SIMD or mini-parallel implementation can provide similar performance gain as well.

Local Deformation: The adaptive hybrid hierarchy was able to handle local deformation, while substaining the KHz update rate. Based on our prototype system implementation of *H-COLLIDE* [GLGT99], we developed an interactive multiresolution modeling and 3D painting system using a haptic interface, called *inTouch* [GEL99], which we will describe next.

3.3 inTouch

inTouch is an interactive multiresolution modeling and 3D painting system with a haptic interface. An artist or a designer can use *inTouch* to create and refine a three-dimensional multiresolution polygonal mesh. Its appearance can be further enhanced by directly painting onto its surface. The system allows users to naturally create complex forms and patterns not only aided by visual feedback but also by their sense of touch.

We chose subdivision surfaces as the underlying geometric representation for our system. This representation enables the user to perform global shape design and multiresolution editing with ease, allows the users to trade off fidelity for speed, and operates on simple triangular meshes. In addition, our system also offers 3D painting capability on arbitrary polygonal meshes with the haptic stylus as an "electronic paintbrush". The contact information output by H-Collide is used for both model editing and painting.

To deform and shape the model interactively, the user simply chooses the edit level (resolution) and attaches the probe to the surface. The real-time haptic display is rendered using *GHOST* and H-Collide. The deformation update process uses the force vector currently being displayed by the PHANToM to move the current surface point at the selected edit level. These geometric changes are then propagated up according to subdivision rules to the highest level of the mesh. The changes are sent across the network to the client application which maintains an identical multiresolution data structure so that it can perform the same operation to update the graphical display. Once the highest level mesh has been modified, the H-Collide and graphical data structures need to be updated to reflect the change. A local deformation algorithm is used to merge the changed triangles with the triangles that were not changed in the H-Collide data structure. The graphical output subsystem also receives the update and proceeds to modify the display lists corresponding to the changed triangles and redraw the screen.

As for 3D painting, H-Collide is used to establish the contact point of the probe with the surface of the object. The probe is then used as a virtual paintbrush with the user's preferred brush size, color, and falloff. The brush size is stretched relative to the amount of force being applied by the stylus, in a manner similar to real painting. Please refer to [GEL99] for more details about the design and implementation of *inTouch*.

4 Ongoing and Future Work

We are currently working on extending the design framework to support a 6-DoF PHANToM 1.5 to manipulate CAD models, nano-structures and flexible surfaces that may deform due to manipulation. Our ultimate goal is to support haptic interaction with complex CAD models for virtual prototyping and engineering design. We plan to continue extending our current algorithmic framework to general haptic devices and to design new hierarchy construction methods for allowing even faster local modification of surfaces, and to work on seamless integration of algorithmic techniques and data structures.

References

[GEL99] A. Gregory, S. Ehmann, and M. C. Lin. *inTouch*: Interactive Multiresolution Modeling and 3D Painting with a Haptic Interface. Technical report, Department of Computer Science, University of North Carolina, 1999.

[GLGT99] A. Gregory, M. Lin, S. Gottschalk, and R. Taylor. H-Collide: A Framework for Fast and Accurate Collision Detection for Haptic Interaction. In *Proceedings of Virtual Reality Conference 1999*, 1999.

[LGLM99] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha", Fast Proximity Queries with Swept Sphere Volumes. Department of Computer Science, University of North Carolina, *Tech. Report #TR99-018*, 1999

[LG98] M. Lin and S. Gottschalk. Collision Detection between Geometric Models: A Survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, 1998.

Providing Haptic 'Hints' to Automatic Motion Planners*

O. Burchan Bayazit Guang Song Nancy M. Amato
{burchanb,gsong,amato}@cs.tamu.edu
Department of Computer Science, Texas A&M University
College Station, TX 77843-3112

Abstract

In this paper, we investigate methods for enabling a human operator and an automatic motion planner to cooperatively solve a motion planning query. Our work is motivated by our experience that automatic motion planners sometimes fail due to the difficulty of discovering 'critical' configurations of the robot that are often naturally apparent to a human observer.

Our goal is to develop techniques by which the automatic planner can utilize (easily generated) user-input, and determine 'natural' ways to inform the user of the progress made by the motion planner. We show that simple randomized techniques inspired by probabilistic roadmap methods are quite useful for transforming approximate, user-generated paths into collision-free paths, and describe an iterative transformation method which enables one to transform a solution for an easier version of the problem into a solution for the original problem. We also show that simple visualization techniques can provide meaningful representations of the planner's progress in a 6-dimensional C-space. We illustrate the utility of our methods on difficult problems involving complex 3D CAD Models.

1 Introduction

Motion planning arises in many application domains such as robotics, virtual reality systems, and computer-aided design. Algorithms for performing fully automatic motion planning would be highly desirable for many of these applications, and as such, have been the object of much research [9]. Despite the large effort that has been spent on this problem, efficient fully automatic solutions are known only for very limited scenarios. Indeed, there is strong evidence that any complete planner (one that is guaranteed to find a solution or determine that none exists) requires time exponential in the number of degrees of freedom (dof) of the robot. Recently, some promising randomized methods have been proposed (see, e.g., [2, 8]). Nonetheless, despite the great amount of effort spent on motion planning [9], there exist many important applications that have resisted automatic solution.

We believe that some such problems could be solved by using user-input to help guide automatic motion planning algorithms. This belief is based on our experience that automatic motion planners sometimes fail due to the difficulty of discovering 'critical' configurations of the 'robot' that are in tight, or crowded, regions of the robot's configuration space but which are crucial configurations in the resulting path. In contrast, such configurations are often naturally apparent to a human observer. On the other hand, automatic methods are very good at computations that human operators find cumbersome and/or overly time consuming, e.g., detailed computations necessary to fully determine a continuous path.

In this work, we consider how to incorporate the strengths of both a human operator and an automatic planning method. In particular, we investigate how haptic and visual interfaces can be used to enable a user and an automatic motion planner to cooperatively solve a motion planning query. Haptic interfaces enable the user to feel and naturally manipulate the virtual objects, which provides valuable information about the environment not present in visual displays. While haptic devices have been used in some limited situations to generate robot trajectories (see, e.g., [4, 7, 11, 13]), they have not been used in truly cooperative systems involving human operators and more general automatic planners. Advances in this area will have important applications in many areas in addition to motion planning, e.g., in augmented reality training systems where a motion planner and an employee could work together to train the worker to perform complex tasks.

*This research supported in part by NSF CAREER Award CCR-9624315 (with REU Supplement), NSF Grants IIS-9619850 (with REU Supplement), EIA-9805823, and EIA-9810937, and by the Texas Higher Education Coordinating Board under grant ARP-036327-017. Bayazit is supported in part by the Turkish Ministry of Education.

We consider a scenario in which the human operator manipulates a virtual object using the haptic interface, captures (some) configurations of the object, and passes them to the planner. The planner then attempts to use these configurations to generate a valid motion plan. The planner's progress is communicated to the operator using a visual overlay on the virtual scene. Our goal is to develop methods which are natural for the human operator and beneficial to the automatic planner. Toward this end, we consider the following questions:

- How can the motion planner best utilize the user-generated input?
- What are 'natural' ways for the user to understand the progress made by the motion planner?

We propose and analyze several techniques for incorporating user-generated input in *probabilistic roadmap (*PRM*)* motion planning methods [8]. In particular, we show that simple randomized methods inspired by an obstacle-based PRM (OBPRM) [1] are quite useful for transforming approximate, user-generated paths into collision-free paths. We also illustrate that simple visualization techniques can provide meaningful representations of the planner's progress in a 6-dimensional C-space on the graphical display of the 3-dimensional workspace.

2 Probabilistic Roadmap Methods

The automatic planning method used in our system is the *obstacle-based probabilistic roadmap method (*OBPRM*)* [1], which is a representative of the class of planners known as *probabilistic roadmap methods* (PRMs) [1, 6, 8]. Briefly, PRMs, use randomization (usually during preprocessing) to construct a graph in C-space (a *roadmap* [9]). Roadmap nodes correspond to collision-free configurations of the robot. Two nodes are connected by an edge if a path between the two corresponding configurations can be found by a local planning method. Queries are processed by connecting the initial and goal configurations to the roadmap, and then finding a path in the roadmap between these two connection points.

We believe PRMs are good prospects for cooperative human/computer planning due to their need for improvement in crowded environments and because they are amenable to incremental construction. For example, human operators could generate configurations to be included in the roadmap or sequences of configurations to connect different connected components of the roadmap.

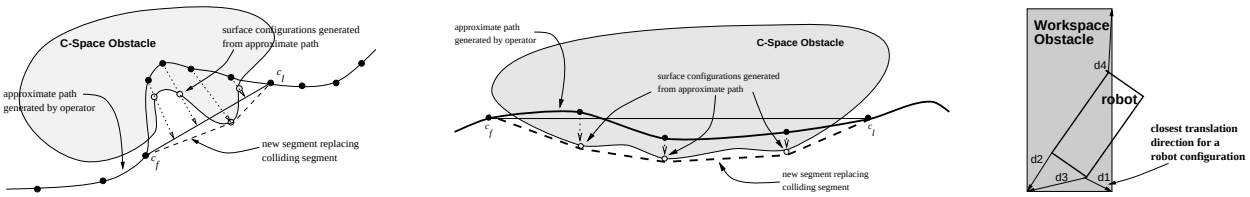
For the human operator to help the automatic planner, he must first understand where the planner is having difficulty. This is, however, a challenging task since most planners, including OBPRM, work in C-space, which is a higher dimensional space than the graphical interface available for display. The naive method of simply displaying a configuration in the the workspace is acceptable if only a few configurations will be shown. However, if many configurations must be displayed simultaneously (e.g., a roadmap), then the resulting scene could be as difficult to understand as a high-dimensional configuration space. Thus, what is needed are methods of 'projecting' configurations into the workspace.

We have implemented two such methods. In the first, we represent each robot configuration by a single point in the workspace, e.g., the positional coordinates of a reference point on the robot's local frame of reference such as the center mass. Connections between configurations are displayed as edges between their projections. Since multiple configurations can project to the same workspace representation, we use colored edges to differentiate roadmap components. A problem with this approach is that all orientation information is lost, and this can be very important in crowded environments. Our second method remedies this problem by displaying a reduced-scale version of the robot, in its actual orientation, instead of a single point.

3 Generating and Processing Haptic Paths

The operator's role is to capture configurations that will be useful for the planner. For example, the operator could help the planner by viewing the workspace representation of the evolving roadmap, and capturing paths of configurations that will enable the planner to connect different roadmap components. In our implementation, the operator generated sequences of configurations (*paths*) by attaching the haptic device to the virtual robot and manipulating it in the VE. The system recorded these configurations at regular intervals, and then passed them to the planner. The configurations generated by the operator could be **free** (i.e., collision-free configurations of the robot), or **approximate** (some penetration of the robot into the obstacles is allowed, which can be viewed as 'dilating' the robot's free space [5]). An advantage of working in a dilated free space is that it allows the operator more freedom of movement.

The planner's goal is to use the operator-generated paths to improve the connectivity of the roadmap. If the operator has collected a free path, this can immediately be added to the roadmap. However, if an approximate path



(a) (b) (c)

Figure 1: Pushing approximate paths to free space: (a) simple push, (b) shortest push, (c) workspace assisted push.

was collected, then the planner can incorporate only the free portions of this path and must attempt to 'push' the colliding portions to the free space. The motivation behind our methods is that even though a node is in collision, there should be a free node nearby since the path was user-generated and so is expected to be 'close' to a 'good' configuration. Thus, the main challenge is determine the best direction in which to 'push' the colliding segment. We have experimented with several different methods for performing this deformation which are illustrated in Figure 1.

The general scenario we consider is as follows. We assume there is a single colliding portion of the path; if there is more than one, each one can be considered separately. We use c_f and c_l to denote the free nodes immediately preceding and following the colliding segment, and we let n_c denote the number of configurations in the colliding segment. Generally, we select $n_{\text{push}} = \lfloor c \cdot n_c \rfloor$ of the colliding configurations to push to the free space, for some constant $0 < c \leq 1$. Our first method, *simple push*, attempts to push the colliding configurations towards the straight-line in C-space connecting c_f and c_l, while the *shortest push* method trys to find the closest free configuration to each colliding configuration. Our third method, *workspace-assisted push*, identifies nearby pairs of robot and obstacle vertices and translates the robot away from the obstacle vertex towards the robot vertex.

Another way to utilize the approximate path is to use iterative pushing. The basic idea is to first solve a simpler version of the original problem (one may do this, for example, by "shrinking" the robot), and then use the resulting solution path as an 'approximate path' input for the harder problem. The process can be applied iteratively until the original version of the problem is solved. The initial input path could be haptically generated, or simply one found by an automatic planner. Note that this approach is similar to the idea of building a roadmap in a dilated free space [5].

4 Haptic Interaction for Motion Planning

Our prototype system consisted of a 3-dof PHANTOM haptic device [10], an SGI O2 graphics workstation (graphics display) and an SGI Octane (computation server). The graphics keeps track of the position updates of the PHANTOM finger tip, and the PHANTOM generates force-feedback using collision/penetration information from the computation server. Our haptic-interaction applications were developed using the C++ *General Haptic Open Software Toolkit* (GHOST SDK) [12]. The operator can use the PHANTOM to manipulate a single rigid object in the virtual scene. Haptic interaction is achieved by the following steps:

Haptic interaction is achieved by the following steps: (i) track motion of the user, (ii) detect collision between the user controlled probe (virtual finger tip) and the virtual objects, (iii) compute the reaction forces in response to contact and motion, and (iv) exert forces on the PHANTOM.

In the current system, the user can turn on or off the collision detection feature. Also, the forces can be applied to the PHANTOM in three modes: maximum-force, half-force and zero-force. This variable strategy enables users to choose between freedom and restriction of movements. For example, in the half-force case if the robot is close to the obstacle, the user will be aware of the closeness while still being able to move inside the obstacle if he chooses. However, in the maximum force case, he would be forced to move away from the obstacle in case of the collision. The zero-force case is the same as when collision detection is turned off.

To achieve realistic feedback, all steps should be executed with 1 kHz frequency. However, in our applications involving complex CAD models, the collision detection rate was usually around 10 Hz, which is 100 times slower than needed. Thus, to achieve realistic interaction, we need to determine some reasonable feedback to apply during the 99 idle collision detection updates. Our solution was to use a heuristic to decide if a collision occurred during these idle periods. Since we collect approximate paths anyway, a heuristic is an acceptable solution and actually it performed very realistically. Basically we used the last computed free configuration, and the minimum distance (clearance) vector from the robot to an obstacle as our references. When waiting for a result from the collision detection routine, we projected the vector between the current configuration and the last free configuration onto minimum distance vector,

(a) (b) (c)

Figure 2: The flange problem is shown in (a) and (b): (a) a colliding configuration from the user-generated approximate path, and (b) the resulting collision-free configuration it was pushed to. For the alpha puzzle (c), we show a narrow passage configuration found by the workspace directed push method.

Comparison of Different Approaches			
Env	Method	Running Time(secs)	
		Gen/Push	Connect
Flange .85	OBPRM	304	119
	Haptic Push	28	1.7
Flange .95	Haptic Push	416	2653
	Iter .85 Path	86	18
Flange 1.0	Iter .95 Path	586	1100

Comparison of Different Approaches			
Env	Method	Running Time(secs)	
		Gen/Push	Connect
Alpha 1.5	OBPRM	30	1649
	Haptic Push	20(9.4)	309(3.6)
Alpha 1.2	Haptic Push	24(6.8)	274(3.0)
	Iter 1.5 Path	31(10)	420(5.0)

Table 1: Node generation/pushing and connection times for each method to successfully solve the query in the specified environment. Values shown are complete processing times; numbers in parenthesis show only time spent processing haptic input. The label Iter.85.Path indicates that the iterative push used the solution path for the .85 version as input, and similarly for others.

and if this projected vector was greater than the minimum distance, we assumed a collision occurred. The details of this approach can be found in [3].

5 Experimental Results

We evaluated the various pushing methods described in Section 3 in terms of their ability to generate useful free configurations. The results reported here were obtained using two environments: the *flange* and the *alpha puzzle*. The flange environment consists of a fixed rectangular part with a circular opening (the obstacle, 990 triangles) and a curved pipe (the robot, 5306 triangles) that must be inserted into the opening in the obstacle (see Figure 2(a).) The alpha puzzle environment consists of two identical twisted, intertwined tubes (one the obstacle and one the robot, 1008 triangles each); the objective of the puzzle is to separate the tubes (see Figure 2(c).) (We worked with slightly easier versions of the models, which were obtained by scaling the pipe by a factor of .85 in the flange, and the obstacle tube by a factor of 1.2 in one dimension in the alpha puzzle.) For both environments, all three methods were tested on the same user-generated path.

Our experiments showed that `simple push` and `shortest push` are fairly successful with more than 90% of the generated nodes pushed in the right direction (i.e., toward the narrow passage in workspace). We note that for both these methods to be most useful, the user-generated configurations must be close to the desired free configurations, and moreover, they must be sampled at a high enough frequency so that the resulting free configurations can be connected to complete the path.

In terms of the `workspace assisted` method, we observed that this method has a success rate of around 60-70%. This is expected since this method may try many unpromising directions since it selects vertex pairs based only only proximity information. However, some of the nodes it found were critical in terms of the connectivity.

In an overall comparison of the pushing methods, the winner was `shortest push` since it had the fastest node generation time and the highest success rate with a small number of nodes being generated (which is an advantage in the connection phase). The detailed results of the experiments can be found in [3].

The experiments also helped us to compare haptically-enhanced motion planning and fully automatic motion planning. The results are shown in Table 1. In both environments, the haptic hints not only enabled us to solve the problems much faster, but also allowed us to solve harder problems than we were able to solve using the fully automatic method alone.

The results for the flange are most dramatic. Our fully automatic planner was only able to solve the .85 version in a reasonable amount of time, but starting with the haptically generated path, we were able to solve the original problem quite quickly using the iterative pushing technique. Moreover, the fast solution times for the iterative method show that using a valid solution path for an easier version of the problem as the starting point (approximate path) for a harder version is a very useful technique for the flange. Finally, we note that the hints (i.e., the approximate paths) also reduced the solution time dramatically. For example, the generation of the roadmap for the flange .85 version took 423 seconds with the automatic planner, and only 30 seconds for the haptically enhanced version.

For the alpha puzzle, we observed similar results. That is, the haptic hints greatly reduced the solution times and enabled us to solve harder problems than with the fully automatic planner. One difference noted was that the iterative method works much better for the flange than for the alpha puzzle. This can be explained by the different geometrical structures of the two problems. Although it appears as if the iterated method actually took longer than the direct method, this is heavily dependent on the quality of the approximate path collected, and the results above indicate that we collected a relatively better path for the 1.2 version than for the 1.5 version, i.e., the operator happened to do a better job in the 1.2 version.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 155–168, 1998.

[2] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, 1991.

[3] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. Technical Report 99-021, Dept. of Computer Science, Texas A&M University, Oct 1999.

[4] G. E. Hovland and *et al.* Skill acquisition from human demonstration using a hidden markov model. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2706–2711, 1996.

[5] D. Hsu, L. Kavraki, J-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998.

[6] D. Hsu, J-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2719–2726, 1997.

[7] M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2700–2705, 1996.

[8] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[9] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[10] T. H. Massie and J. K. Salisbury. The PHANToM haptic interface: A device for probing virtual objects. In *Int. Mechanical Engineering Exposition and Congress, DSC 55-1*, pages 295–302, Chicago, 1994. C. J. Radcliffe, ed., ASME.

[11] B. J. McCarragher. Force sensing from human demonstration using a hybrid dynamical model and qualitative reasoning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 557–563, 1994.

[12] Sensable Technologies, Inc. *GHOST Software Developer's Toolkit Programmer's Guide Version 1.21*.

[13] C. P. Tung and A. C. Kak. Automatic learning of assembly tasks using a dataglove system. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1–8, 1995.

Providing reliable force-feedback for a virtual, echographic exam of the human thigh

Diego d'AULIGNAC and Remis BALANIUK[1]
SHARP Project - GRAVIR/INRIA Rhône Alpes - France
Email : Diego.D_Aulignac@inrialpes.fr, Remis.Balaniuk@inrialpes.fr

Abstract

Echographic images are collected by applying pressure on the thigh with a probe emitting and receiving ultrasounds. This makes it possible to visualise the state on the vein in the thigh, which if it is healthy will collapse under the influence of the external pressure. However, if it does not collapse it may indicate the presence of a thrombosis. Therefore, from the force applied and the echographic image received, the practitioner may be able to make a diagnosis. We have implemented the dynamic model of a human thigh that corresponds satisfactorily to real stress-strain data collected on a human thigh. Presently echographic image generators exist, that given a deformation will be able to simulated a range of pathologies that can be encountered. The aim is to use our dynamic model in an echographic simulator, which allows the practitioner to train himself to a reasonable competence before making a diagnosis on a real patient. For this, haptic return is of paramount importance. Our model, based on a spring-damper network, runs at frequencies of approximately 100 updates per second. This, however, for haptic devices is not sufficient, and may introduce vibration in the force feedback, and therefore, impair the practitioners learning. In this paper we show how we interfaced a Phantom haptic device to this model. Using an approach of fast local modelling we counter the lower update frequencies provided by the dynamic model.

1. Introduction

Echographic image generators [henry-97] use a set of echographic images to construct a more general model that takes into account the orientation and pressure exerted with the echographic probe. Using this generators is possible to build an image that is dependant on the deformation caused by the echographic probe considering additional factors such as the arterial and venous pressure. By modifying these factors we can simulate a set of pathologies on which medical students could be trained for the identification of the latter.

However, this kind of tool does not provide us with a sense of the force needed to provoke this deformation. This notion of force is of paramount importance in the simulation of an echographic exam to be of use for training purposes.

The goal of our work is to lay the groundwork for the development of an echographic simulator with force feedback. In the final system, the trainee will be looking at artificially generated echographic images, and interacting with a computer simulated dynamic thigh model through a haptic interface.

The model used to simulate the dynamics is based on a spring-damper network as described in Section 3. The parameters chosen for this model are derived from measurements on a real thigh as published in [aulignac-99].

This model will only update its state at a maximum frequency of approximately 100Hz. Our aim is to provide force feedback through means of a haptic interface of type PHANToM. These interfaces require a very high update rate for the force and typically this frequency is around 1KHz. However, the physical model that simulates the behaviour of the thigh is not able to provide force values at such rates.

---

[1] Also assistant professor at the Universidade Católica de Brasilia - Brasil

In this paper we present how a local model can bridge the gap between the simulation frequency and the haptic frequency.

2. Overview of the system

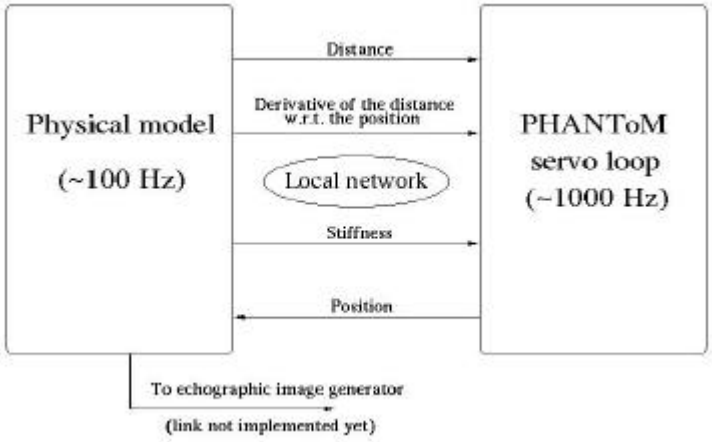


Figure1 : System overview

Figure 1 shows how the physical model and the haptic interface (PHANToM servo loop) are connected via the local network by socket connections. The haptic interface is conceived as an external module of the system simulating the physical model. Based on a simplified local model (see Section 4) the force values are estimated inside the servo loop at the haptic rate.

The physical model of the thigh does not supply force values to the haptic interface. It receives the actual position of the virtual echographic probe, corresponding to the position of the end effector of the PHANToM arm. It computes and returns the minimum distance from the probe to the thigh as well as the partial derivatives of this distance with respect to the probe position. It passes also a linear approximation of the local stiffness of the tissue of the thigh. Finally, it updates its own dynamic state. These steps are repeated in a loop at a rate which directly depends on the physical model computational delay.

In the haptic interface the local model is updated in order to fit the values informed by the physical model. This update is repeated at the same rate of the physical model update. The servo loop rate does not depend on the local model update rate.

3. Description of the physical model

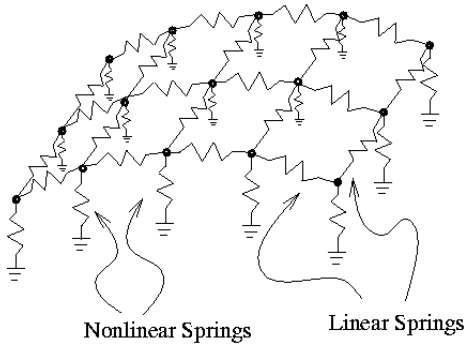


Figure2 : Two layer model of the thigh.

Based on the experimental data and the computational requirements, a two layer lumped element model is chosen. The two layer model is composed of a surface mesh of masses, linear springs and dampers, and a set of nonlinear springs orthogonal to the surface to model volumetric effects by giving normal support to the surface mesh (see Figure 2). The deformation-force relation of the nonlinear springs are of the form :

$$f(x) = \frac{x}{ax + b}$$

The nonlinear spring response is chosen to model the incompressibility of the thigh after a certain deformation. In the model, the values of the surface elements are chosen uniform whereas the parameters of the nonlinear spring vary around the mesh to model heterogeneous nature of the thigh while keeping the number of unknown parameters small.

The parameters of the springs have actually been fitted to match the real deformation measured on a human thigh [aulignac-99]. The typical update frequency of our physical model on a R10000 processor is approximately 100Hz.

Implicit Integration is used for the integration of the dynamic, differential equations as discussed in [baraff-98]. Explicit Newton-Euler takes no notice of wildly changing derivatives and in our case, forces change very rapidly when the position of point on the surface of the thigh changes, and the only solution to stop the integration from diverging is to reduce the time step, and this is not the best thing to do in real-time applications. The general form of the implicit integration (also known as *backward* Euler method) has the following formulation :

$$\Delta Y \left[\frac{1}{\Delta t} I - \lambda (\nabla f)|_{Y=Y_0} \right] = f(Y_0) \qquad (1)$$

where Y is the state vector expressing the positions and velocities of all particles. Thus ΔY represents the change in state and $f(Y_0)$ the derivative at the current state. Further, I is the identity matrix, ∇f the Jacobian matrix derived from the spring configuration of the object, and λ is a constant where $\lambda \in [0,1]$. The resulting square matrix is composed of n^2 elements, where n is the size of the state vector. We use the conjugate-gradient method [press-92] to solve this equation.

The added advantage of using implicit integration, is that we can derive the stiffness parameter that is passed to the local model directly from the Jacobian matrix. Here we use the linear approximation of the rate of change of the force with respect to the position. Even if the forces change non-linearly due to the non-linear springs, a linear approximation at the given time is sufficient for our application.

4. A haptic interface based on a buffer model

The disparity between the physical model update frequency and the haptic rendering frequency can compromise the haptic interface. The PHANToM needs to display forces at rates around 1KHz to maintain a realistic sense of touch. Simple solutions as forces interpolation cause delays in the haptic rendering, and presenting force feedback in the presence of even small delays has been shown to create operator-induced instabilities [richard-96].

To bridge this disparity we use a *buffer model* between the physical model simulator and the haptic device. This approach was firstly proposed in [balaniuk-99]. The haptic interface is conceived as an external module of the physical simulator, able to estimate the contact forces to be displayed using its own simplified physical model. This simple model locally emulates the "real" physical model and can estimate contact forces at the haptic rate.

The buffer model is a generic numerical model, defined by a set of parameters, continuously adapted in order to fit the values informed by the physical simulator. The contact forces estimation is based on a very simple physical simulation, with a point interacting with a non deformable constraint surface defined by the buffer model.

This constraint surface represents the nearest portion of the thigh with respect to the probe. The position and the shape of this surface will be defined by the model parameters.

The constraint surface is defined in the haptic device configuration space. In an ideal framework, the haptic device would offer force and torque feedback, and the proxy would move in a 6-D configuration space. In this case, the proxy motions could be expressed in full probe motions (6 dof : translation and rotation). The PHANToM device we use has no torque feedback, and the proxy is a point in a 3-D configuration space. The proxy motions are expressed only by translations of the probe in the virtual environment (3 dof). The probe does not rotate.

Inside the haptic interface we use a constraint-based framework. The haptic device location inside the configuration space is defined by two values : its effective position and a virtual location, subject to the constraining obstacles. This virtual location, called the "god-object" in [zilles-94] and "virtual proxy" in [ruspini-97], is the place the device would be if the obstacle was infinitely stiff.

The haptic interface is defined by two processes: the model manager and the haptic loop.

The model manager interacts with the physical model, informing the user motions and obtaining the minimum distance between the haptic probe and its closest contact point in the virtual thigh. Using this distance and its derivatives, the manager will adapt the constraint surface to locally fit these values. The physical model informs also the local stiffness in the contact point.

The model manager uses the obtained distance and its partial derivatives to update the buffer model. Using this values and the approach exposed in [balaniuk-99] the model manager adapts the buffer model in order to emulate the physical model nearby the probe position.

The haptic loop follows the constraint based rendering framework. The contact forces are estimated using the constraint surface, the proxy position and the informed stiffness.

To implement this framework, at each time step of the haptic loop we obtain the haptic device configuration and we change the proxy position to reduce its distance to the haptic device position subject to the constraint surface.

Refers to [balaniuk-99] see how the proxy position is determined.

The estimation of forces in the constraint based framework becomes easy after the proxy location is determined. Simple impedance control techniques can be used to estimate the forces, using the given local stiffness.

The general form of the forces estimation can be given by the Hooke's law :

$$f = k(x - p)$$

where k is proportional to the local stiffness, x the proxy position and p the probe position.

Since we have the rate of change of the forces by the way of the Jacobian calculated in equation (1) we can use this information when passing the stiffness parameter. This way the local model will respect the non-linear force/deformation characteristic.

5. Results

Figure 3 shows a user manipulating the PHANToM to interact with the system.

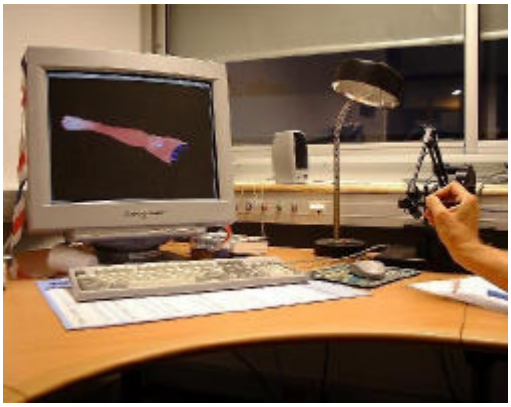


Figure 3 -Using the system

Figure 4 shows the model as it has been built in our simulation system. A force is being applied on the thigh using a probe which provokes a deformation which is in accordance with the measurements taken.

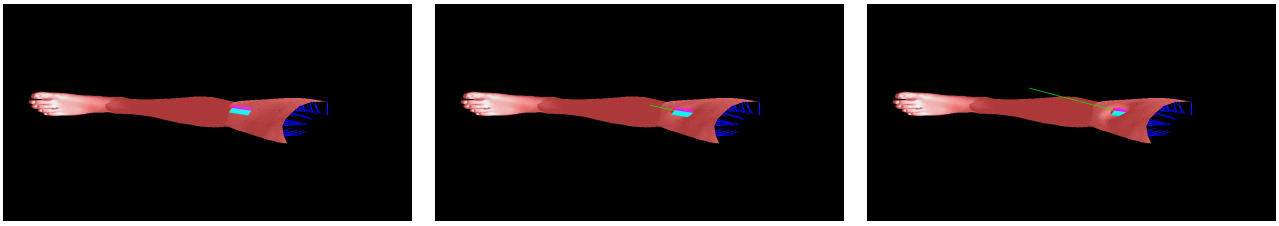


Figure 4 - Deformation of the thigh by applying pressure with the probe.

The force-feedback is smooth and there is no noticeable trembling even when the forces are large.

6. Conclusions

In this paper we have presented a physical simulation system providing graphic and haptic interfaces for a virtual exam of the human thigh. A spring damper model of the human thigh was defined from experimental data. This model is simulated in real-time using an implicit integration method. The haptic interface was conceived using the buffer model approach. Using this approach the haptic rendering rate does not depend on the physical model computational delay. This approach can provide reliable haptic rendering even when the simulation frequency is far lower than the demanded haptic rendering frequency.

The purpose of this work is to contribute for the development of an echographic simulator with force-feedback.

Acknowledgments

Thanks to the GMCAO project at the TIMC lab in Grenoble. This work was partially supported by an INRIA grant within the framework of the AISIM initiative and the France-Berkeley Fund.

Bibliography

[balaniuk-99] R. Balaniuk. Using fast local modeling to buffer haptic data. In *Proc. of the Fourth Phantom User Group Workshop –PUG99*, Boston (US), October 1999.

[baraff-98] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Computer Graphic (Proc. SIGGRAPH)*, pages 43-54, 1998.

[aulignac-99] D. d'Aulignac, C. Laugier and M.C. Cavusoglu. Towards a realistic echographic simulator with force feedback. In *Proc. of the IEEE-RSJ Int. Conf. On Intelligent Robots and Systems*, Kyongju (KR), October 1999.

[henry-97] D. Henry. *Outils pour la modelisation de structure et la simulation dèxamens echographiques.* PhD thesis, Universite Joseph Fourier, Grenoble (FR), 1997.

[press-92] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. Numerical Recipes in C. Cambridge Univ. Press, 2 edition, 1992.

[richard-96] P. Richard, G. Birbent, P. Coiffet, G. Burdea, D. Gomez, and N. Langrana. Effect of frame rate and force feedback on virtual object manipulation. In *Presence*, 1996.

[ruspini-97] D. Ruspini, K. Kolarov and O. Khatib. Haptic interaction in virtual environments. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Grenoble (FR), September 1997.

[zilles-94] C.B. Zilles and J.K. Salisbury. A constraint-based god-object method for haptic display. In Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems, vol. 1, pages 149-150, Chicago ,IL (US),1994.

The Virtual Reality Dental Training System - *Simulating dental procedures for the purpose of training dental students using haptics*

John F. Ranta
Jranta@teneocomp.com

Walter A. Aviles
Aviles@teneocomp.com

Teneo Computing

Introduction

Dental students currently use artificial teeth and jaws, along with real dental instruments, to practice cavity preparation and other procedures. These plastic models lack the level of detail and material properties needed to exactly mirror real life teeth and procedures. Furthermore, real life experiences such as bleeding and tooth extraction cannot be simulated with these plastic training models systems. As such, current training procedures require that dental students gain a portion of their required experience while practicing on patients. This is obviously less than optimal. Furthermore, utilizing classical, visual-only, computer simulations is not acceptable – a significant part of the student's learning is sensorimotor in nature. A "hands-on" curriculum is literally required.

Haptically enhanced simulations, however, can provide the sensorimotor involvement needed for dental training. Moreover, if the simulation can provide a level of haptic realism equal to or better than the artificial-tooth-and-jaw-approach, it promises to be superior to the current physical fixture approach in simulating other aspects and procedures as discussed above. In addition, VR simulation offers "super-reality" training benefits that are not possible with either plastic models or live patients. Thus, the student can repeat and playback procedures many times, precisely measure and quantify their results and even zoom in or work in x-ray mode to see their work in detail.

This paper discusses a project jointly undertaken by the Harvard School of Dental Medicine and Teneo Computing to develop a Virtual Reality Dental Training System (VRDTS) utilizing the PHANTOM™ haptic interface device (See Figure 1). This project is exploring the use of haptic interaction in the dental training curriculum. Both Teneo and Harvard believe that training dental students in certain procedures is an excellent application of the particular 3D haptic capabilities of the PHANTOM. The first phase of this project, the development of a prototype for simple cavity preparation, is discussed. This prototype will be evaluated in the classroom to determine the efficacy of Virtual Reality (VR) techniques for training dental students.

The Virtual Reality Dental Training System

A dentist strongly depends on touch feedback from the tool tip for diagnosis and positional guidance during procedures. In this system we are integrating the PHANTOM

haptic interface with 3D stereo displays and advanced software techniques to create a realistic 3D simulation. The haptic stylus enables the student to orient and operate simulated dental tools. Working on a virtual model in a stereo display, dental students can use a simulated pick to probe a tooth, or a simulated drill to prepare a tooth for cavity repair. Since almost all dental instruments are force-to-a-point tools, these map very well to a PHANTOM-based simulation.

The system consists of a dual Pentium Windows NT workstation with an OpenGL® graphics accelerator. A Phantom Desktop or Phantom Premium with Encoder Stylus provides haptic interaction. An SVGA computer display is used, which provides refresh rates necessary to render stereo images using LCD stereo shutter glasses. The tooth models are rendered volumetrically, which enables the presentation of tooth material as a solid, and provides for the high speed calculation of drilling and filling procedures. The tool head is rendered similarly as a volume, which enables high speed collision detection. A tooth (molar) model, obtained on the Internet as an STL (stereolithography) file [1] is provided in the prototype in both healthy and decayed forms.

The system uses MFC style menus for tool selection and operation, file load/save, view modes and application operations. NDH (non-dominant hand) use of keyboard and mouse is enabled for application interaction.

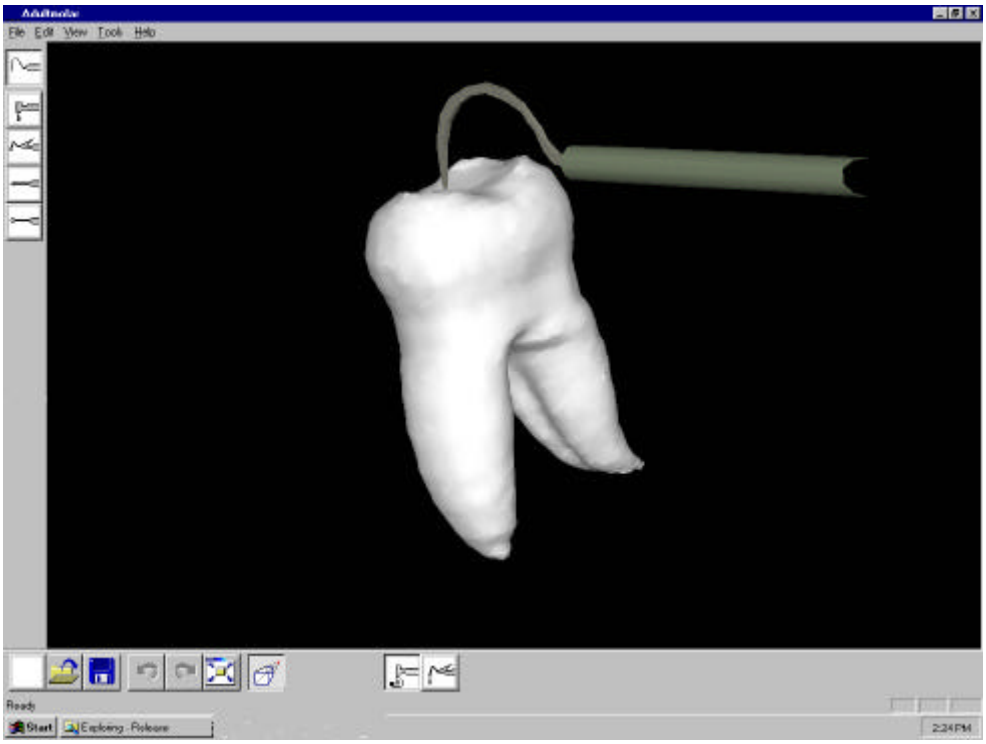


Figure 1 – VRDTS Screen

The Cavity Preparation Simulation

The prototype simulates a simple Cavity Preparation (CP) procedure, wherein the student diagnoses, removes and repairs an occlusal cavity. [NOTE: “Occusal”refers to

the bite surface of a tooth.] The user has 4 instruments simulated in the system (see Figure 2 below) – a pick, a drill (burr), a carrier, and a carver [2].

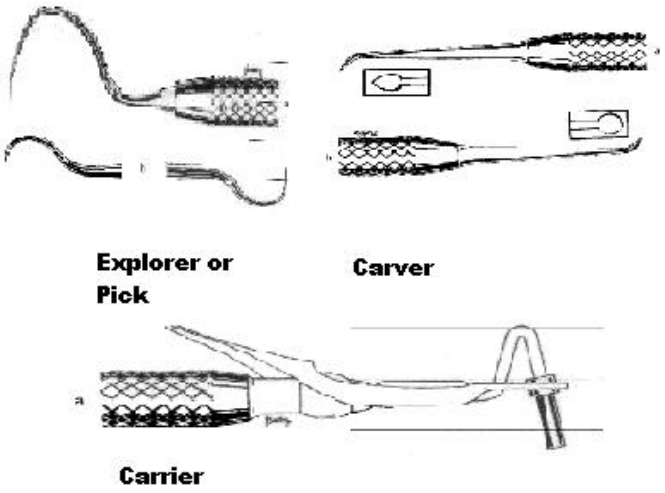


Figure 2 – Dental Instruments

A decayed tooth may be modeled as consisting of 4 different materials, enamel (hard like porcelain), dentin (softer, like ivory), pulp (very soft, like jello), and cavity (somewhat soft, like leather). The system simulates a simple (occlusal) cavity in a molar (see Figure 3).

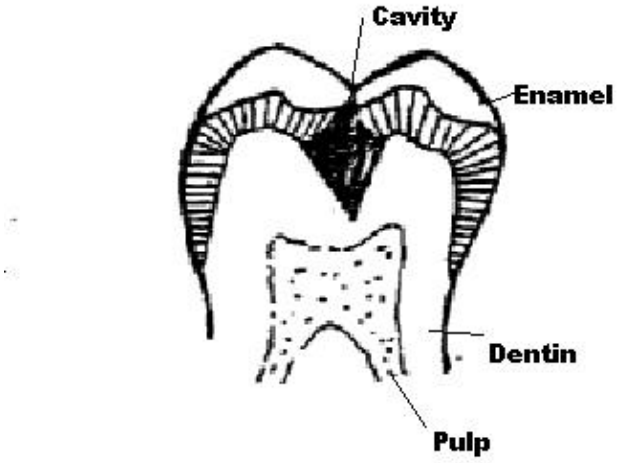


Figure 3 -- Tooth with Cavity

The student explores the tooth surface and detects the relatively softer area of the cavity by probing it with the pick. The haptic interface is used to orient the pick in 6 degrees-of-freedom, and to present the pick tip forces as the tooth is probed. We have experimented with surface spring and friction forces to simulate the feel of enamel.

The user selects the drill to remove the decayed material, ensuring that the walls of the resulting hole are smooth, and that there is a reverse draft angle for retaining amalgam. The PHANTOM is used to position the tool tip and remove the voxels from the tooth model that are intersected by the drill head.

The student selects the carrier, and fills the cavity preparation with amalgam. Here the PHANTOM is used to position the tool tip to add voxels to the model when the stylus button is pressed. Amalgam is a constant volume, highly viscous fluid, which sets (becomes as hard as enamel) approximately 15 minutes after application. The user pushes the amalgam into the cavity preparation until the amalgam has filled it to overflowing. The user selects one of the carvers to contour the amalgam to match the original contour of the tooth.

Discussion

A dentist depends strongly on touch feedback from the tool tip for diagnosis and positional guidance during procedures. The tooth and tool tips have to be presented in sufficient detail to render the fissures and other surface features of the tooth. To date, we have partially implemented the VRDTS prototype. We have rendered models of juvenile and adult molars, with single (enamel) material properties. Further work is needed to implement the multiple material properties of dentin, pulp and cavity. The teeth obtained from the public domain on the Internet do not provide sufficient detail to adequately render surface features. We may need to develop our own higher resolution tooth models to achieve this.

In simulating the 4 dental tools, we have found that the Pick currently comes closest to real time feel and behavior. The interaction of the Pick tip with tooth surfaces is quite realistic. The Drill needs further refinement in simulating cutting resistance, cutting (burr side as well as tip) behavior and accuracy (which is related to the resolution of the tooth model). In addition, drilling realism would benefit by providing haptic rendering of high frequency vibration. The Carrier and Carver tools also need improvements, primarily in the rendering of amalgam properties.

There are shortcomings with the current haptic interface device. There is a need for 6 degree-of-freedom (DOF) force-feedback to enable collision detection of instrument handles. In the future we also anticipate needing 6 DOF force feedback to simulate the torque forces of tooth extraction. However, the PHANTOM haptic interface's current 3 DOF force-feedback capabilities seem to be more than adequate for cavity preparation training.

Finally, there are considerations to address in the setup and configuration of the workspace. We plan to explore visual/haptic collocation, using a Reachin Display [3]. We believe that students will benefit from the added realism of the simulation. We have also found that dentists have a very refined sense of hand position and tool orientation, necessitating accurate initial positioning of the PHANTOM arm and stylus. There is also a need for careful positioning of the PHANTOM in relation to the user's position and arm angle.

Conclusions

In general, our demonstrations show that the Phantom is capable of providing the level of haptic fidelity needed to equal or better the plastic teeth currently used. The real time performance of tool-to-tooth interaction is quite good on our dual Pentium system. Although our development is not complete, and our classroom tests have not yet been run, we feel that the VRDTS system promises to be an excellent alternative to current methods of dental training.

References

[1] Michael Goetz, University of East Anglia, ▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫

[2] Richard S. Schwartz, DDS, Fundamentals of Operative Dentistry, Quintessence Publishing, 1996

[3] Reachin Technology,▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫

Haptic Interaction with Geoscientific Data

Walter A. Aviles
John F. Ranta
Teneo Computing

Background

Developing models of the subsurface environment is a complex and challenging endeavor. Geoscientists and data interpreters must utilize data in the time and spatial domains gathered from a variety of sensors. Typically, this sensor-derived information is noisy and its interpretation requires careful analysis and modeling. This data must then be correlated and interpreted into spatially oriented models in order to for it to be utilized [1]. To further complicate this task, the repercussions of interpretation accuracy in commercial endeavors such as oil exploration are extremely high and often have effects measured in the millions of dollars.

The interpretation of geoscientific data includes five general phases [2]: 1) data preview; 2) initial interpretation; 3) verification of interpretation; 4) correction of the interpretation based on new data and/or further analysis; and 5) the communication of results to others. Historically, all five phases of this inherently multi-dimensional problem have been tackled using tools – non-computer-aided and computer-aided -- that operate on two-dimensional slices of the data set. More recently, 3D surface and volumetric visualization techniques have been applied to aid primarily in steps 1, 3 and 5 of the interpretation process [3].

Adding Haptics -- voxelNotepad

Adding haptic interaction to existing real-time 3D visualization techniques promises to extend the ability of geoscientists to operate in three dimensions during the interpretation process. Haptic interaction allows: 1) the direct and unambiguous selection of individual entities (e.g., voxels) for evaluation and modification; 2) the use of touch feedback to confirm and increase the accuracy of modifications of the data in three dimensions and 3) the use of an additional sensory channel or modality (i.e., touch) for purposes of understanding the data.

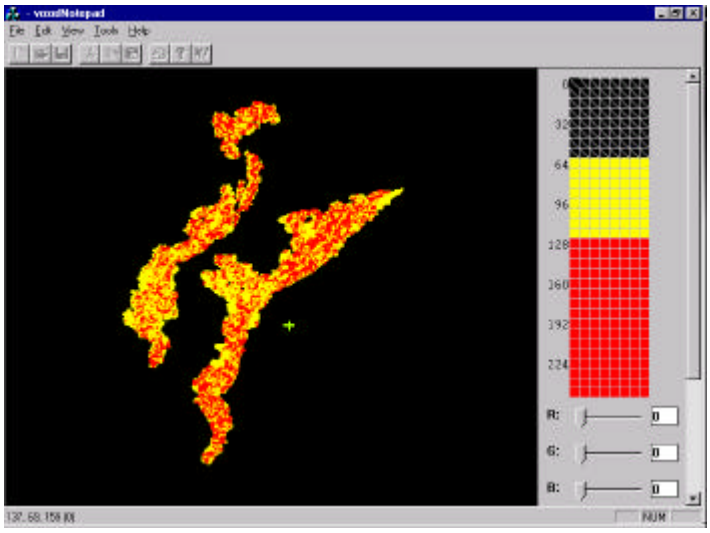


Figure 1 – Geophysical Data and Color/Opacity Map in voxelNotepad

To this end, we are developing a utility known as voxelNotepad™ (Figure 1). As the name suggests, this utility is aimed at providing quick and intuitive interaction with volumetric data. Developed in collaboration with practicing oil company geoscientists and interpreters, voxelNotepad currently provides interactive 3D stereoscopic visualization and haptically-enhanced selection, evaluation and modification of volumetric data on a WindowsNT™ computer workstation outfitted with a PHANTOM™ haptic interface device.

The main features of voxelNotepad are:

- ***Ability to import and export volumetric geoscientific data.*** The utility is designed to work with existing applications and currently supports voxelGeo and Landmark volumetric data formats.

- ***Ability to interactively visualize volumetric data in stereo.*** Frame sequential stereo with controls for camera separation and horizontal parallax is provided. As with most volume visualization tools, data translation, data rotation, data scaling and the mapping of the data to color and opacity is supported.

- ***Ability to interactively feel the volumetric data with a haptic interface device.*** A variety of forces have been explored for volumetric data evaluation purposes. Currently, voxelNotepad provides a viscosity-like force. The magnitude and existence of this force can be mapped to the data values of the volumetric data in a manner analogous to that of color/opacity mapping for visual rendering. Subsequent sections of this paper describe this feedback in more detail.

- ***Ability to directly pick individual voxels.*** A "jack" cursor is driven in 3-space with the haptic device. Coupled with force-feedback, it provides the user with the ability to directly and unambiguously pick individual voxels. This seemingly simple feature has profound repercussions for interacting with the volumetric data. The geoscientist can now precisely examine and change the fine structure (i.e., down to the data values of individual voxels) of the data set while visualizing and working in 3D. Picking a voxel is as simple as reaching out and touching it. Data values are displayed as an individual voxel is "touched". No 2D to 3D transformations on the part of the user are required. The selected voxel may be used as a seed for region modification algorithms or it may be interactively modified as discussed below.

- ***Ability to directly and interactively modify data values.*** Geophysical models are updated on a continuing basis. Oftentimes, these updates require fine-grained changes to the volumetric data. Heretofore, these changes were made in 2D slices that were then composited back into a 3D volume. This process is tedious and prone to error. The voxelNotepad utility allows the data values to be changed directly using a "3D painting" metaphor. The user chooses the value to modify the volumetric data from a palette. This palette displays both the available data values and their current color/opacity mapping. The user then "paints" the volume (i.e., changes the existing data values with the desired data value) by holding down the haptic interface device's stylus button while moving through the data. In order to aid this process, the user feels as well as sees the data values of the voxels to be changed. An unlimited undo/redo capability is provided.

Force-Feedback

Types of Force-Feedback

In developing the voxelNotepad utility, we explored several forms and uses for force-feedback and have currently settled on two basic types of force-feedback. The user may chose either of these or no force-feedback for volume data evaluation and modification operations.

3D Grid Force. The first form of force-feedback is a simple 3D grid. The purpose of this force is to allow the user to very precisely position the cursor in the volume data. The grid force includes the concept of a "dead-band". This tends to avoid the situation of being "flung" out of the detents (i.e., energy wells)

without resorting to drag/viscosity forces that may decrease the perceived precision of the detents and provides extremely precise positioning (0.5 voxel capability independent of haptic resolution).

Viscosity Force. The second form of force feedback is a viscosity-like force that varies with the haptic "density" assigned to a particular data value . This density value is assigned using a map similar to that used for data to color/opacity assignments. The user can set the haptic density value for individual data values. Like visual attribute mapping, there is also a concept of haptic opacity/transparency. Density settings may be overridden and the data made completely haptically transparent by utilizing this control. The visual color/opacity map may also be used to automatically derive the haptic density map.

The feedback force is implemented as a simple spring system where data value density mapping determines how much the spring "anchor" position can move towards the phantom position. This type of force was found by our geoscience colleagues to provide sufficient haptic differentiation between data. This approach has the advantage over a classical viscosity force (i.e., a force proportional to but in opposition to velocity) in that accurate velocity measurements (which require an integration of the haptic interface device position and hence are prone to noise) are not required.

Force-Feedback During Modification

Use of the viscosity force (which varies with the value of the voxel data) during data value modification operations required several refinements.

The first issue addressed was what to do about feeling newly changed data values ***during*** modification (i.e., while the haptic stylus is depressed in modification mode). For example, if haptic density increases with data value and these values are increased during modification, immediately feeling the changed data impedes the motion of the haptic interaction device. This situation is similar to that which would occur if a very quickly setting material were being extruded. The opposite situation also presents problems. Again haptic density increases with data value but the values in a region of data are now decreased. If the changes are NOT immediately felt, then it is difficult to reach the interior voxels of the region. This would be analogous to the situation in which physical material is being removed but confirmation of its removal is somehow delayed in time. In order to meet these conflicting requirements, the user is given control over when changes to data values are ***felt***. Changes to the data values, however, ***occur*** and are ***seen*** immediately.

Three choices are provided: 1) changes are felt as soon as the currently modified voxel is vacated (i.e., immediate feel); 2) changes are felt after a certain number of voxels have been changed during the modification operation; and 3) are felt after the modification operation is finished (i.e., the stylus button is released). These three choices provide users which sufficient control for efficient modification of the volume. The conditions under which any of these settings turned out to be an individual preference so their choice is left to the user.

The second issue concerns the strength of the viscosity force during modification operations. The same settings that allow successful haptic evaluation of data values tend to be too strong during modification. Voxels that exhibit high haptic density impede the movement of the haptic interface device and do not allow the user full freedom of control. Users, however, find that being able to feel the existing data values during modification operations is extremely valuable. This feedback aids in the determination of the extent of the region to be modified and helps to accurately start and end the modification operation. Consequently, users are given control over how much the usual (i.e., evaluation) viscosity feedback force is attenuated during modification.

Force-Feedback and Haptic Resolution

The PHANTOM haptic interface device is an absolute position input device. The most straightforward configuration for the device is to map its position such that its physical workspace encompasses the logical workspace. Situations arise, however, where the user may desire to change this mapping. Users, therefore, are given control over haptic resolution (i.e., the scaling between physical excursions of the haptic interface and distance traveled in voxel space). High-resolution settings are useful for extremely fine control of evaluation and modification operations. In this situation, the entire data set cannot be spanned during an

excursion of the haptic interface device but all portions of the data set can be reached by translating the volume within the haptic interface's workspace. Low-resolution settings, on the other hand, are useful for quick, less precise evaluation of modification of the volume.

It should be noted that the haptic feedback equations for all forces had to take this resolution into account in order to guarantee stability (i.e., stay within the stiffness constraints of the haptic interface device).

Conclusions

A prototype version of voxelNotepad has been completed and is in use by oil company geoscientists and interpreters. Although it is much too early to provide a definitive evaluation of the efficacy of haptic interaction in geoscience, some preliminary observations can be made. First and foremost, users of voxelNotepad in the geoscientific community have found it useful in the interpretation of geophysical data. With voxelNotepad they are able to visualize the volumetric data, as they can with their current tools, but voxelNotepad provides much faster performance for interaction. Furthermore, they are able to do something that they were previously unable to do – controlled, direct modification of volumetric data in 3D. Certain aspects of force-feedback have already proven essential. Specifically, feeling the data during modification operations tends to constrain and determine the extents of the modification operations. Early in the program development there was no such force feedback and users found it extremely difficult to make changes in 3 space without this feedback. Furthermore, viscosity feedback and the overall precision of the haptic interface device, when used at reasonable resolutions, have proven to be precise enough that the 3D-grid force is not often used in practice.

Quantifying the effects of force feedback in this task domain will clearly require further work and evaluation. In addition to evaluating these effects, however, we plan on extending voxelNotepad to 1) handle multi-attribute data sets; 2) model other entities used in geoscience and oil exploration such as well bores, faults and horizons and 3) extend its modification and selection capabilities to include the ability to "lock" the data values of selected regions of the volume.

References

[1] A. Brown. Interpretation of Three-Dimension Seismic Data, American Association of Petroleum Geologists, 1996.

[2] S. Lin, B. Loftin, T. Start, *Virtual Reality for Geosciences Visualization*, Proceedings of Asia Pacific Human Interaction, 1998.

[3] Paradigm Geophysical, http://www.cogniseis.com, 1999.

1999 PUG Attendees

Abbe Cohen	SensAble	acohen@sensable.com
Andrew Hally	SensAble	ahally@sensable.com
Arthur Gregory	UNC-CH	gregory@cs.unc.edu
Arthur Kirkpatrick	University of Oregon	ted@cs.uoregon.edu
Arthurine Breckenridge	Sandia National Laboratories	arbreck@sandia.gov
Bill McNeely	The Boeing Company	bill.mcneely@boeing.com
Brad Payne	SensAble	payne@sensable.com
Brandon Itkowitz	SensAble	bitkowitz@sensable.com
Charlie Lindahl	University of Texas@Arlington	lindahl@uta.edu
Chau Cao	UH VERI/ U. of Houston	cmc@metropolis.vetl.uh.edu
Chih-Hao Ho	MIT	chihhao@mit.edu
Chris Lee	University of Colorado	chris@chs.uchsc.edu
Dan Staples	SensAble	dstaples@sensable.com
David Perlsweig	IBM	dmperl@us.ibm.com
Debraj Basu Mallick	Univ. Iof Illinois at Chicago	debraj@virtual.me.uic.edu
Elaine Chen	SensAble	echen@sensable.com
Elaine Wong	Shell Oil Company	elaine@shellus.com
Florence Morache	WMD/MKT/BDT	florence_morache@ds-fr.com
Frances L. Van Scoy	WV EPSCoR/ West Virginia University	fvanscoy@wvu.edu
Fred Henle	Dartmouth College	henle@cs.dartmouth.edu
Frederic Vacher	WMD/MKT/BDT	frederic_vacher@ds-fr.com
Hiromi Kobayashi	VR Center Yokohama Inc.	vrc@ddd.co.jp
Jered Floyd	MIT	jered@mit.edu
John P. Wilson	Creare Inc.	jpw@creare.com
John Ranta	Teneo Computing	jranta@teneocomp.com
Josh Handley	SensAble	jhandley@sensable.com
Jung Kim	MIT	jungkim@mit.edu
Karl Reinig	University of Colorado	karl@chs.uchsc.edu
Kendall Mauldin	Sandia National Laboratories	kdmauld@sandia.gov
Kenneth Salisbury	MIT/Stanford	jks@robotics.stanford.edu
Lingling Zhang	University of Southern California	linglinz@usc.edu
Mandayam Srinivasan	MIT	srini@mit.edu
Marc Midura	SensAble	mmidura@sensable.com
Matthew Hutchins	CSIRO	Matthew.Hutchins@cmis.csiro.au
Michael Graffeo	Cornell University	mpg2@cornell.edu
Mike Lohse	SensAble	lohse@sensable.com
Nancy Amato	Texas A&M University	amato@cs.tamu.edu
Nikolai A. Grabowski	University of Delaware	grabowsk@eecis.udel.edu
Osman Burchan Bayazit	Texas A&M Univesity	burchanb@cs.tamu.edu
Philip Winston	SensAble	pwinston@sensable.com
Pierre Dupont	Boston University	pierre@bu.edu
Remis Balaniuk	INRIA - Rhone Alpes	Remis.Balaniuk@inrialpes.fr
Richard M.Y. Lee	3D Incorporated, Japan	lee@ddd.co.jp
Robert D. Howe	Harvard University	howe@deas.harvard.edu
Robert J. Kline-Schoder	Creare Inc.	rjk@creare.com
Roger Webster	Millersville University	webster@cs.millersv.edu
Russell Blue	GE Corparate R&D	blue@crd.ge.com
Seong-Rim	University Of Southern California	seongnam@scf.usc.edu
Shibin Yin	University of Texas@Arlington-VTL	shibiny@hotmail.com
Shingo Hirose	Mechanical Engineering Laboratory	hirose@mel.go.jp
Steve Wang	MIT Touch Lab	wangs@mit.edu
Steven Dow	GROK Lab - U. of Iowa	steven-dow@uiowa.edu
Steven E. Linthicum	GE Corparate R&D	linthicu@crd.ge.com
Tamaharu Yoshimizu	Nihon Binary	email@nihonbinary.co.jp
Thomas Debus	Boston University	tdebus@bu.edu
Thomas Massie	SensAble	thomas@sensable.com
Timothy Miller	Brown University	tsm@cs.brown.edu
Tom Anderson	Novint Technologies/ Sandia Nat'l Labs	tom@novint.com
Tomer Shalit	ReachIn Technologies	tomer@reachin.se
Torben Kling-Petersen	Mednet, Goteborg University	kling@mednet.gu.se