Haptic Rendering with the Toolhandle Haptic Interface

by

Craig B. Zilles

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degrees of

Master of Science

and

Bachelor of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1995

Author .....
Department of Mechanical Engineering
May 12, 1995

Certified by . .....
J. Kenneth Salisbury
Principle Research Scientist
Thesis Supervisor

Accepted by .....
Ain A. Sonin
Chairman, Departmental Committee on Graduate Students

Haptic Rendering with the Toolhandle Haptic Interface

by

Craig B. Zilles

Submitted to the Department of Mechanical Engineering
on May 12, 1995, in partial fulfillment of the
requirements for the degrees of
Master of Science
and
Bachelor of Science in Mechanical Engineering

Abstract

This thesis describes the design of the hardware and software for a haptic interface system. A haptic interface allows a human "observer" to explore *and* interact with a virtual environment using the sense of touch. Haptic interface systems include three main components: the haptic interface (usually an electro-mechanical system capable of exerting forces on a user), a model of the environment to be touched, and a rendering algorithm which unites the first two by generating the feedback forces based on the environment model. This thesis focuses on the first and third of these components: a haptic interface, the MIT-Toolhandle, and haptic rendering algorithms for simulating general real-world virtual environments.

The MIT-Toolhandle is a ground-based force-feedback device designed to allow subjects to use tools to interact with virtual environments. One of the difficulties of haptic interfaces is simulating both the human's kinesthetic and tactile system. Tool interactions are interesting because they passively satisfy the human's tactile sense. Since the user holds a physical tool the simulation can be reduced to simulating the interactions between a tool and the environment. The description of the MIT-Toolhandle is accompanied by a history of previous haptic interfaces, and some insight into the psychophysical issues involved in the synthesis of haptic feedback.

The rendering algorithms described fall into two major categories: those for coarse geometry and those for surface effects. We will examine vector field and god-object methods for representing geometry. The weaknesses of the vector field method are uncovered, and we explain how the god-object method handles these difficulties. Lastly, extensions to the god-object method for friction, texture, and surface smoothing will be described.

Thesis Supervisor: J. Kenneth Salisbury
Title: Principle Research Scientist

Acknowledgments

This thesis describes work conducted at Massachusetts Institute of Technology's Artificial Intelligence Lab. This work is supported in part by NAWCTSD contracts, N61339-93-C-0108, N61339-93-C-0083, N61339-94-C-0087 and ONR Grant N00014-93-1-1399.

I would like to thank Ken Salisbury for giving me a chance to do research in such a new and exciting field, and Thomas Massie for the development of the PHANToM on which part of my work is based. I am grateful for the assistance from everyone who contributed to the ideas on which my work is based, including Ken, Thomas, Nick Swarup, Brian Eberman, Dave Brock, Mandayam Srinivasan, and Walt Aviles. I appreciate the work Derek Schulte has done on the machine drawings of the MIT-Toolhandle and in developing tools to be used with the haptic interfaces. I would like to thank Akhil Madhani, Won "Jesse" Hong, Gunter Niemeyer, John Morrell, and Brian Anthony for making my stay at the A. I. Lab pleasurable and productive. I would like to congratulate and thank Patrick Winston and the other members of the A. I. Lab for creating an environment conducive to great works.

My personal sanity has been kept strong over the years by my friends at the Institvte and the brothers of Tau Epsilon Phi. I am continually impressed by the truly amazing people that M.I.T. attracts and am thankful for the contributions they have made to my life. My parents Steve and Connie Zilles and my brother Karl have my gratitude for helping shape the person I am. It was their influence and reinforcement which got me interested in science and engineering and urged me to excel. Finally, I would like to thank my fiancée Julie Hollenback for the generous love and support she has given me over the last four and a half years.

Contents

List of Figures

Chapter 1

Introduction

In recent times there has been an increasing interest in using computers to replicate situations which would be too costly or too dangerous to create otherwise. These so called *virtual environments* are expected to supply all of the sensory information which would be accessible to the user and allow the control the user would expect to have in such an environment. Much of the early effort in this field involved flight simulators which were used to train pilots without subjecting them to the danger of actual flight. With the exponential advances of computational hardware, we've seen a broadening of the applications which are seeking virtual reality solutions.

The availability of visual displays (computer monitors) and computer audio hardware has allowed an abundance of progress in the visual and audio aspects of virtual environment development. Although far from having a complete solution, the level of development in both fields is far beyond that of touch interfaces (haptic interfaces). A haptic interface is a device worn by a human which provides the touch sensations of interacting with virtual object. The device usually consists of an electro-mechanical linkage which can exert a controllable force on a human's hand. A number of reasons have delayed haptic interface research until the present.

1. a lack of understanding of how to communicate haptic effects
2. computational power and the need for real-time software
3. cost of hardware
4. safety issues

Only recently have researchers tried to quantify human haptic perception. Without a good understanding of the human's sensory abilities it is difficult to design systems to stimulate them.

Also, unlike the visual domain, haptic interfaces need to be run in real time; there is no haptic equivalent to a still frame of video. Neither can the computation be done ahead of time (like a ray-traced animation) due to the interactive nature of haptic displays. Computers with sufficient processing power only recently have become a commodity resource.

Robotic hardware is expensive, and cannot exploit the advantages of "economies of scale" in the way that monitors and stereo systems have. Advances in manufacturing have reduced the cost of actuators and sensors, but robotic hardware is still only common in research labs. Only recently

have commercial vendors of haptic interfaces appeared; before it was necessary to design and build your own haptic interface to do haptic rendering research.

Whenever a human subject is in contact with active mechanical system, safety should be a concern. Many haptic interfaces can produce moderate forces and have a sufficient range of motion that a human could be injured. The precautions required may have discouraged research in this area. The safety of the devices themselves has not improved, but modern computers are more reliable then their predecessors.

Without haptic interfaces a user of a virtual environment is really only an onlooker. Most applications of virtual environments require manipulation of or interaction with a virtual environment. In attempts to fulfill this need a number of companies offer "data gloves" which measure hand position and configuration. This allows manipulation of sorts, but in an unnatural manner; it does not draw on the wealth of experience people have with manipulating physical objects. In addition, there is a sense of immersion in a virtual environment that cannot be attained without touch interaction. In fact, often subjects are so immersed that they are startled when the simulation is turned off; "the feeling is similar to sitting down in a chair only to find that it has been pulled out from beneath you." [12]

1.1 A Brief History of Haptics

Some of the first applications for haptic interfaces were in the 1960's and 1970's. [1] A. Michael Noll recognized that haptic modalities of man-machine communication had advantages over graphic modalities for some applications. [16] He designed a Cartesian three degree-of-freedom haptic interface and wrote some simple haptic software to demonstrate the device. Noll's software simulated the 3D contact forces which a user could expect from a point interacting with simple frictionless objects. Noll expected that haptic interfaces could replace graphic interfaces for 3D visualization.

Around the same time, Project GROPE was started at U.N.C. Chapel Hill to create a system for better scientific visualization. [3] Argonne National Labs donated a 7-DOF (6 + grip) remote manipulator they no longer needed. Their first simulated world consisted of a table and a number of blocks which could be manipulated by the user; only forces, no torques, were provided as haptic feedback. At the time, this was the most sophisticated demonstration the computer hardware could support. They estimated that useful visualization tools would require 100 times as much computational power; the project was moth-balled until computers could provide the necessary speed.

In the late 80's computers had improved to the level required for continuing Project GROPE. The 7-DOF manipulator was used to simulate a molecular docking task. In this task the user tries to position a virtual molecule by following the force gradient until the molecules are interlocked in a stable configuration. Although a realistic calculation of this 7-DOF force field is computationally intensive, haptically, it can be applied to the user as simple attractions or repulsions. Project GROPE found that the force feedback increased the speed and performance of subjects attempting the task. This was an important step for the acceptance of haptics, but the task was too specialized to have provided research into a broad cross-section of haptic effects.

A more complete set of haptic effects was studied by Margaret Minsky using the *Sandpaper* system. [13, 14] The Sandpaper system consisted of a 2-DOF joystick with force feedback for simulating

[1] Margaret Minsky's PhD thesis [13] includes a more complete, yet terse list of the research of haptic interfaces.

texture. Multiple subjects arranged the textures by roughness in a consistent manner, suggesting that users were able to recognize the textures enough to discriminate and classify them. More generally, Minsky investigated making more complex haptic models from combining textbook objects: linear springs, dampers, and masses.

In the last couple of years the interest in haptic interfaces for virtual reality has picked up dramatically. There are far too many projects to mention all of them. The following represents a cross-section of the work being done.

In the early 90's, the Mechanical Engineering Laboratory in Japan started what could be the first work in simulating complex surface geometry. [9] They seem to have modeled reasonably simple objects, but they took a surface representation approach which is portable to higher complexity objects. The work covers both point-object interactions and object-object interactions.

At Northwestern University, J. Edward Colgate has been investigating passivity in the human-machine control loop and the hardware and software requirements for stiff virtual walls. [5] Much of this work involves single degree of freedom haptic interfaces. Single DOF interfaces avoid the problems associated with complex geometry and allow concentration on the individual haptic effects.

Grigore Burdia's research group has been developing a non-ground-based force feedback glove at Rutgers University. [4] Using pneumatic cylinders and LVDT's, the Rutgers Dextrous Master can provide forces against grasping objects. Although quite an improvement over traditional (passive) data gloves, the glove displays only grasp forces and cannot display net forces. Forces due to gravity or inertia can not be displayed, and the user's hand can not be stopped from passing through virtual objects.

At the University of British Columbia, a number of haptic interfaces have been built including a 6-DOF magnetically levitated joystick. Magnetic levitation permits a low-inertia and low friction design, but the workspace of the joystick will probably limit its utility as a mainstream haptic interface. S. E. Salcudean has been researching the emulation of stiff walls and surface effects (including friction). [17]

In 1993, Thomas Massie and Ken Salisbury developed the PHANToM, a 3-DOF finger controller with a moderate workspace and good force bandwidth. Our group at the A.I. Lab has used the PHANToM to approach issues including surface effects, arbitrary shaped objects, and the complex impedance blending methods which sophisticated surgical simulations and other training aides will require. [11, 18]

1.2 Thesis Overview

This thesis endeavors to illuminate two areas of active research in haptic displays: haptic interface design and haptic rendering.

Since haptic display is really the combination of a haptic interface with a human's perception system, it is important to understand the haptic capabilities of the human. A qualitative discussion of what humans can perceive is provided in chapter 2 to outline the design requirements for haptic interface design. A more quantitative approach is taken in the description of the MIT-Toolhandle haptic interface. The kinematics of the MIT-Toolhandle are described in detail.

Haptic rendering refers to the software for the haptic interface which computes the feedback force based on the history of motions. The largest difficulty in rendering is the inherent compliance of

the haptic interface, and the uncertainty of its virtual location which this causes. In chapter 3, two methods of haptic rendering will be considered: the vector field method, and the constraint-based god-object method. The limitations of the vector field method will be exposed, and an explanation of how the god-object method avoids these problems will follow.

Haptic rendering deals most directly with object geometry, but is also the basis for surface effects like friction and texture. The constraint-based god-object method will provide a location in the virtual environment where the user is currently touching. With this location in hand implementing surface effects is greatly simplified; for this reason the surface effects proposed in chapter 4 will be described as extensions to the god-object algorithm.

Chapter 2

Haptic Interfaces and the Human Haptic system

2.1 Human Factors

Synthesis of high performance haptic interface systems requires some understanding of the capabilities of the human haptic system. The following is an over-simplification of the issues, but contains some good guidelines and rules of thumb. For a more rigorous treatment of psychophysics, we recommend reading the literature in that field. [6, 20]

2.1.1 Touch modalities

The major goal of a haptic interface is to provide the sensations a user would experience if he/she were to touch a virtual environment. This problem has a number of issues, the most notable of which involve how humans sense.

Humans have two main systems for sensing touch: tactile and kinesthetic. The tactile system consists of nerve endings in the skin which respond to pressure, warmth, cold, pain, vibration, and itch. The abundance of these mechano-receptors in the fingers and hands allow us to accurately discriminate between different stimuli. The tactile system allows us to sense local geometry, rough texture, and thermal properties just from static contact. When we slide our fingers over surfaces we can learn much about the surface's fine texture, friction characteristics, and some information about the local stiffness and viscosity of the surface. Attempts to stimulate this modality have proliferated in designs of tactile display arrays. These arrays usually consist of closely arranged groups of "pins" which can be individually actuated. [19]

The kinesthetic system refers to the collection of receptors in the muscles, tendons, and joints which allow perception of the motion and forces upon ones limbs. The brain interprets the signals from these receptors to estimate the position and contact force at the end of the limb. Through these forces the presence of a virtual object is most directly perceived. Although statically very little can be sensed from a "crude" force, the spatial distribution of force vectors can provide information including geometry, surface properties, and dynamic properties. Force feedback joysticks and exoskeletons stimulate this modality by providing "crude" forces to parts of the body.

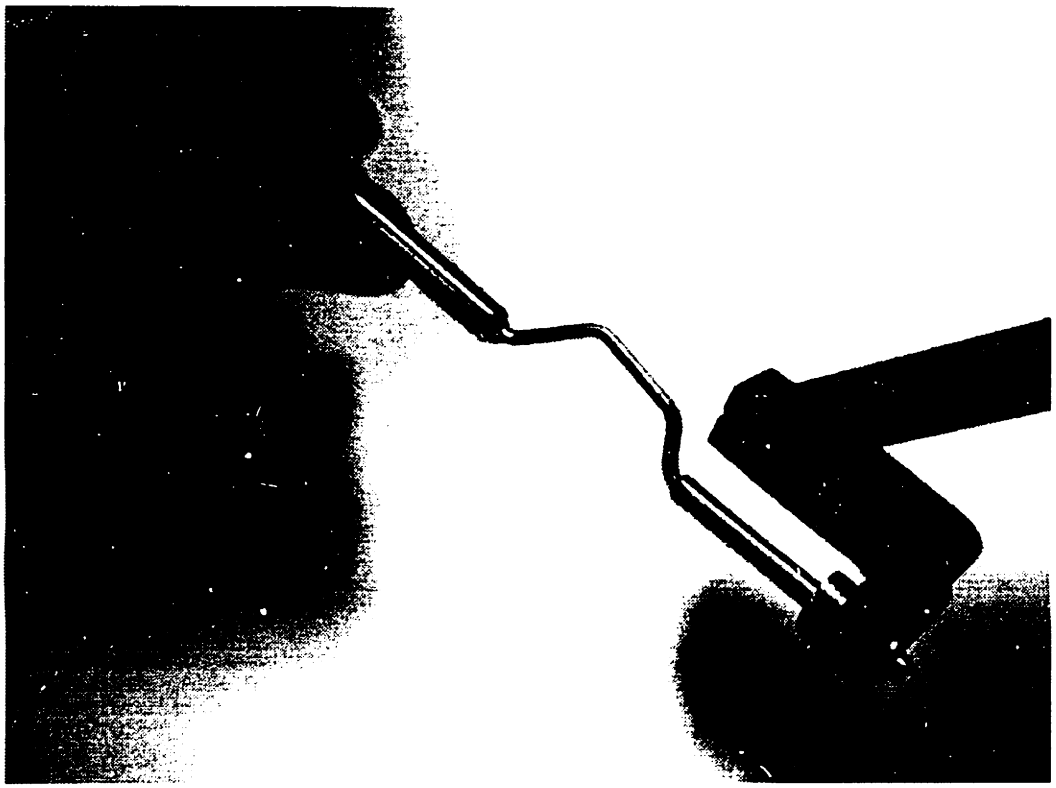

Figure 2-1: A dentist's tool inherently presents the tactile "feel" of a real tool

These force feedback interfaces come in two varieties: ground-based and body-based. Ground-based devices, like joysticks, are physically attached to something stable which provides the reaction forces necessary to apply a net force to a user. Body-based (typically gloves and exoskeletons) use a connection point on another part of the wearer to provide the force. Body based devices can be portable because they require no attachment to the world, but this portability can be a drawback. A portable exoskeleton will not be able to stop the wearer from passing through a virtual wall. Similarly, an exoskeletal glove could provide the forces required to simulate grasping an object, but could not simulate the weight of the object.

We'd like to create devices which would stimulate both sensory modalities, but modern tactile stimulation arrays can't provide the fidelity required and are too large and bulky to incorporate into a high performance haptic interface. Because tactile sense is an important part of haptic interaction, we followed another promising approach, that of the tool-handle interface. A tool-handle interface provides the tactile feeling of holding a tool because the user holds an actual tool. The problem is then reduced to correctly simulating the interaction of the tool with the environment. Since a tool will often interact as a small patch or even a single point we have greatly reduced the complexity of the simulation.

The tool-handle approach is not applicable to all haptic interactions, but tools are used in many tasks which require haptic proficiency. A surgeon's scalpel, an artist's paint brush, a handy man's screwdriver, a draftsman's pencil, and a fencer's foil are a few examples where a tool-handle system could easily provide the correct sensations. This thesis focuses in part on the design of a ground-based tool-handle haptic interface.

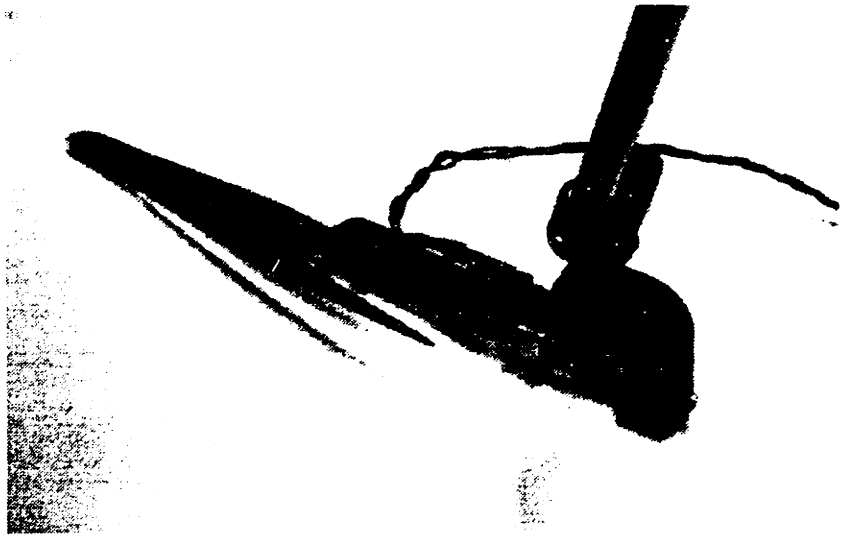

Figure 2-2: A pair of tweezers for picking up virtual objects

2.1.2 Human Perception

We will never be able to synthesize touch sensations with the quality that the real world has. Luckily, the human haptic system, like the human visual system, has its limits. Computer graphics relies on the fact that the humans have a flicker fusion frequency around 30-60 Hz; discrete images presented above this frequency can not be discriminated from a continuous image. Likewise, if we know the limits of the human haptic system we can design systems to take advantage of them.

Humans have poor discrimination of position. The location of the hands are computed in the brain from the joint angles in the arm and wrist. "The just noticeable difference (JND) is about 2.5 degrees for finger joints, 2 degrees for wrist and elbow and about 0.8 degrees for the shoulder." [20] These uncertainties mean that throughout an arm's range of motion a human can really only discriminate position to within about an inch without an external reference.

Given a reference to compare against the human does much better. A pinch grasp to gauge length has a JND around 5-10 percent of the reference length. [6] It will be important to have the virtual objects correctly registered with respect to each other, especially if the user has multiple fingers in the virtual environment.

Humans are competent, but not astonishing, at comparing force magnitudes; the JND's for force magnitude are around 5-15 percent of the reference force. Characterization of compliance is still rather lacking. JND's for compliance can range from 5-99 percent. [6] "For deformable objects with rigid surfaces held in a pinch grasp, the JND for compliance is about 5-15 percent when the displacement range is fixed, increases to 22 percent when it is varied randomly, and can be as high as 99 percent when cues arising out of mechanical work done are eliminated." [23] The mechanism which humans use to sense compliance is not well understood.

One study found that the minimum stiffness which humans can't distinguish from a solid surface was around 25N/mm. [24] We've found that when subjects use less than a few pounds (about 10N) of force they are willing to "believe" that surfaces which are less stiff (by an order of magnitude) are solid. This leads one to believe that this feeling of solidity might involve the sensing of displacement more than the sensing of a force-displacement relation.

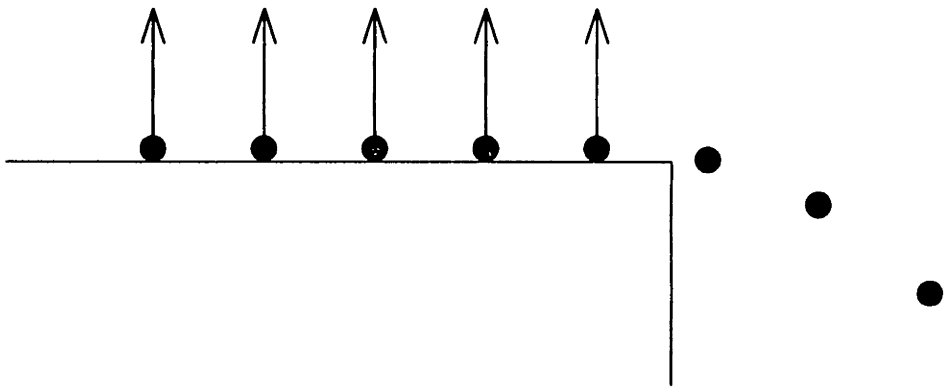

Figure 2-3: A finger sliding across a table receives a discontinuity in force when the end of the table is reached

Humans are extremely sensitive to vibration and discontinuities of force. One type of mechano-receptors, the Pacinian Corpuscle, in the finger tips (and elsewhere in the body) acts as an accelerometer and measures the high frequency component of the motion of the hand. Due to their presence humans can "distinguish vibration sequences of up to 1 kHz through the tactile sense." [6] It is important that our haptic system not have spurious vibrations below this frequency or they will eclipse the signal trying to be transmitted. A discrete signal (like our feedback force) is often perceived by a human as the combination of a continuous signal and noise with a frequency content near the sampling rate. This is one of the reasons that our high performance haptic systems run their servo loops above 1kHz.

The cognitive interpretation of forces appears not to be taking place at such a high frequency. "The bandwidth of the kinesthetic sensing system has been estimated to be 20-30Hz." [6] This means that the positions of objects in the virtual environment might not have to be updated as often as the force on the haptic interface as long as it does not induce vibration.

Some discontinuities in force which are desired because they accurately model the interaction with the environment. For example, when a finger is sliding across a table and it reaches the end; suddenly the contact force disappears (Figure 2-3). These cues are important, and without tactile arrays they are the only method to simulate many contact situations. Without the use of our tactile receptors our sense of sharpness and texture come from the variation of the force direction and magnitude.

2.1.3 Vision vs. Haptics: The Battle in the Mind

Haptics are important for manipulation tasks, but vision is still the simplest way to comprehend complex scene geometry. Even high performance haptic interfaces only give feedback to a couple of finger pads; no modern system can stimulate the whole hand. Using only a single finger without the aid of mechano-receptors it is very difficult to perceive a scene of moderate complexity. Once the user has a mental picture of the virtual environment the visual display is not so important, but the visual display is the best way to get the mental picture.

Graphics can play more than just a supporting role to haptics. Visual feedback can overshadow[1]

[1] *overshadowing*: In classical conditioning (psychology) if more than one stimuli are provided during a conditioning, usually the most noticeable *overshadows* the others and only that one develops the conditioned response. Our meaning

haptic feedback. Experiments have shown that if visual feedback is different than haptic feedback, subjects tend to rely on the visual feedback. The perceived stiffness of a button can be changed by only varying the visual response. [21]

Without force cues the kinesthetic system can't accurately position the limbs, so the brain has come to rely on the vision system for fine positioning. Due to this lack of kinesthetic perception of global location, humans are not sensitive to small discrepancies in the registration of the haptic and visual location of objects. In addition, subjects of our two handed simulations tend to use visuals to coordinate one hand with the other rather than using their kinesthetic sense. The uncertainty of the relative location of the two hands seems to be equal to the uncertainty of one hand to a global location.

Humans are capable of making simple transformations between the haptic display and the graphics display. With a little experience humans are able to internalize the map of motion in one space to the motion in another space. The most prevalent example of this is the computer mouse which maps a horizontal physical motion to a vertical perceived motion. This requires a mental translation and a 90 degree rotation. It has been our experience that more complex transformations are extremely difficult. Although benefits will be seen if the visual display is collocated with the haptic display (with mirrors or a head mounted display), this human ability allows standard computer monitors to be used for visual feedback.

2.1.4 The Haptically Challenged

Similar to the sight and hearing impaired there seems to be a group of people who aren't responsive to some haptic stimuli. In demonstrations of our apparatus we have come across a few people who cannot seem to comprehend its actions. One subject claimed that the device was pushing on her when the environment was completely static; she could not recognize that she was pushing against object and it was merely resisting her motion. These people seem otherwise normal and have no trouble doing normal day to day manipulation tasks like eating, writing, getting dressed, etc.

I do not believe that the cause of this inability is known yet, but since our devices only affect a subset of touch senses the symptoms are clear. Normal subjects can recognize basic shapes (spheres and cubes) using only haptic feedback. These haptically challenged subjects seem not to be able to interpret a spatial distribution forces as a shape.

2.1.5 Design Issues for Ground-based Haptic Interfaces

We can categorize our knowledge of human perceptions into design criteria for designing haptic interfaces. [11] We'd like to be able to simulate the spectrum of forces between static contact and instantaneous impact. We'll see that a large force range, a high force bandwidth, and a high maximum stiffness will be needed to attain those goals.

A static contact will simply yield a constant force, but simulating the presence of objects requires more than static contacts. To make virtual objects perceptible it is important that they feel very different than free space; this implies that the maximum force the device can provide has to be much larger than the force needed to move the haptic interface through free space (the friction force).

lies between the technical and common usages of the word.

Our experience has found that subjects compare the local impedances in different directions to determine the presence and shape of an object. It is important for the force sources not to saturate in the device's operating region. Once saturation has occured, the impedance in the direction of the surface's normal is the same as those tangential to the surface. Lastly, to make the object feel solid the stiffness of the device should be as high as possible.

True impact forces contain an infinite frequency spectrum, but the human is only sensitive to frequencies below 1 kHz. To correctly replicate impacts the device should be capable of producing forces on the user with frequency components in this range. Such a high force bandwidth requires the servo rate to be above 1 kHz and the haptic interface to be structurally stiff.

To have these characteristics, the device must be low in friction and inertia and have low backlash, good sensor resolution, and high mechanical stiffness. Force range, bandwidth, and stiffness are highly coupled in their dependencies on the device and on the servo rate of the simulation.

2.1.6 Control methods

Some researchers have postulated that "in order to achieve such high performance without mechanical instabilities, robust and adaptive closed loop control of the devices is necessary." [6] We have not found this to be the case. Our devices do not sense force, are open loop (they use a human to close the loop), and utilize simple control laws (Hooke's Law (F=kx) or single sided PD control). We have been able to create reasonably sophisticated and high performance haptic environments.

In addition, the problems which need to be addressed for haptic interfaces are different than those for control. The most notable distinction between haptics and robotics it that the mechanism's trajectory cannot be known ahead of time; we can make no predictions about the current acceleration. Methods like inertia compensation are not useful. Another common robotics technique is to compensate for friction by adding extra torque at moving joints, but to implement this correctly really requires knowledge of the trajectory. If we assume the user will continue moving in the same direction we could add a small force in the direction of motion to counter friction; really we are only exchanging friction for inertia. These control limitations put even more pressure on the mechanical designer to remove inertia and friction from the design.

Some of the literature in haptic interfaces concentrates on the development of control techniques to increase the stiffness of walls. [5, 17] Although we've found that high haptic stiffness is not necessary for the perception of solid objects, the research in this field seems to be orthogonal to research into rendering methods. When both fields become mature they should be easy to integrate.

2.2 MIT-Toolhandle

The MIT-Toolhandle is an extension of the PHANToM [11] haptic interface using the tool handle paradigm (described in Section 2.1.1). The MIT-Toolhandle uses the same basic kinematics as the PHANToM but has a larger workspace and is capable of larger forces.

2.2.1 The PHANToM

The PHANToM (shown in Figure 2-4) is a finger controller with 3 active degrees of freedom (DOF's) and a 3 DOF passive gimbal. The device can produce a force in any direction on the user's finger, but

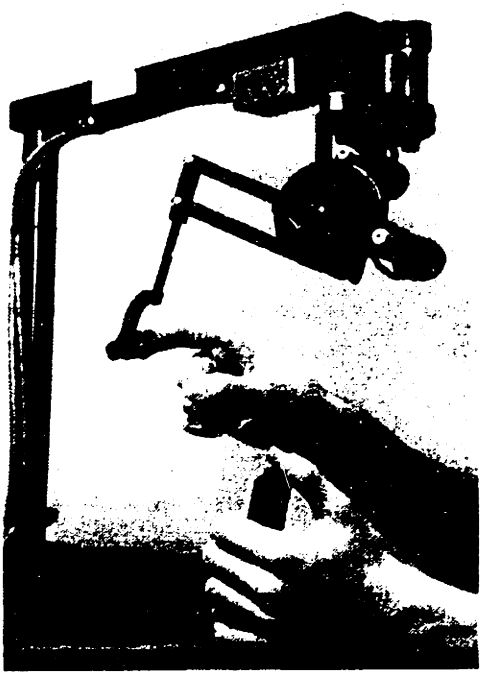

Figure 2-4: The PHANToM haptic interface – the kinematic basis for the MIT-Toolhandle (photo courtesy of Thomas Massie and Thomas Wiegand)

the finger can always freely rotate. The PHANToM haptic interface permits users to feel the forces that arise from point interactions with simulated objects. The point interaction paradigm greatly simplifies both device and algorithm development while permitting bandwidth and force fidelity that enable a surprisingly rich range of interactions.

The PHANToM is powered by three motors a and sensed by three shaft encoders. Cable transmissions provide a back-drivable "gear" reduction without backlash. Since stiff transmissions can be made from cables, high force bandwidth is attainable. Two of the motors are positioned such that they passively balance the mechanism; no gravity compensation needs to be done in software.

The workspace of the PHANToM is about 12cm x 12cm x 24cm (5 x 5 x 10 inches). The PHANToM was designed to allow the user to keep their fore-arm stationary. The major pivot is placed in line with the user's wrist.

2.2.2 MIT-Toolhandle Design Criterion

The requirements for a generic tool-handle interface are different than the capabilities of the PHANToM. Many tools (knives, paint brushes, screwdrivers) aren't used with the fore-arm braced against another object. These tasks are usually difficult to accommodate with the PHANToM because of its small workspace. The MIT-Toolhandle was designed to allow wrist and elbow motion; the workspace extends to almost wherever a stationary standing or seated user can reach.

By allowing motion of a greater part of the arm we expect subjects to use larger muscle groups. Larger muscle groups requires a larger maximum force from the device, but the user should be more tolerant of inertia and friction. The quality of the device will be partially masked by the inertia of the user's own arm.

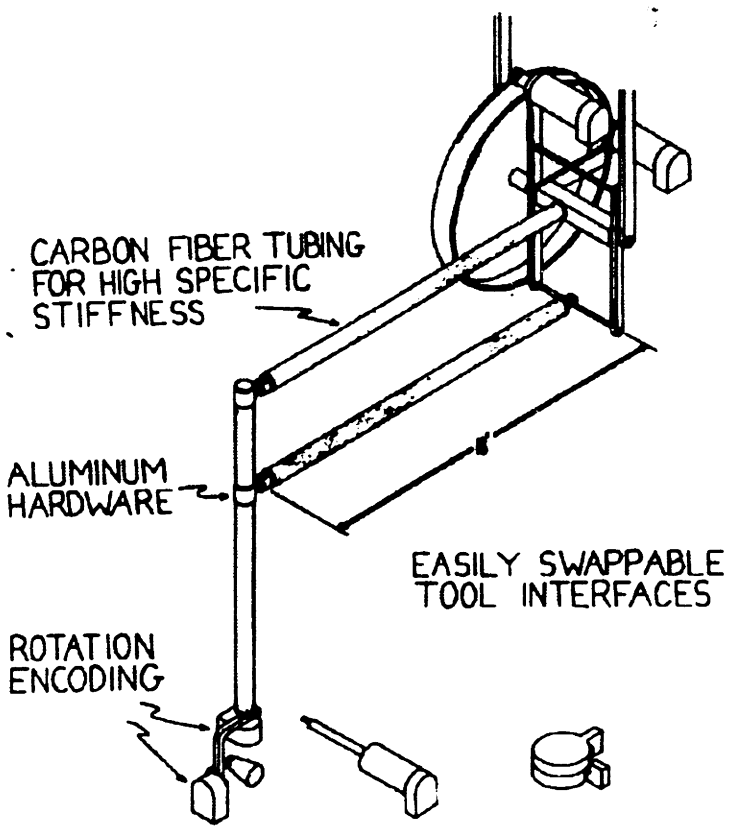

Figure 2-5: A sketch of the MIT-Toolhandle

Along with a large force range, high force bandwidth, and high stiffness it was desired that the MIT-toolhandle would be easy to use and easy to manufacture. It should be able to accommodate a variety of tools, and the tools should be easy to swap with little or no adjustment. To minimize cost, parts should have simple geometry and require minimal material removal.

High quality DC motors were used to attain a large force range. The maximum torque from the motors is one hundred times larger than the friction torque. When free cable lengths are minimized cable transmissions are very efficient transmissions; the most significant energy loss is due to the hysteresis from bending and stretching the cables. Ball bearings were used at every joint to keep friction low.

To keep the structure's natural frequency as high as possible,[2] carbon fiber tubing was employed for the link elements. Carbon fiber composites can have specific stiffnesses above those of most metals. Machined aluminum end caps are attached to the tubes with a special epoxy. This simplifies machining because only the end caps (not the whole link) need to be machined.

Encoders with higher resolution than those used by the PHANToM were employed because the servo stiffness of the PHANToM is currently limited by its encoder resolution. When the gain of the PHANToM is set too high it limit cycles between encoder ticks. In addition, higher position resolution will provide a better velocity estimate.

Inherent damping might be a boon to the stability of the device and allow a higher servo stiff-

[2]It is my opinion that the structural stiffness is currently limiting the servo stiffness of the device. When the gain is set too high a natural mode of the device is excited and the device begins to vibrate. Future revisions should probably replace the injection molded plastic parts with aluminum, and use larger diameter carbon fiber tubing

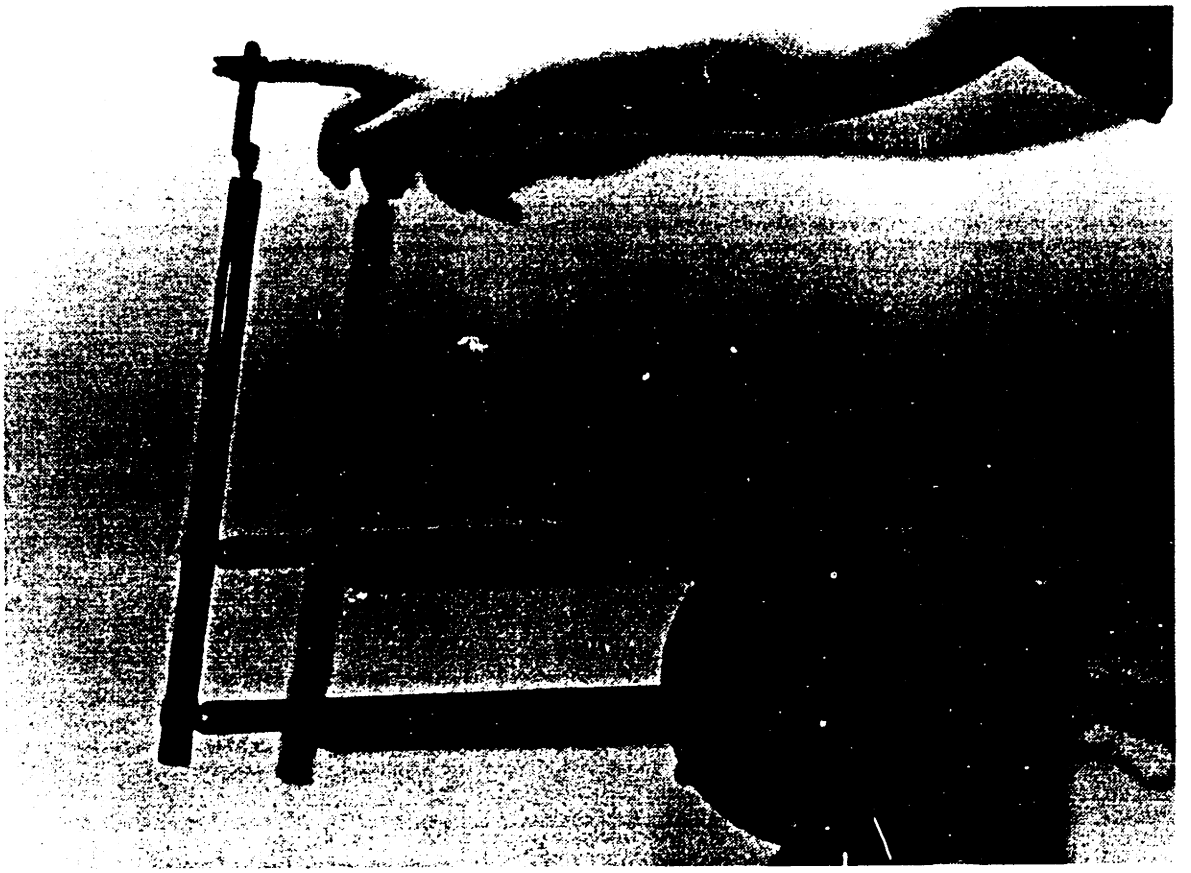

Figure 2-6: The MIT-Toolhandle: a haptic interface for the workspace of the human arm.

ness [5], but could have disastrous effects on the force bandwidth. To overcome the physical damping, negative damping could be supplied in software. As was pointed out in Section 2.1.6 we can really only trade viscosity for inertia, and either of these will reduce the force bandwidth of the device.

The useful work volume of the device is about 80cm x 40cm x 40cm (30in x 15in x 15in). The device can momentarily put out 20N (4lbf) of force at the endpoint. The encoder resolution is such that the space descritization is about .00075inches/encoder tick near its home position. At the edges of the useful work space, the maximum displayable stiffness is 4N/cm (2.5lb/in).

2.2.3 MIT-Toolhandle Kinematics

The kinematics of the MIT-Toolhandle differ slightly from that of the PHANToM. The pivots where the horizontal links meet the vertical link are not in line with the first gimbal axis.

The kinematics are presented using Denavit-Hartenberg notation.[3] [1] The Denavit-Hartenberg notation provides a sequence of transformations using a minimum of parameters to completely describe the kinematics. Six 4x4 matrices are used to transform both orientation and location.

The transformation between one frame and the next is specified by 4 parameters:

a_i the length of the common normal (the distance between the Z_{i-1} and Z_i axes measured along the X_i axis)

[3] We stray from the Denavit-Hartenberg notation for the last transformation because it is common for the final rotation to be sensor-less. This structure of transformations allows A_5^0 and A_6^0 to be very similar.

d_i the distance from the X_{i-1} and X_i axes measured along the the Z_i

α_i the angle between the Z_{i-1} and Z_i axes measured about the X_i axis

θ_i the angle between the X_{i-1} and X_i axes measured about the Z_i axis

The transformation matrix from one frame to the next can be found by substituting these parameters into equation 2.1.

$$A_i^{i-1} = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

The Denavit-Hartenberg parameters for the MIT-Toolhandle are as follows:

Frame Number	a	d	α	θ
1	0	0	$\pi/2$	θ_1
2	$L1$	0	0	θ_2
3	$L2$	0	$-\pi/2$	θ_3
4	0	$L3$	$\pi/2$	θ_4
5	0	0	$\pi/2$	θ_5

where L_1 is 43.8cm (17.25in), L_1 is 1.9cm (.75in), and L_3 is 45.7cm (18in). These parameters lead us to the following frame transformations. Figure 2-7 shows the locations and orientations of the intermediate frames.

$$A_1^0 = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$A_2^1 = \begin{bmatrix} cos\theta_2 & sin\theta_2 & 0 & L1 * cos\theta_2 \\ sin\theta_2 & -cos\theta_2 & 0 & L1 * sin\theta_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$A_3^2 = \begin{bmatrix} cos\theta_3 & 0 & sin\theta_3 & L2 * cos\theta_3 \\ sin\theta_3 & 0 & -cos\theta_3 & L2 * sin\theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$A_4^3 = \begin{bmatrix} cos\theta_4 & 0 & sin\theta_4 & 0 \\ sin\theta_4 & 0 & -cos\theta_4 & 0 \\ 0 & 1 & 0 & L3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

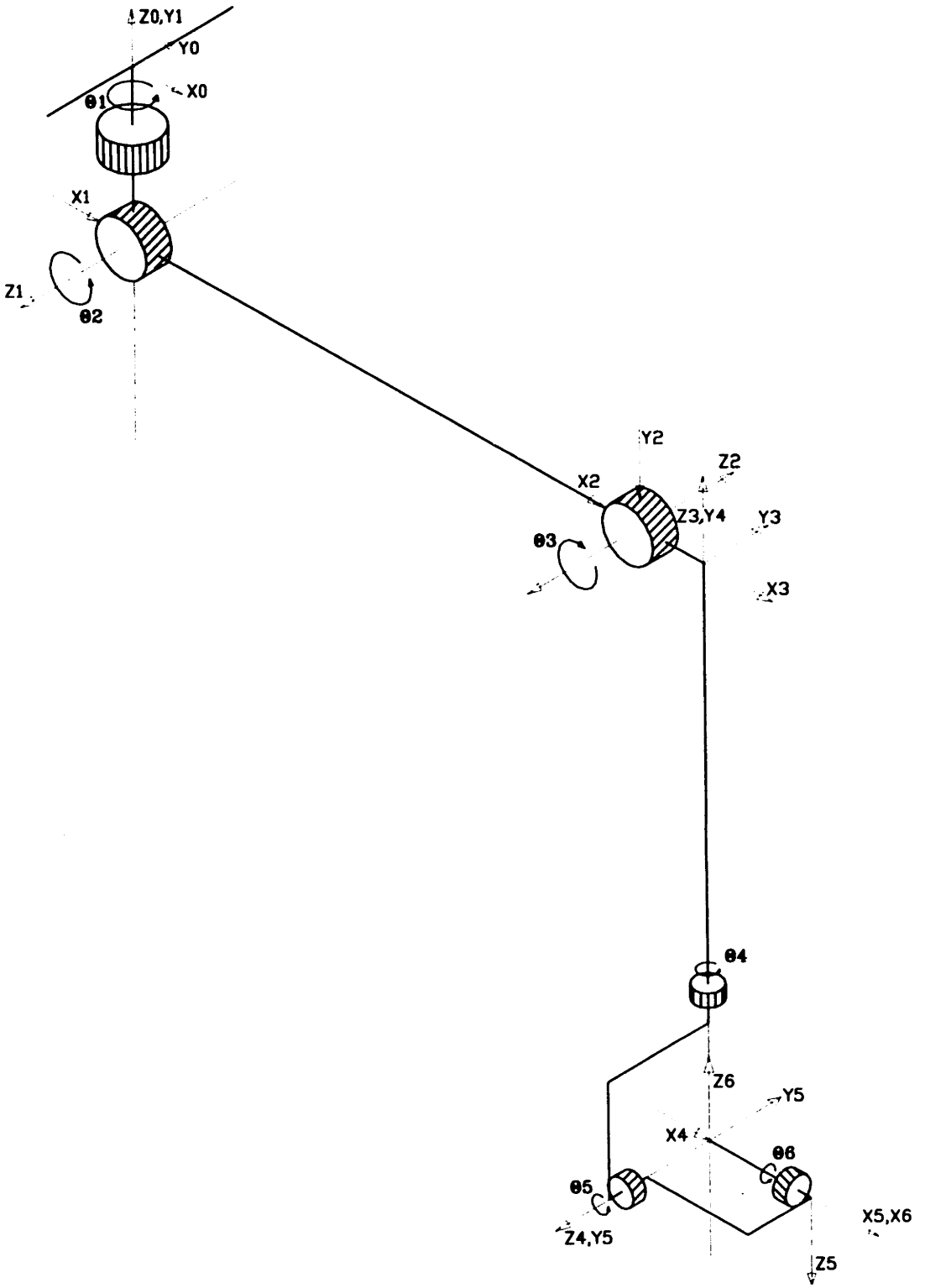

Figure 2-7: Coordinate frames for the MIT-Toolhandle

$$A_5^4 = \begin{bmatrix} cos\theta_5 & 0 & sin\theta_5 & 0 \\ sin\theta_5 & 0 & -cos\theta_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$A_6^5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -cos\theta_6 & -sin\theta_6 & 0 \\ 0 & sin\theta_6 & -cos\theta_6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

2.3 Conclusion

Humans have two modalities of touch sensing: kinesthetic and tactile. The kinesthetic system has trouble accurately sensing position, and the brain often relies on vision for accurate positioning. The tactile system is extremely sensitive to local shape and texture and will be difficult to stimulate in a controllable fashion. The tactile system is responsive to frequencies of at least 1kHz.

A tool-handle haptic interface stimulates both of these modalities. By holding a physical tool the mechano-receptors can be passively satisfied, and the force feedback of the device can concentrate on the kinesthetic system. In addition, there are many meaningful tasks which more directly involve a tool interaction than direct fingertip interaction.

The MIT-Toolhandle is a haptic interface based on the PHANToM using the tool-handle paradigm. With a larger workspace and a higher maximum force, the MIT-Toolhandle is capable of simulating situations which the PHANToM is not. One such situation which has been implemented is two hand manipulation.

Users fatigue more quickly using the MIT-Toolhandle than they do using the PHANToM. This is not unexpected given that tasks which use the full arm are more tiring than ones which allow bracing against an object. Tasks which can be accurately simulated with a PHANToM scale haptic interface should definitely utilize the PHANToM's advantages, but some tasks don't lend themselves to its design.

The MIT-Toolhandle has been used extensively to study haptic rendering methods. Using the "point in space" paradigm, algorithms to display arbitrary geometries, friction, and texture have been developed.

Chapter 3

Rendering Algorithms

A haptic interface provides us the means to provide feedback forces on a user in a controlled manner. Since we have the flexibility to display any force, we need an algorithm to determine which force should be applied. This is the purpose of a haptic rendering algorithm.

The work described below is concerned with replicating the forces felt by humans when interacting with real objects.[1] We will concern ourselves with such issues as shape, surface compliance, and texture which are the staples of human haptic perception. Our methods have been implemented for a haptic interface which can control the 3 translational degrees of freedom. The user's interactions with the virtual environment are reduced to those of a *point* interacting with 3 dimensional objects. The term ***haptic interface point*** will be used to describe the endpoint location of the physical haptic interface as sensed by the encoders.

3.0.1 Goals of a Haptic Renderer

Shape is the most fundamental haptic characteristic of any object and will be the central part of the haptic renderer. Clearly, it makes little sense to display texture in a volume of space; texture should be felt on the surface of objects. Simply stated the haptic renderer should be capable of displaying arbitrary 3 dimensional objects[2], like the one shown in Figure 3-1. These objects should be able to have concave portions as well as convex portions. The haptic representation should be able to be generated from a standard object file format (e.g. AutoCad's DXF, WAVEFRONT's OBJ).

In addition, the haptic renderer should be extensible to include surface effects. The algorithm should supply a framework which simplifies the addition of other haptic effects. Lastly, to simplify scene design we attempted to design a system which allows complex objects to be built by overlapping (logical ANDing) a number of simple objects. The haptic renderer should be robust to the presence of multiple objects.

[1] as opposed to simulating an alternate scale like Fred Brooks's work in molecular docking or an "enhanced reality" environment with haptic cues not found in the real world.

[2] The methods described tend to represent objects as polygonal approximations to the desired shape. With a fine mesh and edge smoothing these objects can be indistinguishable from continuous objects. The author deems this to be satisfactorily arbitrary.

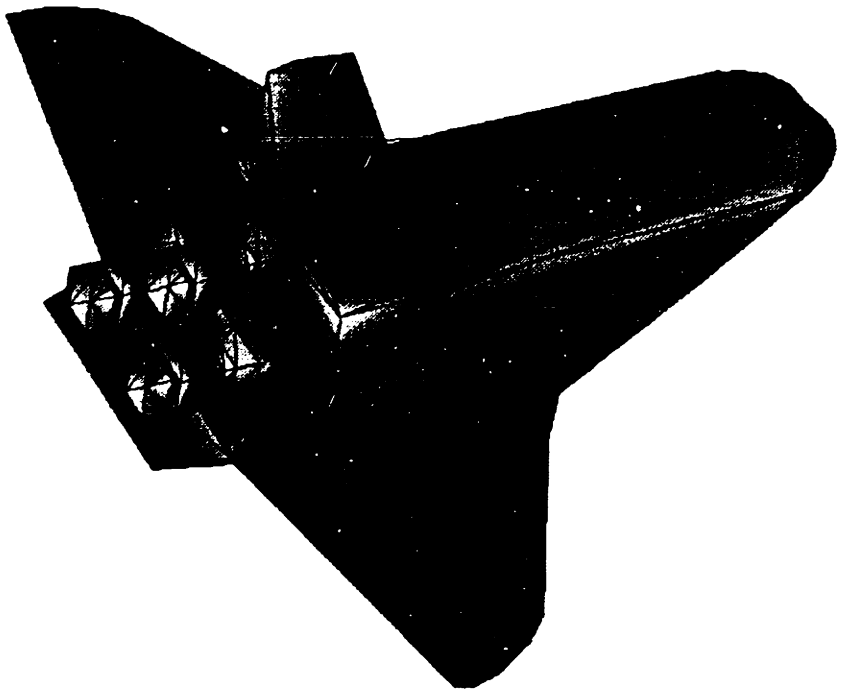

Figure 3-1: A space shuttle consisting of 616 polygons, currently renderable by the god-object algorithm

3.0.2 Problems in Haptic Rendering

As in any physical simulation, one of the major problems in haptic rendering is collision detection. With our point among objects paradigm, collision detection is straight forward, but can still be computationally intensive. The human haptic system requires a servo loop on the order of 1 kHz; if complex models are to be simulated then sophisticated collision detection algorithms are needed. [10] The rendering algorithms themselves are very local in nature and can be decoupled from the complex collision detection algorithms, as they are in this thesis.

One difficulty in haptic rendering is due to the inherent mechanical compliance of haptic interface devices. Since the maximum "displayable" stiffness of any virtual object is limited, the haptic interface point often penetrates into a simulated object a greater distance than would be possible in real life. In typical demonstrations with our system, the user presses the haptic interface as much as a half an inch into the virtual object when he is stroking the surface. Because humans have poor position sense, this usually goes by unnoticed. Although this volume penetration is not perceptible by the user it can make the logic of generating proper interaction force vectors difficult. When a user is in contact with the center of a large flat surface, it is obvious that the direction of the force should be normal to the plane. As the boundary between surfaces is approached, the choice of which virtual surface the user should be touching becomes ambiguous. Multiple paths, meriting different feedback forces, can be taken to reach the same internal location. Figure 3-2.a shows two possible paths to reach the same location in an square; without a history we do not know which path was taken.

Two solutions to this ambiguity are examined. A short description of vector field methods in Section 3.1 will illustrate their limitations and illuminate the necessity for the god-object method. The implications of the constraint-based god-object method will be demonstrated in Section 3.2.

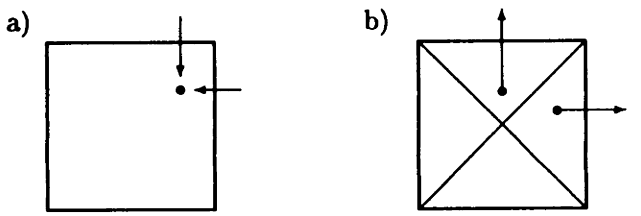

Figure 3-2: Generation of contact forces using volumes (the small dot represents the haptic interface point)

Section 3.3 discusses some of the implementation issues for the god-object method which may be applicable to other representations.

3.1 Vector Field Methods

Vector field methods are a classification for any method whose feedback force can be computed by knowing only the location of the haptic interface point. These force fields can be pre-computed and contain no notion of history. These methods use volumes to compute forces rather than the more physically real effects of interacting with surfaces.

One method handles the location ambiguity (mentioned above) by subdividing the object volume and associating a sub-volume with each surface. [12] In two dimensions, the vector field method subdivides the square's area and assumes that the user entered from the closest edge (Figure 3-2.b). The force vectors are normal to the edge and proportional to distance penetrated. This method causes a sensation of "sharpness" to be felt at corners due to the sudden force discontinuity from passing from one region into another. This can be useful when truly sharp corners are desired, but confusing when transitions are made accidentally.

This method works for simple geometric shapes because it is reasonably easy to construct these subspaces by hand. In addition, any shape that can be described with an equation can easily be modeled. For example, spheres display a feedback force in the direction of the vector pointing from the sphere's center to the haptic interfaces point with a magnitude that is a function of the distance the point has penetrated the sphere's surface. [12] The inherent simplicity of these methods has allowed interesting work in dynamic objects and surface effects, [18] but the methods are not flexible enough to allow arbitrary geometries.

The drawbacks of vector field methods are:

1. It is often unclear which piece of internal volume should be associated with which surface.
2. Force discontinuities can be encountered when traversing volume boundaries.
3. Small and thin objects do not have the internal volume required to generate convincing constraint forces.

It is pretty obvious how to create sub-volumes to make a haptic representation of a cube, but it is less clear for a complex object like the space shuttle shown in Figure 3-1. The complexity of this geometry problem can not be under emphasized.

We've found that subjects often traverse volume boundaries accidentally in simulations using the vector field method. It can be seen from Figure 3-2.b that when the haptic interface point is pressed

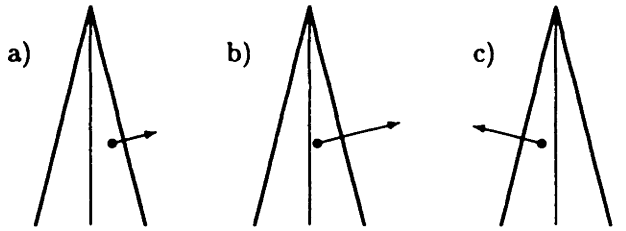

Figure 3-3: Pushing through thin objects in vector field method

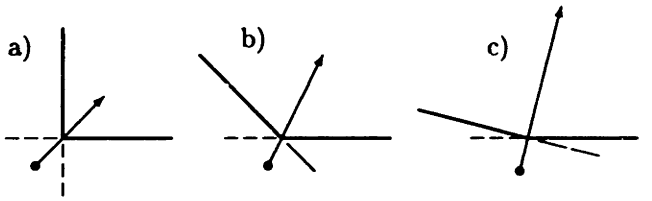

Figure 3-4: Force summation for multiple objects in vector field methods

straight into the side of cube, it is possible to make the transition to another side of the cube. To some extent the size of the cube is a function of how hard it is being pressed on. These side-effects are unexpected and tend to confuse subjects. This problem is even more severe with thin objects.

3.1.1 Problems with Thin Objects

Vector field methods also break down when thin objects are to be rendered. Due to the limited servo and mechanical stiffnesses, the haptic interface point must travel somewhat into the object before enough force can be applied to make the object feel "solid". When this distance becomes greater than the thickness of an object, the vector field model produces unrealistic sensations. As shown in Figure 3-3.a user touches surface and feels small force. As he pushes harder he penetrates deeper into the object (Figure 3-3.b), until he passes more than halfway through the object where the force vector changes direction and shoots him out the other side(Figure 3-3.c). The algorithm has assumed that the user entered through the other side. This effect takes place at any convex corner.

One possible solution is to keep a history of contact occurrences so we know the surfaces our haptic interface point has passed through. With a history of old locations it would be clear which surface is meant to be touched and, therefore, which force should be displayed; unfortunately, this method can rapidly become numerically cumbersome. It is important to have a compact representation in both use of memory and processing time. In Section 3.2 1. we describe the use of a "god-object" as a compact representation of history.

3.1.2 Problems with Multiple Objects

Although not a requirement of a rendering algorithm, we'd like to be able to overlap simple objects to create more complex objects. This is not feasible with vector field methods. In regions where object intersections occur, we might consider computing the net reaction force by vectorially adding contributions from each object's force field in hope that it will generate the correct sensations at corners and edges. This will not always compute the correct force.

When a user is in contact with more than one such object simultaneously, the net surface stiffness

can be larger than that of either surface alone. On objects meeting at perpendicular surfaces, the forces can be summed because the distance into the solid is the vector addition of the two orthogonal components (Figure 3-4.a). However, as the angle made by the surfaces is increased (Figure 3-4.b), the resulting reaction force magnitude (and corresponding apparent stiffness) becomes too large. Finally when the surfaces are almost parallel the force approaches twice that of either surface alone (Figure 3-4.c). Surfaces intersecting at acute angles are less stiff than either surface alone. The system is no longer robust; these stiffness discontinuities could cause the maximum stable stiffness to be exceeded.

Generating believable forces requires directly determining the distance the haptic interface point has penetrated an object and the surface(s) it has passed through to arrive at its current position. The constraint-based method described in Section 3.2 computes this distance directly.

3.2 The God-object Algorithm

We will define and explain the reasoning behind the god-object representation and how it can be utilized in rendering polyhedral representations of objects, in Section 3.2.1. The term god-object has been previously used in a similar spirit to describe a virtual object controlled by a human user in physical simulations. [7]

Using the history (the god-object location calculated in the previous servo cycle) and the current haptic interface point, a set of surfaces currently impeding motion can be found. A discussion of constraints is given in Section 3.2.2.

Lagrange multipliers are used to find the new location of the god-object during contact with a virtual object. The god-object's new location is chosen to be the point which locally minimizes the distance between the god-object and the haptic interface point, subject to the constraint that the god-object is *on* a particular surface. The mathematics of this method are explained in Section 3.2.3

3.2.1 God-objects

As we saw in the previous section, a number of problems arise from the penetration of the haptic interface point into virtual objects. We know the haptic interface point cannot be stopped from penetrating the virtual objects, but we are free to define additional variables to represent the *virtual location* of the haptic interface. This location is what we will call the *god-object.* [25]

We have complete control over the god-object; we can prevent it from penetrating any of the virtual objects and force it to follow the laws of physics in the virtual environment. The god-object is placed where the haptic interface point would be if the haptic interface and object were infinitely stiff. Because the god-object remains on the surface of objects, the direction of the force should never be ambiguous. This allows a more realistic generation of the forces arising from touching an object. In particular, this method is suitable for thin objects and arbitrarily shaped polyhedra.

In free space, the haptic interface point and the god-object are collocated, but as the haptic interface moves into an object the god-object remains on the surface. The god-object location is computed to be a point on the currently contacted surface such that its distance from the haptic interface point is minimized. This assumes the god-object moves across the surface without being impeded by friction. Inclusion of friction is a simple extension which is described in Section 4.1.

Even when friction is being modeled, the god-object algorithm positions the god-object as if there was no friction.

By storing additional state variables for the position of the god-object (one variable for each degree of freedom of the apparatus), we can keep a useful history of the object's motion in a compact manner. In our work with a three-degree-of-freedom haptic interface, the god-object is a point needing only three coordinates to fix its location.

Once the god-object location is determined, simple impedance control techniques can be used to calculate a force to be displayed. Stiffness and damping elements applied between the haptic interface point and the god-object will model the local material properties. Hooke's law transfers directly into Equation 3.1 where k is the stiffness of the surface.

$$F_{stiffness} = k(x_{god-object} - x_{haptic-interface}) \tag{3.1}$$

The damping force should be based on the motion of the haptic interface point relative to the motion of the god-object. Since we don't want to impede motion tangential to the surface, only the motion along the surface normal $\hat{N}$ should be used. Equation 3.2, in which c is the damping coefficient, computes this force. The damping force should only be applied when it stiffens the surface; the surface should not resist the withdrawal of the haptic interface. Currently this algorithm has only been implemented on static objects. The motion of the god-object can be discontinuous; a stable method for determining the god-object's velocity will be needed.

$$F_{damping} = c((\dot{x}_{god-object} - \dot{x}_{haptic-interface}) \cdot \hat{N})\hat{N} \tag{3.2}$$

$$F_{normal} = F_{stiffness} + F_{damping} \tag{3.3}$$

Section 4.2.2 demonstrates how stiffness and damping can vary between objects and across an object. These coefficients can be varied arbitrarily as long as they do not exceed the maximum displayable values of the device as limited by servo stiffness and stability. In addition, non-linear stiffnesses could be used to give surfaces more interesting sensations, such as the click of a button.

3.2.2 Constraints

Although we are interested in simulating volumes, we interact with those volumes on their surfaces. In general, it is more convenient to represent objects by their surfaces. To simplify the mathematics of the problem, only planar surfaces are used.

In this work we have found that a good first-cut haptic representation can be derived from the same polyhedral geometry used to represent objects for visual rendering. Straight-forward lists of vertices, edges and facets, as are found in standard polyhedral representations, are sufficient to permit use of the god-object algorithm. This is particularly convenient in that it enables haptic rendering of a large body of existing visually render-able objects.

A mesh of triangular elements is used because it is the most fundamental, and assures that all of the nodes are coplanar. Graphic models do not require the exactness required by haptic models, so it is not uncommon to find objects with four-noded surfaces where the nodes are not coplanar. The problems caused by such surfaces can be avoided by using a triangular mesh. In addition, since a plane is completely determined by three points, we can move nodes *on the fly* and still have

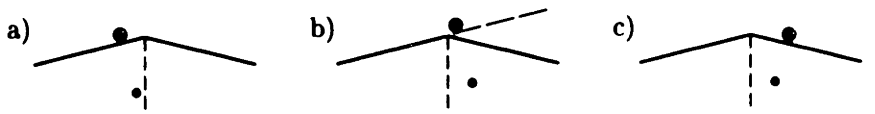

Figure 3-5: God-object motion between convex surfaces takes place in two steps (the small dot represents the location of the haptic interface, and the larger dot represents the god-object)

geometrically acceptable surfaces. Moving nodes due to applied forces is one way to implement deformable surfaces.

Using a polygonal representation of objects makes collision detection simple. For an infinite surface (a planar constraint), we will denote the surface as *active* if the old god-object is located a positive (in the direction of the outward pointing surface normal) distance from the surface[3], and the haptic interface point has a negative distance to the surface (i.e. on the other side). This creates surfaces as one-way constraints to penetration[4].

When the surfaces are not of infinite extent, then for a surface to be active, we also require that the god-object contact take place *within the boundaries* of the surface. A line can be traced from the old god-object to the new haptic interface point. If this line passes through the facet (within all of the edges) then that facet should be considered as active.

When touching convex portions of objects, only one surface should be active at a time. The transition of the god-object between two surfaces sharing a convex edge requires two steps. While the first surface is active (Figure 3-5a), the contact point must stay on the plane of the surface, but not necessarily within the boundaries; the first step places the god object over the second surface (Figure 3-5b), but still in the plane of the first surface. In the next servo loop the first surface will no longer be active and the god-object can then fall to the second surface (Figure 3-5c). The times and distances involved are small enough to cause only an imperceptible and transient distortion of shape.

When probing a concavity, multiple surfaces can be active simultaneously. When touching the concave intersection of two planes, both constraints are active, and the god-object's motion should be restricted to a line (the intersection of both planes). When in contact with the intersection of three surfaces, all three will be active, and the net constraint becomes a point (the intersection of all three planes). At the intersection of more than three surfaces, only three will be active at any one time, and the user will still be constrained to a point. Once we identify one active constraint, we can temporarily limit our search to neighboring surfaces.

Additional care must be taken when surfaces intersect at an acute angle (as viewed from the "outside" of the surfaces) to form a concavity. If the user presses into one such surface and slides

[3]Due to the round-off error present in any digital system the newly computed god-object position might be infinitesimally below the virtual surface. If no precautions are taken the god-object will cross (and no longer be restrained by) the virtual surface. To prohibit such behavior, we add a small constant to the calculated distance from the god-object to the virtual surface. Conceptually, this is equivalent to using another plane just below the virtual surface for computation of god-object distances.

[4]This method treats surfaces as single-sided surfaces respecting the fact that objects are formed from a closed set of surfaces. There is no physical reason to touch both the inside and outside of an object simultaneously. In some respects it is actually beneficial because a simulation can begin with the haptic interface in any configuration; if the haptic interface starts inside a virtual object the user can simply move out of the object as it were free space. In vector field methods a "workaround" has to be introduced for this contingency.

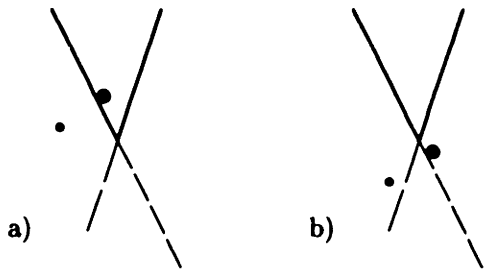

Figure 3-6: Special precautions need to be taken for concave intersections (The large dot represents the position of the god-object and the small dot represents the haptic interface)

along it, it is possible for the god-object to cross the other constraint surface before the haptic interface point does (Figure 3-6). The god-object will cross to the negative side of the surface before the haptic interface will, and the constraint will not be activated (Figure 3-6b). Once the god object crosses a wall, it will be free of that constraint; therefore, we must not allow the crossing.

One solution is to iterate the whole process. The first iteration will find a set of active constraints and calculate a new god-object location. Using the "new" god-object location as the haptic interface point, the constraints of neighboring surfaces are checked again to see if any additional surfaces should be active. If additional constraints are found, then another "new" god-object location is computed. This iteration continues until no new constraints are found. This iteration process requires very little time, since the number of possible constraints in the neighborhood of a contact is very small. The maximum number of iterations is equal to the maximum number of possible simultaneous constraints.

3.2.3 God-object Location Computation

When a set of active constraints has been found, Lagrange multipliers can be used to determine the location of the new god-object. Equation (1) gives the energy in a virtual spring of unity stiffness, where x, y, and z are the coordinates of the god-object and x_p, y_p, and z_p are the coordinates of the haptic interface point. Constraints are added as planes because they are of first order (at most) in x, y, and z in the form shown in equation (2). The general case for our 3-degree-of-freedom haptic interface involves 3 constraints; for less constrained instances, zeros are used to replace unused constraints giving a lower order system.

$$Q = \frac{1}{2}(x - x_p)^2 + \frac{1}{2}(y - y_p)^2 + \frac{1}{2}(z - z_p)^2. \tag{3.4}$$

$$A_n x + B_n y + C_n z - D_n = 0 \tag{3.5}$$

The new location of the god-object is found by minimizing L in equation (3) below by setting all six partial derivatives of L to 0. Because the constraints are of first order and Q is of second order, the differentiation is very simple. In fact, the differentiation can be done symbolically and the actual coefficient values can be just substituted at runtime. In the general case we have six variables (x, y, z, l_1, l_2, l_3), which will give us six partial derivatives organized into a set of simultaneous equations:

$$
\begin{aligned}
L &= \tfrac{1}{2}(x - x_p)^2 + \tfrac{1}{2}(y - y_p)^2 + \tfrac{1}{2}(z - z_p)^2 \\
&\quad + l_1(A_1 x + B_1 y + C_1 z - D_1) \\
&\quad + l_2(A_2 x + B_2 y + C_2 z - D_2) \\
&\quad + l_3(A_3 x + B_3 y + C_3 z - D_3)
\end{aligned} \tag{3.6}
$$

$$
\begin{bmatrix}
1 & 0 & 0 & A_1 & A_2 & A_3 \\
0 & 1 & 0 & B_1 & B_2 & B_3 \\
0 & 0 & 1 & C_1 & C_2 & C_3 \\
A_1 & B_1 & C_1 & 0 & 0 & 0 \\
A_2 & B_2 & C_2 & 0 & 0 & 0 \\
A_3 & B_3 & C_3 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix} x \\ y \\ z \\ l_1 \\ l_2 \\ l_3 \end{bmatrix}
=
\begin{bmatrix} x_p \\ y_p \\ z_p \\ D_1 \\ D_2 \\ D_3 \end{bmatrix} \tag{3.7}
$$

The matrix, equation (4), has a number of useful properties. It is symmetric, the upper left hand corner (3x3) is always the identity matrix, and the lower left hand corner is always a null matrix, it should never require row swapping, and should always be invertible.[5] Because of these properties it requires only 65 multiplicative operations (multiplies and divides) to solve for x, y, and z. In the 5x5 case (when there are only 2 constraints), it requires only 33 multiplicative operations, and in the 4x4 case (a single constraint), it takes only 12. When no constraints are active, the god-object is located at the position of the haptic interface. Also because of its symmetry, and the identity and null portions, we only need variables for 15 of the 36 matrix locations and 6 for the right side vector.

3.3 Implementation Details

Although the implementation details are not as important as the god-object algorithm itself, there are a few which merit discussion. Section 3.3.1 describes virtual object structures and how multiple copies of the same virtual object can share the same model information. This object structure is efficient with respect to object motion, but moving objects require a slightly different active surface selection scheme (Section 3.3.2). Section 3.3.3 explains the active surface detection algorithms in more detail and a mechanism for hidden surface removal.

3.3.1 Virtual Objects as Instantiations of Models

The renderer requires a file or a group of files to describe the scene to be simulated. Each line of our scene-file names a .plg[6] file and gives its location in the global coordinate frame. The .plg file is a minimal object description file consisting of a name, a number or vertices, a number of surfaces, the list of vertices, and the list of surfaces. Each node entry consists of its x, y, and z coordinates; the nodes are numbered from 0 to n-1. The surface entry consists of the color, the number of nodes, and then the node numbers clockwise as viewed from the front of the surface. The following is an example of a .plg file for a small box.

```
box 8 12
15 15 20
15 -15 -20
```

[5] It is important to make sure that none of the surfaces are parallel because this will cause the matrix to be singular.

[6] In the current version of the renderer we are moving towards the more accepted WAVEFRONT's .obj file format.

```
15 15 -20
15 -15 20
-15 15 -20
-15 15 20
-15 -15 20
-15 -15 -20
0x1E80  3 0 1 2
0x1E80  3 0 3 1
0x1E80  3 0 2 4
0x1E80  3 4 5 0
0x1E80  3 3 0 5
0x1E80  3 5 6 3
0x1E80  3 7 6 4
0x1E80  3 6 5 4
0x1E80  3 6 1 3
0x1E80  3 6 7 1
0x1E80  3 7 4 2
0x1E80  3 2 1 7
```

This file is parsed and a list of nodes, edges, and surfaces are created to represent the object. The software model is built in its own local coordinate frame.

The instantiation of this model is a separate data structure which has a pointer to the object model. This object has a position, a velocity, an orientation, an angular velocity, a scale, and a material. From the position, orientation, and scale, a transformation matrix can be computed to transform back and forth from the local coordinate frame of the model to the global coordinate frame of the virtual environment. Since the haptic renderer only worries about interactions with a point, we can do the active surface detection in the coordinate frame of the model. This means that we won't have to transform all of the points, edges, and surfaces of the model each time it moves; only the haptic interface point will need to be transformed. Once candidate active surfaces are found they will have to be transformed to the global coordinate frame because we may be in contact with surfaces from multiple objects at the same time.

Since it is not uncommon for a scene to have more than one of an object (like chips and resistors on a circuit board, or plates and chairs in a kitchen) our architecture has been designed to allow multiple "object instantiations" to share the same geometric model. This should greatly reduce the amount of memory required for scenes with redundancy. This scheme only works for rigid objects since a deformation in one of the objects would cause all similar objects to be deformed. The algorithm could be written to allow objects to share models until they are deformed, at which point they make a private copy (a copy-on-write policy).

A material could be associated with an object to provide specific information about density, impedance properties, friction properties, and texture properties. By keeping material properties separate from geometry we increase the likelihood that either can be reused. This implies that the materials files should be stored in a manner that allows sharing between objects. Sharing material properties might not be feasible if they vary across the surface of an object; texture maps (Section 4.3) could be shared, but the mapping from surface coordinates to texture coordinates could not.

3.3.2 Moving Objects

If the objects are capable of moving they should keep track of their velocity and angular velocity so their position and orientation can be updated. From the current position and orientation a new transformation from the global coordinate frame can be found.

When virtual objects move, the active surface detection algorithm for the god-object method changes slightly. If we simply update the transform, the old god-object location might be now beneath the surface it was *on* during the last servo cycle. Special precautions have to be taken to assure that the god-object doesn't inadvertently cross object boundaries when the objects are moved. We want to keep the god-object at it's old location in the model's coordinate frame, so the transformation from the previous servo cycle is stored and used that one to find the god-object location. The new transformation is still used to find the haptic interface point's new location in the local coordinate frame.

Objects are accelerated by forces coming from interactions with other objects and with the god-object. The rendering algorithm must compute the forces the haptic interface is applying to each object. Since the god-object could be in contact with multiple objects, the force from the god-object has to be decomposed by surface. Since the planes being touched are not necessarily orthogonal, we cannot simply project the god-object force onto each of the planes. In the most general case we'll have to solve a three by three matrix to find the contribution from each surface. If the surfaces are being shaded, then the force vector won't necessarily be able to be broken down into vectors normal to the surfaces; another means will have to be employed to compute the forces on each object.

3.3.3 Hidden Surface Removal

As was mentioned in Section 3.0.1, one of the goals of the god-object method is to be able to build up complex scenes by overlapping (logical AND'ing) simpler objects. When multiple objects are in close proximity a naive active surface detection algorithm will return too many surfaces. We need a hidden surface remover to assure that we only touch the closest valid surfaces.

A simple example of the problem is a small box on top of a larger box (Figure 3-7.a). When pressing down on the top of the small box we might cross the top surfaces of both boxes (Figure 3-7.b). Our constraint checker is likely to mark both of these surfaces as "active", but, obviously, we aren't touching both. If both surfaces are entered into the Lagrange engine there will be a zero pivot. It is important to have a second step which scrutinizes the candidate surfaces and tosses out the inappropriate ones.

Similar to hidden surface removal methods in graphics we only want to render the object closest to the observer. In haptics the observer is the god-object rather than a camera, as in graphics. This surface removal involves finding redundant surfaces (surfaces with the same or similar surface normals) and calculating which is closest to the god-object. Only the closest surfaces are kept. Hidden surface removal for haptic interfaces is less computationally intensive than methods like z-buffering for graphics because there are usually only one or two haptic interfaces rather than thousands of pixels on a screen.

The list of non-redundant surfaces should be ordered (by distance from the god-object) from nearest to farthest and applied to the god-object one at a time. Beyond redundant surfaces we can find surfaces which could technically be active, but still shouldn't be considered. One example is

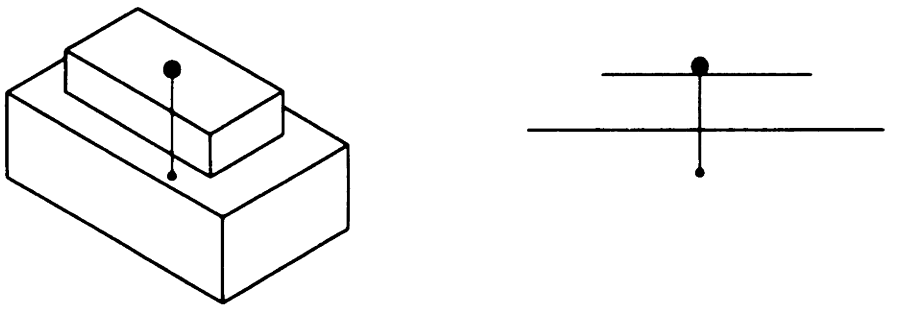

Figure 3-7: A situation requiring hidden surface removal

present in the interaction with two overlapping rectangular solids at an obtuse angle. The user could press into the top of the block and cross the top planes of both blocks. They are not parallel so the hidden surface remover hasn't removed it, but if both surfaces are entered into the Lagrange engine then the god-object will be pulled sideways to intersection of the two planes.

By adding constraints one at a time we can avoid this problem. We compute the god-object location using just the closest surface and check if the new god-object location is valid. If this new god-object crosses virtual object surfaces then the nearest surface crossed can be added to the matrix and the computation redone. This will assure that we don't over-constrain the god-object.

3.4 Conclusion

Due to the inherent compliance in haptic interfaces, the current location of the haptic interface point isn't sufficient to generate a correct feedback force. Some history of the haptic interface point's motion needs to be maintained to avoid the ambiguity of feedback force.

To keep a compact representation of history, we propose the concept of a god-object algorithm. A god-object algorithm keeps track of a position on the object's surface which can be thought of the virtual location of the haptic interface's endpoint. A feedback force can be easily generated from the location of the god-object and the haptic interface point using simple control laws. With the god-object location calculated during the previous servo cycle and the current location of the haptic interface point, a new god-object location can be computed.

The constraint-based god-object method is an efficient god-object method for arbitrary rigid polyhedra. By using standard object file formats, an extensive set of existing objects are available for rendering. In addition, since the god-object method returns a position rather than a computed force, the algorithm is easily extensible to include friction, smoothing, and texture.

Chapter 4

Extensions to the God-object method

The god-object method in itself only handles shape, but it provides a solid framework for surface effects. Rather than just computing a feedback force, the god-object method computes a location on the virtual object's surface where the user can thought to be touching. Surface effects are dependent on the surface location, so having the god-object's location greatly simplifies their implementation.

This chapter explores some common surface effects and how they can be incorporated into the god-object architecture.

4.1 Friction

The most important haptic effect beyond geometry is friction. Without friction a user is mostly limited to probing, and exploring a virtual environment; with friction one can manipulate virtual objects and change the virtual environment in a straight-forward manner. Friction can also be used to give the feel of doing work; this friction algorithm has been used to simulate the sensation of driving screws with a screwdriver haptic interface (Figure 4-1).

The friction that we require is a static friction; viscous drag is not sufficient. In a virtual environment, a user should be able to support an object using only the friction force without the object slipping. Since viscous friction can only provide a force when there is a relative velocity, it does not fit our needs.

Friction models with stiction, in general, have two states of contact: stiction and kinetic friction. In the stiction state the user has made contact with an object but has not provided enough tangential force to "break away"; when enough force is applied a transition is made to the kinetic friction state. In the kinetic friction state a force is applied in a direction to impose the direction of motion. All friction forces are applied in directions tangential to the normal force.

For a general implementation of the static state:

$$\Delta x = (x_{stiction-point} - x_{haptic-interface}) \tag{4.1}$$

$$\Delta x_{tangential} = (\Delta x - (\Delta x \cdot \hat{N}) \cdot \hat{N}) \tag{4.2}$$

$$F_{friction} = k\Delta x_{tangential} + c\Delta \dot{x}_{tangential} \tag{4.3}$$

where k is the stiffness, c is the viscosity, and $\hat{N}$ is a unit vector normal (outward pointing) to the surface. Equations 4.3 and 3.3 can be used to find the the total feedback force (Equation 4.4).

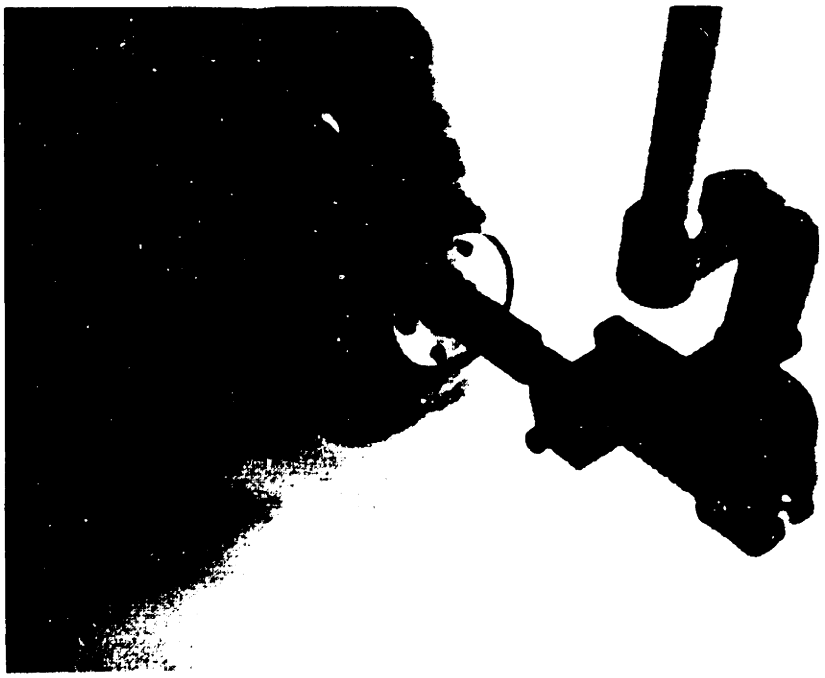

Figure 4-1: A 1 degree-of-freedom active screwdriver for driving virtual screws

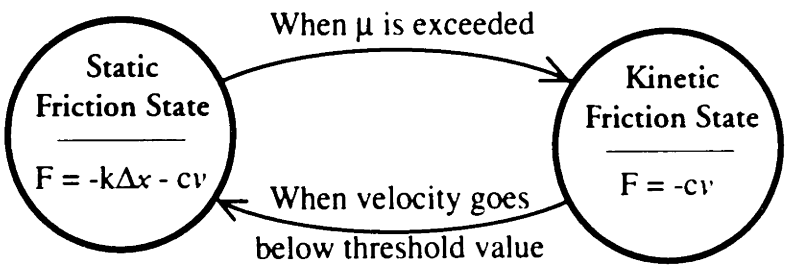

Figure 4-2: State transition diagram for viscous friction model.

$$F_{total} = F_{normal} + F_{friction} \tag{4.4}$$

The transition from static to kinetic friction should be made when:

$$F_{friction} >= \mu F_{normal} \tag{4.5}$$

where μ is the coefficient of friction.

One published friction model uses viscous friction in the kinetic state. [17] Their two state friction model makes the transition back to the stiction state at a certain velocity threshold (Figure 4-2). Since position is descritized it is difficult to accurately determine the direction of the velocity vector when the velocity is small. By using viscosity for kinetic friction they are assured that only small forces will come from small velocities so any errors in calculation of velocity direction should go unnoticed.

We've chosen to model friction using a Coulomb (velocity independent) friction model. The

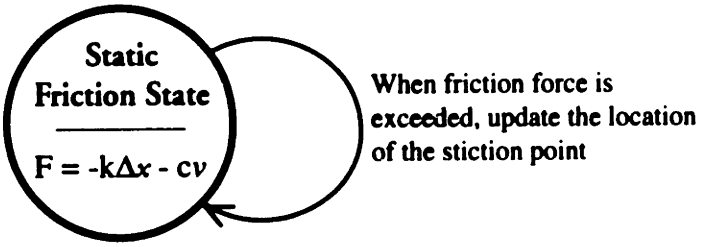

Figure 4-3: State transition diagram for coulomb friction model

magnitude of the kinetic friction force is only a function of normal force and coefficient of friction, μ. The friction force is applied to oppose the direction of motion. At small velocities we will have poor discrimination of velocity direction but the force magnitude can still be large; this will cause vibration if we use a traditional two-state model. Since only the direction of the velocity vector is needed, and not the magnitude, our current position with respect to the stiction point will give us a more stable and accurate direction for the friction force.

This algorithm can be visualized as a convict with a ball and chain around his ankle. As long as convict stays within the length of the chain the ball is motionless[1]. When the convict tries to go beyond the length of the chain, the ball dragged behind.

The coulomb friction implementation really only has one state (Figure 4-3). The new stiction point is placed on the line between the current position and the old stiction point. It is placed at such a distance from the current position so that the force on the user is equal to maximum friction force (Equation 4.6). This method really eliminates the dependence on the velocity signal; we are no longer susceptible to the noise in the velocity signal which could cause vibration.

$$\Delta x_{tangential} = \frac{\mu}{k} F_{normal} - \frac{c}{k} \Delta \dot{x}_{tangential} \tag{4.6}$$

4.1.1 Friction with Slip

This implementation is very stable, but in this incarnation it is impossible for a user to tell whether he/she is slipping relative to an object or not. If an object is being supported from the sides, the friction forces are constant and equal to the gravitational force on the object. When slipping is taking place, we've specified that a constant force is applied to the user as a criterion for placement of the stiction point. These two sensations are tactilely the same.

In the real world there is usually some vibration associated with slipping. To mimic this in our virtual model we use two different coefficients of friction. One μ is used for checking to see if we should update the location of the stiction point, and another, slightly lower μ is used for calculating the new location of the stiction point. This means that each time a new stiction point is placed the friction force is lower, and a bit of distance will have to be traveled to break away again. When a constant relative velocity is specified we get the saw-tooth wave seen in Figure 4-4. When the stiction point is moved, the new position is calculated using a coefficient of friction smaller than the one used to check if the maximum friction force has been exceeded. When a constant velocity is specified a vibration is felt due to the saw-tooth like variation in friction force.

[1] in this case the chain is elastic and is constantly pulling the convict towards the ball.

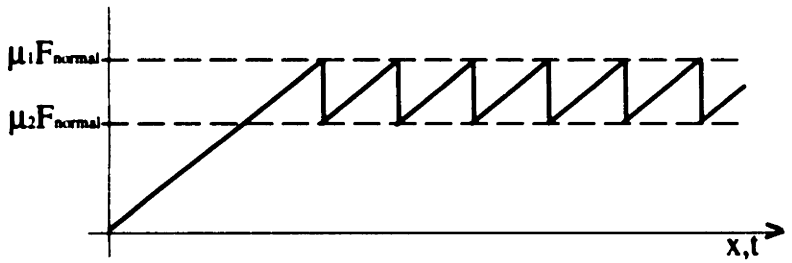

Figure 4-4: Sawtooth vibration from friction model with two friction coefficients

The god-object method makes implementing this algorithm trivial. Given the location computed by the god-object method we can determine what the normal force should be. Since the god-object is always on the virtual object's surface, there is no non-tangential motion of the god-object. Equation 4.2 can be simplified to Equation 4.7.

$$\Delta x_{tangential} = (x_{stiction-point} - x_{god-object}) \tag{4.7}$$

If the normal force is 0 then the stiction point should be collocated with the god-object. Otherwise the position of the stiction point should only be updated (moved toward the god-object) when the distance from the god-object to the stiction point exceeds the distance from the haptic interface point to the god-object multiplied by μ. The stiction point can then be used (instead of the god-object) for graphical display and force computation (Equation 4.8).

$$\Delta F_{total} = k(x_{stiction-point} - x_{haptic-interface}) \tag{4.8}$$

The stiction point acts as a god-object for the god-object; there are distinct similarities in the two algorithms. They both are necessary because of the inherent compliance in haptic interfaces. The device can't know which direction it should push on the user, until after he has already tried to move. Both algorithms try to decode the position of the user in the virtual environment (and from that what force should be applied) from the position of the of the haptic interface point.

4.2 Interpolation Algorithms

It is not uncommon for the properties of an object to vary across its surface. There are many mechanisms for continuous variation of a parameter, but interpolation between nodal values is an obvious choice for slowly varying properties. Interpolation has low storage requirements. Surface smoothing can by implemented by interpolating surface normals, and more sophisticated objects can be rendered by varying object impedance across an object.

4.2.1 Surface Smoothing

The constraint-based god-object method simulates rigid polyhedra, but few objects in real life can be accurately described as rigid polyhedra. A model could be refined enough that it would be

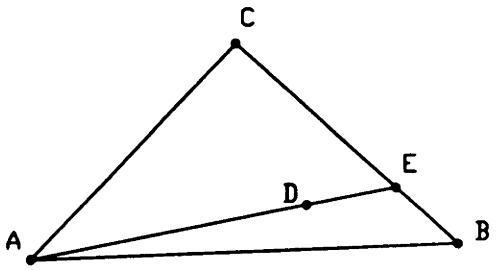

Figure 4-5: Interpolation method for continuous variation of parameters

indistinguishable (to a human) from a continuous object, but such a model would require extensive memory and processing time. An algorithm for smoothing the intersections of the polygons would provide seemingly continuous objects with a coarse mesh. This section will describe a smoothing algorithm that has been used with the constraint-based god-object method.

In the best of worlds, we'd like to be able to model objects using curved surfaces, but these options are computationally intensive. [15] Non-Uniform Rational B-Splines (NURBS) surfaces would be a pretty obvious candidate due to their wide acceptance in the design and geometric modeling industries. Collision detection with such a surface is expensive. Because there is no simple transformation from global coordinates to surface coordinates, we have to iterate to find the point of the surface closest to our point in global coordinates.

Due to all of the difficulties involved with curved surfaces, the constraint based god-object method employs planar surfaces but interpolates surface normals between nodes. Section 2.1.2 explains that humans have a rather poor sense of position but are extremely sensitive to discontinuities of force direction. Our assumption is that if we can smooth the force normals, then the actual shape of the object is secondary.

In the real world the surface normal of the object is the derivative of the surface, necessitating a smooth surface to have a continuous surface normal. With virtual objects there is no such constraint; the surface normal can be decoupled from the shape of the object.

Our smoothing algorithm is very reminiscent of Phong shading used in graphics. [8] A surface normal is associated with each node in each polygon. The appropriate surface normal can then be computed for any location (point D) in the polygon by interpolating between the normals at its three nodes (A, B, and C). The interpolation can be done projecting the vector $\overline{AD}$ to the line $\overline{BC}$ to find the point E (Figure 4-5). The surface normal at point E can be found by adding a contributions from node B and C:

$$\hat{E} = \frac{|\overline{BE}|}{|\overline{BC}|}\hat{B} + \frac{|\overline{CE}|}{|\overline{BC}|}\hat{C} \tag{4.9}$$

The normal at point D can then be found by interpolating between the normals of points A and E:

$$\hat{D} = \frac{|\overline{AD}|}{|\overline{AE}|}\hat{A} + \frac{|\overline{DE}|}{|\overline{AE}|}\hat{E} \tag{4.10}$$

It should be noted that the interpolation scheme is moderately simple because the god-object has provided a location on the surface which we can compare to nodal locations. This algorithm smoothes objects rather well, but only works correctly for a modest range of angles. When the angle

between surfaces gets much less than 150 degrees, there is a noticeable change in displayed surface normal when the boundary is crossed. As was shown in Figure 3-5, there can be a discontinuity in the motion of the god-object if the angle between surfaces is large. The interpolation scheme relies on continuous motion of the god-object to generate a continuous surface normal. This limitation in the shading algorithm is not much of a drawback. Usually the models required for aesthetic graphics are refined enough to meet our haptic needs.

4.2.2 Stiffness and Viscosity/Impedance

Interpolation can be used to vary stiffness and damping in a continuous fashion.[2] Each node of an object can be assigned its own stiffness and damping. The impedance at any point on the object can be interpolated in exactly the same way the surface normals were.

One subtle thing we will miss with such a simple implementation of impedance blending is energy conservation. [22] It will be possible to press in at a location with a small stiffness, move laterally, and then let the object push you back out at a location where the stiffness is greater. We have allowed "passive" objects to do work on the user. The severity of this effect is directly related to the rate of change of stiffness. This phenomena should not make the device go unstable, so the only problem will be the conceptual understanding of the user. With friction, texture, and surface shading, it will be very difficult to evaluate the work done through a closed loop path. A more pure approach is currently being undertaken to assure that the net work around a loop will be zero. [22]

4.3 Surface Mapping Algorithms

For effects that vary greatly over a small range of motion, interpolation between nodal values is not sufficient. Texture is such an effect. Our haptic representation of texture can take much of its form from the graphics method of texture mapping. A texture map consists of a square data pattern described in texture coordinates (u,v). The process of texture mapping involves translating a position in surface coordinates (s,t) to texture coordinates (u,v) to find the value on the map which applies to the current location. The most difficult part is determining which locations on the texture map correspond to the nodes of the object. Though originally exploited for texture, this method is applicable to many effects.

The texture map mechanism could also be used to vary stiffness and damping. Using a texture map would give the object designer much more control at the cost of increased storage. If a model has a simple geometry (e.g. a cube) but a complex pattern of stiffness or viscosity, texture mapping these variables would save from having to refine the mesh. Most realistic objects will require a sufficiently fine mesh that texture maps for stiffness and viscosity should not be necessary. The impedance only specifies the position the user ends up for a specified force and the position sense of humans is rather poor.

4.3.1 Texture Concepts

Like in graphics, a haptic effect which does require this exactness is texture. As we've hinted at before (in Section 2.1.1) texture is really a phenomena which is sensed with the mechano-receptors

[2] Interpolation for non-linear stiffness equations should be possible assuming it can be made to stay stable.

in your finger tip; the human kinesthetic system can not perceive motions on the scale of texture. Realistic texture would require a tactile array display. Although our hardware doesn't include such a display, we can still give some sense of texture by other means.

To simulate texture without a tactile display, we need to stimulate the human tactile system in a secondary manner, through vibration. We can vary the force applied to user as function of position (or more specifically relative position to an object being stroked). The neurons we are trying to stimulate are perceptive to high frequencies (10-1000Hz); if the "period" of this variation is too large then it will be hard to perceive.

Extensive work on texture has been done with a two degree-of-freedom haptic interface including an attempt to categorize textures. [13] This work is important for understanding the types of texture that can be simulated, but it is not directly applicable to 3-dimensional objects. Using the graphics method bump mapping we take an inherently 3-dimensional approach.

4.3.2 Bump Mapping

In the graphics world, bump maps are used to correctly display the illumination of a bumpy surface. Since the color of a pixel is highly dependent on the orientation of the surface it is important to have the correct surface normal for each pixel. Bump maps are much like texture maps, but instead of associating a color with a location in (u,v) it associates a small displacement to be applied in the surface's normal direction.

These small displacements give rise to a new shape and this new shape means that the surface has new surface normals. A good approximation to the new normal is: [2]

$$\bar{N}_{new} = \bar{N} + \frac{B_u(\bar{N} \times \bar{P}_t) - B_v(\bar{N} \times \bar{P}_s)}{|\bar{N}|} \tag{4.11}$$

where $\bar{N}$ is the surface normal, B_u and B_v are the partial derivatives of the bump map with respect to the u and v directions, and $\bar{P}_s$ and $\bar{P}_t$ are the partial derivatives of the equation $\bar{P} = [x(s,t), y(s,t), z(s,t)]$ in the s and t directions. The new surface color can be computed using the lighting which would fall on a pixel with the newly computed surface normal.

This method works remarkably well for graphics applications, but haptic implementation is somewhat different. The slight variation of height due to the bump map might not be sensible by the user, but this information is directly available. The magnitude of the feedback force vector can be increased or decreased slightly depending on the entry in the bump map. The new normal direction can be used with the magnitude computed by the god-object algorithm as long as the variation in normal direction are small. If much more than ± 5 deg are used the system is likely to be unstable. A bump-map extension to the god object should be able to simulate materials like wood, sand paper, or rusted metal, but anything much more significant would probably have to be encoded into geometry.

This was implemented in a limited form for use with the god-object algorithm. Surfaces were covered by with a periodic texture. A 3x3 transformation matrix was computed based on surface position and applied to the normal force vector. We found this to be capable of reproducing sandpaper-like textures.

4.4 Conclusion/Summary

Many of the ideas explored in graphical rendering, including shading and texture mapping, are applicable to haptic rendering. There are some major differences between graphical methods and haptic methods. Graphical methods need to redraw a whole screen of pixels every 30-60 Hz while our haptic methods need to update the force on a single point at about 1kHz.

All of these effects require a surface location to generate their effect. In graphics computations, rays of light always interact with the surface of the object[3] so the surface location is always known. Haptic interfaces can penetrate into the volume of an object so a method is needed to determine its location on the object's surface. The constraint-based god-object approach satisfies this need.

[3] with the exception of translucent objects.

Chapter 5

Conclusion

This thesis describes the two major components required for a haptic display system: the haptic interface and a haptic rendering algorithm. The MIT-Toolhandle is large workspace haptic interface which uses the tool-handle approach to haptic rendering. The tool-handle approach simplifies the haptic rendering problem to simulating the interaction between a tool and a virtual environment.

The constraint-based god-object described is possibly the first attempt to haptically render arbitrary polygonal objects. Using a history of previous motion, the location of the god-object (a representation of the user in the virtual environment) can be found. Since the god-object does not penetrate objects there is no ambiguity in how the user is interacting with the virtual object. Also, because the god-object algorithm returns a position in the virtual environment rather than directly computing a force, the algorithm is very flexible. One of many methods can be used to determine the feedback force without having to change the god-object algorithm itself. The god-object algorithm is easily extensible to include surfaces effects.

Using our knowledge about human haptic perception we derived a number of surface effect algorithms from similar work done for graphical rendering. Surface shading makes polygonal object feel like smooth continuous objects, and texture mapping can make an object feel like a recognizable material. In addition, static and dynamic friction models have been implemented to allow manipulation of virtual objects. These models correctly simulate the vibration associated with objects slipping from one's grasp.

5.1 Performance

In its current implementation the constraint based god-object algorithm is written in C++ and compiled with Gnu's g++ under Linux. Running on a 66Mhz Pentium PC it is capable of rendering objects with up to about 1000 surfaces at a servo rate of about 1kHz. The algorithm is currently limited by the amount of collision detection required for an object of that size; collision detection now makes up about 90 percent of the computation taking place. With a state of the art collision detection scheme it is not unreasonable to expect an order of magnitude increase in the size of renderable objects.

5.2 Future Work

The haptic surface effects described in this thesis have been implemented in proof of concept demonstration, but have yet to be integrated together. This integration is not a trivial step because the effects are highly coupled in their output, the magnitude and direction of the feedback force. Each additional surface effect acts as a transformation on the feedback force. The challenge will be to determine in what order these transformation should be applied and which components of the force vector should be transformed.

In addition, the god-object method currently only handles rigid objects. Many objects with which we interact on a daily basis are compliant. It was hinted at in Section 3.2.2 that we could update the positions of the nodes and surfaces based on forces from the user. The computation involved in updating the data structures required by the god-object algorithm would not be prohibitive, but computing realistic motion of the nodes is likely to be computationally intensive. Having an underlying finite element model could enable correct calculation of these deflections, but not with current hardware of reasonable cost. FEM solutions are probably more accurate than is necessary; it may be that other algorithms will provide the necessary balance between fidelity and efficiency.

Bibliography

[1] Haruhiko Asada and Jean-Jacques E. Slotine. *Robot Analysis and Control.* John Wiley and Sons, 1985.

[2] J. F. Blinn. Simulation of wrinkled surfaces. In *SIGGRAPH*, pages 286–292, 1978.

[3] Fredrick P. Brooks, Ming Ouh-Young, James J. Batter, and P. Jerome Kilpatrick. Project GROPE - haptic displays for scientific visualization. In *Computer Graphics*, volume 24-4, pages 177–185, Dallas, Texas, August 1990. ACM.

[4] G. Burdia, J. Zhuang, E. Roskos, D. Silver, and N. Langrana. A portable dextrous master with force feedback. *Presence*, 1(1):18–28, 1992.

[5] J. Edward Colgate and J. Michael Brown. Factors affecting the Z-range of a haptic display. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3205–3210, 1994.

[6] Nat Durlach et al., editors. *Virtual Reality: Scientific and Technological Challenges*, chapter 4: Haptic Interfaces. National Academy of Sciences, Washington D.C., December 1994. Report produced for the National Research Council.

[7] Paul Dworkin and David Zeltzer. A new model for efficient dynamic simulation. In *Proceedings Fourth Eurographics Workshop on Animation and Simulation*, pages 135–147, 1993.

[8] James Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics: Principles and Practice.* Addison Wesley, 1992.

[9] Tetsuo Kotoku, Kiyoshi Komoriya, and Kazuo Tanie. A robot simulator with force generating function - configuration space approach. *SICE*, 1992.

[10] Ming C. Lin, D. Manocha, and John Canny. Fast collision detection between geometric models. *Tech Report TR93-004, Department of Computer Science, University of North Carolina*, 1993.

[11] Thomas Massie. Design of a force reflecting fingertip stimulator. Bachelor's thesis, Department of Electrical Engineering and Computer Science, MIT, May 1993.

[12] Thomas Massie and J. Kenneth Salisbury. The PHANToM haptic interface: A device for probing virtual objects. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Chicago, IL, November 1994. Proceedings of the ASME Winter Annual Meeting.

[13] Margaret Minsky. *Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display.* PhD thesis, Media Lab, MIT, June 1995.

[14] Margaret Minsky, Ming Ouh-Young, Oliver Steele, Fredrick P. Brooks, and Max Behensky. Feeling and seeing: Issues in force display. In *Symposium on Interactive 3D Graphics*, volume 24-2, pages 235–270. ACM SIGGRAPH, March 1990.

[15] Michael E. Mortenson. *Geometric Modeling.* John Wiley and Sons, 1985.

[16] A. Michael Noll. Man-machine tactile communication. *Society for Information Display Journal*, July/August 1972. reprinted in the Jul/Aug 1978 issue of Creative Computing.

[17] S. E. Salcudean and T. D. Vlaar. On the emulation of stiff walls and static friction with a magnetically levitated input/output device. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 303–309. Proceedings of the ASME Winter Annual Meeting, 1994.

[18] J. Kenneth Salisbury, David Brock, Thomas Massie, Nitish Swarup, and Craig Zilles. Haptic rendering: Programming touch interaction with virtual objects. In *Proceedings of the ACM 1995 Symposium on Interactive 3D Graphics*, Monterey CA, April 1995.

[19] K. B. Shimoga. A survey of perceptual feedback issues in dexterous telemanipulation: Part ii. finger touch feedback. In *Proceedings of VRAIS*, pages 271–279, Seattle, WA, 1993.

[20] Mandayam A. Srinivasan. Virtual haptic environments: Facts behind the fiction. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, June 1994.

[21] Mandayam A. Srinivasan, David L. Brock, G. Lee Beauregard, and Hugh B. Morgenbesser. Visual-haptic illusions in the perception of stiffness of virtual haptic objects. Manuscript in preparation, 1995.

[22] Nitish Swarup. work in progess. Master's thesis, Department of Mechanical Engineering, MIT, Expected August 1995.

[23] Hong Tan, X. D. Pang, and Nat Durlach. Manual resolution of length, force, and compliance. In *Advances in Robotics*, pages 13–18. Proceedings of the ASME Winter Annual Meeting, 1992.

[24] Hong Tan, Mandayam Srinivasan, Brian Eberman, and B. Chang. Human factors for the design of force-reflecting haptic interfaces. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. Chicago, IL, November 1994. Proceedings of the ASME Winter Annual Meeting.

[25] Craig Zilles and J. Kenneth Salisbury. A constraint-based god-object method for haptic display. In *Proceedings of the IROS conference on Robotics.* IROS, 1995.