Haptic Interaction with Deformable Objects Using Real-Time Dynamic Simulation

by

Nitish Swarup

B.S.M.E., Massachusetts Institute of Technology (1993)
B.S.E.E., Massachusetts Institute of Technology (1995)

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

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

September 1995

Author . . . .
Department of Mechanical Engineering
August 11, 1995

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

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

Haptic Interaction with Deformable Objects Using Real-Time Dynamic Simulation

by

Nitish Swarup

Submitted to the Department of Mechanical Engineering
on August 11, 1995, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This thesis discusses the design and implementation of an interactive deformable media simulation (IDMS) that provides real-time visual and force feedback by means of a graphics display and haptic interface. A finite element modeling approach is used to represent structural and viscoelastic properties and simulate dynamic behavior. Media is represented as a discrete network of masses and constitutive mechanical elements, such as springs and dashpots. Real-time simulation is achieved by (a) using a surface representation coupled with a surface restoring force to encapsulate volumetric properties; (b) conserving computation power by rendering media local to the interaction point; and (c) evaluating the simulation within the topology of the model. Haptic interaction forces that satisfy causality and passivity are generated using simple collision detection techniques based on point intersections. Results of fundamental interaction tasks including palpation, stroking, and plucking are discussed within the IDMS framework and are shown to be highly representable of real interactions.

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

Haptic Interaction with Deformable Objects Using Real-Time Dynamic Simulation

by

Nitish Swarup

Submitted to the Department of Mechanical Engineering
on August 11, 1995, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This thesis discusses the design and implementation of an interactive deformable media simulation (IDMS) that provides real-time visual and force feedback by means of a graphics display and haptic interface. A finite element modeling approach is used to represent structural and viscoelastic properties and simulate dynamic behavior. Media is represented as a discrete network of masses and constitutive mechanical elements, such as springs and dashpots. Real-time simulation is achieved by (a) using a surface representation coupled with a surface restoring force to encapsulate volumetric properties; (b) conserving computation power by rendering media local to the interaction point; and (c) evaluating the simulation within the topology of the model. Haptic interaction forces that satisfy causality and passivity are generated using simple collision detection techniques based on point intersections. Results of fundamental interaction tasks including palpation, stroking, and plucking are discussed within the IDMS framework and are shown to be highly representable of real interactions.

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

Acknowledgments

The MIT Artificial Intelligence Laboratory has been a dynamic research environment to mature and partake in. I am indebted to all those who made my experience fluid and intellectually stimulating. In particular, I wish to thank my fellow group colleagues: Catherine Anderson, Brian Anthony, David Brock, Brian Eberman, Jesse Hong, Craig Latimer, Akhil Madhani, Thomas Massie, Thomas Moyer, John Morrell, Gunter Niemeyer, and Craig Zilles.

To my advisor Ken, for being an inspiring mentor and friend.

David Rahn provided substantial support in developing network and graphics software for IDMS. Derek Schulte designed and built the instrumented tweezers. Thomas Massie designed the Phantom haptic interface.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. This material is based upon work supported under a National Science Foundation Graduate Fellowship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

> *"Jennifer Roberts, the mother, is training to become a surgeon and is at her [Virtual Environment] station studying past heart operations.*
>
> *She previously spent many hours familiarizing herself with the structure and function of the heart by working with the virtual-heart system she acquired after deciding to return to medical school and to specialize in heart surgery. This system includes a special virtual-heart computer program obtained from the National Medical library of physical/Computational Models of Human Body systems and a special haptic interface that enables her to interact manually with the virtual heart. Special scientific visualization subroutines enable her to see, hear, and feel the heart (and its various component subsystems) from various vantage points and at various scales. Also, the haptic interface which includes a special suite of surgical tool handles for use in surgical simulation analogous to the force-feedback controls used in advanced simulations of flying or driving), enables her to practice various types of surgical operations on the heart. As part of this practice, she sometimes deliberately deviates from the recommended surgical procedures in order to observe the effects of such deviations. . . ."*[5, p25]

Virtual environments, as exemplified in the vision above, are systems that integrate computers, man-machine interfaces, and human operators to synthesize perceptual immersion into artificial reality. These systems have the potential to allow humans to create design prototypes quickly and inexpensively; to train in complex, hazardous environments; to operate on remote tasks through transmission of sensorimotor signals; and create artificial environments that are physically unrealizable. Clearly, the environment being modeled must be accurately represented in order to be perceptually acceptable as a substitute for reality. Furthermore, understanding of the human psychophysical system is necessary since virtual environments transfer information to and from the human operator through sensorimotor signals. These requirements impose strict constraints on the bandwidth, fidelity, and non-obtrusiveness of the virtual environment. With continuously increasing growth in computational speed and hardware, however, virtual environments are becoming feasible as powerful tools to assist humans.

An important modality of virtual environments that has received little attention relative to visual and auditory forms of feedback is *haptics*, or the touch channel. Haptics is unique from vision and audition because both sensing and actuation occur on the environment. The ability to haptically explore *and* manipulate objects greatly enhances the human's sense of immersion in a virtual environment [5]. A synergistic effect occurs when visual, auditory, and haptic cues are combined and presented to the user. This coupling between different sensorimotor channels is meaningfully exploited in software to facilitate and enhance representation of multi-modal scenes.

Advances in haptic interaction have been primarily limited by the performance of haptic interfaces. Haptic interfaces typically are electromechanical devices that interface with the user's hand or finger and transmit position and force information bidirectionally between the human and the virtual environment. Programming the force behavior of the haptic interface as a function of position is equivalent to controlling the mechanical impedance felt by the user [8]. Design constraints on haptic interfaces are motivated to achieve psychophysical transparency between the desired programmed behavior of the virtual environment and the human mechano-sensorimotor system. Inertia and friction, for example, in a haptic interface only serve to corrupt the transmission of sensorimotor signals and reduce fidelity. The device used in this thesis, the PHANToM haptic interface [11], is a high-performance ground-based haptic interface with three active translational and three passive rotational degrees of freedom. The PHANToM uses a lightweight cable-driven parallel linkage to transmit haptic signals to and from the human fingertip (Section 5.1.2). Because of the high bandwidth and fidelity exhibited by the device, the Phantom succeeds in satisfying the transparency requirement.

In addition to hardware, an intrinsic part of a modeled virtual environment is the software representation and simulation of objects that comprise it. Three major components of a dynamic environment simulation are listed below.

1. *Dynamics simulation.*
2. *Collision detection.*
3. *Contact force modeling.*

Haptic display is primarily concerned with generation of interaction forces (item 3 above). Because visual and haptic display of object characteristics share similar attributes, some mathematical techniques from the established field of computer graphics [6], for example collision detection (item 2), can be borrowed to provide insight into methods for haptic display. Although parallels between visual and haptic simulation exist , the high bandwidth of the human haptic system requires a considerably faster display of information, particularly at the local areas of contact [5]. Furthermore, many significant mechanical attributes such as surface friction, texture, and impedance are not appropriately addressed in the framework of graphics. Thus, haptic display necessitates a fundamentally separate perspective for simulation design.

A coherent framework for haptically representing shape, surface properties, and bulk properties of rigid objects is discussed in [21]. Interaction force algorithms have been developed for rigid polyhedra that are based on static geometry constraints. The perception of a rigid surface is conveyed if the programmable stiffness is made high relative to the human's proprioceptive sense for discerning stiffness. A point-based paradigm is described in [26, 27] which computes interaction forces based on Hooke's law when virtual contact is made. Another constraint-based display system utilizes edge contact models in addition to vertex constraints to generate interaction forces [9, 10].

A smaller subclass of objects in our environment are better classified as deformable media. Examples include rubber, foam, and biological tissue. Deformable objects are computationally more complex to simulate due to the dynamic behavior they exhibit in their local reference frames. Haptic interaction is simplified in rigid object simulations because, apart from rigid body dynamics, items 1 and 3 above are decoupled. In other words, assuming all objects are grounded in the environment, interaction forces with a grounded rigid object do not *affect* the state of that object. Simulation of deformable objects, conversely, requires tight computational coupling of items 1 and 3 in order to display accurate forces to the user. The deformation force and geometry of a model is determined by a *combination* of interaction modeling and calculation of internal reaction forces.

This thesis describes a prototype virtual environment system, *IDMS (Interactive Deformable Media Simulator)*, for haptically and visually interacting with deformable objects in real-time. A discrete simulation model is used to represent media by a network of energy storage and dissipative elements connected to point masses, or atoms. Volumetric properties are achieved by (1) using a two-layer thick slab filled with tetrahedral elements to represent the surface of the object and (2) grounding atoms on the external surface of the slab to their respective equilibrium positions. Real-time interaction is achieved by assuming local deformation near the point of interaction; computation power is conserved by only simulating those tetrahedral elements that lie within a bounding sphere centered at the interaction point. Models are stored using `plg` and `fem` file formats (Appendix B) to visually and haptically describe object properties, respectively. Generation of models is achieved by modifying preexisting models through either programmed or manually controlled forces. The following section discusses the motivation behind developing IDMS.

1.2 Motivation

Integrated haptic and visual simulation of deformable media has potential applications in a broad range of fields. Applications in two primary fields are briefly addressed below.

- *Medicine*
  Virtual environments will play an important role as high-performance training tools in medicine. Medical students will be able to learn about the human anatomy interactively by probing sophisticated models ad infinitum, without accessing human cadavers or invading the privacy of live human models. By equipping a haptic interface with an application specific tool such as a scalpel, students will be able to practice surgical procedures, which require a large degree of hand-eye coordination, on virtual models. Modeling interaction with biological tissue in these procedures requires real-time simulation of deformable media.

- *Computer-Aided Design*
  Current state-of-the-art computer-aided design (CAD) tools lack the ability to truly interact with three-dimensional objects. Three-dimensional user input is transmitted

through two-dimensional projections of various views and a non-force-reflecting stylus. The ability to both explore and manipulate with three-dimensional force-reflecting haptic interfaces will allows designers to utilize their creativity much more effectively. Engineers could easily mold virtual clay to construct prototypes and proof their ideas before deciding on a design. The ability to interact with tangible three-dimensional objects will give engineers a heightened sense of confidence before spending precious money on tooling and manufacturing. The existing CAD primitives of cutting, copying, and pasting combined with the haptic channel of interaction will provide engineers with powerful creative tools.

1.3 Organization

- Chapter 2 discusses the domain of deformable media that is addressed by this thesis. Requirements for a real-time interactive deformable media simulator are described. Lastly, a review of literature related to deformable object simulation is provided.

- Chapter 3 describes and justifies the finite element modeling approach used to capture the significant physics of deformable media. Several features of the approach are described in correlation with the requirements of Chapter 1.

- Chapter 4 describes the mathematical framework used for updating the state of the model dynamics. Stability is discussed in terms of the model physical parameters and the simulation time step.

- Chapter 5 discusses the hardware and software architecture that comprise the implementation of IDMS. The overall system loop that concurrently executes haptic, environment, and visual processing is also described. Algorithms for generating interaction force and simulating local dynamics are presented. Lastly, techniques for automatically and manually generating models are discussed.

- Chapter 6 presents results from fundamental haptic interactions with IDMS. Visual and haptic information is presented for each interaction to illustrate the versatility and functionality inherent to IDMS.

- Chapter 7 presents a summary of the features of IDMS and offers conclusions for further research that will enhance the performance and functionality of IDMS.

Chapter 2

Problem Domain

2.1 Scope

Within the perceptible range of the human proprioceptive system, deformability qualitatively encompasses a vast spectrum of objects ranging from highly compliant materials such as soft foam to relatively stiff media such as highly viscous gels. Somewhere along this deformability spectrum lies a band of objects that is meaningful and feasible for real-time dynamic simulation in a virtual environment. As described in the Introduction, a major application area of this research is surgery simulation, which requires realistic modeling and simulation of biological media. Without loss of generality, biological media provide a meaningful and useful context for addressing the issues inherent in designing an interactive deformable media simulation. A broader, perhaps unrelated, range of deformable objects can be simulated using this context as a coherent perspective. The following assumptions are made about the material properties of biological media to assist in the motivation of design requirements for such a system and incorporation into a virtual environment setting.

1. *Local influence*
   Many interactive medical procedures such as suturing, incising, palpating, dissecting, and retraction primarily have local yet complex influence on the biological object. The forces induced by these interactions, however, do not substantially propagate to affect the *global* state of the object. This result is fundamentally due to the highly viscoelastic, overdamped behavior of tissue and the high mechanical coupling with the skeleton through connective tissue in the surrounding environment.

2. *Nonhomogeneity*
   Biological media are characterized by a nonhomogeneous composition of materials with potentially discontinuous variations in mechanical impedances. Some forms of tumorous tissue, for example, exhibit much stiffer mechanical properties than healthy tissue. Underlying bone is perhaps better characterized by a rigid representation than a deformable one. Furthermore, surface properties such as friction may vary considerably depending on the tissue type.

2.2 Requirements

This section focuses on the design requirements of an interactive deformable media simulation environment given the context described above.

2.2.1 Real-time interactivity

Real-time response in a virtual environment is derived by the bounds on perceptual immersion defined by the human sensorimotor system. Each sensorimotor modality of the human has information refresh rates that need to be satisfied or else the perceptual illusion of reality will not be conveyed. The lower bound for visual modality requires a throughput rate greater than 8 to 10 frames per second and a maximum delay of 0.1 sec [5, p.250]. These values are primarily for static environments with slowly moving objects. Environments which contain higher frequency motion require frame update rates on the order of 60 Hz to achieve visual realism. The haptic modality requires considerably faster update rates to establish sufficient performance. Specifically, the bandwidth of tactile and force feedback within the virtual environment should exceed 1 kHz [5, p.184].

2.2.2 Passivity

An important but often overlooked property desirable of an interactive simulation is passivity of work interaction. Passivity implies that no net positive work transfer occurs from hardware or software elements that lie *between* the simulation and the human during an interaction. A passive object in the virtual environment should not appear active to the human. The environment dynamics may or may not be passive depending on the nature of the environment; an example of an active environment is muscle tissue in which energy is being produced internal to the environment. Although strict passivity is perhaps too conservative a requirement [2, 3] for haptic interaction, it is a useful robustness criteria and measure of haptic display performance.

2.2.3 Nonhomogeneity

As exemplified above in 2.1 for the case of biological media, nonhomogeneous material properties such as density, stiffness, and viscosity need to be sufficiently captured by the modeling representation in order to accurately convey changes in mechanical impedance felt by the haptic interface. Humans rely on the perceived continuity of change in force to discriminate between internal boundaries of different media. For example, doctors learn to detect tumors by palpating and stroking tissue, relying heavily on perceptual changes in stiffness to identify their presence.

2.2.4 Reformability

As mentioned above in the context of biological media, surgical procedures involve cutting, removing, and joining. Algorithmically, the simulation should provide functionality for performing such operations on the virtual model. More importantly, the model representation should be flexible enough to allow for such real-time alteration of object characteristics using the haptic and visual channels provided.

2.2.5 Data file format

A file format for coherently storing, reading, and writing model data by the simulation is an important design requirement. A predefined standard for storing physical parameters and geometry allows for a virtually infinite set of models to be defined easily. The use of a file format also allows for creation of models by other software tools and conversion from other model definition protocols used by other simulations. Libraries of models can be created to represent systems that are better represented as a hierarchical collection of independent objects, an obvious example being the human body.

2.2.6 Haptic scanning

In the context of visual images, scanning converts the real visual properties of an image into a representable data format for later processing. Analogously, the notion of *haptic scanning* is introduced. A real object would be haptically scanned for geometric, mechanical, and material properties to produce an artificial copy that would serve as a haptic representation for that object. This process would allow virtual models to be easily generated from existing real objects. Although the mechanics of haptic scanning are perhaps not appropriately addressed by the simulation itself, the modeling approach used in representing objects for the simulation should be rich enough and amenable to incorporation of data from haptic scanning.

2.3 Previous approaches

This section describes previous research relevant to deformable object simulation.

Barr [1] describes an approach that permits global and local deformations of solid geometries. Tangent and normal vectors of the deformed model geometry are achieved by applying a transformation matrix to the tangent and normal vectors of the undeformed geometry. A series of deformation operations (including tapering, twisting, bending, and scaling) can be performed on undeformed solid primitives by concatenating transformation matrices in a hierarchical fashion. These methods, however, are not based on underlying physical models and hence do not model nonhomogeneous physical behavior or allow for generation of haptic interaction forces.

Terzopoulos [23] formulates a method for modeling deformable objects using elasticity theory. Partial differential equations are derived using classical mechanics and solved using finite element modeling techniques. External forces are added to represent gravity, fluid forces, and collisions with rigid objects. A similar framework is used by Terzopoulos [24] to address the inelastic deformation behaviors of viscoelasticity, plasticity, and fracture. The resulting simulation of object dynamics are realistic and accurately represent natural physical behavior. The finite element calculations used in each of these methods, however, are prohibitive for real-time simulation and thus do not satisfy the interactivity requirement.

Pentland [16, 17] presents a visually real-time (15 Hz) solid modeling system that allows users to mold “clay” by applying external virtual forces. A family of superquadrics are used as the basic volume primitives for describing the object geometry. The finite element method is utilized to derive the matrix differential equations describing the dynamic behavior of each object. Computational expense is minimized by using a modal method to only simulate the precomputed low-frequency modes of the system. The approach, however, does not address generation of haptic interaction forces nor real-time simulation at haptic refresh rates (≈ 1 kHz). Furthermore, interactive reformability does not fit within the modal-based framework since modification of object structural properties would require expensive recomputation of the modal vectors.

Pieper [18] describes a system for animating human facial tissue using a finite element approach to globally model tissue behavior. The structural and geometric topology is represented using a discrete simulation model, a network of discrete masses and mechanical elements stored in graph form. This representation, similar to the one used in this thesis, is appropriate for describing nonhomogeneous physical parameters and nonlinear mechanical relationships. The graph method is also highly amenable to reformability, facilitating addition and removal of nodes and links during run-time. Pieper's approach for modeling facial tissue, however, is not appropriate for modeling volumetric behavior of deformable objects. Furthermore, haptic interaction issues are not addressed. Pieper [19] also discusses a system for simulating plastic surgery that is based on finite element modeling. A global stiffness matrix is utilized for solving the differential equations. This method, however, is aimed at task-level analysis and similarly does not address display of information through the haptic channel.

Cover [4] presents a methodology for interactive deformation of surfaces by using the concept of active contours, essentially energy-minimizing splines. The surface is discretized as a mesh of points and spring-like connections, each point containing information about its direct neighbors and the external force being applied to the point. A "home" force is also applied that acts to restore each point to its equilibrium position. A gradient descent approach is used to update the point positions in the mesh in an energy-minimizing manner. The internal energy of the contour, however, is globally controlled by only two parameters that affect elasticity and flexibility. Hence, local variation in physical characteristics are not captured by the active contour. Furthermore, it is not clear whether this approach is amenable to easy incorporation of haptic scan data. Lastly, real-time interactivity is not addressed in the framework of haptics.

Each of these methods are missing fundamental features that are necessary to satisfy the requirements for true simultaneous haptic *and* visual interaction with deformable media. The following chapter discusses the modeling approach used in IDMS and appeals to the requirements listed above.

Chapter 3

Physical Modeling

Modeling of deformable objects is indeed a nontrivial task. Unlike rigid objects, deformable media have significant internal dynamics and thus a state-based approach *must* be used to solve for interaction forces. A physically-based approach grounded in applied mechanics ensures that the model will capture the real world idiosyncrasies of deformable media. A molecular level model, however, is clearly not necessary for a virtual environment setting in which perceptual and cognitive abilities limit the range of information that can sensed. Thus, a viable model must capture sufficient properties to prove perceptually realistic while maintaining enough simplicity to allow for real-time simulation. The following chapter describes the fundamentals behind the modeling approach used in IDMS and the features that make the approach highly amenable for haptic interaction.

3.1 Approach

The modeling approach used in IDMS utilizes a combination of finite element modeling (FEM) theory and surface encapsulation of volumetric properties to convincingly convey the feeling of deformation to the user. In mathematics, a modal decomposition separates a dynamic system into a linear superposition of normal eigenvectors. Carrying this modal decomposition metaphor over to physical modeling, the haptic model only needs to represent the most significant modes of deformation to the user. The model is comprised of a two-dimensional layer of finite tetrahedral structural elements, networked together to form the surface of the deformable model (see Figure 3-1). Tetrahedral elements (dashed line) are used because they are structurally stable and have low mass density, requiring minimal discrete elements per unit volume to represent them. The encapsulation of volumetric effects is achieved by adding a restoring force between the top surface (bold line) of the tetrahedral mesh and the equilibrium position of the top surface. The user only interacts with the top surface of the mesh, both visually and haptically. The notion of a *haptic window* is introduced to allow simulation of object dynamics *local* to the interaction point. Only media that falls within this geometrically shaped window is simulated in order to generate real-time response. Combined, these features serve to haptically and visually display the significant modes of deformation to the user.

A discrete simulation model (DSM) is chosen to represent the surface mesh described

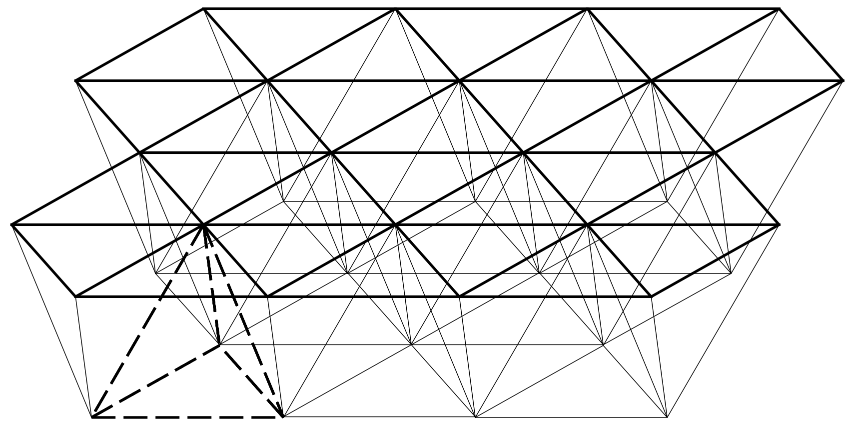

Figure 3-1: A fragment of media represented by a two-dimensional layer of tetrahedral elements (dashed line). Volumetric effects are simulated by applying a restoring force between the top surface (bold line) and the equilibrium position of the top surface.

above. A DSM is a discretized network of lumped parameter elements and atoms that approximates the physical behavior of a continuum media. In the present context of deformable objects, atoms are point masses connected to neighboring atoms through constitutive mechanical elements. Each tetrahedral element is represented by four link elements and four bounding atoms, shared with neighboring tetrahedra. The DSM is stored in graph form to facilitate alteration of the model topology. Simulation of the DSM state is accomplished by a multi-step algorithm that is evaluated *within* the topology of the model. During each time step, forces are summed for each mobile atom based on external forces and internal forces from neighboring elements. The two state variables of position and velocity for each atom are then numerically integrated resulting in the DSM state for the next time step.

The restoring force between the top layer of the mesh and the equilibrium position is achieved within the framework of the DSM. Essentially, a massless immobile *home* atom is added at the *equilibrium* position of each atom located on the surface of the mesh. A connecting *home* element is added between the surface atom and its respective home atom. As the virtual mesh is deformed interactively, the superposition of forces produced by home elements in tension compactly simulates the volumetric compression of the continuum media. Figure 3-2 displays the equilibrium and deformation states of a media fragment, explicitly showing the home elements and atoms for the surface atoms that are significantly deformed[1].

The following section describes the features of the modeling approach mentioned above in light of the requirements of an interactive deformable object simulation described in Chapter 2.

[1]Note that during a normal IDMS session, only surface elements and surface atoms are visually rendered. Home elements, home atoms, and interior atoms and elements are visually hidden.

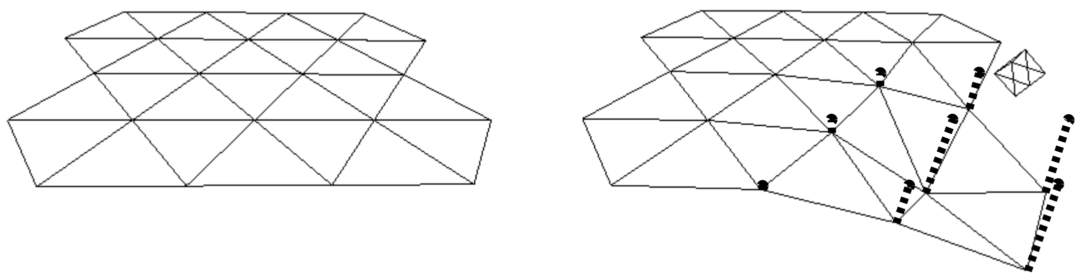

Figure 3-2: A fragment of dsm media in equilibrium is shown on the left. A sample deformation and the home atoms and elements are explicitly shown at right.

3.2 Features

3.2.1 Surface representation

Assuming small deformations, haptic interaction can be viewed as primarily a surface phenomena. Although the objects being modeled are clearly three-dimensional volumes, the interactions being modeled can be approximated as occuring near the object surface. As discussed below, a surface representation of haptic properties is spatially and computationally efficient compared to volumetric models of object physics.

Spatial properties

From a spatial perspective, a surface representation is highly desirable. The amount of geometric and haptic data required to describe the object scales in proportion to the surface area instead of the object volume. Furthermore, the process of haptically scanning in real object properties will likely be implemented by measuring surface impedance and material properties through the use of an instrumented force-measurement device. Mathematically speaking, this procedure can be described as projecting multivariable data captured from a measurement input space onto the geometric output space. Given that measurement data will spatially cluster about the surface of the object, the use of a surface model for representing haptic properties facilitates the data projection and registration process. Conversely, use of a volumetric model would imply projection of the measurement space onto a *higher* dimensional space, requiring the use of interpolation or other data-insertion techniques to fill the output space.

The home element described above is a discretized encapsulation of the local surface impedance and volumetric properties in the modeled continuum object. In the current generation of IDMS, each home element is a second order representation of the surface impedance, storing elastic and viscous properties. The spatial density of home elements is governed by the sampling size of tetrahedral elements used. Borrowing principles from discrete-time signal processing [15], a larger resolution of tetrahedra will capture higher spatial frequency content of the continuum media. A larger resolution, however, increases data storage and imposes a greater computational load on the haptic rendering algorithm for displaying the data.

Computational properties

A surface representation allows for computationally tractable rendering of virtual objects. Real-time simulation of objects using volumetric finite elements throughout the object are still intractable even on supercomputers. A surface model reduces the computational complexity from requiring $O(n^3)$ operations for a volume representation to only $O(n^2)$ operations for a surface representation. Given this reduction, desktop workstations can suffice as real-time simulation engines.

3.2.2 Local simulation

Current haptics technology primarily involves environment interaction with fingers or tools that have localized mechanical effect. Just as a graphics simulation only renders those objects that fall within the viewing window of the monitor, a haptics simulation need only render those effects that are spatially local to the interaction point to minimize computation effort. This haptic "window" is not as spatially rigid as the rectangular viewing space of a traditional graphics display because physical elements throughout the model contribute some work transfer to the user. Thus, the haptic window must selectively include enough elements to the point of diminishing marginal return for computational cost versus perceptual realism.

A DSM facilitates local simulation of dynamics. As explained in more detail in Section 5.5.2, local simulation is achieved by only simulating media that lies within a sphere centered at the interaction point; the rest of the model is immobilized and hence not unnecessarily simulated during each time step. The size of this sphere is empirically chosen such that stroking along the surface of the virtual object is haptically smooth. Although not implemented in the current generation of IDMS, a slower background process could be added to update global dynamics. This parallelism of different time scale processes is amenable to the concurrent run-time model used in the software implementation of IDMS (Section 5.2).

3.2.3 Nonhomogeneity

Nonhomogeneity of model physical characteristics are compactly represented in the DSM framework. Variations in density can be achieved varying atom mass values appropriately. Nonhomogeneous stiffness and viscosity is similarly achieved by varying the physical parameters of DSM elements. Furthermore, the DSM framework conveniently allows for regions of higher resolution meshing of tetrahedral elements, especially where surface properties contain higher spatial frequency.

3.2.4 Nonlinearity

In addition to nonhomogeneity, many materials exhibit nonlinear mechanical behavior. Using a solid mechanics model, the regime below the yield stress of a material can be represented by linear relations; above this point, the material becomes plastic and nonlinear stress-strain behavior results. Given the graph form (Section 3.3) used to store the DSM, each element may contain a nonlinear constitutive relation $\mathbf{F}(l, \dot{l})$ to produce an internal force. Additional state variables may be added per element as necessary for other force generating relations. For simplicity of computation and modeling, however, all the models used in this thesis use linear relations.

3.2.5 Boundary conditions

Boundary conditions can be easily applied by immobilizing atoms. Each atom data structure contains a boolean specifying the boundary condition, fixed or free, of the atom (see Section 3.3). The DSM simulation algorithm saves computation power by only updating the state of those atoms that have a free boundary condition. Also, element forces are generated only for elements that are bounded by at least one mobile atom.

3.2.6 Reformability

The graph-based method (Section 3.3) and data structures used to represent the DSM greatly facilitate appending and removing atoms and elements from the model. Different interaction procedures can be implemented by producing different effects on the topology of the graph. For example, removal of material can be achieved by deleting atoms and elements from the graph. Slicing, a mass-conserving procedure, can be accomplished by removing elements along the cutting path. Gluing and joining multiple models together is possible by adding elements between existing atoms of the models.

3.2.7 Parallel processing

The DSM approach scales well with increased processing power, both in single and multi-processor computer architectures. More processing power translates into a greater number of atoms and elements that can be simulated at required haptic servo rates, allowing rendering of more complex models. In a multi-processing system, each processor can be assigned a region of the model to simulate. Run-time model data would be stored in central, shared memory to facilitate data coherency and access between processors.

3.2.8 Data file format

The `fem` and `plg`[2] file formats described in Appendix B are the off-line storage protocols for representation of model data used in IDMS. All initial conditions, physical parameters, and topology information used in constructing the data structures and connectivity are completely contained in these files.

[2] The `plg` file format is a subset of the more descriptive Wavefront Technologies `obj` file format, which will be integrated into later iterations of IDMS

3.3 Data representation

Storing the DSM data in graph form allows for flexibility and efficiency in computation. Each node in the graph represents an atom and each link represents an element. In conventional FEM techniques, matrices are used to store state and material property as the simulation progresses. Often these matrices are sparsely filled, thus occupying an unnecessary amount of memory. For a model with N atoms and two state variables per atom, use of matrix storage amounts to $O(4N^2)$ storage. Graph storage, conversely, grows linearly with N and is only $O(N)$. Graph storage is highly amenable to the design requirement of reformability. As material is removed, added, or joined during the simulation, nodes and links in the graph can be correspondingly manipulated. Matrix-based methods are essentially limited to static topologies, because appending and deletion are algorithmically difficult and time-consuming.

In addition to the spatial advantages of linear storage and malleability, graph methods are time efficient relative to matrix methods. Complex computationally-intensive numerical techniques are necessary to invert matrices. Assuming the best case scenario for matrix methods, a matrix multiplied by a state vector must occur during every simulation loop. This operation entails $O(8N^3)$ operations. Conversely, solving for the state using graph methods only requires $O(N+L)$ operations, where L is the number of elements in the model.

The following sections describe the data structures that comprise a DSM.

3.3.1 Atom

The data structure, $\mathcal{A}$, for the atom is shown in Table 3.1[3]. The variable num contains the atom's number in the list of atoms for the parent model, $\mathcal{M}$. A list of neighboring element structure addresses is maintained to facilitate in calculation of external forces on the atom. The surface condition sc holds a boolean determining whether the atom is located on the exterior of the model (external), hence visually and haptically sensable, or on the interior (internal). The boundary condition bc is also a boolean determining whether a position constraint is being imposed (fixed) or not (free). Finally, the spatial condition spc specifies if the atom is inside or outside the haptic window.

3.3.2 Element

The element data structure is shown in Table 3.2. The num contains the element's number in the list of elements under the parent model, $\mathcal{M}$. The physical parameters for the element are stored in variables k, b, and l_{eq}. Structure addresses $\&\mathcal{A}^0$ and $\&\mathcal{A}^1$ of the element's two bounding atoms allow the state variables $\dot{l}$ and l to be calculated. The state variables combined with the physical parameters are used to evaluate the internal element force $\mathbf{f}$ during each simulation loop.

3.3.3 Model

The model data structure, $\mathcal{M}$, is the root node of the DSM graph and stores individual lists of the N atoms and L elements that comprise it (Table 3.3). As a preprocessing step, a list

[3]ANSI C notation is used where appropriate for addressing.

$\mathcal{A}$	
num	atom number
m	mass
$\mathbf{p}$	absolute position
$\mathbf{p}_{eq}$	absolute equilibrium position
$\mathbf{x_1}$	differential position
$\mathbf{x_2}$	differential velocity
$\dot{\mathbf{x}}_\mathbf{1}$	time derivative of $\mathbf{x_1}$
$\dot{\mathbf{x}}_\mathbf{2}$	time derivative of $\mathbf{x_2}$
$\mathbf{f}$	net force applied on atom
$\&\mathcal{E}^0, \&\mathcal{E}^1, \ldots, \&\mathcal{E}^{k-1}$	list of neighboring elements
bc	boundary condition (fixed or free)
sc	surface condition (external or internal)
spc	spatial condition (local or non-local)

Table 3.1: Data structure for atom.

$\mathcal{E}$	
num	element number
k	stiffness
b	viscosity
l	length
$\dot{l}$	time derivative of length
l_{eq}	equilibrium length
$\mathbf{f}$	net internal force
$\&\mathcal{A}^0, \&\mathcal{A}^1$	list of neighboring atoms

Table 3.2: Data structure for element.

$\mathcal{M}$	
`name`	model name
$\&\mathcal{A}^0, \&\mathcal{A}^1, \ldots, \&\mathcal{A}^{N-1}$	list of model atoms
$\&\mathcal{A}^0, \&\mathcal{A}^1, \ldots, \&\mathcal{A}^{NS-1}$	list of model surface atoms
$\&\mathcal{E}^0, \&\mathcal{E}^1, \ldots, \&\mathcal{E}^{L-1}$	list of model elements

Table 3.3: Data structure for model.

of the NS atoms which are on the surface of the model is also included in the model data structure. This list of surface atoms facilitates evaluation of collision detection algorithms by reducing the size of the search space; atoms that are beneath the surface need not be searched for potential interaction with the user.

Chapter 4

Mathematical Framework

4.1 Simulation

A state-space approach is used to analyze and simulate the model dynamics. Briefly, in a state-space approach, the mathematical model is represented by a series of first order differential equations. The order of a dynamic system is determined by the number of initial conditions that need to be specified. In the DSM approach, there are two state variable vectors, $\mathbf{x_1}$ and $\mathbf{x_2}$, for each of the N atoms in the model, resulting in at least a $6N$ order system for the entire model. $6N$ is a lower bound on the order of the system because the constitutive relations used to calculate internal element forces may be nonlinear. The set of differential equations that defines the dynamic process of a model can be written as

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) \tag{4.1}$$

where $\mathbf{x}$ is a $6N$ dimensional state vector of the process. Although the physics defined in the element constitutive relations are time-invariant, they have the potential to be nonlinear functions of $\mathbf{x}$. Thus, the lack of linear time invariance (LTI) in the model precludes use of the conventional convolution integral approach for solving the state of the dynamic system.

As will be explained later, the actual run-time order of the model is much less because only certain atoms are simulated each time step, depending on the position of the haptic interface. This "localizing" simulation feature due to use of a haptic window is fundamental to realizing real-time deformation response. During the simulation, a multi-step algorithm shown in Figure 4-1 is used iteratively to update the state of the model. Each block of the diagram is explained in detail below.

1. *Calculate external forces*

$$\mathcal{A}^i \!\rightarrow\! \mathbf{f} = \sum \mathbf{f}^i_{ext} \tag{4.2}$$

The external force on each mobile atom, $\mathcal{A}^i$, in the DSM is calculated, as shown in Equation 4.2, and stored in the atom's data structure. This step essentially finds the input component $f(\mathbf{u})$ in Equation 4.1. The external input can be supplied by a variety of sources. Field forces such as gravity ($\mathbf{f}^i_{ext} = m^i\mathbf{g}$) and air resistance ($\mathbf{f}^i_{ext} = -m^i\dot{\mathbf{x}}^i_{\mathbf{1}}$) may be applied to each mobile atom. Interaction forces are applied locally to those

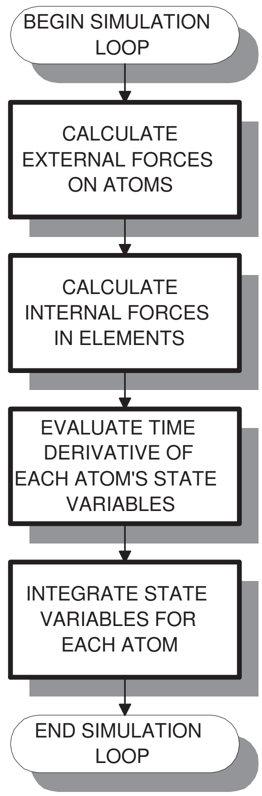

Figure 4-1: Algorithm for updating the DSM state.

surface atoms that come into contact with the haptic interface (see Section 5.5.1). The versatility of the DSM allows other external forces to be easily added as the environment model becomes more complex. Examples include interaction with rigid environment walls, collision between multiple deformable objects, and surface friction at the contact patch.

2. *Calculate internal element forces*
Once all external forces on mobile atoms are calculated, internal element forces need to be evaluated and stored in each element's data structure, $\mathcal{E}$. This step is spatially solving for the potentially nonlinear component $f(\mathbf{x})$ *in the topology of the model*. The constitutive element relations, nonlinear or linear, are determined during the modeling process for the particular media being simulated. Currently, each element is composed of a linear spring and dashpot in parallel and hence apply only linear elastic and viscous forces, respectively. The following equations are used to update each element's state.

Equation 4.3 finds the difference vector, $\mathbf{s}$, between the absolute positions of the two bounding atoms $\mathcal{A}^1$ and $\mathcal{A}^0$.

$$\mathbf{s} = \mathcal{E}^i \to \mathcal{A}^1 \to \mathbf{p} - \mathcal{E}^i \to \mathcal{A}^0 \to \mathbf{p} \tag{4.3}$$

The element's length, l, for the current time step is found by taking the magnitude of $\mathbf{s}$, as shown in Equation 4.4.

$$\mathcal{E}^i{\rightarrow}l = |\mathbf{s}| \tag{4.4}$$

The unit direction vector, $\hat{\mathbf{s}}$, between the element's adjoining atoms is consequently found by Equation 4.5.

$$\hat{\mathbf{s}} = \frac{\mathbf{s}}{\mathcal{E}^i{\rightarrow}l} \tag{4.5}$$

The scalar velocity of the element's length, $\dot{l}$, is then found by applying a first order, backwards difference filter to the scalar length data for that element (Equation 4.6).

$$\mathcal{E}^i{\rightarrow}\dot{l}_n = \lambda(\mathcal{E}^i{\rightarrow}l_n - \mathcal{E}^i{\rightarrow}l_{n-1})/T + (1-\lambda)\mathcal{E}^i{\rightarrow}\dot{l}_{n-1} \tag{4.6}$$

The magnitude of stiffness force present in the element is found by multiplying the element's stiffness constant, k, with the difference between current element length, l, and element equilibrium length, l_{eq}. The stiffness force vector is found by scaling $\hat{\mathbf{s}}$ with the stiffness force magnitude (Equation 4.7).

$$\mathbf{f}_k = \mathcal{E}^i{\rightarrow}k(\mathcal{E}^i{\rightarrow}l - \mathcal{E}^i{\rightarrow}l_{eq})\hat{\mathbf{s}} \tag{4.7}$$

The viscous force is similarly found by multiplying the element's viscous constant, b, with $\dot{l}$, and consequently scaling $\hat{\mathbf{s}}$ by this amount (Equation 4.8).

$$\mathbf{f}_b = \mathcal{E}^i{\rightarrow}b(\mathcal{E}^i{\rightarrow}\dot{l})\hat{\mathbf{s}} \tag{4.8}$$

Finally, the net force internal to the element is found by summing up the vector force contributions of the various constitutive elements (Equation 4.9).

$$\mathcal{E}^i{\rightarrow}\mathbf{f} = \mathbf{f}_k + \mathbf{f}_b \tag{4.9}$$

3. *Evaluate time derivatives of state variables*
   The next block evaluates the time derivatives of the state variables $\mathbf{x_1}$ and $\mathbf{x_2}$ for each of the mobile atoms in the model. Equation 4.10 assigns the velocity, $\mathbf{x_2}$, to the time derivative of position, $\mathbf{x_1}$.

$$\dot{\mathbf{x}}_{\mathbf{1}}^i = \mathbf{x_2}^i \tag{4.10}$$

   The acceleration, $\dot{\mathbf{x}}_{\mathbf{2}}^i$, is set equal to $\frac{1}{m^i}$ multiplied by the sum of forces acting on $\mathcal{A}^i$ as shown in Equation 4.11. The sum of forces is comprised of external forces $\mathcal{A}^i{\rightarrow}\mathbf{f}$, calculated in Step 1, and internal forces $\sum_{j=0}^{k-1}\mathcal{E}^j{\rightarrow}\mathbf{f}$, calculated in Step 2. Only the k elements that connect to $\mathcal{A}^i$ are used in calculating the internal forces. This step completes the vector differential equation defined in Equation 4.1.

$$\dot{\mathbf{x_2}}^i = \frac{1}{m}(\mathcal{A}^i{\rightarrow}\mathbf{f} + \sum_{j=0}^{k-1}\mathcal{E}^j{\rightarrow}\mathbf{f}) \tag{4.11}$$

4. *Integrate state variables*
   Several numerical algorithms exist for integrating ordinary differential equations of the

form

$$\dot{\mathbf{x}} = f(\mathbf{x}, t), \tag{4.12}$$

where the function f is *known* [20]. These algorithms can be broadly grouped into the following three classes.

(a) Predictor-corrector

(b) Bulirsch-Stoer

(c) Runge-Kutta

The dynamic process of a DSM, however, obeys the differential equation shown in Equation 4.1, which has dependence on a *causal* input forcing vector $\mathbf{u}$. Given that the virtual environment simulation is progressing concurrently with non-deterministic user interaction, haptic forces can *only* be calculated for the same time interval as is being simulated.

Although predictor-corrector (PC) methods do not require evaluations of f at time steps future to the current one (and hence obey causality), PC methods are only appropriate for very smooth equations. Given the potential for discontinuous human input through the haptic interface, PC methods are not desirable. Similar to PC methods, the Bulirsch-Stoer (BS) method is only appropriate for differential equations containing smooth functions on either side. Furthermore, the BS method requires evaluations of f at later time steps, violating the causality criteria.

Runge-Kutta methods, of the algorithms listed above, are the most amenable to integrating nonsmooth data. An nth order Runge-Kutta method makes use of n function evaluations to calculate the state of the next time step. Although used in the literature primarily for pedagogical purposes, the Euler method ($n = 1$) satisfies causality and is computationally efficient since the additional computational cost of re-evaluating the DSM loop for higher order Runge-Kutta methods is prohibitive. Currently, the Euler method has proven sufficient for simulating a wide range of material properties in IDMS.

Equations 4.13 and 4.14 integrate state variables $\mathbf{x_1}$ and $\mathbf{x_2}$, respectively, using Euler's method and the simulation step size, T.

$$\mathbf{x_1}[n+1] = \mathbf{x_1}[n] + \dot{\mathbf{x}}_\mathbf{1}[n]T \tag{4.13}$$

$$\mathbf{x_2}[n+1] = \mathbf{x_2}[n] + \dot{\mathbf{x}}_\mathbf{2}[n]T \tag{4.14}$$

Equation 4.15 updates the absolute position $\mathbf{p}$ for each atom by adding the differential position $\mathbf{x_1}$ to the absolute equilibrium position $\mathbf{p}_{eq}$.

$$\mathbf{p}[n+1] = \mathbf{p}_{eq} + \mathbf{x_1}[n+1] \tag{4.15}$$

4.2 Stability

The numerical stability of the DSM simulation is governed by a combination of stability of the continuous-time model, step size T, and the discrete-time integration method used. Aside from guaranteeing a stable response, addressing stability is important for maximizing computational efficiency. A time step that is chosen too large for the given physical parameters will reduce simulation speed unnecessarily, hence compromising haptic fidelity.

Given the infinite set of complex three-dimensional topologies that can be simulated using IDMS, it is impossible to produce a stability criteria for all models.Much insight can be derived, however, from analyzing a lumped parameter model of a media fragment and using the resulting intuition in designing global models. Stability of the models used in this thesis was determined by mathematically analyzing a fragment of DSM media, creating a global object with this media, and then empirically testing the object haptically for proper feel. This procedure is repeated as necessary to converge on a perceptually convincing, yet stable model.

The system shown in Figure 4-2 is a one-dimensional simplification of the haptic window described in Section 3.2.2. Each mass is connected by a spring-dashpot element to its nearest neighbors and its own equilibrium position, hence modeling the connecting and home elements, respectively. The ends of the system are mechanically grounded, thus representing the boundary condition at the haptic window border. This simple 6th order system is an excellent vehicle for illustrating the stability of the discrete-time window model as a result of varying physical parameters and T.

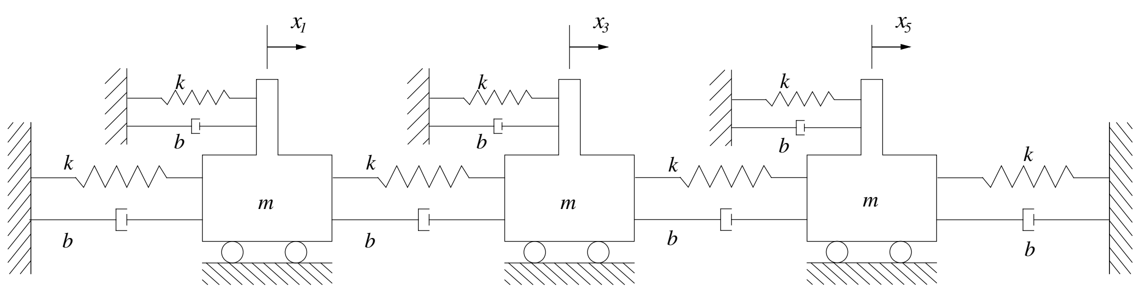

Figure 4-2: 6th order system model of haptic window.

The continuous time differential equation for the plant is

$$\dot{\mathbf{x}} = A\mathbf{x} \tag{4.16}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-3k}{m} & \frac{-3b}{m} & \frac{k}{m} & \frac{b}{m} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k}{m} & \frac{b}{m} & \frac{-3k}{m} & \frac{-3b}{m} & \frac{k}{m} & \frac{b}{m} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k}{m} & \frac{b}{m} & \frac{-3k}{m} & \frac{-3b}{m} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \tag{4.17}$$

are the system matrix A and state vector $\mathbf{x}$ respectively.

Using an Euler integration method, the discrete-time equation describing the state update is

$$\mathbf{x}[n+1] = (AT + I)\mathbf{x}[n] \tag{4.18}$$

where T is not to be confused as a matrix. The stability of the system is governed by the eigenvalues of the discrete-time system matrix, $A_z = AT + I$. The following sections describe the tradeoff between performance and stability in the z-plane using a root locus approach. Nominal unitless parameter values of the media fragment are $k = 15.0$, $b = 0.45$ and $m = 0.002$ and the time step is $T = 0.001$ sec, resulting in an underdamped system. In the following figures, 'x' marks the location of the nominal pole positions. In each subsection, one parameter is varied while holding the others fixed.

4.2.1 Stiffness

As shown in Figure 4-3, increasing the element stiffness k decreases the margin of stability in the z-plane for a fixed value of T. Stiffer systems have higher natural frequencies and thus require smaller time steps for stable numerical simulation. Nevertheless, at $T = 0.001$ sec (1 kHz update rate), the range of representable stiffnesses is still greater than 10:1.

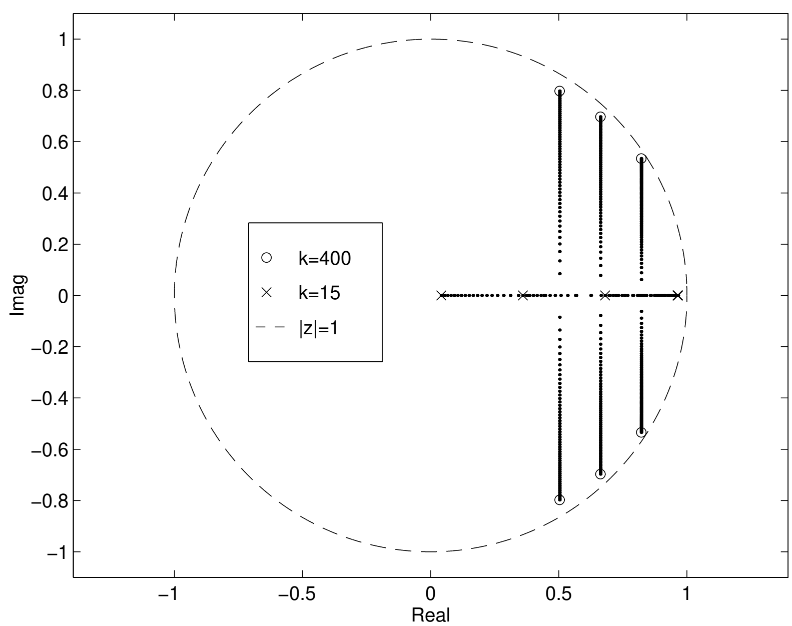

Figure 4-3: Z-plane root locus of haptic window: $15 < k < 400$.

4.2.2 Mass

Decreasing the atom mass m forces the poles to the left on the real axis (see Figure 4-4, eventually pushing them outside the unit circle. From a performance perspective, however, lighter masses are desirable because they reduce the amount of high frequency momentum transfer during interaction with the surface. Larger masses result in the perception of haptic "noise" as atom masses transition on and off the master interaction point (see Section 5.5.1).

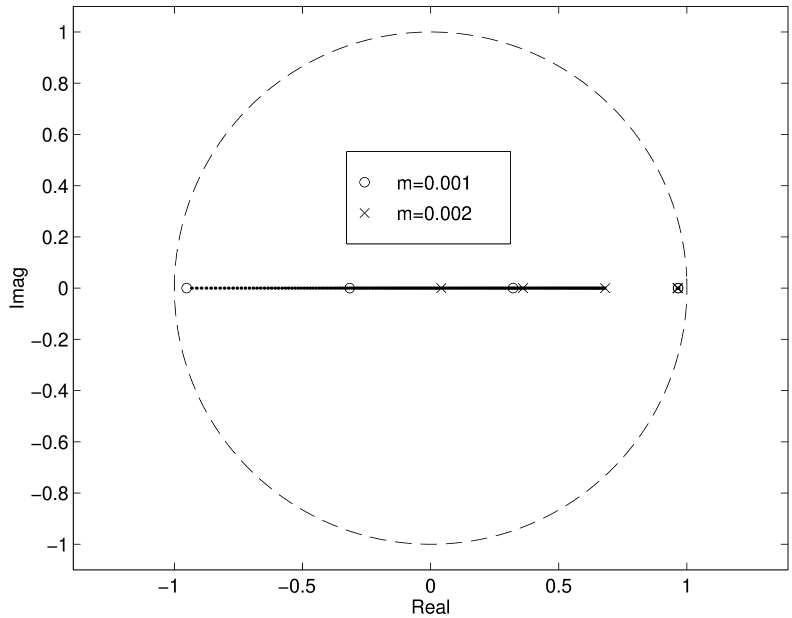

Figure 4-4: Z-plane root locus of haptic window: $0.001 < m < 0.002$.

4.2.3 Viscosity

In the continuous-time system, decreasing the viscosity b moves the poles closer to the jw axis while increasing the viscosity stabilizes the system. In the discrete-time case, lowering the viscosity similarly forces the poles towards the unit circle, for example, shown at the value of $b = 0.1$ in Figure 4-5. Larger values of b increase the frequency content of the plant dynamics, also tending to destabilize the system ($b = 0.9$). For the nominal value of $b = 0.45$, a viscosity range of approximately 10:1 exists.

4.2.4 Time step

From a performance perspective, a minimal time step T is desired in order to increase the servo rate of the haptic display, hence improving the bandwidth of force display. Furthermore, from a stability viewpoint, increasing T shifts the poles to the left on a path leaving the unit circle, as shown in Figure 4-6. A smaller time step, however, limits the number of atoms and elements that can be simulated each time step, reducing the computable spatial resolution for a given window size.

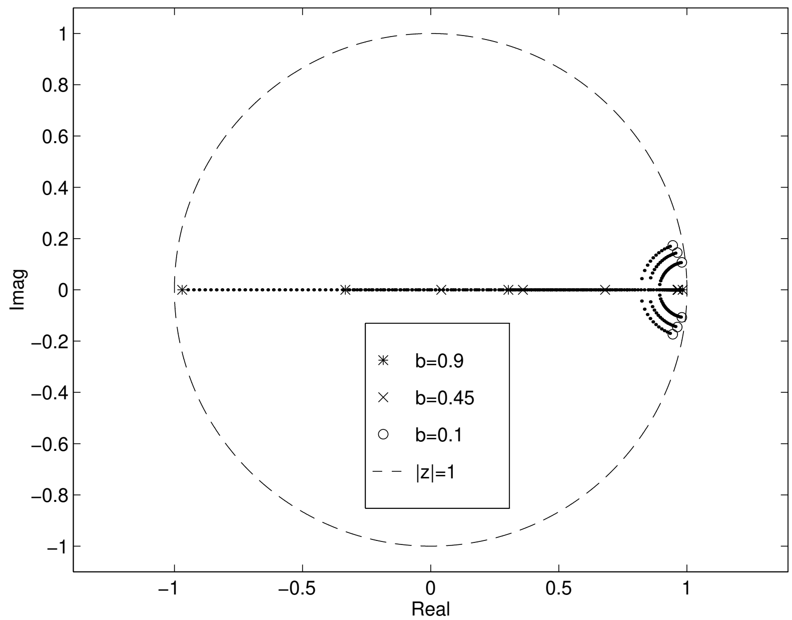

Figure 4-5: Z-plane root locus of haptic window: $0.1 < b < 0.9$.

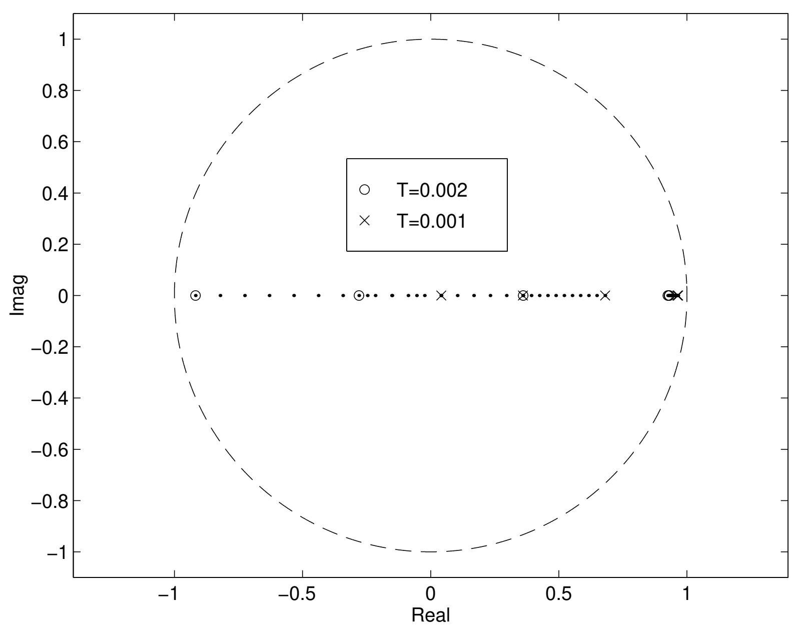

Figure 4-6: Z-plane root locus of haptic window: $0.001 < T < 0.002$.

Chapter 5

Implementation

The framework described in the last chapter provides the mathematical core for processing the DSM simulation. Integration of the DSM simulation into the IDMS system requires a real-time computer architecture to meet the computational constraints of an interactive virtual environment. Sections 5.1 and 5.2 describe the hardware and software architectures, respectively, that provide the computational backbone for IDMS. Section 5.3 explains the flow of processing from the initial state of obtaining model data files to the state of real-time interaction. Section 5.4 proceeds to describe the real-time system loop. Section 5.5 explains the algorithms and data structures used to generate an interaction force both on the environment and the user. Lastly, Section 5.6 describes the current methods of generating data models for simulation in IDMS.

5.1 System Architecture

Several high-performance components comprise the hardware architecture that IDMS runs upon. The choice of architecture is heavily influenced by the concurrent run-time model used in simulating IDMS (see Section 5.2). Figure 5-1 illustrates the IDMS system architecture and the flow of information between the components. Each of the components and their connectivity is described below.

5.1.1 Human

The human delivers position to and receives force-feedback from the haptic interface. Visual feedback is delivered to the human by the visual engine through a color display and the optional use of StereoGraphics CrystalEyes shutter glasses, which provide a stereo display of the screen to the user. In the current IDMS virtual environment setting, the user places herself behind the haptic interface and the visual display is positioned directly behind the haptic interface. Mathematically speaking, the origins of the user, haptic interface, and visual display reference frames are placed roughly in a collinear fashion and the depth axes of the three frames are approximately parallel. The visual display is positioned as close as possible behind the haptic interface without interfering with the workspace of the device. Based on empirical trials of different plans, this spatial layout allows the user to register haptic and visual signals in a perceptually coherent manner.

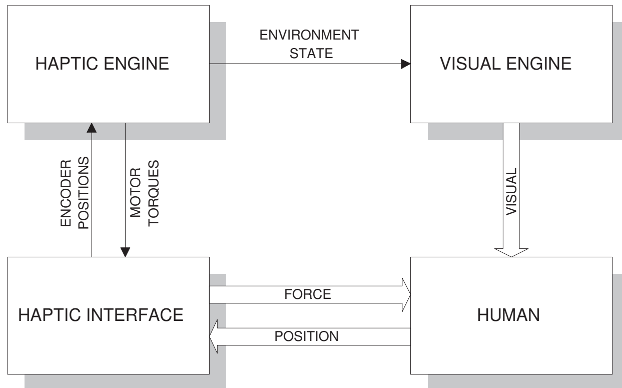

Figure 5-1: System architecture for IDMS. Wide arrows represent flow of sensorimotor signals; thin arrows represent flow of electronic data.

5.1.2 Haptic interface

The haptic interface provides information about the Cartesian position of the endpoint in the form of encoder data for each of the haptic interface axes. Motor torques applied to the joints of the haptic interface are mechanically transmitted through the device's structure, resulting in an endpoint force applied to the user. The Phantom haptic interface is used in IDMS to serve these purposes and is described below.

The Phantom haptic interface [11, 12] shown in Figure 5-2 is a three-degree-of-freedom force-reflecting interface that relies on a cable-driven transmission and well-balanced mechanics for high fidelity transmission of position and force signals. The design of the Phantom allows users to execute wrist-centered motions with a large workspace. Multiple Phantoms can be interfaced with the hand allowing multi-finger interactions with the virtual environment. The endpoint of the Phantom is a passive three degree-of-freedom gimbal which allows for torque-free rotations. This feature enables the Phantom to be treated as a point force source at its endpoint. Various tools, passive or active, can be inserted into the gimbal to model different types of interactions with the world. The simplest of these tools, a thimble, is shown more clearly in the Phantom schematic in Figure 5-3 and allows the user to apply and receive point forces at the fingertip. In the context of surgical simulation, a surgical tool such as a scalpel or tweezer may be inserted into the gimbal to reflect the desired training situation. For example, Figure 5-4 shows an instrumented passive tweezer mounted to the gimbal. Binary contact information is provided through an electrical switch that is closed when both tweezer sides are touching.

5.1.3 Haptic engine

The haptic computational engine is responsible for running processes that simulate the dynamics of the virtual environment (environment process), servo the haptic interface (haptic process), and update the visual process on the visual engine with the current environment state. As explained in Section 5.2, the haptic and environment processes are run as one com-

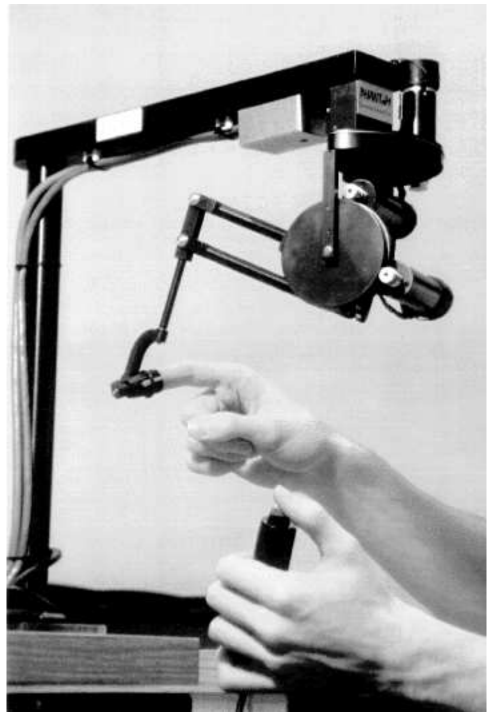

Figure 5-2: The Phantom haptic interface.

bined process because of the inherent interactive coupling between the environment dynamics and the haptic interface. A 120 MHz Pentium-based computer running Linux, a unix-based multitasking operating system, is used to serve these purposes.

5.1.4 Visual engine

The visual engine is responsible for displaying the state of the virtual environment at a sufficient refresh rate to the human visual system. A 200 MHz Silicon Graphics Indigo II Extreme workstation, used as the IDMS visual engine, is designed for fast graphics computation because its architecture has an integrated graphics pipeline for processing low-level graphics routines. The visual processes for IDMS use the Silicon Graphics OpenInventor library toolkit built upon the OpenGL protocol for platform independent graphics rendering. All the figures in this text which display visually rendered scenes from IDMS depict the viewer window that is standard to the OpenInventor library. Incorporated into this viewer are basic viewing primitives such as zooming, panning, rotating, lighting, and shading that are easily controllable by the user. Lastly, data is communicated between the haptic and visual engines over Ethernet using a UDP packet level connection, ensuring high throughput rates of data transfer.

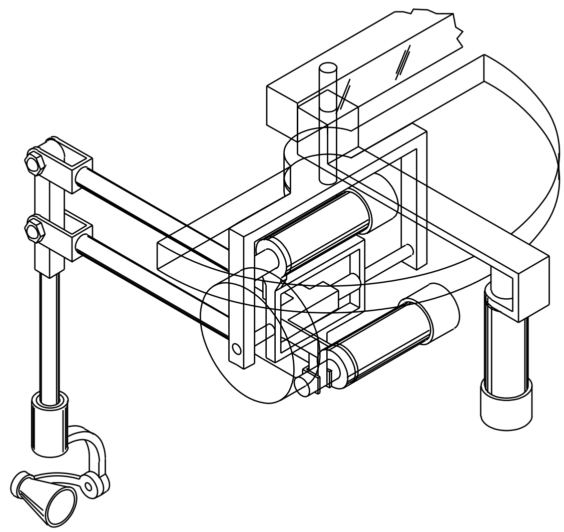

Figure 5-3: Schematic of Phantom haptic interface (Courtesy of David Brock, MIT Artificial Intelligence Laboratory).

5.2 Run-time Model

A concurrent [5, p259] model is utilized as the run-time software architecture for IDMS (see Figure 5-5). Under the concurrent model, separate operations in the environment run in parallel on different processors. In IDMS two main processes, which handle three discrete subprocesses, run concurrently on the haptic and visual engines. The *client* process, running on the haptic engine, is comprised of the *environment* and *haptic* processes and updates at approximately 1 kHz. The environment and haptic processes are lumped together as one process because the haptic interaction force is applied to the haptic interface and the environment dynamics concurrently (see Section 5.4). The *server* process, which runs on the visual engine and is equivalent to the *visual* process, receives environment data from the client process and renders the visual state of the environment at approximately 30 Hz. In order to decouple loading effects between the client and server processes, a separate *shared-memory* process runs on both the haptic and visual engines to handle interprocessor data communication. The client process updates the shared-memory process on the haptic engine with *differential* changes in environment data, hence minimizing latency in the time-critical client loop. The two individual shared-memory processes run synchronously using a handshaking protocol and serve to transfer the entire visual state of the environment at 30 Hz. Lastly, the server process runs synchronously with the shared-memory process on the visual engine and hence also updates the screen at 30 Hz. Because Linux is not a preemptive operating system, the shared-memory process on the haptic engine is currently initiated by the client process at 30 Hz. The use of a preemptive real-time operating system, such as LynxOS, would allow each process to be time-sliced during each unit of processing time, allowing for more transparent decoupling between processes (see Section 7.2.1).

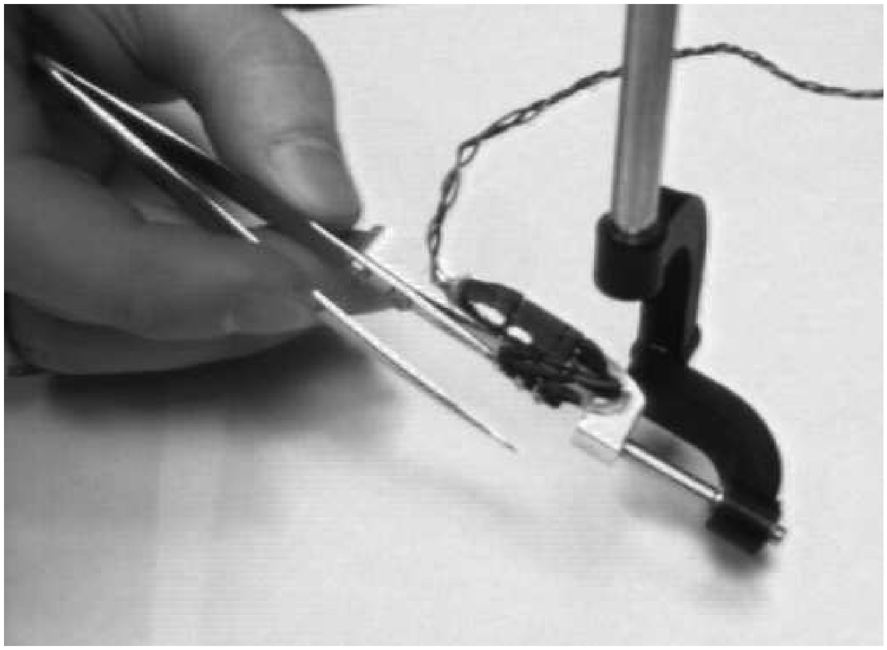

Figure 5-4: Instrumented tweezers attached to Phantom gimbal.

A concurrent architecture is appealing as a run-time model for IDMS for the following reasons.

1. *Modal decomposition*
   By separating haptic and visual software onto two platforms, issues specific to each sensorimotor modality can be addressed and optimized independently. Assuming modular programming techniques are used, changes to visual rendering software will not affect the performance or integrity of the haptic rendering software. Furthermore, each platform can be tailored to the specific hardware needs of the modality it represents. The haptic engine, for example, requires bus slots for expansion cards to accommodate interfacing with strain gauges, amplifiers, encoders, and other analog and digital input/output. Because of the extensive commercial hardware available for the PC platform and the memory-mapped I/O on the bus, hardware interfacing is greatly facilitated. Furthermore, other modalities, such as audition, can be addressed independently of currently represented modalities without necessarily constraining hardware or software.

2. *Parallel processing*
   The use of additional processors increases computational power significantly by parallelizing computation. Furthermore, each processor can be matched independently to the required refresh rate of the modality it represents, resulting in decoupling between rendering rates for different modalities. As stated in Section 2.2, an adequate haptic display requires at least 1 kHz bandwidth to convincingly transmit tactile and force

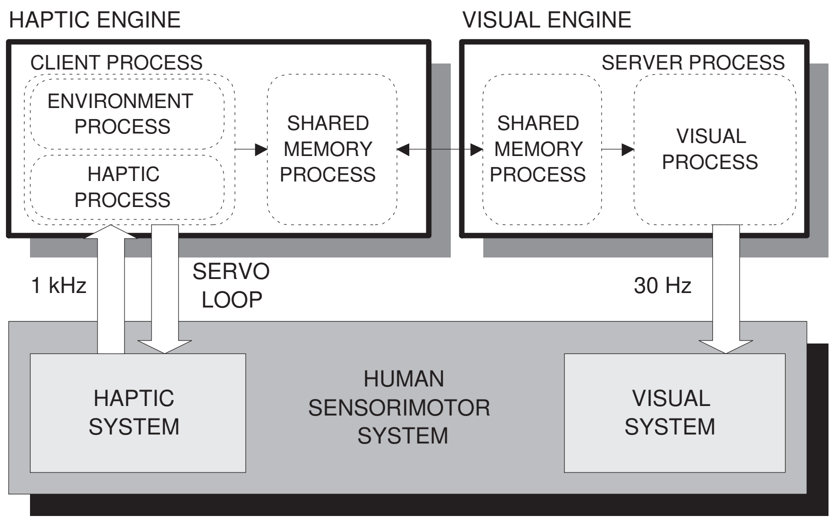

Figure 5-5: Concurrent model of run-time processing for IDMS.

information. Conversely, a visual display only requires at minimum a throughput rate of 8 to 10 Hz and a latency of 0.1 seconds. Since the haptic process is part of a feedback control system that includes the haptic interface, it is desirable to minimize computation delay in order to maximize force display bandwidth. Computation specific to the visual modality is separated onto the visual engine, thus parallelizing computation of *and* decoupling loading effects between the haptic and visual processes.

5.3 Process flow

An IDMS session follows the process flow diagram shown in Figure 5-6. The database of `plg` and `fem` files is accessible to both the haptic and visual engines[1]. The left sequence is run by the haptic engine; the sequence on the right is executed by the visual engine. When both sequences have completed, the real-time interactive simulation begins.

The client process parses the `fem` file and constructs a complete model data structure, $\mathcal{M}$, described in Section 3.3. Once the model is created, the client process proceeds to set the initial conditions of the model in the environment. Although this procedure is currently embedded within the simulation, a shell could be written that allows the user to interactively set the initial conditions of the model. The haptic interface is then calibrated and geometrically registered with the reference frame of the environment. Finally, the shared-memory process on the haptic engine is initiated as a child process by the client and the user is prompted to begin the simulation.

In parallel, the server (visual) process initiates a viewer window for graphics display and

[1]Each model is comprised of a `plg` and `fem` file sharing the same filename prefix, for example `cube.plg` and `cube.fem`.

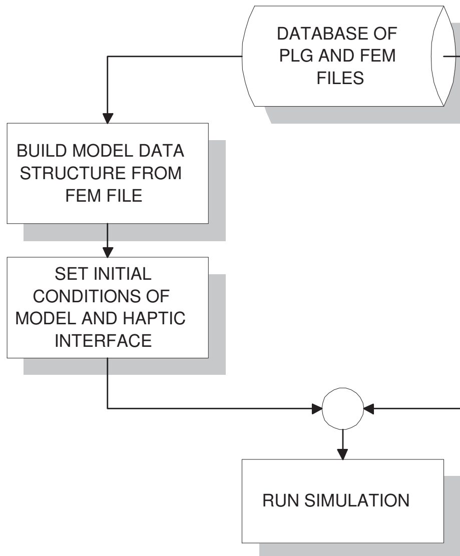

Figure 5-6: Process flow of an IDMS session. The sequence on the left is executed by the haptic engine; the sequence on the right is executed by the visual engine. The interactive simulation loop begins when both sequences have finished.

forks off the shared memory process on the visual engine as a child process. When the client process reads the `fem` file, the server is notified and the corresponding `plg` file is read from the database. This `plg` file is used to create an object data structure native to OpenInventor for storing topology and geometry information. The server then displays the object in the viewer window. The user proceeds to orient the object in the viewer window so that it is visually registered with the haptic interface. Graphical editors and menus allow the user to easily change visual properties such as material color, display mode (shaded, wireframe, or vertices), and lighting direction. Stereo viewing is a built-in option to the OpenInventor viewer window and can be selected dynamically[2]. Once the visual conditions are set, the user signals the client process to begin the IDMS simulation loop. Viewing conditions can be changed if necessary at any time during the simulation.

5.4 System Loop

Figure 5-7 depicts in detail the computation blocks that are executed and the processes they occupy during each system loop of IDMS. The DSM simulation blocks are each represented by a thick border in the environment process. As mentioned in Section 5.2, the environment and haptics processes are combined as one process because both are synchronized to the interaction force computation block (Section 5.5.1). Because only local dynamics are

[2]If stereo viewing is chosen, the user can interactively choose the offset value between the left and right images presented to the shutter glasses.

simulated, the environment dynamics and haptic display can be updated simultaneously and still meet required perceptual refresh rates for the human haptic system. As referred to in Section 3.2.2, however, the next design iteration of IDMS might involve a second environment process to update global dynamics at a slower refresh rate. In this situation, the first loop in the diagram would still be updated quickly, thus maintaining haptic fidelity, and the second loop would be updated less often for atoms and elements external to the haptic window.

5.5 Interaction Force Calculation

5.5.1 Master Interaction Object

The subsystem of IDMS that calculates the interaction force, the master interaction object (MIO), is the software element connecting the simulation and the haptic interface. The MIO *models* in software the contact interaction between the human and the object surface. The MIO inputs position information from the haptic interface and the DSM simulation and outputs a force signal back through the haptic interface and the DSM simulation. Each haptic interface that is connected to IDMS is assigned a different MIO.

In system engineering terminology, the MIO is a software entity representing a linear time-invariant (LTI) two-port element. There are energy requirements imposed on the MIO by the virtual environment. These requirements are enumerated below and refer to Figure 5-8, which shows the power flow between each of the two-port elements in IDMS.

1. *Causality*
   In order to preserve integral causality on each of the atom masses in the DSM, only forces can be applied as inputs to them. Given that the haptic interface only outputs position information, the MIO must convert position input into force output to satisfy the causality constraint in the DSM simulation. Similarly, the haptic interface can only apply force on the user. Thus, the MIO must also convert position output from the DSM simulation into force input for the haptic interface. Similar to a two-port gyrator, the MIO transforms an effort variable input from port A into a flow variable output at port B; furthermore, a flow variable input at port B is transformed into an effort variable output at port A. Unlike a gyrator, however, the MIO uses position input from both the first port (haptic interface) *and* the second port (DSM simulation) to generate equal and opposite force output on both ports. Lastly, the MIO can *store* energy but a gyrator must conserve power at both ports, hence it is unable to accumulate energy.

2. *Passivity*
   As mentioned in Section 2.2, an important property for a virtual environment is *passivity* along the transmission path between the simulation and the human. Implementation of the MIO using software models of passive physical elements will ensure passive characteristics. Early experiments with potential field based modeling of passive environments approaches utilized a non-passive MIO, resulting in the undesirable perceptual feel of an active environment.

3. *Stiff dynamics*
   Depending on its representation, the MIO can have internal dynamics. In the context

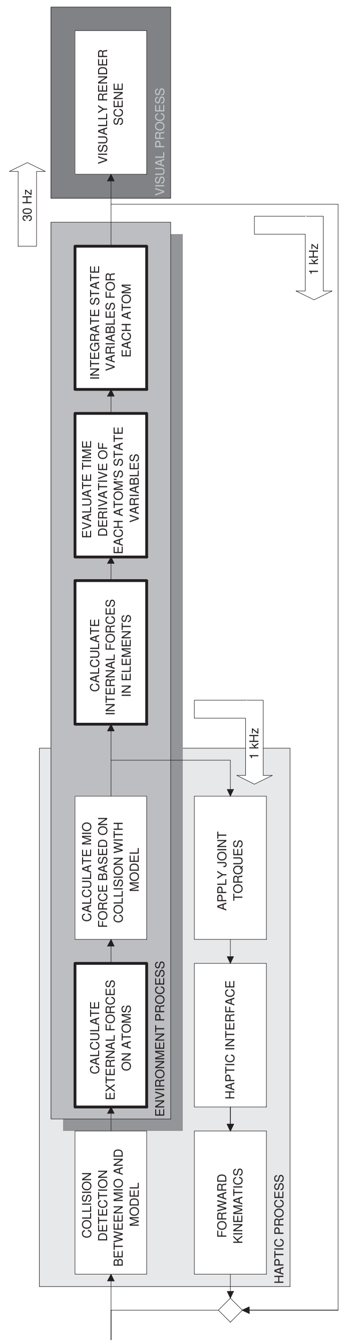

Figure 5-7: Breakdown by process of overall system loop. The environment and haptics processes are synchronized to run at the same speed because the external force is applied to the haptic interface and the simulation dynamics concurrently.

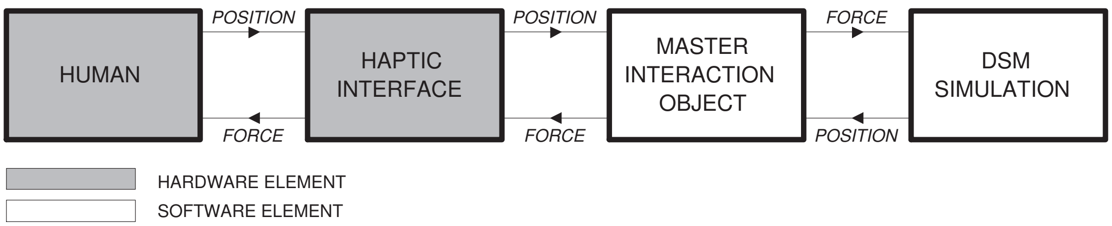

Figure 5-8: Power flow between two-port elements in IDMS.

of deformable media, it is desirable for the interaction element to have a much stiffer surface impedance than that present in the model, in order to prevent filtering or attenuation of deformation force during contact. If the poles of the interaction element are too closely located to those of the model being simulated, the human may not be able to distinguish the dynamics of the interaction element from the dynamics of the simulated media.

Specifically, the MIO is defined by a static geometric shape that is dynamically centered at the endpoint of the haptic interface. This shape represents the geometry of the real-world tool being modeled for interaction. The minimum size of the MIO is determined by the lowest spatial resolution of atoms present on the model surface. Too small a size for the MIO will generate a nonsmooth force during stroking. Too large a size, however, increases the number of atoms and elements that need to be simulated to generate an interaction force, thus increasing computation load. These tradeoffs need to be considered carefully when determining the spatial resolution and MIO size for a particular modeled environment.

During contact, atoms that are positioned internal to the geometric boundary of the MIO receive an external force. A reaction force is applied to the haptic interface equal to the negative sum of external forces applied on DSM atoms by the MIO. Figure 5-9 illustrates a typical interaction sequence using, for example, a spherical MIO shape. Currently, each atom receives a force proportional to its internal penetration distance normal to the geometric surface, simulating an elastic contact. Other force generating functions are possible as long as they satisfy the energy requirements enumerated above. A simple collision detection algorithm is used to find those atoms of the model that are internal to the MIO boundary. Collision detection with the MIO is computationally efficient because only point intersections need to be found. Increasingly complex geometries can be created for the MIO as a function of processing power and more efficient collision detection algorithms. The following example illustrates the MIO representation of fingertip interaction.

Example 1: MIO representation of fingertip

Modeling the fingertip as a haptic interaction element is accomplished by representing the MIO as a spherical object. A sphere is an ideal shape for the MIO because collision detection and force generation are fast to compute. The data structure, $\mathcal{MIO}$, shown in Table 5.1 is used to store parameters and variables for a spherical MIO of radius r with surface stiffness

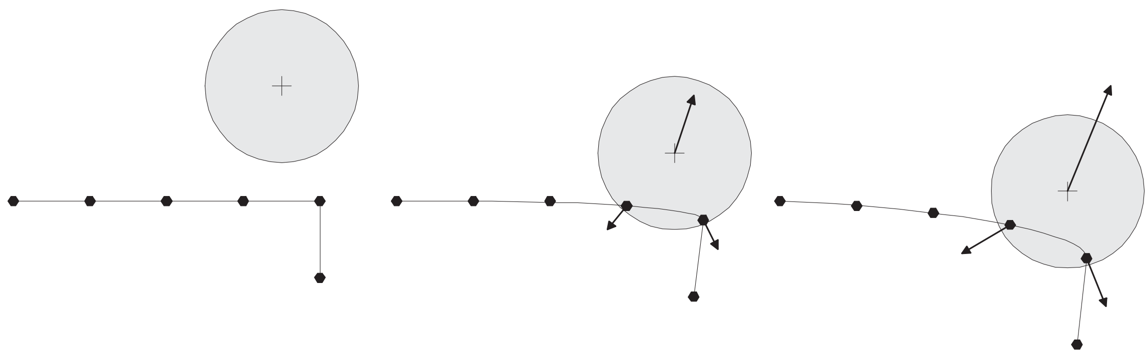

Figure 5-9: Illustration of an interaction between a spherically-shaped MIO and the surface of a model.

k.

$\mathcal{MIO}$	
$\mathbf{p}$	absolute position of haptic interface endpoint
$\mathbf{f}$	net force applied on environment
k	boundary stiffness
r	radius of interaction sphere
r_l	radius of local sphere

Table 5.1: Data structure for MIO representation of human fingertip.

The following algorithm computes the collision detection and calculation of interaction force using a spherical MIO. This algorithm comprises the top three computation blocks of the haptic process in Figure 5-7. A look-up table of square root values is used to reduce square root computation time by a factor of 50%.

$\mathcal{MIO} \to \mathbf{f} = \mathbf{0}$
for $i = 1 : NS$
  $\mathbf{d} = \mathcal{A}^i \to \mathbf{p} - \mathcal{MIO} \to \mathbf{p}$
  $r = |\mathbf{d}|$
  if $r < \mathcal{MIO} \to r$
    $\mathbf{f} = \mathcal{MIO} \to k(\mathcal{MIO} \to r - r)/(r)\frac{\mathbf{d}}{|\mathbf{d}|}$
    $\mathcal{A}^i \to \mathbf{f} += \mathbf{f}$
    $\mathcal{MIO} \to \mathbf{f} -= \mathbf{f}$
  end
end

Note that currently all NS atoms are scanned for possible intersection with the MIO. The integration of three-dimensional spatial data structures (octrees) into IDMS would greatly reduce the search space of potential collisions. Off-line, the model would be subdivided into a tree of octants to a given spatial resolution. A preprocessing step would assign each *surface* atom to a leaf octant based on the atom's equilibrium position. During run-time, the search space for MIO collision is consequently reduced to leaf octants that are children of the smallest octant which entirely contains the MIO.

5.5.2 Local simulation

As stated in section 3.2.2, the feature of IDMS enabling real-time simulation is the ability to simulate the region of DSM media *local* to the interaction point. Local simulation is an option to IDMS; some procedures such as model generation utilize the DSM algorithm on *all* model atoms and elements and hence require the local simulation option to be turned off. The DSM framework allows for this local simulation to be easily implemented. The variable r_l in the $\mathcal{MIO}$ data structure contains the radius of the sphere, centered at the center of the MIO, that spatially separates atoms and elements into local and non-local entities. Atoms are defined as local if their absolute position lies within the radius of the local sphere, as illustrated in the following algorithm. Note that internal as well as external atoms can be defined as local; hence all N atoms are scanned.

for $i = 1 : N$
  $\mathbf{d} = \mathcal{A}^i{\rightarrow}\mathbf{p} - \mathcal{MIO}{\rightarrow}\mathbf{p}$
  $r = |\mathbf{d}|$
  if $r < \mathcal{MIO}{\rightarrow}r_l$
    $\mathcal{A}^i{\rightarrow}\mathsf{spc} = \text{LOCAL}$
  end
end

If local simulation is on, this algorithm is executed in tandem with the MIO collision detection algorithm in the first computation block of the haptic process. Atoms are first tested for inclusion in the local sphere before testing for collision with the interaction sphere to save computation. Once the local atoms have been defined, the simulation space of the four blocks of the DSM algorithm are each modified as shown below.

1. *Calculate external forces*
   $\forall \mathcal{A}^i \ (0 < i < N-1) \ s.t. \ \mathcal{A}{\rightarrow}\mathsf{spc} == \text{LOCAL}$

2. *Calculate internal element forces*
   $\forall \mathcal{E}^i \ (0 < i < L-1) \ s.t. \ \mathcal{E}{\rightarrow}\mathcal{A}^0{\rightarrow}\mathsf{spc} == \text{LOCAL} \ or \ \mathcal{E}{\rightarrow}\mathcal{A}^1{\rightarrow}\mathsf{spc} == \text{LOCAL}$

3. *Evaluate time derivatives of state variables*
   $\forall \mathcal{A}^i \ (0 < i < N-1) \ s.t. \ \mathcal{A}{\rightarrow}\mathsf{spc} == \text{LOCAL}$

4. *Integrate state variables*
   $\forall \mathcal{A}^i \ (0 < i < N-1) \ s.t. \ \mathcal{A}{\rightarrow}\mathsf{spc} == \text{LOCAL}$

Note in block two, elements are indirectly defined as local if either one of their bounding atoms is local. As mentioned above, the use of spatial data structures such as octrees to store atom location would greatly reduce time spent in searching through the list of model atoms for local ones.

5.6 Model Generation

5.6.1 Draping

The flexibility of IDMS allows simulation of any model that adheres to the `fem` and `plg` file formats. An important concern to be addressed is the creation of such models for use in IDMS. Currently, the main approach to model generation in IDMS is based on *draping* a free-form rectangular viscoelastic "skin" slab onto a parametrically defined rigid form, and then saving the equilibrium state of the slab as the newly defined model. The same DSM engine (Section 4.1) is used to simulate the physics of state transition from the initial condition to the equilibrium position. Although this process is *not* intended to be real time, it is of no consequence because the haptic display channel is turned off during the draping procedure.

Draping forces are generated by applying attractive forces between atoms in the slab and the surface of the parametric object. Thus, the skin slab wraps itself around the object in an energy-minimizing fashion similar to a sheet of positive charge conforming to the exterior of a negatively charged object. Home elements are added to each of the NS surface atoms on the slab once the state is saved, in effect "hardening" the state of the skin. Finally, the new model is written to disk as `plg` and `fem` files for later use by IDMS. Although the resulting surface is not closed in a mathematical sense, the surface area produced is sufficiently large for meaningful interactions.

The three stages of draping are described in detail below using the example of a rectangular slab and a spherical form. The procedure is illustrated in the six frames shown in Figure 5-10.

Stage 1: Set initial conditions

During stage one, the initial conditions of the slab are set manually so that it is centered above the center of the rigid form and exterior to the surface of the rigid form.

Stage 2: Apply attractive forces between slab and form

Once the slab is positioned according to desired initial conditions, the DSM engine is started and the attraction forces are applied between the slab and the form, illustrated in the following algorithm. The spherical form has a radius R_f, surface stiffness k_f^i, attractive stiffness k_f^o, and a viscosity b_f. An atom external to the form surface receives an attractive force proportional to its distance away from the surface with stiffness k_f^o. The material parameters k_f^i and b_f are empirically determined to produce a highly inelastic response from collision between slab atoms and the form surface, consequently minimizing settling time.

Let $\mathbf{c}$ be center of sphere
for $i = 1 : N$
  $\mathbf{d} = \mathcal{A}^i \rightarrow \mathbf{p} - \mathbf{c}$
  $r = |\mathbf{d}|$

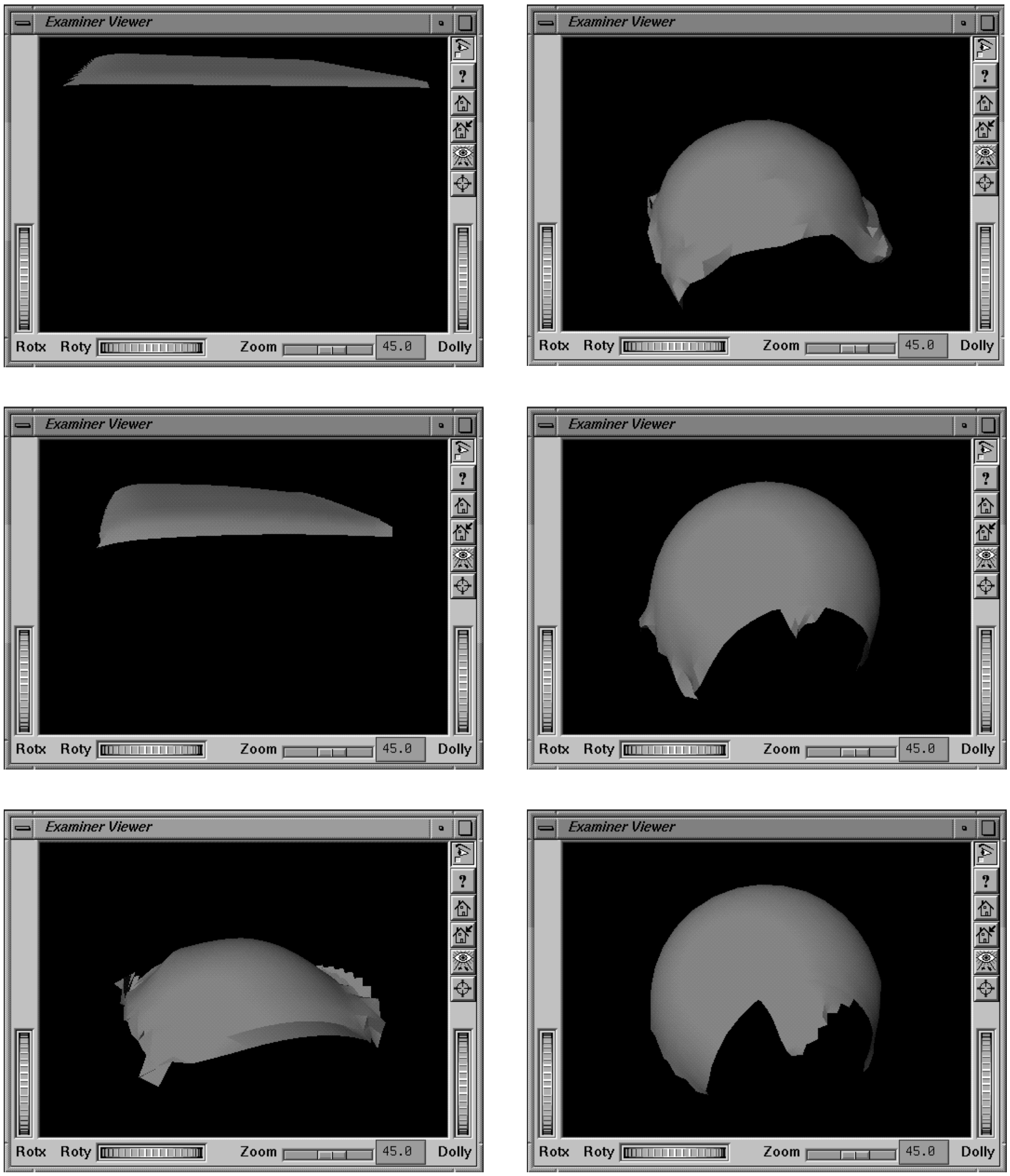

Figure 5-10: Example depicting three stages of draping using a rectangular slab and a spherical form (from top to bottom, left to right): (1) initial state, (2-5) attraction, and (6) equilibrium.

if $r > R_f$ *apply attractive forces*
$\mathcal{A}^i \rightarrow \mathbf{f} += -k_f^o (r - R_f) \frac{\mathbf{d}}{|\mathbf{d}|}$
else *apply contact forces*
$\mathcal{A}^i \rightarrow \mathbf{f} += k_f^i (R_f - r) \frac{\mathbf{d}}{|\mathbf{d}|} - b_f (\mathcal{A}^i \rightarrow \dot{\mathbf{x}}_1)$
end

Stage 3: Attach home elements and save to disk

When stage three is reached, the slab state is saved and home elements are added to each surface atom. Finally, the new spherically-shaped model is written to disk as two files, the `plg` file and the `fem` file.

for $i = 1 : N$
$\mathcal{A}^i \rightarrow \mathbf{p}_{eq} = \mathbf{p}$
if $\mathcal{A} \rightarrow \mathsf{sc} ==$ EXTERNAL
Add new home atom $\mathcal{A}^{h_i}$
$\mathcal{A}^{h_i} \rightarrow \mathbf{p}_{eq} = \mathcal{A}^i \rightarrow \mathbf{p}_{eq}$
$\mathcal{A}^{h_i} \rightarrow \mathsf{bc} =$ FIXED
Add new home element $\mathcal{E}^{h_i}$ between $\mathcal{A}^i$ and $\mathcal{A}^{h_i}$
end
end
Output `temp.fem` and `temp.plg` files

5.6.2 Other methods

Another approach to model generation is *molding*. Using the haptic display channel, the human can manually create shapes by manipulating, squeezing, depressing, and pinching existing models. Geometrically simple shapes such as prisms and flat sheets, that can be easily generated algorithmically, are used as initial models to mold. This approach is only limited by the human's creativity in model generation. Other model generation techniques are also possible. These include using commercial software tools such as AutoCAD or ProEngineer for automatically creating surface finite element meshes for arbitrary topologies. Furthermore, standalone algorithms that *mathematically* tessellate the surface of rigid objects with the viscoelastic tetrahedral elements should be feasible.

Chapter 6

Results

The following sections illustrate the features, performance, and versatility of IDMS by describing results from fundamental interaction tasks. Visual results from each interaction are presented as a sequence of screen images from the IDMS display[1]. Force and position histories of the MIO object are also provided to qualitatively and quantitatively express the haptic display performance.

6.1 Basic Interactions

Each of the basic interactions in this section operate on the homogeneous DSM media fragment shown in Figure 3-2. The haptic display is refreshed at approximately 700 Hz and the visual display is updated at 30 Hz. The Cartesian axes of the fixed IDMS reference frame are shown explicitly in the first frame of Figure 6-1. All position measurements are taken with respect to this reference frame. The element parameters are $k = 15$ and $b = 0.2$ and the atom mass is 0.002. The top surface of this fragment has dimensions of 4.00 inches along x and 3.46 inches along y. The surface stiffness of the MIO was empirically found to be $\mathcal{MIO}{\rightarrow}k = 400$, approximately 27 times the stiffness of the elements in the media. Given the relatively high stiffness value of the MIO, the deformation force is transmitted from the simulation to the user with negligible filtering of information.

During initial trial sessions with IDMS, a transparent sphere was used to visually render the MIO object. Interestingly, this action was visually disconcerting because part of the sphere would disappear beneath the top surface of the mesh due to atoms penetrating the MIO surface. Users tend to focus on the media deformation itself rather than the MIO object while in contact. Consequently, a small diamond shaped object was used instead because it provided an unobstructed view of the deformation and removed the disturbance due to visual interference between the MIO and the media. Taking into account the DSM element length of 1 inch, the radius of the MIO was empirically optimized to be $\mathcal{MIO}{\rightarrow}r = 1.5$ inches for modeling the human fingertip. Although this radius is approximately three times the radius of a fingertip, users do not perceive the actual MIO size due to the small diamond-shaped visual

[1] In the following figures where appropriate, the viewer display is shown in reverse video and all objects are shown in wireframe to clarify deformation results

representation. The absence of collocation between the haptic and visually displays and the poor kinesthetic discriminability of the human [5, p170] allow the user to successfully register haptic and visual cues using primarily haptic force and visual deformation information.

6.1.1 Palpation

A typical finger palpation sequence is depicted in Figure 6-1. The force and position histories for the MIO are shown in Figure 6-2. As expected, f_x and f_y are negligible relative to deformation force normal to the surface, f_z. The peak value for f_z reached is approximately 1.7 N. Note that f_z becomes nonzero at approximately $z = 1.5$, which is the radius of the MIO.

The force versus displacement relationship applied to the haptic interface for a palpation interaction is shown in Figure 6-3. The measurement data is fitted in a linear least squares sense to illustrate the stiffness behavior of the media. The stiffness increases initially as more atoms come into contact with the MIO and then transitions into a linear function as the number of intersection atoms levels off. The curve illustrates that the contribution of deformation force by home elements of intersection atoms is dominant relative to the force provided by non-intersection atoms. The use of a haptic window exploits this rapidly diminishing contribution of force by conserving processing power for local atoms.

6.1.2 Stroking

In contrast to palpation, stroking involves motion of the MIO lateral to the surface while it is in contact with the media. A sample stroking sequence is shown in Figure 6-4. At an approximately fixed position of $y = 1.5$ inches, the MIO object is brought into contact with the media surface, laterally moved at constant z in the direction of increasing x, and then brought out of contact with the media. In order to illustrate the effect of speed on lateral force interaction, two cases differing in lateral surface speed are discussed below.

Slow lateral surface speed

As shown in Figure 6-5, between $t \approx 1$ sec and $t \approx 3$ sec, the MIO object lowers and deforms the media surface, increasing the normal force to a value of $f_z \approx 1.3$ N. As the MIO accelerates in the positive x direction across the media, the opposing lateral force f_x increases to -0.1 N due to the inertia of atoms along the MIO path that needs to be overcome. As the MIO decelerates at $x \approx 3$ inches, the lateral force changes sign to reflect the buildup of inertia behind the MIO object.

Fast lateral surface speed

Figure 6-6 displays the MIO force and position histories for the same stroking sequence shown in Figure 6-4, with the exception that the surface is traversed three times while the MIO is in contact. Although the average normal force f_z is similar to that of the previous case, the maximum lateral force f_x reaches a maximum magnitude of almost 2 N during each traversal due to the increased lateral speed. Similarly, the lateral force perpendicular to the traversal

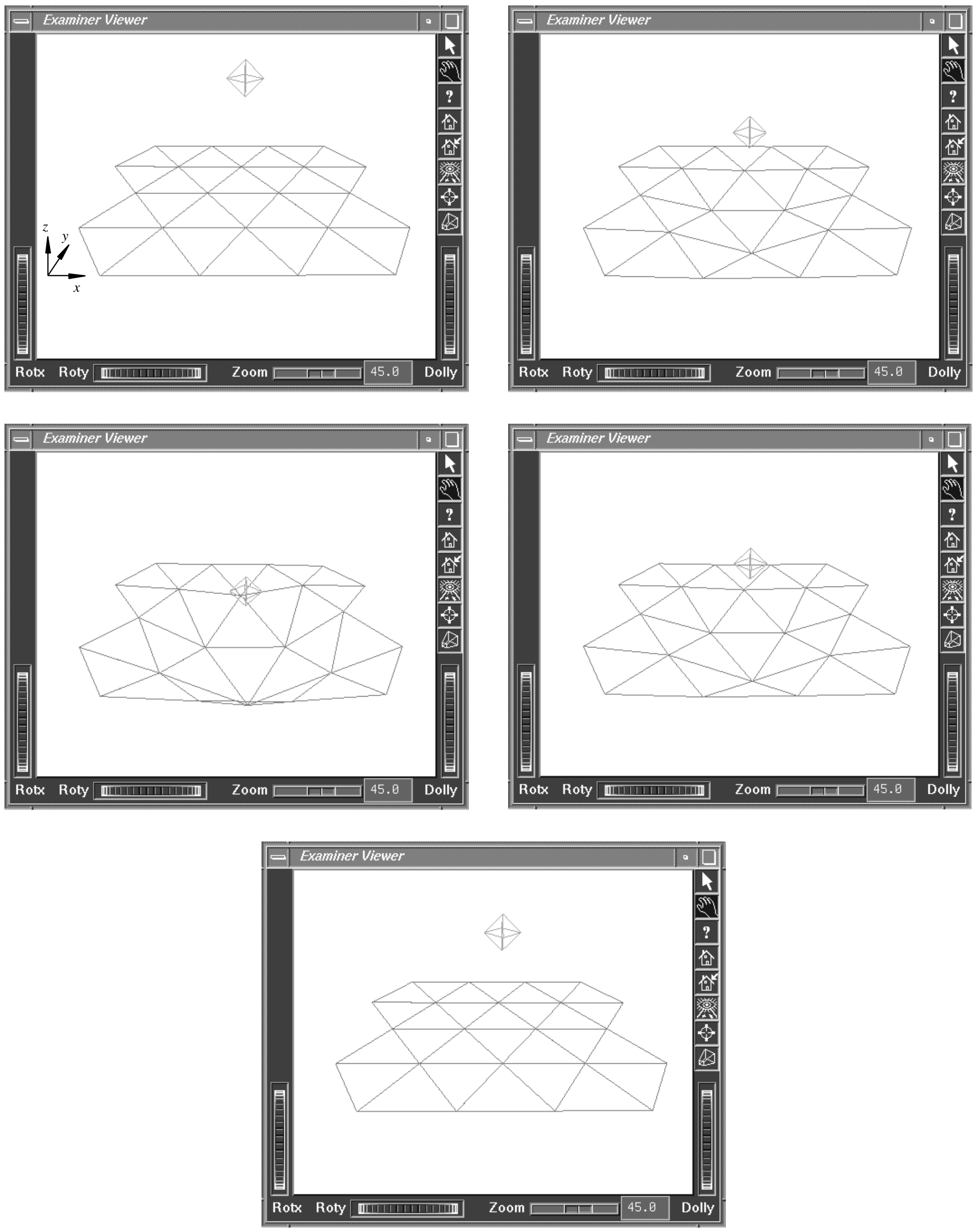

Figure 6-1: Sequence of visual frames for palpation interaction (from left to right, top to bottom).

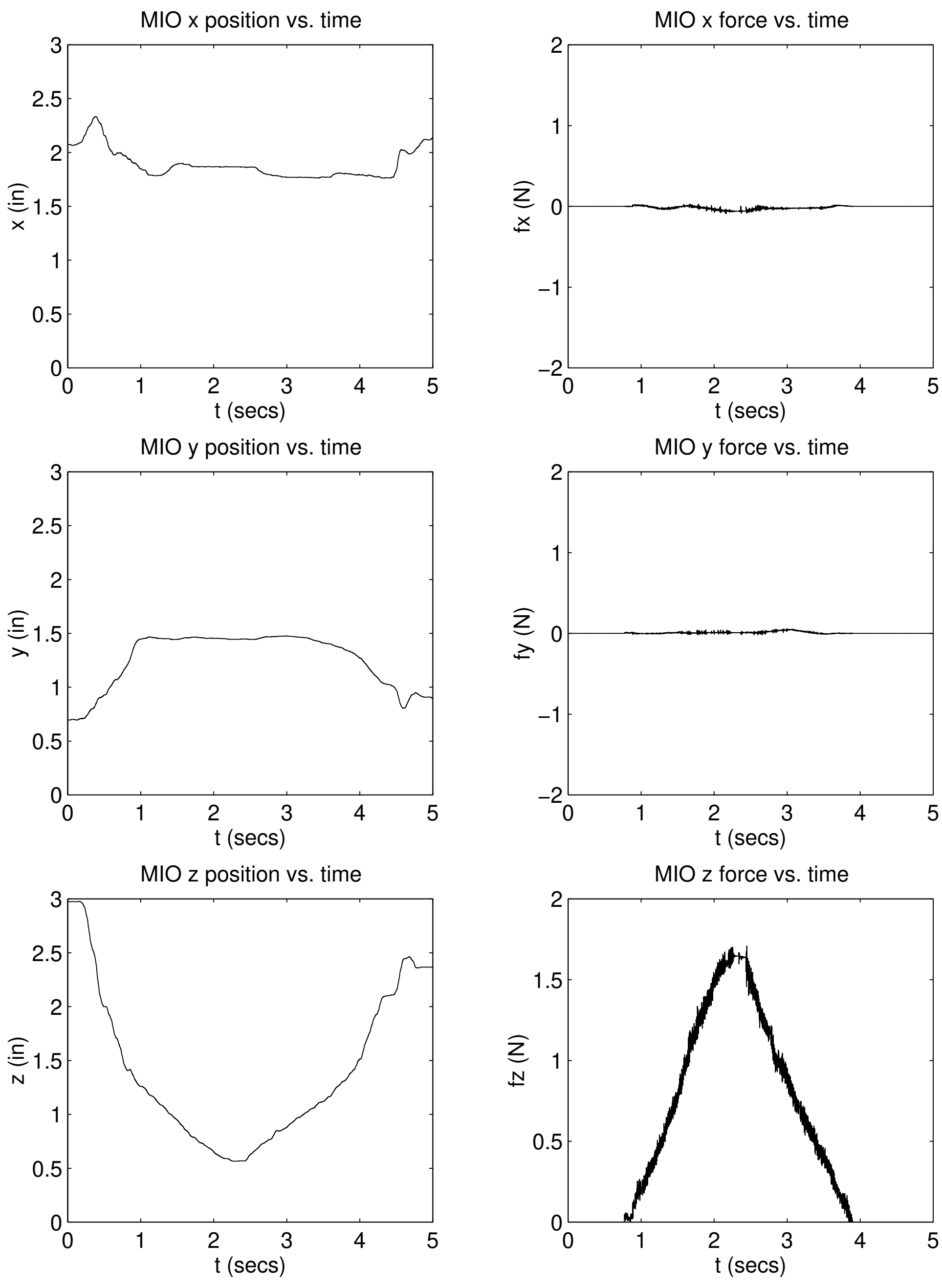

Figure 6-2: Position and force histories for the MIO during palpation.

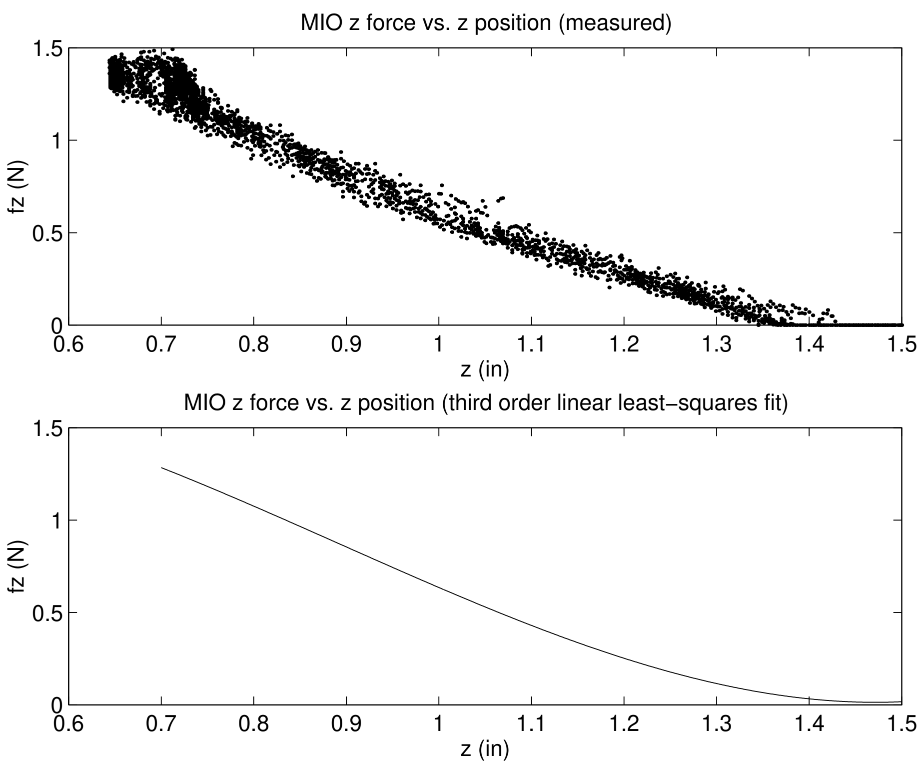

Figure 6-3: Interaction stiffness normal to surface for palpation: (top) measured, (bottom) linear least squares fit.

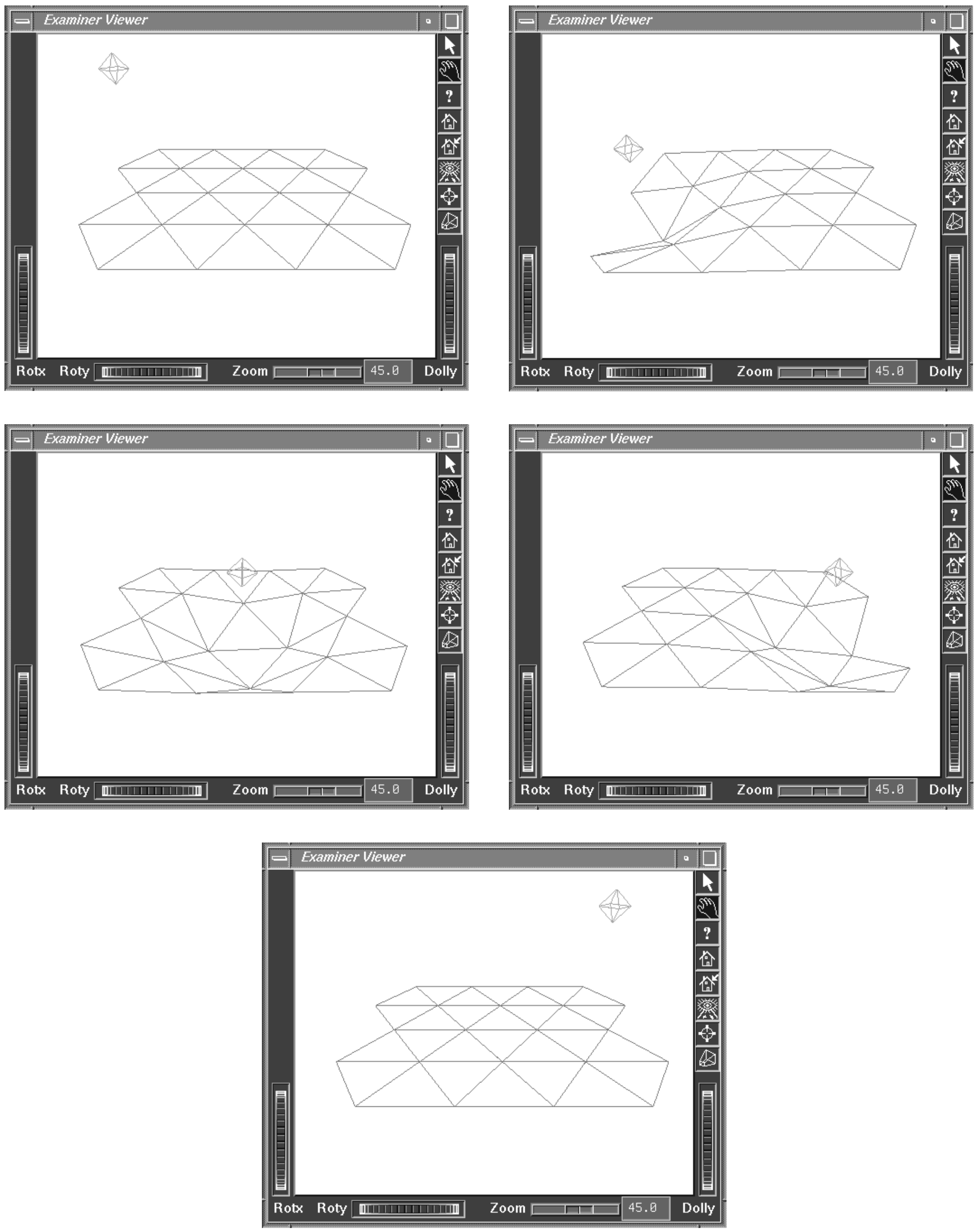

Figure 6-4: Sequence of visual frames for stroking interaction (from left to right, top to bottom).

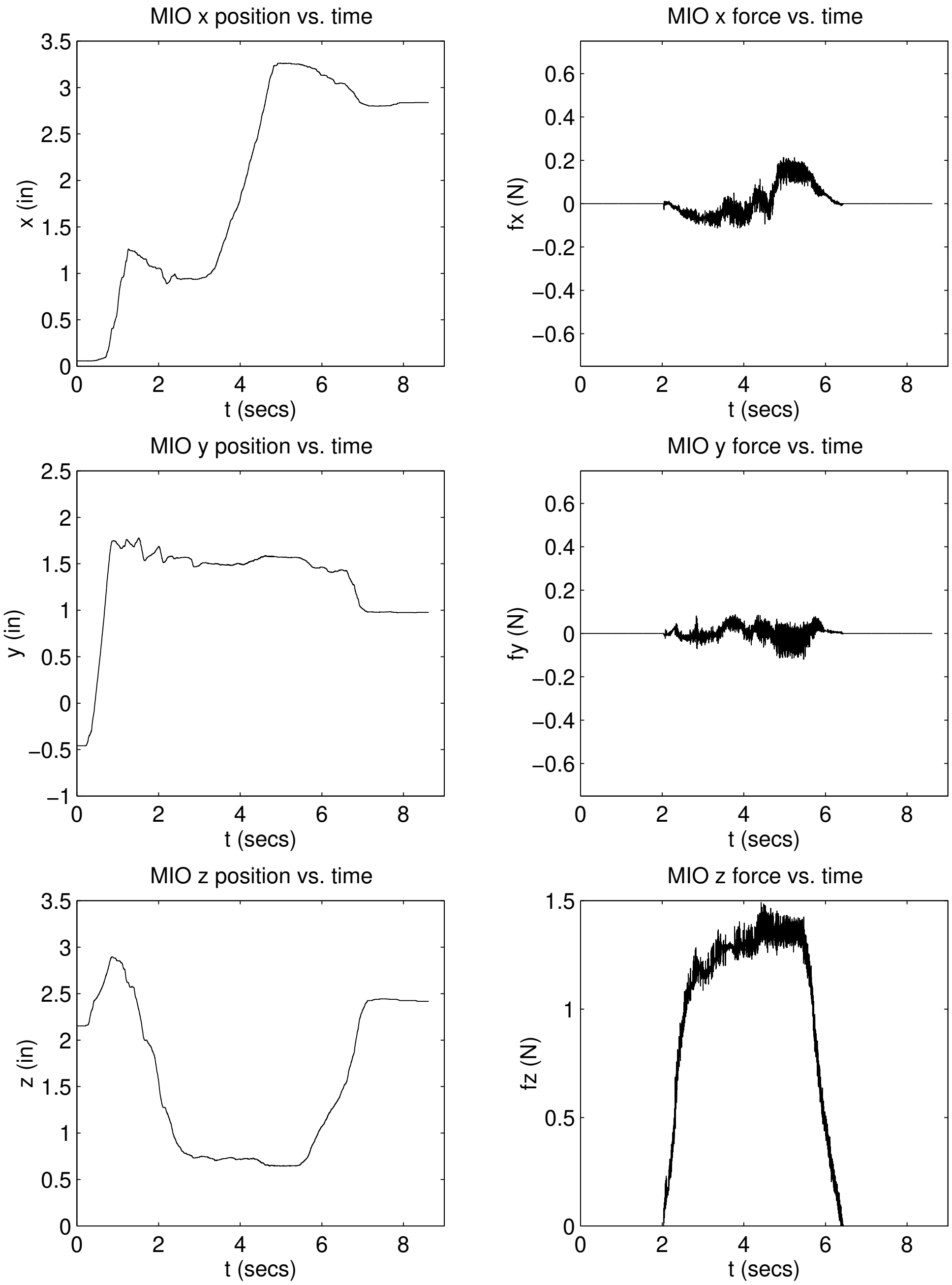

Figure 6-5: Position and force histories for the MIO during a stroking interaction using a slow lateral surface speed.

path, f_y, also increases dramatically because of the rapid transfer of momentum to atoms as they are pushed aside.

6.1.3 Plucking

Using the tweezers shown in Figure 5-4 as an interaction tool, the surface can be plucked, manipulated, and held. The MIO is slightly modified to model the grasping ability of the tweezers as follows. If the MIO is in contact with the surface when the tweezers are closed, a sorting algorithm is executed to find the nearest atom, of those that are in contact with the MIO, to the center of the MIO. A new element $\mathcal{E}$ is created that connects the nearest atom and the center of the MIO with an equilibrium length equal to the distance between the atom and the MIO center at the time of closure. The stiffness of this element is assigned the same stiffness as that of the MIO surface to simulate a rigid attachment with the atom. If the tweezers are opened, the connecting element is deleted, hence allowing the contact atom to return to equilibrium. A sample plucking sequence is shown in Figure 6-7.

Figure 6-8 shows the position and force histories of the MIO during an example plucking sequence with a media fragment. While the MIO is in contact with the media, the user closes the tweezers hence creating the connecting element between the nearest contact atom and the MIO center. When the MIO is no longer in contact with the surface, the MIO force represents the connecting element force. At $t \approx 4$ sec, the tweezers are opened and the MIO force instantly falls to zero magnitude. There is a slight perturbation in MIO position as the human stabilizes after receiving a step function in force.

6.2 Local Simulation

The sequence shown in Figure 6-9 illustrates the stroking interaction with a higher resolution media fragment. The curvature of the fragment was generated by draping it onto a spherical form and consequently saving the state of the new model. The local simulation option is enabled using a haptic window radius of $\mathcal{MIO}{\rightarrow}r_l = 2.25$ inches. Although the haptic display refresh rate suffers slightly in this particular example due to the increase in spatial density of atoms, the refresh rate is largely uncoupled from the size of the media fragment. Recall from Section 5.5.2, some coupling is still present because the collision detection algorithm searches the entire atom list for potential collision atoms. As described previously, however, complete uncoupling would occur if octrees were used to spatially segment the DSM model.

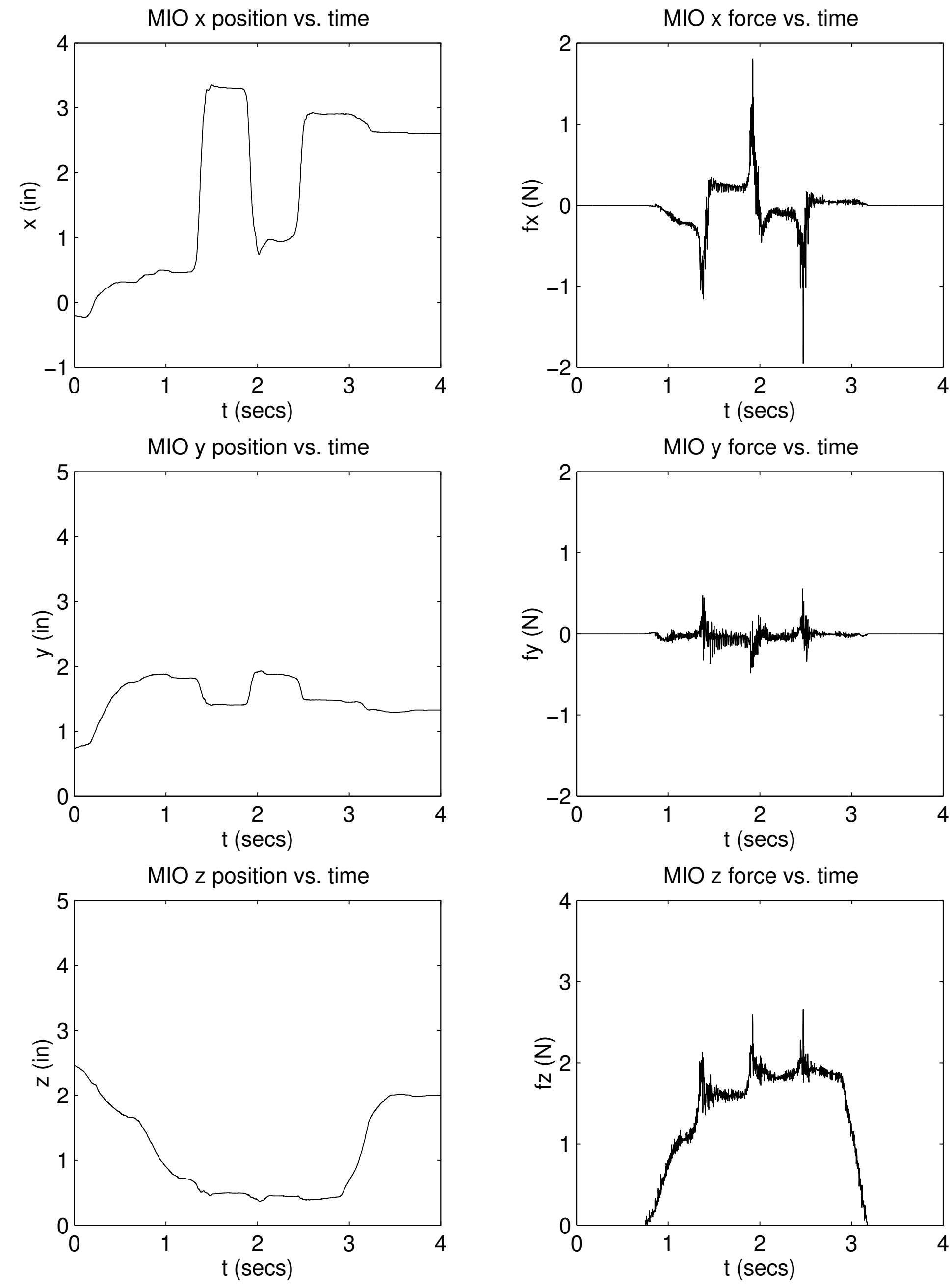

Figure 6-6: Position and force histories for the MIO during a stroking interaction using a fast lateral surface speed.

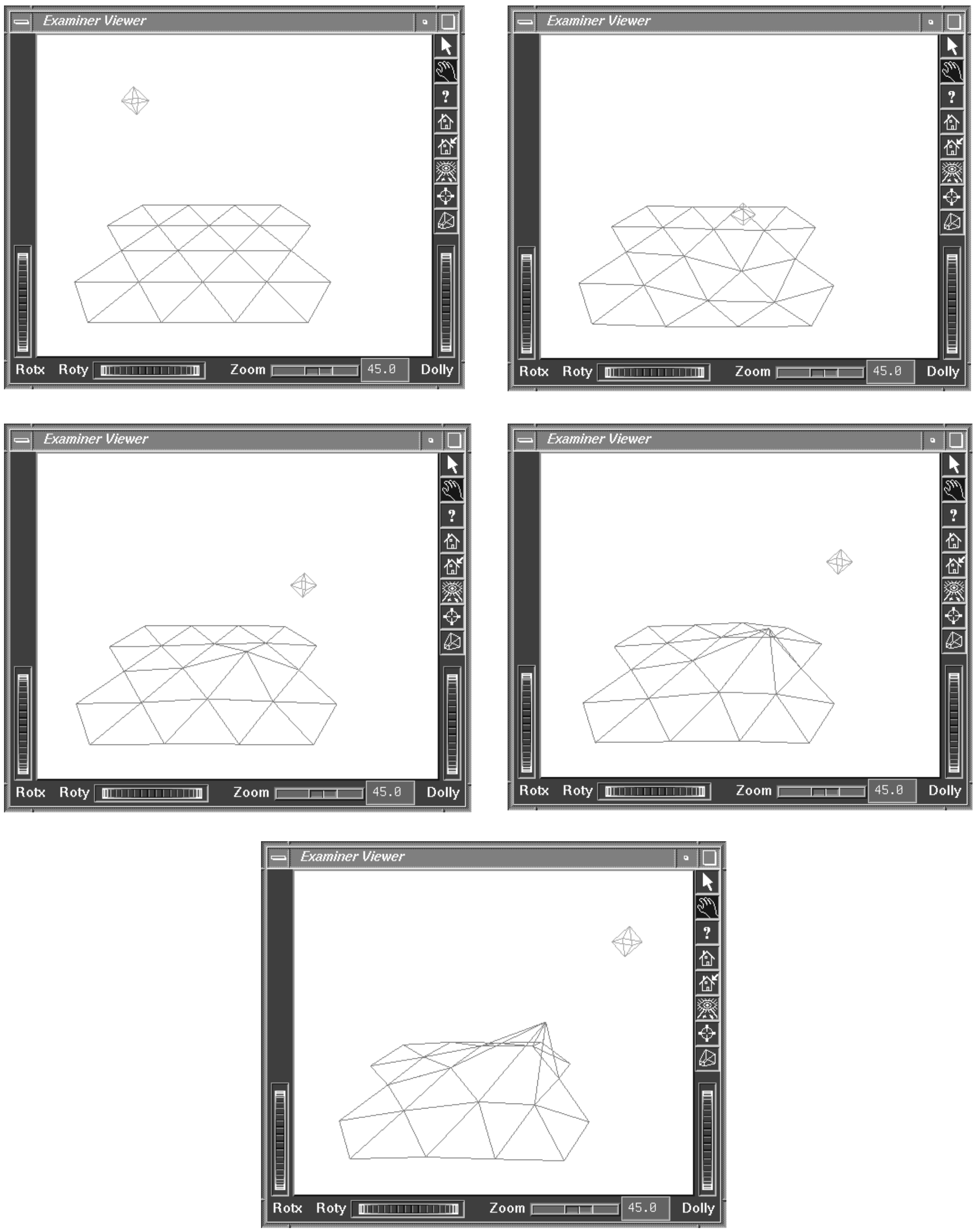

Figure 6-7: Sample plucking sequence using an instrumented pair of tweezers (from left to right, top to bottom).

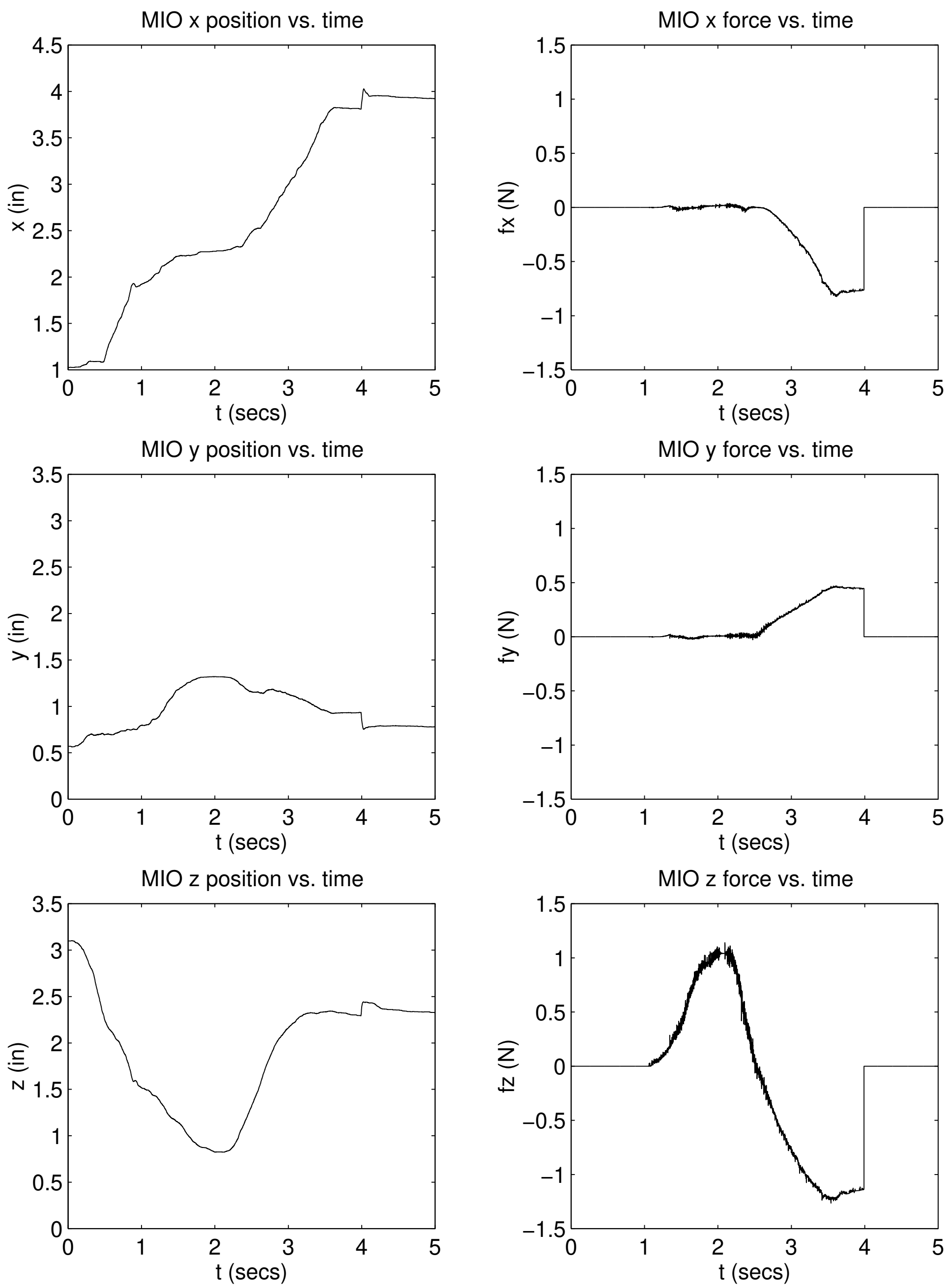

Figure 6-8: Position and force histories for the MIO during a plucking sequence.

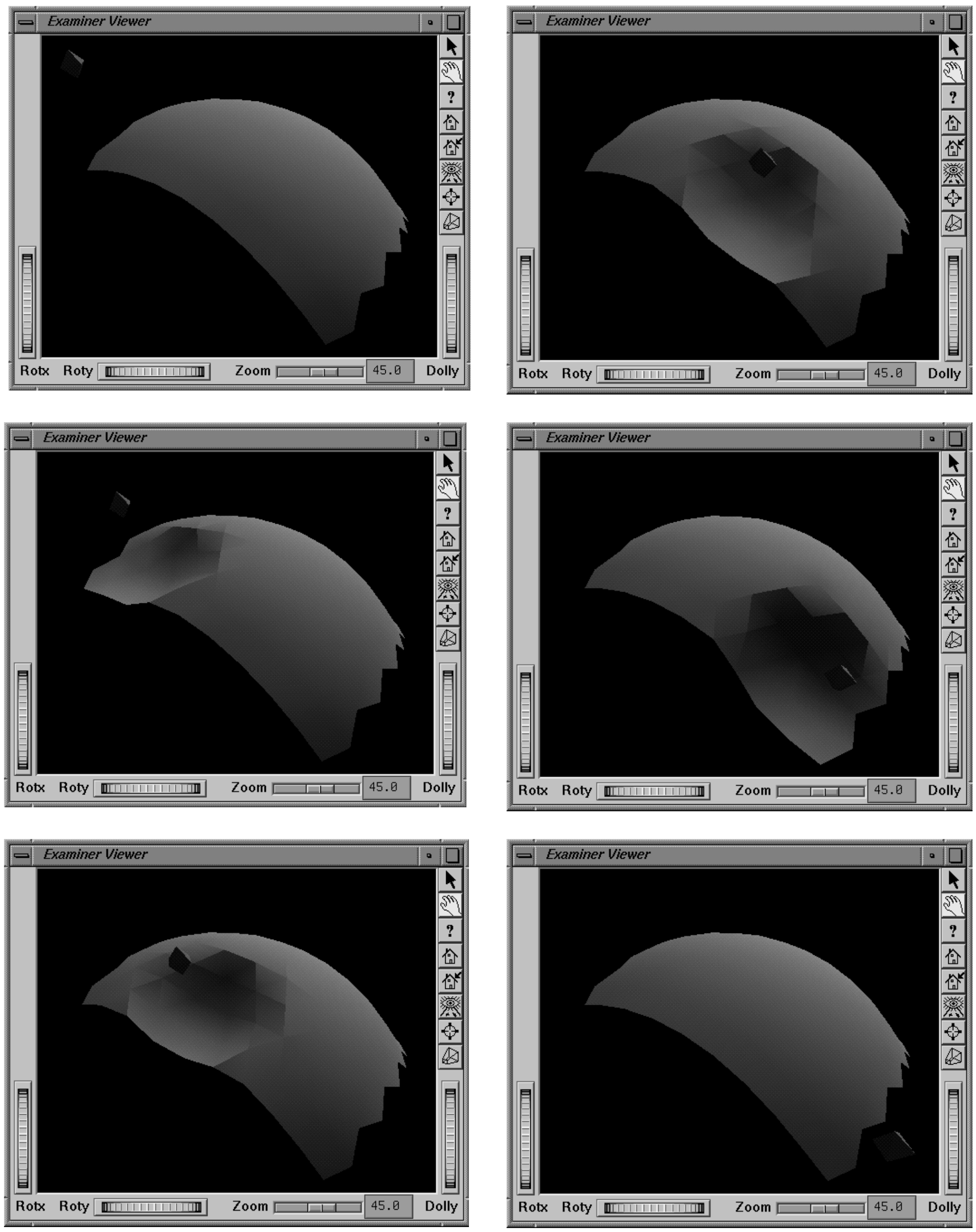

Figure 6-9: Stroking sequence with a media fragment using the local simulation option (from top to bottom, left to right).

Chapter 7

Conclusions

7.1 Summary

In summary, Interactive Deformable Media Simulator (IDMS) is a real-time dynamic simulation that allows users to interact both haptically and visually with deformable media in a virtual environment setting. Haptic feedback is provided through the Phantom haptic interface and visual feedback is presented by OpenInventor based graphics software on the Silicon Graphics Indigo II Extreme workstation. A finite element approach is utilized to model the structural properties of deformable media. Real-time computation is achieved by (a) using a surface representation coupled with a surface restoring force to encapsulate volumetric properties; (b) conserving computation power by rendering media local to the interaction point; and (c) evaluating the simulation within the topology of the model. The discretized representation of the media facilitates creation, modification, and storage of model data. The Master Interaction Object (MIO) is a fixed geometric shape that efficiently calculates interaction force based on point intersection and surface penetration distance normal to the MIO boundary. Other interaction modalities such as tweezers are easily integrated into IDMS by appropriate modification of the MIO force generation algorithm. The following section describes further research that would enhance the performance and functionality of IDMS.

7.2 Future Work

7.2.1 Computational architecture

The DSM framework in IDMS is highly amenable to a parallel processing architecture. Additional processors could be assigned mutually exclusive sections of media for simulation purposes. Model data would be stored in common memory to ensure data coherency and to facilitate communication of data between processors. Furthermore, a real-time preemptive operating system such as LynxOS for the haptic engine is critical to evenly distribute processing between the haptic and environment processes and the shared memory process. Currently, under Linux, the haptic process signals the shared memory process to begin a data transfer to the visual engine at 30 Hz. Although not a factor for simple models, the delay induced by transferring visual information for complex models may become haptically noticeable at the 30 Hz frequency. A real-time operating system will allow for context switching between processes at much higher service rates than possible in Linux, removing the need for synchronization between haptic and shared memory processes.

7.2.2 Dynamics Simulation

In order to achieve real-time response, local simulation is utilized by IDMS to calculate interaction force. For the class of media addressed in this thesis, this technique is sufficient for creating realistic interactions. Media in which local interactions are more globally propagated throughout the media may require additional concurrent dynamics processes to update the model state. These "global dynamics" processes may run at slower update rates than the "local dynamics" process depending on the frequency content present in the media. This synchronous execution of multiple time-scale dynamics processes fits easily into the concurrent run-time architecture of IDMS.

7.2.3 Spatial data structures

Integrating spatial data structures into IDMS will greatly improve computation performance. Currently, the entire list of surface atoms is traversed to check for collision detection with the MIO. A more efficient approach would be to segment the model into a multi-resolution spatial hierarchy. Octrees and bounding boxes would easily serve these purposes. This procedure could be done as a preprocessing step to IDMS or could be incorporated into the `plg` and `fem` data file formats.

7.2.4 Haptic interactions

In addition to the fundamental haptic interactions of palpation, stroking, and plucking, other more complex haptic interactions should be modeled and implemented. Each interaction may involve independent modification of the interaction force algorithm or require a special tool to be inserted in the Phantom gimbal in order to model the real world interaction. Examples of potential interactions include sewing, suturing, piercing, and incising. Surface effects such as friction and stiction could also be implemented as external forces applied to individual atoms and the MIO.

7.2.5 Model generation and filling

In order to exploit the variety of models that IDMS can simulate, more advanced techniques need to be created for generating model geometries and assigning values to physical parameters for atoms and elements. As described in Section 3.2.1, a haptic interface with an instrumented end effector could be used to measure surface impedance and geometry data. The home elements provide a compact representation of surface impedance and hence methods for assigning home element values based on measurement data need to be developed. Depending on the context, these *scanning* and *filling* techniques may be required to be nondestructive to preserve the state of the real object.

7.2.6 Visualization

Currently, the media is visually displayed in either shaded or wireframe graphics modes. In the shaded mode, each polygon specified by the `plg` file is uniformly shaded based on lighting direction. In wireframe representation, only each surface element is displayed. In each of these modes, the surface smoothness suffers for large deformations due to the spatial discretization of atoms. Two-dimensional parametric interpolation polynomials could be incorporated utilizing the positions of the surface atoms as control points to provide a visually smooth deformation. B-splines are especially suited for local deformation because they utilize a set of blending functions that *locally* influence shape changes, depending only on neighboring control points to produce changes in curvature [14].

Depending on the object being modeled, texture mapping may add significant realism to the visual display. Skin, for example, could be texture mapped onto biological objects to enhance visual feedback. The SGI Indigo II Extreme platform cannot handle real-time texture mapping and thus more powerful computation hardware would be necessary. The SGI Reality Engine, a multiprocessor based hardware architecture, would ideally serve these and other graphics intensive purposes.

Appendix A

Notation

Symbol	*Meaning (units if applicable)*
IDMS	Interactive Deformable Media Simulator
DSM	Discrete Simulation Model
MIO	Master Interaction Object
$\mathbf{p}$	absolute position
$\mathbf{p}_{eq}$	absolute equilibrium position
$\mathbf{x_1}$	differential position
$\mathbf{x_2}$	differential velocity
$\mathbf{f}$	force
num	number in sequence
bc	boundary condition (fixed or free)
sc	surface condition (external or internal)
spc	spatial condition (local or non-local)
m	mass
k	stiffness
b	viscosity
l	length
l_{eq}	equilibrium length
r	radius
r_l	local radius
R_f	spherical form radius
k_f^i	form surface stiffness
k_f^o	form attractive stiffness
b_f	form surface viscosity
T	simulation step size (sec)
$\mathcal{A}$	atom data structure
$\mathcal{E}$	element data structure
$\mathcal{M}$	model data structure
$\mathcal{MIO}$	MIO data structure
$\rightarrow$	dereferencing operator, $\mathcal{A}\rightarrow m$ is the mass stored in atom $\mathcal{A}$
N	number of model atoms

L	number of model elements
NS	number of model surface atoms

Appendix B

Data File Formats

B.1 `plg` files

B.1.1 Description

The `plg` file description is one of several widely used data file formats for describing three-dimensional polyhedra. These file formats are used for a variety of purposes including computer-aided design, computer graphics, and modeling. A brief description of the `plg` file format is provided here. The `plg` file format is essentially a subset of the substantially more versatile `obj` file format provided by Wavefront Technologies [25], that additionally allows for free-form curves and surfaces, material properties, nested grouping of objects, and many other graphics attributes.

The format of a `plg` file, shown in Figure B-1 is a listing of vertices and polygons that comprise the polyhedra, each listed on a separate line. The first line of the file contains, in order, the ASCII name of the object, the number of vertices, and the number of polygons. Each vertex entry has the x, y, and z coordinates that define its position. Each polygon entry contains the number of sides and a list of vertex numbers that comprise the polygon, the first vertex being numbered 0. A whitespace separates the first line, the listing of vertices, and the listing of polygons.

{name} {nv} {np}

{x} {y} {z}
⋮
{x} {y} {z}

{number of sides} {v1} {v2} ...{vn}
⋮
{number of sides} {v1} {v2} ...{vn}

Figure B-1: `plg` file format.

B.1.2 Example: hex_4x5_home.plg

The plg file for the fragment of media shown in Figure 3-1 is shown below.

```
hex 22 28

0.500 0.000 0.000
1.500 0.000 0.000
2.500 0.000 0.000
3.500 0.000 0.000
0.000 0.866 0.000
1.000 0.866 0.000
2.000 0.866 0.000
3.000 0.866 0.000
4.000 0.866 0.000
0.500 1.732 0.000
1.500 1.732 0.000
2.500 1.732 0.000
3.500 1.732 0.000
0.000 2.598 0.000
1.000 2.598 0.000
2.000 2.598 0.000
3.000 2.598 0.000
4.000 2.598 0.000
0.500 3.464 0.000
1.500 3.464 0.000
2.500 3.464 0.000
3.500 3.464 0.000

3 0 1 5
3 1 2 6
3 2 3 7
3 4 5 9
3 5 4 0
3 5 6 10
3 6 5 1
3 6 7 11
3 7 6 2
3 7 8 12
3 8 7 3
3 9 10 14
3 10 9 5
3 10 11 15
3 11 10 6
3 11 12 16
3 12 11 7
```

```
3 13 14 18
3 14 13 9
3 14 15 19
3 15 14 10
3 15 16 20
3 16 15 11
3 16 17 21
3 17 16 12
3 18 14 19
3 19 15 20
3 20 16 21
```

B.2 `fem` files

B.2.1 Description

The `fem` file format was developed for IDMS to describe networks of atoms and connecting constitutive elements. This format changes as increases in modeling complexity within IDMS require more physical parameters or initial conditions to be specified.

The specific format of the `fem` file is shown in Figure B-2. The first line of the file contains, in order, the name of the object, the number of atoms, and the number of elements. Each atom entry has the x, y, and z coordinates that define the atom's initial position; the atom mass; the boundary condition (fixed or free); and the surface condition (on surface or not). Each element entry contains the numbers of the two atoms the element connects, the element stiffness, and the element viscosity. The first atom in the atom list is numbered zero. A whitespace separates the first line, the listing of atoms, and the listing of elements.

```
{name} {na} {ne}

{x} {y} {z} {mass} {boundary condition} {surface condition}
⋮
{x} {y} {z} {mass} {boundary condition} {surface condition}

{a1} {a2} {stiffness} {viscosity}
⋮
{a1} {a2} {stiffness} {viscosity}
```

Figure B-2: `fem` file format.

B.2.2 Example: `hex_4x5_home.fem`

The `fem` file for the fragment of media shown in Figure 3-1 is shown below. Note that home atoms and home elements are included.

```
hex 58 141

0.500 0.000 0.000 0.002000 1001 2001
1.500 0.000 0.000 0.002000 1001 2001
2.500 0.000 0.000 0.002000 1001 2001
3.500 0.000 0.000 0.002000 1001 2001
0.000 0.866 0.000 0.002000 1001 2001
1.000 0.866 0.000 0.002000 1001 2001
2.000 0.866 0.000 0.002000 1001 2001
3.000 0.866 0.000 0.002000 1001 2001
4.000 0.866 0.000 0.002000 1001 2001
0.500 1.732 0.000 0.002000 1001 2001
```

```
1.500 1.732 0.000 0.002000 1001 2001
2.500 1.732 0.000 0.002000 1001 2001
3.500 1.732 0.000 0.002000 1001 2001
0.000 2.598 0.000 0.002000 1001 2001
1.000 2.598 0.000 0.002000 1001 2001
2.000 2.598 0.000 0.002000 1001 2001
3.000 2.598 0.000 0.002000 1001 2001
4.000 2.598 0.000 0.002000 1001 2001
0.500 3.464 0.000 0.002000 1001 2001
1.500 3.464 0.000 0.002000 1001 2001
2.500 3.464 0.000 0.002000 1001 2001
3.500 3.464 0.000 0.002000 1001 2001
0.500 0.000 0.000 0.002000 1000 2000
1.500 0.000 0.000 0.002000 1000 2000
2.500 0.000 0.000 0.002000 1000 2000
3.500 0.000 0.000 0.002000 1000 2000
0.000 0.866 0.000 0.002000 1000 2000
1.000 0.866 0.000 0.002000 1000 2000
2.000 0.866 0.000 0.002000 1000 2000
3.000 0.866 0.000 0.002000 1000 2000
4.000 0.866 0.000 0.002000 1000 2000
0.500 1.732 0.000 0.002000 1000 2000
1.500 1.732 0.000 0.002000 1000 2000
2.500 1.732 0.000 0.002000 1000 2000
3.500 1.732 0.000 0.002000 1000 2000
0.000 2.598 0.000 0.002000 1000 2000
1.000 2.598 0.000 0.002000 1000 2000
2.000 2.598 0.000 0.002000 1000 2000
3.000 2.598 0.000 0.002000 1000 2000
4.000 2.598 0.000 0.002000 1000 2000
0.500 3.464 0.000 0.002000 1000 2000
1.500 3.464 0.000 0.002000 1000 2000
2.500 3.464 0.000 0.002000 1000 2000
3.500 3.464 0.000 0.002000 1000 2000
0.500 0.289 -0.817 0.002000 1001 2000
1.500 0.289 -0.817 0.002000 1001 2000
2.500 0.289 -0.817 0.002000 1001 2000
0.000 1.155 -0.817 0.002000 1001 2000
1.000 1.155 -0.817 0.002000 1001 2000
2.000 1.155 -0.817 0.002000 1001 2000
3.000 1.155 -0.817 0.002000 1001 2000
0.500 2.021 -0.817 0.002000 1001 2000
1.500 2.021 -0.817 0.002000 1001 2000
2.500 2.021 -0.817 0.002000 1001 2000
0.000 2.887 -0.817 0.002000 1001 2000
```

```
1.000 2.887 -0.817 0.002000 1001 2000
2.000 2.887 -0.817 0.002000 1001 2000
3.000 2.887 -0.817 0.002000 1001 2000

0 1 15.000000 0.200000
1 2 15.000000 0.200000
2 3 15.000000 0.200000
4 5 15.000000 0.200000
5 6 15.000000 0.200000
6 7 15.000000 0.200000
7 8 15.000000 0.200000
9 10 15.000000 0.200000
10 11 15.000000 0.200000
11 12 15.000000 0.200000
13 14 15.000000 0.200000
14 15 15.000000 0.200000
15 16 15.000000 0.200000
16 17 15.000000 0.200000
18 19 15.000000 0.200000
19 20 15.000000 0.200000
20 21 15.000000 0.200000
0 4 15.000000 0.200000
0 5 15.000000 0.200000
4 9 15.000000 0.200000
5 9 15.000000 0.200000
1 5 15.000000 0.200000
1 6 15.000000 0.200000
5 10 15.000000 0.200000
6 10 15.000000 0.200000
2 6 15.000000 0.200000
2 7 15.000000 0.200000
6 11 15.000000 0.200000
7 11 15.000000 0.200000
3 7 15.000000 0.200000
3 8 15.000000 0.200000
7 12 15.000000 0.200000
8 12 15.000000 0.200000
9 13 15.000000 0.200000
9 14 15.000000 0.200000
13 18 15.000000 0.200000
14 18 15.000000 0.200000
10 14 15.000000 0.200000
10 15 15.000000 0.200000
14 19 15.000000 0.200000
15 19 15.000000 0.200000
```

```
11 15 15.000000 0.200000
11 16 15.000000 0.200000
15 20 15.000000 0.200000
16 20 15.000000 0.200000
12 16 15.000000 0.200000
12 17 15.000000 0.200000
16 21 15.000000 0.200000
17 21 15.000000 0.200000
0 22 15.000000 0.200000
1 23 15.000000 0.200000
2 24 15.000000 0.200000
3 25 15.000000 0.200000
4 26 15.000000 0.200000
5 27 15.000000 0.200000
6 28 15.000000 0.200000
7 29 15.000000 0.200000
8 30 15.000000 0.200000
9 31 15.000000 0.200000
10 32 15.000000 0.200000
11 33 15.000000 0.200000
12 34 15.000000 0.200000
13 35 15.000000 0.200000
14 36 15.000000 0.200000
15 37 15.000000 0.200000
16 38 15.000000 0.200000
17 39 15.000000 0.200000
18 40 15.000000 0.200000
19 41 15.000000 0.200000
20 42 15.000000 0.200000
21 43 15.000000 0.200000
44 0 15.000000 0.200000
44 1 15.000000 0.200000
44 5 15.000000 0.200000
45 1 15.000000 0.200000
45 2 15.000000 0.200000
45 6 15.000000 0.200000
46 2 15.000000 0.200000
46 3 15.000000 0.200000
46 7 15.000000 0.200000
47 4 15.000000 0.200000
47 5 15.000000 0.200000
47 9 15.000000 0.200000
48 5 15.000000 0.200000
48 6 15.000000 0.200000
48 10 15.000000 0.200000
```

```
49 6 15.000000 0.200000
49 7 15.000000 0.200000
49 11 15.000000 0.200000
50 7 15.000000 0.200000
50 8 15.000000 0.200000
50 12 15.000000 0.200000
51 9 15.000000 0.200000
51 10 15.000000 0.200000
51 14 15.000000 0.200000
52 10 15.000000 0.200000
52 11 15.000000 0.200000
52 15 15.000000 0.200000
53 11 15.000000 0.200000
53 12 15.000000 0.200000
53 16 15.000000 0.200000
54 13 15.000000 0.200000
54 14 15.000000 0.200000
54 18 15.000000 0.200000
55 14 15.000000 0.200000
55 15 15.000000 0.200000
55 19 15.000000 0.200000
56 15 15.000000 0.200000
56 16 15.000000 0.200000
56 20 15.000000 0.200000
57 16 15.000000 0.200000
57 17 15.000000 0.200000
57 21 15.000000 0.200000
44 45 15.000000 0.200000
45 46 15.000000 0.200000
47 48 15.000000 0.200000
48 49 15.000000 0.200000
49 50 15.000000 0.200000
51 52 15.000000 0.200000
52 53 15.000000 0.200000
54 55 15.000000 0.200000
55 56 15.000000 0.200000
56 57 15.000000 0.200000
44 47 15.000000 0.200000
44 48 15.000000 0.200000
47 51 15.000000 0.200000
48 51 15.000000 0.200000
45 48 15.000000 0.200000
45 49 15.000000 0.200000
48 52 15.000000 0.200000
49 52 15.000000 0.200000
```

```
46 49 15.000000 0.200000
46 50 15.000000 0.200000
49 53 15.000000 0.200000
50 53 15.000000 0.200000
51 54 15.000000 0.200000
51 55 15.000000 0.200000
52 55 15.000000 0.200000
52 56 15.000000 0.200000
53 56 15.000000 0.200000
53 57 15.000000 0.200000
```

Bibliography

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

[2] J. Edward Colgate and J. Michael Brown. Factors affecting the z-width of a haptic interface. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994.

[3] J. Edward Colgate and Gerd Schenkel. Passivity of a class of sampled-data systems: Application to haptic interfaces. In *Proceedings of the American Control Conference*, 1994.

[4] Steven A. Cover et al. Interactively deformable models for surgery simulation. *IEEE Computer Graphics and Applications*, 13(6), November 1993.

[5] Nathaniel Durlach and Anne S. Mavor, editors. *Virtual Reality: Scientific and Technological Challenges.* National Academy of Sciences, Washington, D.C., 1994.

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

[7] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1989.

[8] Neville Hogan. Controlling impedance at the man/machine interface. In *IEEE International Conference on Robotics and Automation*, pages 1626–1631, 1989.

[9] Tetsuo Kotoku, Kiyoshi Komoriya, and Kazuon Tanie. A robot simulator with force generating function-configuration space approach. In *Proceedings of the International Symposium on Measurement and Control in Robotics*, pages 805–810, 1992.

[10] Tetsuo Kotoku, Kouichi Takamne, and Kazuo Tanie. A virtual environment display with constraint feeling based on position/force control switching. *IEEE International Workshop on Robot and Human Communication*, 1994.

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

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

[13] Margaret Minsky et al. Feeling and seeing: Issues in force display. In *Computer Graphics, Proceedings of SIGGRAPH Symposium on 3D Real-Time Interactive Graphics*, volume 24, 1990.

[14] Michael E. Mortenson. *Geometric Modeling.* John Wiley and Sons, New York, 1985.

[15] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-time Signal Processing.* Prentice Hall, Englewood cliffs, New Jersey, 1989.

[16] Alex Pentland et al. The thingworld modeling system: Virtual sculpting by modal forces. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*, 1990.

[17] Alex P. Pentland. Computational complexity versus simulated environments. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*, 1990.

[18] Steven D. Pieper. More than skin deep: Physical modeling of facial tissue. Master's thesis, Massachusetts Institute of Technology, School of Architecture and Planning, January 1989.

[19] Steven D. Pieper. *CAPS: Computer-Aided Plastic Surgery.* PhD dissertation, Massachusetts Institute of Technology, School of Architecture and Planning, February 1992.

[20] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, New York, NY, 1992.

[21] 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.

[22] Gilbert Strang. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, San Diego, 1988.

[23] D. Terzopoulos et al. Elastically deformable models. *Computer Graphics, Proceedings of SIGGRAPH*, 21(4):205–214, July 1987.

[24] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Compter Graphics, Proceedings of SIGGRAPH*, 22(4):269–278, August 1988.

[25] Wavefront Technologies, Santa Barbara, CA. *Wavefront Advanced Visualization Software Manual, Appendix B1: Object Files (.obj).*

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

[27] Craig Zilles and J. Kenneth Salisbury. A constraint-based god object method for haptic display. In *Proceedings of IROS-95*, Pittsburgh, August 1995.