More Than Skin Deep: Physical Modeling of Facial Tissue

by

Steven Donald Pieper

B.A., University of California at Berkeley (1984)

Submitted to the Media Arts and Sciences Section
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1989

Signature of Author .... [illegible]
Steven Donald Pieper
Media Arts and Sciences Section
January 13, 1989

Certified by .... [illegible]
David L. Zeltzer
Associate Professor of Computer Graphics
Thesis Supervisor

Accepted by .... [illegible]
Stephen A. Benton
Chairman, Departmental Committee on Graduate Students

More Than Skin Deep: Physical Modeling of Facial Tissue

by

Steven Donald Pieper

Submitted to the Media Arts and Sciences Section
on January 13, 1989, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis describes a computer system for animating human facial tissue using a mathematical model of the stresses and strains caused by deformation of the tissue. The model is based on networks, the nodes of which represent reference points within the tissue, and the arcs of which are spring-like constraints which model the mechanical behavior of the tissue between the reference points. The networks can be assembled to approximate the anatomy of facial tissue. The system simulates the action of layers of soft tissue as they interact with underlying hard tissue, internal muscle forces, and external forces such as gravity. The ability to combine the effect of these forces has been missing in previous animation models of facial tissue. The system is part of a project to develop real-time animation systems in which the simulated objects respond as if they were real objects. An application of the physical model of facial tissue and real-time manipulation of simulated objects includes a simulator for planning surgery and training plastic surgeons. The techniques described here can also be used to simulate facial tissue for computer animated human figures and create more realistic facial expressions than available in previous animation models. The facial tissue simulation system comprises several modules of an interactive simulation environment, called *bolio*, which is also briefly described here.

Thesis Supervisor: David L. Zeltzer
Title: Associate Professor of Computer Graphics

Contents

Chapter 1

Introduction

Physical simulations based on mathematical models have become an increasingly important application of computer technology in fields such as architectural and mechanical design where the reaction of a given structure to real-world forces is a critical design consideration. A goal of the research described here is to extend existing physical simulation techniques to the realm of biological tissue and explore the possibilities of physically-based simulation of human tissue, particularly the soft tissue of the face, for application in the fields of plastic surgery and computer animation, which would benefit from a predictive model of tissue behavior. Biological tissue presents perhaps the ultimate challenge of physical modeling, since it is at once a complex load-transferring structure and a living system. Living tissue is highly non-uniform in composition, and its response to forces is influenced by the age and health of the skin as well as its short- and long-term history of deformation. Conventional physically-based simulation techniques which were developed to predict the behavior of construction and manufacturing materials must be extended to model this behavior.

In this thesis, I describe a prototype system for modeling soft tissue (e.g., skin, fat, muscles, and tendons). The system models the interaction of multiple layers of tissue, the forces introduced by muscle contraction within layers of tissue, the motion of soft tissue as it travels over hard tissue (bone), and the effect of external forces such as gravity. The system uses a model based on networks of interacting force constraints acting on point masses. The network is arranged in a three layer lattice structure to model the skin surface and two layers of underlying fascia. Force constraints between the layers model the action of connective tissue and the volumetric effects of fatty tissue.

In addition to the investigation of the mathematical and other representational issues, a goal of this research has been to explore the implementation of real-time interactive physical simulation of biological tissues. Real time "human-in-the-loop" computer simulation has been applied successfully to operator training for land, sea, air, and space craft. In these systems, the evolving state of the simulation is presented to the user, usually through computer graphics. The user interacts with the simulation as if it were the real situation, and the simulator responds with the results of those actions. This thesis describes a real-time dynamic simulation system which allows a user wearing a DataGlove[1] to interact with the mass and spring elements which form the basis of the tissue simulation system. A user is able to "reach" into the simulated world and "grab" objects; if the grabbed object is connected to other objects via springs, then the movement is transferred along the springs to the other objects. The real-time spring example is built using the primitive components developed for the tissue model. Faster graphics and simulation hardware will extend the possibilities of real-time interaction to include more complex tissue models.

A future application of real-time interaction with complex biological models will be a *surgical* simulator. Such a simulator will allow a surgeon to explore various surgical techniques and observe the response of the patient's soft tissue. This would be of importance for plastic and reconstructive surgery, particularly surgery on the face, where the patient's appearance after surgery is an important measure of success. Since correct functioning of the face is defined in terms of its deformation under the applied forces of the facial muscles, gravity, etc., a simulator which models the dynamic behavior of the tissues would be a valuable tool. An important test of a physical model will be its ability to generate realistic images of the patient's face in various expressions. If a model of the patient's face built from scanned data (MRI, CT, or ultrasound) can be made to mimic the patient's expressions purely through the action of simulated facial muscles, then the model should be able to predict the way the same muscle actions will form expressions on the post-operative

[1]The DataGlove is an instrumented glove made by VPL Research, Inc., which digitizes the wearer's hand position and orientation as well as the flex angles of the wearer's fingers. Further description of the glove is given in section 4.5.2.

face. Accurate tissue simulation would provide a means of designing an operation for the maximum benefit of the patient. Since the predicted appearance of the patient would exist on a computer screen rather than merely in the surgeon's imagination, the patient would have both an opportunity to choose among possible surgical procedures and have a realistic idea of the expected results.

This thesis explores the fundamental requirements of a system to satisfy the long-term goal of developing such a surgical simulator. The remainder of this thesis is organized as follows:

- Chapter 2 reviews the medical, mathematical, and computational literature in order to describe the problem of predicting the behavior of soft tissue and the technologies available to address it;
- Chapter 3 outlines the desirable properties of a simulator, describes a mathematical model for simulating the behavior of soft tissue, and discusses how the current prototype addresses the ideal goals;
- Chapter 4 describes bolio, an interactive simulation system which uses a gesture-based user interface to control real-time simulations. Bolio serves as a testbed for the tissue simulation system;
- Chapter 5 presents a description of the soft tissue simulation module, how it approximates the action of biological materials, and how it has been added to bolio;
- Chapter 6 contains examples of simulations performed on the system and a comparison of the results with studies of the behavior of real tissue; and
- Chapter 7 proposes extensions to the model which will lead in the direction of practical application of this research.

Chapter 2

Background

To formulate an engineering solution to a problem, it is necessary to define the requirements of the problem and the techniques available to address the problem. Predicting the results of surgery on soft tissue requires an understanding of the properties of the tissue itself, and of the surgical techniques applied to it. The techniques available to address the problem include mathematical formulations which model the physical behavior of materials, computational implementations of those models, and interactive computer systems which allow experimentation with the parameters of the model and display of the results. The literature relevant to both the problem and the solution techniques is reviewed in this chapter.

2.1 The Processes to be Modeled

This section looks at the physical processes which are to be modeled in a surgical simulator. The subject matter is broken into three sections: the anatomical structure of soft tissue, the mechanical response of this tissue to applied loads and deformation, and the types of surgery which might be simúlated.

2.1.1 Anatomy and Function of Soft Tissue Layers

The soft tissue of the body has a complex microstructure of cell types and interacting networks which form layers over the underlying skeleton and internal organs.[1] The relative

[1]Material in this section is drawn primarily from *Gray's Anatomy*[23].

density of cell types and the thickness of the layers varies considerably over the surface of the body according to the functional requirements of the tissue in the region, as well as the age and health of the individual. Aspects of this microstructure and its functions are reviewed here; chapter 3 describes how much of this detail has been incorporated in the currently implemented system.

The outer layer of soft tissue on the body is the skin. The skin is divided into the *epidermis* or *cuticle* layer, the *dermis* or *cutis* layer, and the *subcutaneous* cellular tissue. The skin is a defensive covering which insulates the body from the environment in many ways. Its functional interactions include limiting evaporation, providing frictional and gripping properties, providing thermal insulation, and limiting damage from mechanical interaction with the physical environment, parasites, and predators.

The epidermis is made mainly of *keratin*, generally in the form of dead cells flattened in the plane of the surface[21]. In most parts of the body, the epidermis is thin and its cells are easily separated. On the palms and the soles of the feet, the epidermis is much thicker and stronger. The epidermis has nerve fibrils in its lower levels in a density depending on the sensitivity of the region, but is non-vascular (has no internal fluid transport mechanisms).

The dermis is made mostly of bundles of collagen fibers (70-80% of dry weight and up to 30% of wet weight[21]), much like the fibers in tendons, only arranged in a three dimensional weave. The rest of the dermis includes elastin fibers, numerous blood vessels, lymphatics (a fluid uptake system), and nerves. Like the epidermis, the thickness of the dermis varies over the body.

Underneath the dermis lies the *superficial fascia* (also called the *hypodermal layer*), which consists of adipose tissue (fat cells) distributed in a network of connective fibers, as well as the networks of vessels and nerves traveling near the surface of the body. The connective tissue is mostly collagen arranged in a lattice with fat cells distributed in the spaces; this structure expands to accommodate growth during fat deposition while retaining its shape and mechanical properties[13]. The fat in the superficial fascia is a poor conductor of heat and therefore it insulates the body. Tubes carry fluid from the sweat glands to pores on the skin surface, and hair follicles serve as ductways for secretions from the subcutaneous glands. Much of the texture and appearance of the skin is regulated by secretions of *subum* from the subcutaneous glands which are particularly dense on the forehead and nose. The glands and hair follicles are embedded in the subcutaneous cellular tissue. The intercellular fluid is referred to as *ground substance.*

Beneath the superficial fascia lies the *deep fascia* which coats the bones. This layer is mainly formed of aponeuroses, which are flat or ribbon-like tendons, which are made mainly of collagen. Between these two layers are the muscles. For most of the muscles of the body — for example, those which move articulated joints — both the *origin* (which generally refers to the immobile end of the muscle) and the *insertion* (the mobile end) connect to deep fascia tissue. The superficial fascia serves as the connection point for muscles which move the skin surface, such as the muscles of the face.

The muscles are composed of *fasciculi* (bundles) of fibers which are themselves composed of myofibrils, which are bundles of myofilaments. The myofilaments are composed of the proteins myocin and actin, which move past each other in response to nerve impulses[46]. Sensory receptors detect stretching and tension in the muscle and provide feedback to the higher level control units of the nervous system which coordinate action.

2.1.2 Tissue Mechanics

The amount and nature of soft tissue deformation under applied loads has been the subject of a number of biomechanical investigations [28][13][21][62][56]. These investigations approach the problem of tissue deformation from an engineering point of view similar to that taken by researchers in Materials Science and Continuum Mechanics[3], but the results obtained from experiments on living tissue show greater complexity than those on structural materials for construction and manufacturing, the materials for which much theory has been developed. These complexities make the application of analytical techniques more difficult.[2]

Skin Mechanics

As a unit, the soft tissue of the skin is *viscoelastic* in its responses to *stress* (force or load) and *strain* (deformation or stretch), meaning that it has properties of both elastic solids and

[2]See section 2.2.2 for a discussion of analytical theories of deformation.

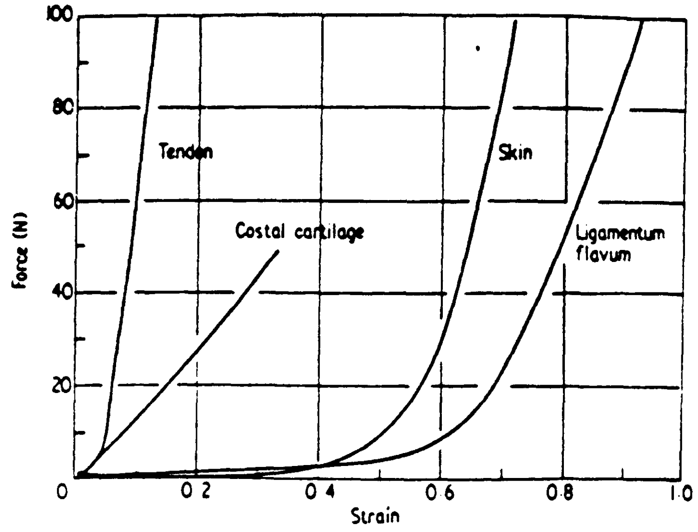

Fig. 4. Force–deformation relations for a selection of excised human tissues tested in uniaxial tension. Apart from costal cartilage which is sensibly linear, other tissues display an initial lax response with a gradual transition into a stiffer, nearly linear response. The magnitude of strain at the transition is the principal variable.

Figure 2-1: Experimental stress/strain relations.[28]

viscous liquids. The elastic nature of soft tissue refers to its storage of energy and tendency to return to its rest shape when the load is removed. The relationship between load and deformation is non-linear, even in the range of deformations commonly encountered in living subjects. Figure 2-1 shows relationships between stress and strain in soft tissue which are typical under uniaxial tension. Tissue is viscous in that the internal force generated due to a deformation is dependent not only on the *amount* of deformation but also on the *rate* of deformation.

Several experimental phenomena display the viscoelastic nature of soft tissue[62][56][28]. *Hysteresis* refers to a change in the response of the material under cyclic loading and unloading such as that shown in figure 2-2.[3] *Stress relaxation* is the reduction in the force opposing a deformation held constant in time; figure 2-3 illustrates this effect. Related to stress relaxation, *creep* is the increase in strain over time in a material under the influence

[3]Figures 2-2 through 2-5 show the results of *in vivo* measurements of the mechanical properties along a 5cm by 1cm island of piglet skin which remained attached to the subcutaneous tissue[62].

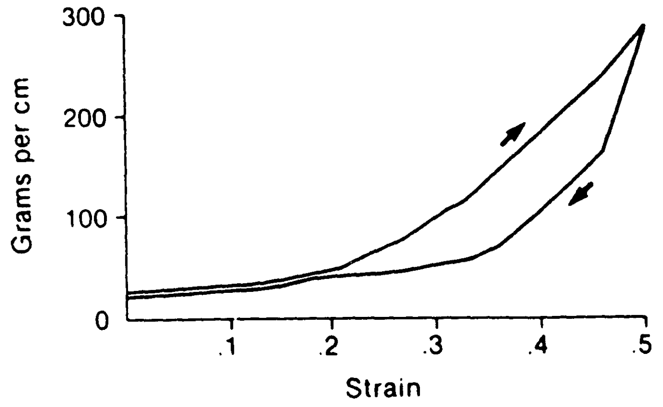

Fig. 4. Loading versus unloading force elongation curves of a skin island, demonstrating hysteresis with separation of the two curves.

Figure 2-2: Hysteresis.[62]

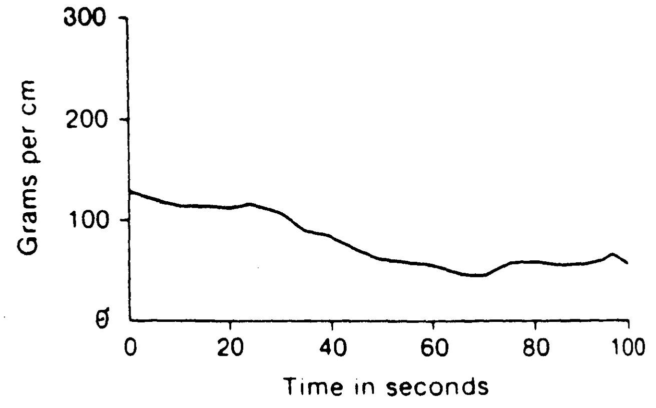

Fig. 5. Stress relaxation, tension versus time at a constant strain of .2 in the skin island. A 5 cm length of skin was extended 1 cm, and resulting tension monitored for 100 seconds.

Figure 2-3: Stress relaxation.[62]

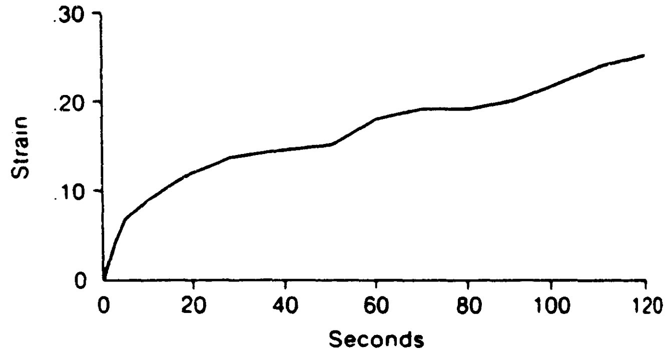

Fig. 6. Creep. Strain versus time measured in a 5 cm skin island with a constant tension of 100 g/cm.

Figure 2-4: Creep.[62]

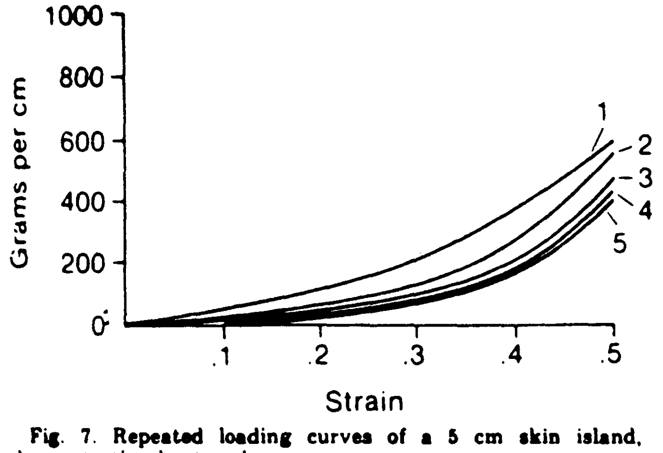

Fig. 7. Repeated loading curves of a 5 cm skin island, demonstrating hysteresis.

Figure 2-5: Pre-conditioning.[62]

of a constant load; figure 2-4 illustrates the creep phenomenon. A phenomenon related to hysteresis is *pre-conditioning*, in which repeated applications of the same load result in different deformation responses as shown in figure 2-5.

In addition to these properties, soft tissue can be distinguished from the materials which have commonly been the subject of mechanical analysis is that the tissue is, in fact, made up of living cells and that biological factors influence the mechanical response of tissue in many ways[28]. The type of tissue and its location on the body must be taken into account, along with the age and health of the subject.

The mechanical properties of skin are the result of the interaction of the component cells. The elastic response is thought to have two cellular bases: elastin and collagen. The behavior of elastin, which is a major component of blood vessels, is very similar to an ideal rubber[38] with an essentially linear stress/strain curve over a wide range of deformation. Collagen, the material of tendons, has a much stronger stress response to applied load and has a more limited range of deformation. The anatomical relationship between these materials in hypodermal tissue, as described in the last section, might also explain the experimental stress/stain curves. The low stress response to small strains may be due purely to the stretching of the elastin, since the collagen is arranged in a deformable lattice structure. The sudden increase in stress may be due to the stretching of collagen once it is aligned with the deformation[13]. The pattern of the collagen lattice is not symmetric and tends to form contour lines of fibers with common orientation. These lines correspond to lines of anisotropic deformation of the skin called Langer's lines[12].

The fat cells and the ground substance, which are composed mostly of water, account for the viscous properties of skin. They also account for the behavior of tissue under compression, where the collagen lattice on its own would merely collapse. Instead, since the water in the fat and ground substance is incompressible, they are forced out perpendicular to the line of compression; this phenomenon is called the *poisson effect.* Through this process, fibers perpendicular to the compression are actually extended and therefore resist the compression. This accounts for the qualitative similarity between experimental stress/strain curves for compression and extension[28]. Extension of tissue also causes contraction along the plane perpendicular to the line of extension, which can also be attributed to the volume conserving properties of the fat and ground substance.

The composition of soft tissue of the face changes with age. In the dermis of a newborn, there is a large amount of collagen compared to the amount of elastin, but this ratio inverts in old age so that elastin is present in much higher concentrations than collagen[31]. In addition, the skin becomes thinner with age due to a general loss of adipose tissue (fat) bringing the skin into contact with the deep layers. Aging also causes the support structures which hold the shape of the face to retract, and the soft tissue sags as a result. Folds form along the lines of skin adherence and muscle insertion causing characteristic wrinkle lines. The overall decrease in volume of the cranial structure and the lack of compensating structural integrity are visible in the aging face.

Biological and physiological considerations are particularly crucial in the analysis of wound closure and healing[58]. Typical wound healing involves the generation of collagen *fibroblasts* (the precursor of connective tissue) across the wound. The rate of healing and the type of scar developed depend on the size and shape of the wound, its location on the body, and its orientation with respect to skin tension lines, as well as the age and health of the patient, and the method of wound closure. Sutures, skin tapes, skin clips, and wound adhesives are all used, sometimes in combination, to hold the wound edges together while healing occurs. The placement of sutures (stitches), on the surface and in the subcutaneous layers, is a surgical tool for reshaping tension lines of the skin and guiding the healing process[64].

Muscle Mechanics

Active muscle fibers have a length at which they exert their maximum force for a given amount of nervous stimulation. The amount of force generated by the muscle falls off if it is stretched or compressed beyond a region of approximately 10% of the length which generates the maximum force; the force eventually falls off to zero within approximately 70% of the maximal force length[57]. Although the muscles themselves exert an inelastic force, they are in series with tendons which stretch to absorb the change in length of the muscle and transfer forces to tissue at the insertion site[66].

Inactive muscles are governed by an exponential stress/strain relationship similar to that shown by collageous soft tissue. Muscles also show a non-linear damping with respect to shortening velocity[35]. Muscles contract without changing volume.

Sources of Mechanical Data

It should be noted that obtaining reliable data for living human subjects is obviously difficult (especially for measures such as skin breaking point!) so experiments are often carried out on cadavers, laboratory animals, and extra tissue excised during surgical procedures. The mechanical properties of cadaver tissue are different than living skin[62], perhaps due to the amount of fluid in the composition of living skin. Mechanical properties are slightly different in all animals depending on such factors as density of hair, thickness of skin, and composition of the tissue layers underlying the skin. Piglet skin is similar to human skin in these respects and has therefore been the subject of research[62][13]. These difficulties become even more pronounced in obtaining experimental results for new techniques of surgery, since patients must not be unduly exposed to the risks of new procedures. There is rarely enough excised skin to test complex surgical procedures.

2.1.3 Plastic Surgery Techniques

Applications of plastic surgery include repairing lesions caused by disease, replacing skin lost to burns or abrasions, rebuilding features misshapen by birth defects or injury, and removing excess tissue to reduce the visual effects of aging.[4] The plastic surgery literature provides a wealth of descriptive material about the structure and properties of facial tissue, as well as experimental data regarding the types of facial tissue, deformities, and abnormalities in facial structure and the reaction of tissue to injury and surgery. The rest of this section looks at some of the most common techniques of plastic surgery and how their implementation is influenced by mechanical considerations. Even the most minor applications of these techniques can be surprisingly effective in their ability to recreate the normal appearance and functioning of body parts.

[4]The material in this section is drawn mainly from *Plastic Surgery*, by William Grabb and James Smith[64].

Fig. 1-2. Elliptical excision. If the ellipse is too short (A), dog ears (arrows) will form at the ends of the closed wound. The correct method is shown in B.

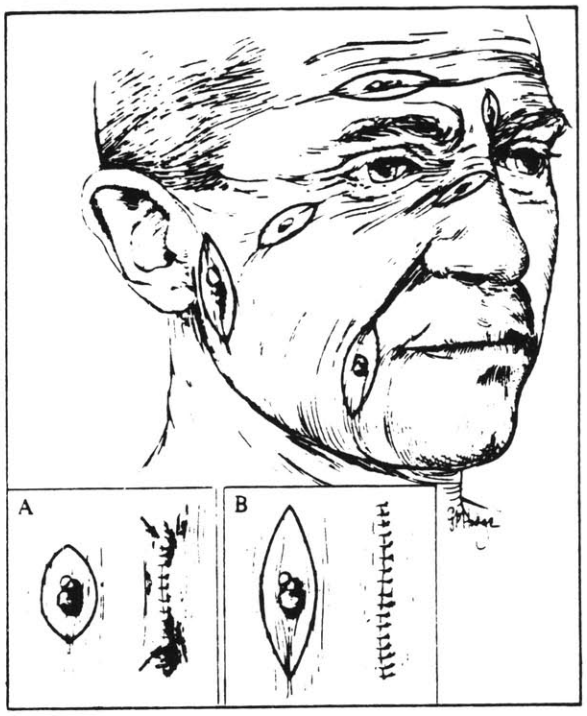

Figure 2-6: Example elliptical excisions.[64]

Closures and Excisions

A common application of plastic surgery is the removal of skin lesions caused by disease. The typical method of removing tissue is the double-convex lenticular, or "surgeon's ellipse", as shown in figure 2-6. The amount of skin which can be removed with the elliptical excision technique varies widely and depends on the age of the patient and the type of skin. The stress on the healing wound and the visibility of the resulting scar depend on the orientation of the major axis of the lenticular with respect to Langer's lines, wrinkle lines, and the relaxed skin tension lines. Elliptical excisions at right angles to these lines are more likely to leave a noticeable scar than excisions parallel to these lines.

The removal of tissue and subsequent closure of the opening accomplishes a redistribution of tissue to replace the diseased section. If this takes place over too small an area, the tissue being redistributed will bunch up, resulting in "dog ears" around the closure site.

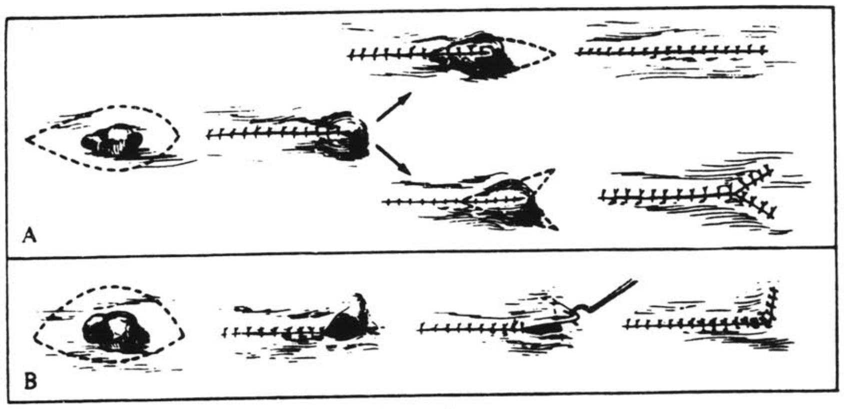

Fig. 1-3. A, two methods of removing a dog ear caused by making the elliptical excision too short. B, method of removing dog ear caused by making one side of the ellipse longer than the other.

Figure 2-7: Approaches to eliminating dog ear effects.[64]

This problem can be addressed by increasing the length of the major axis, or by removing some of the tissue which is bunching up. Figure 2-7 shows some approaches to eliminating dog ear effects.

Other excision techniques may be used depending on the circumstances. *Wedge excisions*, for example, are performed on free margins of skin such as the ears, lips, nostrils, and eyelids. Other excision shapes are also used depending on the location and shape of the lesion.

Grafts

Skin grafting is the process of completely removing a patch of skin from a donor site and transplanting it to a recipient site to replace skin lost due to lesions or burns. Problems can arise with skin grafts due to the differences in skin thickness of donor and recipient sites.

Grafts of different thicknesses are taken depending on the circumstances. *Preauricular* (in front of the the ear), *postauricular* (behind the ear), *supraclavicular* (neck), or scalp skin grafts are often used to repair small facial defects. For larger grafts, skin from the thigh or abdominal wall can be used. Split thickness grafts of skin from the upper arm are often used to repair the upper eyelids, because it is pliable enough to fold and unfold as the eyes are opened and closed.

The pattern for the graft must be determined such that it will fit the recipient site correctly *after* the skin has healed. Skin grafts undergo two types of contraction: *primary* and *secondary*. Primary contraction occurs immediately after the graft has been cut from the donor site. The amount of contraction depends on the thickness of the graft, with thicker grafts shrinking more than thinner grafts. It is overcome during surgery by strongly suturing the graft to the recipient site. Secondary contraction occurs as scar tissue forms between the graft and the recipient site. This contraction also varies with the thickness of the graft, as well as with the stiffness of the skin at the recipient site.

Flaps

Skin flaps are an alternate method for repairing wounds too large for direct closure. The technique involves moving tissue from one part of the body to another while maintaining some attachment which permits blood flow from the donor site. Although grafts are the simpler technique, flaps are used when the recipient site has poor blood supply, the skin thickness must be closely matched, or the wound needs to be reopened for later surgery on underlying structures. The difficulties of flaps arise due to the importance of planning the continued blood supply and the mechanical stresses placed on the skin due to rearrangement of large areas of tissue.

Z-plasty

The Z-plasty technique is used to change the length of skin in a desired direction, to change the direction of a scar so that it lies along a skin line, or to rotate the axis of skin regions in the manner of a skin flap but without removing tissue. The technique involves exchanging the triangular wedges formed by a "Z" shaped incision. Each wedge has one edge which lays on a limb of the "Z" and one edge which lays on the central member. When the flaps are exchanged, the "Z" shape will have been reflected and its central member will be formed by the two edges which had been on the limbs. The edges which had been on the central member are now attached to the surrounding tissue, in effect exchanging the length of the central member with the distance between the terminal ends of the limbs. Note that for this procedure all three sections of the "Z" must be of equal length. The effect of the Z-plasty depends on the angle between the central member and the limbs of the "Z". If the angle is very small, then the distance between terminal ends of the limbs is almost the same as the length of the central member; if the angle is large, the distance between the terminal points will be larger than the length of the central member (see figure 2-8).

Face Lifts

Plastic surgery is also applied to counteract the visual effects of the aging process. As mentioned above, a number of factors contribute to the look of old age, including wrinkles and a loss of firmness in the skin. The face lift is a surgical technique to address these signs of age by removing a portion of the skin so that what remains is under greater tension — flattening the wrinkles and holding in place any sagging skin. The technique described by Smith[64] involves an incision on each side of the face and around the ear. Tissue (skin and fat) is excised from the sides of the face.

Another important aspect of face lift surgery is removing any slack from the support structures of the face. With age, the transmission of muscle action to the skin is impeded by the loss of tension in the tendons and aponeuroses. A horizontal *plication* (folding over) of the support structure in front of the ear is used to increase tension in the underlying tissues. The superficial musculo-aponeurotic system, or *SMAS*[59], is a term referring to a layer of hypodermal tissue in the lower face and neck made up of fascia, aponeuroses, and muscles, which is the subject of the plication procedure. The exact anatomical definition of the SMAS has been disputed in the plastic surgery literature[18][26], but the plication procedure has proved useful clinically for making the nasolabial fold less deep and for facilitating the transmission of muscle actions in the lower face[64][32].

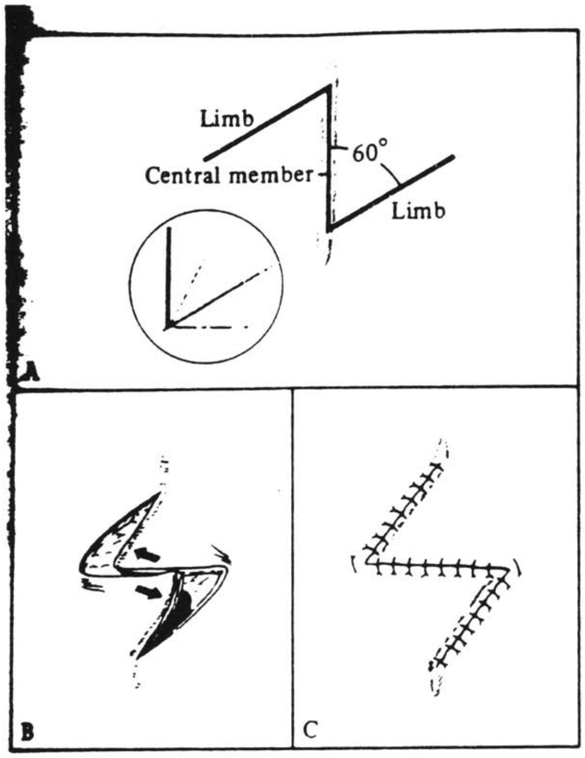

Fig. 1-40. The classic 60-degree-angle Z-plasty. Inset shows method of finding the 60-degree angle by first drawing a 90-degree angle, then dividing it in thirds by sighting. The limbs of the Z must be equal in length to the central member.

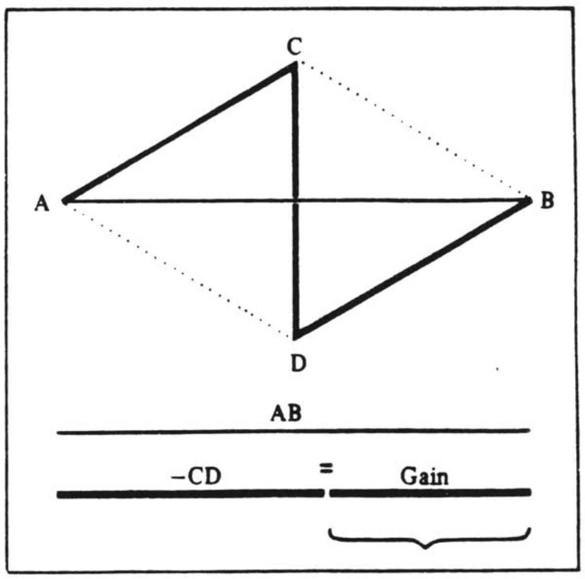

Fig. 1-41. Calculating the theoretical gain in length of the Z-plasty. The theoretical gain in length is the difference in length between the long diagonal and the short diagonal. In actual practice the gain in length has varied from 45 percent less to 25 percent more than calculated geometrically, due to the biomechanical properties of the skin [283, 284].

Table 1-1. Z-plasty, Angles, and the Theoretical Gain in Length

Angles of Z-plasty	Theoretical Gain in Length
30°–30°	25 percent
45°–45°	50 percent
60°–60°	75 percent
75°–75°	100 percent
90°–90°	120 percent

Figure 2-8: Example Z-plasty plans.[64]

Other Plastic and Reconstructive Surgery

Beyond the basic procedures described here are advanced techniques to reconstruct damaged body limbs and facial features. In many of these techniques, the surgeon must reorganize tissue to restore function, for example, folding over a patch of skin from the forehead to replace a missing nose or using a rib to replace a cheek bone. Other techniques involve replacing missing tissue with prosthetic implants, in which case both the implant and the surgery must be designed according to the details of each case.

Obviously, these procedures are exceedingly complex and require extensive skill and planning. The surgeon must take into account the patient's unique anatomy, the nature of their injury or congenital defect, the overall age and health of the patient, and the structural and mechanical requirements of the reconstructed feature. The ability to successfully address these issues is a complex skill attained through years of education and experience.

2.1.4 Summary of Processes to be Modeled

It is clear from the material reviewed so far in this chapter that human soft tissue is a complex composite material consisting of a number of important component substances (elastin, collagen, and ground substance being the most important) and that its behavior is non-linear and is influenced by both anatomical factors (from the inhomogeneous microarchitecture of skin layers to the anisotropic lines of stress in its undeformed state) and biological factors (such as the changes in structure and composition of the skin with age and health). It is also clear that all of these factors are important considerations for the successful application of plastic surgery techniques to correct abnormalities in form and function of the soft tissues. A major goal of this thesis has been to evaluate these complexities and identify techniques which can address them. The next section looks at how these complex phenomena can be modeled.

2.2 The Techniques Available for Modeling

The second half of the engineering analysis of the problem is to assess the tools and techniques available to address the problem and to review past efforts to use those tools and techniques in the same or related problem areas. The tools and techniques available to the problem of soft tissue simulation include computer graphics and animation techniques to display the resulting shapes of the simulated tissue, the theory of deformable materials, and the computational techniques through which the theory can be implemented in an interactive animated environment. Past efforts to apply computer tools to medicine have included recovery of geometric structure from non-invasive scanning, animated display of internal structure, and surgical simulations using physical modeling.

2.2.1 Past Computer Animation Facial Models

Four major techniques have been applied to the animation of faces. *Rotoscoping* extracts the shape and position of facial features from a sequence of images of a real face. While this approach provides a direct means of describing the tissue deformation, it is difficult to accurately model the transitions between recorded expressions. *Keyframing*, the standby technique of computer animation, is often used to interpolate images between frames of a rotoscoped sequence or of expressions which are created by hand. *Parameterized* facial models have been developed to deal with the "degrees of freedom problem" encountered in keyframing approaches. Recent attempts have emphasized *muscle parameterization* in an attempt to achieve more realistic movement. The next several sections review relevant work in facial animation and structural modeling from the computer graphics literature. It is important to note that while techniques for facial animation and physical modeling have both been progressing, no work has yet been presented which attempts to combine these fields. In this thesis, I have attempted to make that combination.

Parke, 1972

In his 1972 report, Fred Parke, a pioneer in the field of facial animation, described a system he developed at the University of Utah for animating faces[39]. His technique included extraction of expressions from pairs of photographs of a model with polygons painted on her face. Data of the model displaying a range of facial expressions were used as keyframes for Gouraud shaded animations. To simulate inertial drag on the facial features, Parke used cosine interpolation rather than linear for his keyframing. Another interesting technique he applied was the doubling of polygon vertices to introduce shading discontinuities along the nasolabial fold.

Parke, 1982

Parke continued his work at The New York Institute of Technology and described many useful facial animation techniques in 1982[40]. Parke separated facial parameters into those controlling the face's *conformation* or structure and those controlling the face's *expression.* Characters were defined by selecting values of the conformation parameters, and animations were formed by keyframing the expression parameters. Parke reports that interesting animations results can be obtained with as few as 15 expressive parameters. He also notes that viewers expect more realistic motion from graphical models which look more realistic.

Platt and Badler, 1981

Platt and Badler approached the problem of facial animation from the perspective of notation and automatic recognition[43]. Their model is based on the AUs (action units) of FACS (the facial action coding system). FACS, which was developed by psychologists and sociologists looking for "universal" facial expressions of emotion, describes 6 basic expressions which are recognizable even by members of remote tribes who have not been exposed to the media of other cultures[11]. The expressions are described in terms of collections of muscles (AUs) working together to influence the position of facial features. Platt and Badler used a spring-based skin model to propagate the effect of muscles to the rest of the face. Since their system used a one-layer tissue model, it was unable to represent the volumetric properties of facial tissue under the influence of the muscle actions.

Waters, 1987

Another recent work on facial animation, by Waters at Middlesex Polytechnic in England, is also based on FACS and was motivated by a desire to create a platform for controlled experiments in lip reading[61]. Waters developed a muscle model which directly displaces the nodes on the skin surface as a function of muscle tension. Waters' technique works well for modeling the action of the jaw and the lips, however it does not accurately model areas which are under the influence of multiple action units.

Magenat-Thalmann, Primeau, and Thalmann, 1988

As part of their "human factory" project, Magenat-Thalmann, Primeau, and Thalmann have put together a facial animation system based on what they call *abstract muscle actions* (AMA)[30]. The vertices of the polygon mesh representing the face are altered by the AMA procedures in a manner corresponding to an abstraction of the action of muscle groups in that area of the face. The AMA procedures are unique to the character being animated.

Terzopoulos, Platt, Fleischer, and Barr, 1987

While Terzopoulos et. al. did not work specifically on the problem of facial animation, their work on elastically deformable models for animation is of particular interest for the problem of animating facial tissue[54]. The model they present uses a simplified version of elasticity theory which is specialized for isotropic homogeneous materials. They describe solution techniques for the equations of deformation and motion. Their approach falls in the category of discrete simulation models which are described in section 2.2.2, and is very similar to the model used in this thesis.

Unsolved Problems of Facial Animation

Several difficult problems encountered in attempting to generate realistic facial animation have been described by the researchers listed above and left unsolved in the literature. These difficulties include modeling the flow of muscles over bone sheets, the influence of jaw actions, the movement of cartilaginous areas (specifically, the nose), the interactions of the tongue with the lips and cheeks, the puffing and sucking of the cheeks according to changes of air pressure in the mouth, the fatiguing of the skin under repeated deformation due to the viscous and plastic nature of the skin, modeling of facial blemishes, teeth, and hair, and control of eyeball motion. The physical simulation techniques described in this thesis were developed to address many of these issues to the extent required to make a simulator for plastic surgery.

2.2.2 Predictive Models of Deformable Materials

The problem of predicting the behavior of deformable materials under the influence of various loads has been addressed mathematically in two quite different ways. The classical approach is analytical; the more modern approach is primarily numerical. The former provides a closed form solution into which are plugged the specific details of a given problem. The latter provides a method of building a specific instance of the problem out of a set of generic building blocks which, as a whole, exhibit the behavior described by the former. These two approaches are outlined below.

Analytical Models from Continuum Mechanics

This approach relies on the tools of mathematical analysis. To model deformations, the entire body is described by an encompassing set of equations into which are substituted equations describing the specific instance of the problem.

The two essential measures which describe the deformation of a material are the strain and the stress[3][2]. *Strain* refers to the amount of stretch of the material at any given point and in any given direction and is usually expressed as the ratio of the change in length to the original length. *Stress* is the distribution of force per unit area generated to oppose a given strain, or, inversely, the distribution of force required to obtain a given state of strain. Both of these measures can be represented as *second-order tensor fields* over the region occupied by the body; that is, for any point in the material a second-order tensor is defined. A second-order tensor is essentially a 3x3 matrix independent of a specific coordinate frame. Multiplying a direction vector (first order tensor) by these second-order tensors defines the stress and strain respectively on a surface through the material the normal of which is defined by that vector.

The physical properties of the material are incorporated into the model by way of the function which relates stress and strain, which in its general form may be quite complex. If the material is *anisotropic*, then the function depends on the orientation of the stresses and strains with respect to the material. If the material is *inhomogeneous*, then the function also depends on the coordinates within the material. If the material is *viscous*, then the function depends on the derivatives of the stresses and strains with respect to time. If the material is *plastic*, then the function depends on the past history of deformation of the material. As noted in section 2.1.2, experimental results show that the behavior of soft-tissue exhibits *all* of these complexities as well as complexities related to its biology, such as regrowth after injury, and changes in composition with aging.

Since, in the analytical approach, the problem remains in symbolic form through the course of the solution, simplification techniques can be applied at any stage to take advantage of any special knowledge about the form of a given problem. The most common example of this simplification is the restriction that the amount of deformation in a problem will be small with respect to the size of the object under consideration (the case of *infinitesimal strains*). When this assumption can be shown to hold, some terms in the equations may be dropped or linearized without significant loss of accuracy. Other simplifications are applied, for example, when all the forces in the problem are known to be parallel to a given vector, or when the material is assumed to be homogeneous and isotropic. Analytical infinitesimal theory is an accurate and powerful tool in the analysis of structural materials for manufacturing and building; however, a simplified model which works well for concrete is clearly inappropriate for modeling the complex behavior of biological soft-tissue. Formulating an analytical description of this more complex behavior would be extremely difficult, and the resulting equations are likely to be analytically intractable[1].

Discrete Simulation Models

The discrete simulation approach is an attempt to avoid the formulation difficulties of the analytical approach and to make problems amenable to solution using digital computers. The approach describes the properties of individual elements of the material and the interaction between elements and their neighbors. To model a specific instance of the problem, the elements are assembled into the shape of the material to be modeled, and the behavioral properties assigned to the elements reflect the physical properties of the corresponding material. The behavior of the elements is then simulated to discover the behavior of the entire system.

This approach is at the core of the *Finite Element Method* (FEM)[49], in which the state variables of all the elements and relationships between the elements are assembled in a system of matrices. FEM can be applied to a wide range of problems such as heat exchange, fluid flow, and distributions of electrical and magnetic fields on surfaces and solids. By applying a common formulation to these problems, a variety of standard techniques can be applied to the matrices in order to increase the accuracy and efficiency of the solution. The choice of solution technique can also be tailored for a given problem type.

Most of the problems with analytical methods that are due to the complex behavior of soft tissue can be addressed through variations on the finite element method. These variations make formulation of the problem more difficult and solution less efficient. Two problems remain, however, which complicate the application of the traditional finite element method for deformation problems on biological tissue. The first problem is numerical and is due to the fact that in deformation problems the sample points are moving in relation to one another (as opposed to problems in thermodynamics, for example, in which the point samples remain stationary). Because of this, movement estimates of quantities such as strain, in which the components of stretch are measured with respect to the coordinate axes, can become inaccurate as the strain becomes large.

The second problem with traditional FEM is computational and is due to the fact that the elements are assembled into a single linear system which must be solved each time step. The number of elements grows according to the density of sampling of the material — in problems relating to solids, the number of uniformly sized elements grows with an inverse cube relationship to the size of each element — and the size of the state matrix grows in the square of the number of elements. Since solving the system of equations essentially requires inverting this matrix — a problem whose complexity grows by the cube of the size of the matrix in the worst case — it isn't hard to see that this approach can quickly consume both memory and time resources. Although the matrix describing the connectivity of the elements is sparse, it is difficult to take advantage of this sparsity. Even when special sparse matrix techniques are used, the size of the matrices presents serious computational difficulties.

An alternative to the classical FEM technique is to represent the material in graph form rather than matrix form. In this formulation, sample points in the material are the nodes of the graph, and the relations form the arcs. Each additional sample point adds a node to a graph rather than a row and a column to a matrix, and the number of relations also grows linearly with the number of elements, since each node only relates to its neighbors. There is a two-stage simulation process in this approach. First, each relation is evaluated using current states of the point samples and its contribution to the next state of the point sample is recorded at the node which represents the sample. Second, each node is updated to a new position based on the influences of all its relations. These two steps are repeated for each time step of the simulation. The execution time and memory space requirements for this scheme are linear in the number of point samples plus the number of relations (the number of relations itself being linear in the number of point samples as noted), resulting in a more efficient representation of the problem, although choice of a time step and integration technique are critical to the success of the method. This modeling technique was chosen for the research described in this thesis and is described more fully in the section 3.2.1.

2.2.3 Interactive Simulation Systems

The simulation process for animation consists of three stages: *designing* the model and setting the initial conditions and any time-varying boundary conditions, *simulating* the evolving state of the model, and *displaying* the results of the simulation. In an *interactive simulation*, these stages are iterated at high speed, and the boundary conditions can depend on user input in addition to any predefined boundary conditions. Interactive simulation systems of this type are limited by the amount of computation available for the simulation and the display of the simulation results.

Interactive simulation has traditionally been associated with flight simulation, where the user's manipulation of airplane controls is mapped to changes in the view of a simulated environment. Although some physical simulation is required to determine the plane's path, the major bottleneck in such a system is the graphical task of continually redisplaying a view of the world. The existence of this bottleneck for flight simulation (and other applications) has spurred the development of specialized graphics hardware which greatly reduces the time spent rendering and/or allows it to go on in parallel with the simulation process.

A recent driving simulation system developed by Evans and Sutherland Corporation[10] is the most advanced example to date of the combination of complex dynamic simulation

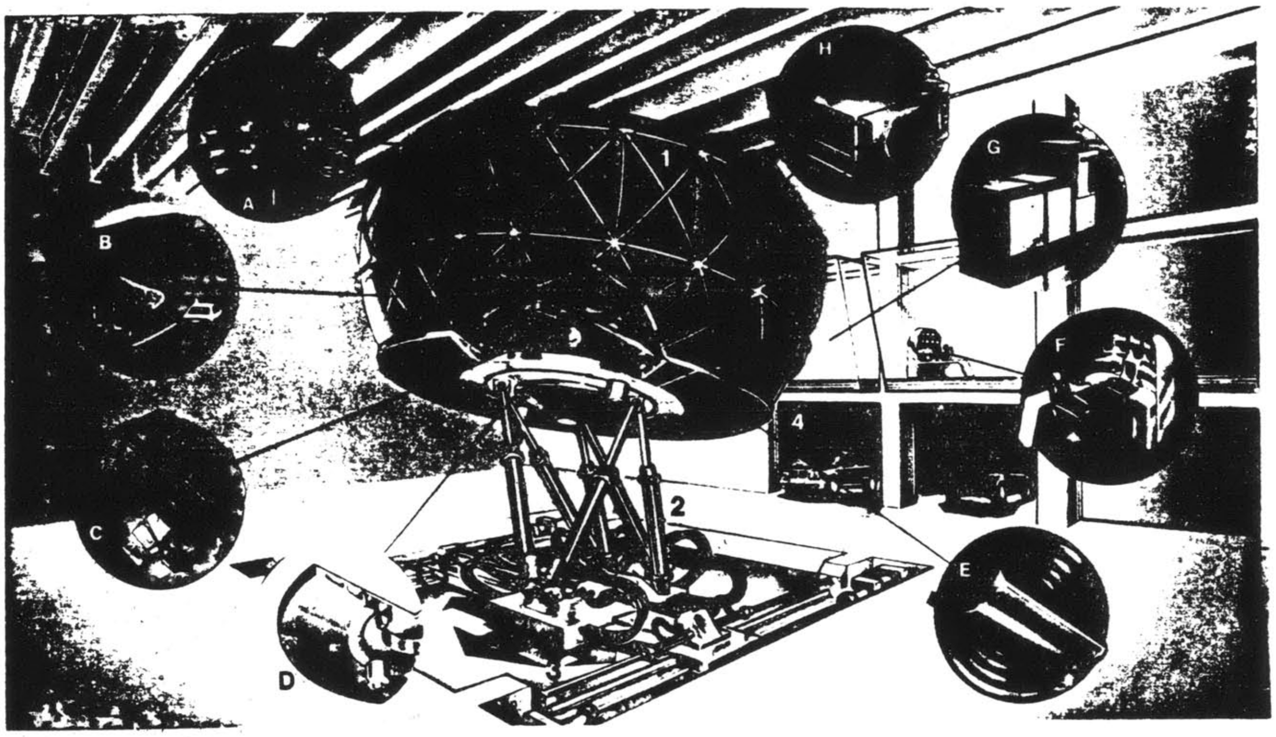

Figure 1 - Driving Simulator System

Figure 2-9: The driving simulator system from Evans and Sutherland.[10]

and realistic rendering in an interactive environment. Their system was developed in order to provide an accurate and repeatable method of evaluating automotive designs while minimizing the number of prototypes required. The test driver sits in a car on a motion controlled platform surrounded by a dome onto which are projected realistic shaded graphics of the simulated environment. The car is equipped with sensors which monitor the driver and inform the simulation computer of the state of the system, and force-feedback devices which simulate the feel of the car in the steering wheel, the brake pedal, and the gear shift lever. The system also generates the sound of the engine, gear whine, and passing traffic. The dynamic interaction of the engine, drivetrain, transmission, suspension, and tire/road interaction is all calculated in real-time and in response to the driver's actions by an Alliant FX/8 eight-processor parallel computer running a recursively formulated articulated rigid body dynamics simulation optimized for parallel execution.

While the Evans and Sutherland group has built specialized input and output devices (the instrumented car and domed motion platform), a group at the NASA Ames Research Center has put together a system which can be used for a variety of simulation tasks[16][17]. The system combines a DataGlove input device[5] and a head-mounted stereo display, both of which track the user's motion. The position and orientation of a head tracker is used to control the computer graphics views displayed in each of the user's eyes. The hand tracker provides position, orientation, and flexion values which are used to render the user's hand in the simulated environment. The input/output system also includes stereo sound and voice synthesis, as well as voice recognition and gesture recognition to increase the user's "situational awareness". The applications envisioned by the NASA group include *telerobotics* to control remote devices and vehicles in either a high-level supervisory mode or a *telepresence* mode, where the operator is provided with sensory feedback to approximate actual presence at the remote task site. An example application of telerobotics investigated by the NASA team is remote control of a surgical robot on a space station or space craft. If surgery is required during an extended mission, the robot could perform the operation under the control of an earth-based specialist. Simulators for surgery have also been investigated. Another application is in *information management* for control of complex processes such

[5]See section 4.5.2 for a description of the DataGlove.

as functioning of a space station. In this application, process status is presented as objects in the simulated environment and user actions are used for data manipulation and control functions.

2.3 Medical Applications of Computers

The vast complexity of biological systems presents a number of difficult tasks to the medical community for which computer tools are beginning to prove useful. The following sections look at a few of these techniques which may have application in development of surgical simulators.

2.3.1 Scanning and Reconstruction

An important application of computer technology in medicine today is the reconstruction of three dimensional models of patients from collections of scanned slices obtained from CT (*Computerized Tomography*) and MR (*Magnetic Resonance*) scanning. These reconstructions provide physicians with a non-invasive method of visualizing the structures inside the body for planning surgeries and monitoring postoperative progress[27][36]. Computer graphics techniques such as color, translucence, light-source shading, and surface reconstruction have been used to improve the viewer's appreciation of complex anatomy[29][8]. Application of these scanning techniques to physical simulation model building is examined in section 3.1.2.

2.3.2 A Hand Biomechanics Workstation

Thompson et. al.[55] have built an interactive simulation environment for planning hand surgery. The system uses data derived from CT and MR scans of a patient's hand and a kinematic model of the interaction between muscles, bones, and tendons to present a display of the simulated hand. Through menus and knobs, a surgeon can manipulate joint angles, and observe the resulting tendon and hand movement. The system is not based on a dynamic physical simulation of hand movement, so it is not possible to pull on a tendon and observe the resulting movement of the fingers. However, the system does present a graphical representation of the torque acting at the hand joints as the hand is moved. The Thompson group's work is important because it is an attempt to apply real-time three dimensional graphics and a simulation model in a clinical setting with real patient data. From their initial results, they emphasized the following important considerations:

- Tools must be provided for incorporating scanned data into the simulation model. Their system includes an interactive segmentation tool for extracting the contours of individual bones from the scanned data.
- The system must be user-friendly. The primary users of their system are surgeons who are generally not expert computer users, but who need fast, accurate results.
- The system must be real-time. The ability to display three dimensional motion under user control is invaluable for the accurate perception of anatomy.

Kinematic and Dynamic Analysis of Hip Surgery

The Computer-Aided Surgical Simulation project *CASS* system being developed by Mann et. al. addresses osteotomy surgery (hip replacement) by analyzing forces and torques in the musculoskeletal system during activities such as walking, running, or jumping[7]. The system has two components: TRACK and NEWTON. TRACK, records a patient's movement by combining information from infrared camera observing the patient who is wearing a suit studded with LEDs and from measurements obtained from a force-sensitive plate in the floor. NEWTON performs inverse-dynamics based on the information supplied by TRACK, and a model of the masses and inertias of the legs, arms, and other skeletal parts derived from MRI and CT scans. The combined system is used to predict the effectiveness of prosthetic hip joints. The project has also experimented with *in vivo* force measurement of the human hip by installing a hip prosthetic hip with 10 pressure sensors into an osteotomy patient. They then recorded the data through the stages of rehabilitation. The CASS project is an important example of the application of engineering analysis to clinical problems.

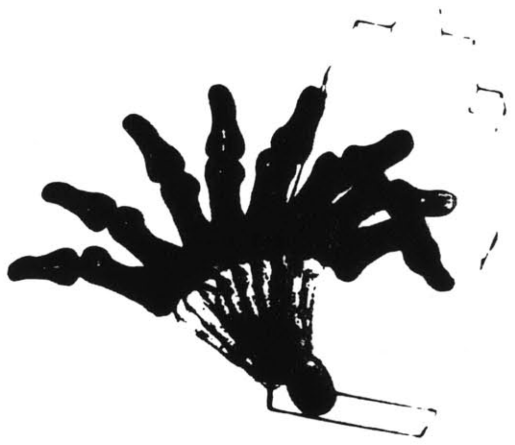

Figure 12: A multiple exposure photograph showing the standard thumb positions which are sequenced through to display a moment arm summary.

Figure 2-10: Time lapse display of simulated thumb movement.[55]

2.3.3 Physical Simulations of Soft Tissue

Although the field is still very new, three previous researchers have applied computer simulation techniques to the analysis of soft tissue mechanics. All previous models are used to find static equilibrium solutions to wound closure problems. The next chapter describes the properties of an ideal physical simulation system and the prototype system developed to test algorithms for tissue simulation. These examples of previous work all illuminate important aspects of the problem which should be considered in the development of future systems.

A Finite Element Model of Tissue Deformation, Larrabee, 1986

Larrabee developed a finite element model to describe and predict the deformation of tissue and correlated it with actual experiments performed on piglets[62]. His system models the skin as a two dimensional membrane made up of triangular elements in which the vertices have spring connections (which may be turned off) to fixed points on an immobile base plane. The triangular elements are governed by linear stress/strain relations from classical elasticity theory. The equilibrium state of deformation for the model is found through the solution to the matrix system defined by the finite element technique. The model has been used to model elliptical wound closures and rectangular flap advancement. The elliptical closure simulations were compared to the results from piglet studies and showed "a fairly realistic approximation of a clinical wound closure".

A number of simplifications are inherent in this model. Larrabee points out the following: linear stress/strain relations, no initial stress in the skin, the model is two-dimensional, and the model is not viscoelastic.

A Three Dimensional Simulation of Wound Closure, Gordon et. al., 1987

A model developed by Gordon et. al. simulates deformation in a three dimensional grid of point samples, each of which is connected by springs to its nearest neighbors[20]. A discrete simulation method was used to solve for the movement of the skin elements by calculating the net force acting on each of the elements and moving the element a small amount in the direction of that force. Skin incisions are modeled by breaking spring connections between elements, which resulted in the typical elliptical defect. To experiment with placement of sutures for optimal closure, sutures were simulated by reattaching points on opposite edges of the wound. The experiments showed that sutures placed closer to the ends of the incision and further from the wound edge resulted in the most uniform distribution of force. Separation of underlying layers of tissue in wounds closed with superficial sutures was also observed in the experiments.

A Finite Element Analysis of Surgery of the Human Facial Tissues, Deng, 1988

Deng presents a thorough analysis of the application of the finite element method to facial tissue and a description of surgical simulations performed on scanned patient data[9]. A three-layer model of facial tissue is described in terms of the geometrical and mechanical properties which were fed into ADINA (Automatic Dynamic Incremental Nonlinear Analysis), a standard engineering finite element software package. Deng describes the measurement of mechanical properties of the skin from experimental results and the extraction of geometric descriptions of the external shapes of faces from a 3D laser digitizer. Simulated surgical procedures included closure of elliptical excisions, removal of tissue to reduce dog-ear effects, analysis of the effect of Langer's lines on the rest shape of circular puncture wounds, and removal of excess tissue from a scanned section of cheek.

Deng's work is important because it describes the mathematical basis for application of FEM analysis to the problems of facial modeling. Deng also demonstrates a method for capturing patient data to form models which can be used in the analysis stage. Deng does not address the issue of muscle modeling or dynamic analysis of the face. An important issue which was not explored in her work is the surgeon's interaction with the simulation model for defining surgeries and evaluating results.

Chapter 3

Functionality of an Interactive Tissue Simulator

The last chapter looked at the anatomy and behavior of biological soft tissue, simulation techniques available to model its behavior, and computer systems which allow real-time interaction with simulated systems. This chapter describes the required functionality of a system in which simulations of soft tissue elements can be viewed and manipulated interactively; first in terms of the ideal design of such a system, and second in terms of the prototype system which was built to test these ideas.

3.1 Ideal Properties of a Simulator

In light of the material presented in the last chapter, I propose that a successful facial tissue simulation system for use as a surgical training and planning tool must have the following properties:

- Deformability - to model for shape change under applied forces;
- Formability - to build the model for a specific patient;
- Reformability - to edit the model to simulate surgery;
- Controllability - to model movement due to muscle action; and

- Interactivity - to provide timely feedback to changing input.

Each of these requirements will now be examined in more detail.

3.1.1 Deformability

The system must provide an accurate prediction of the movement and deformation of tissue in response to applied forces. This includes modeling such phenomena as the change in shape of soft tissue as it stretches across or travels over underlying hard tissue, the bulging and folding of skin due to resistance to compression when skin is under load, and the internal forces and deformations due to the action of muscles within the the skin layers. The system must be able to model these phenomena for anisotropic, inhomogeneous material. Other tissue behaviors which may be modeled include the formation of wrinkle lines, the movement of the tongue and its interaction with the cheeks, and the effects of puffing and sucking of the cheeks.

3.1.2 Formability

An inherently difficult problem in computer graphics systems is the definition of models which describe the geometric properties of objects for rendering. The problem is typically addressed by an editor which allows the definition of geometric primitives, collections of which can be used to define more complex shapes. Simulation systems suffer from a similar modeling problem, compounded by the need for a description of the physical properties of an object in addition to geometric properties. The approach taken for structural analysis in manufacturing is to build the simulation model from the original design specification and assign physical properties according to the material to be used, often with the help of an automatic mesh generator to define the nodes for FEM analysis. For biological simulation, this problem is more complicated since the geometric structures are highly complex (not easily generated using a standard geometric editor) and because the material is not of uniform composition and thus the physical properties vary from point to point.

Ultimately, for surgical planning, a physical simulation of soft tissue must be built which conforms to the particular patient. Non-invasive scanning techniques such as MR, CT, or ultrasound can provide a data set of geometric and compositional information about a patient. Automatically generating a physical model from this data is a desirable long-term goal which will require extensions of machine vision technology to the 3D volume sets derived from these scanning techniques. More practically in the near future, human operators using interactive input devices and graphical views of the data can identify key reference points. These reference points can be used to derive some of the required patient information such as the shape and thickness of various tissue layers. This geometric data can be combined with assumptions about the physical properties of tissue types based on biophysical experiments such as those described in section 2.1.2. Another goal for non-invasive scanning techniques is to look at the local structure and composition of the tissue which give rise to mechanical behavior, for example observing the asymmetries in the orientation of the collagen lattice which give rise to the phenomenon of Langer's lines as described in section 2.2.1.

3.1.3 Reformability

Once a model is available which closely conforms in geometric and physical structure to the patient, tools must be available to edit the model in a manner which corresponds to the physical actions performed on the patient. For surgical simulation, this includes:

- breaking physical connections between simulation elements, as in making an incision or undermining tissue layers,
- making new physical connections between previously unconnected elements, as in suturing or plicating,
- moving pieces of simulated material for reconnection, as in flap advancement or grafting, and
- inserting new hard tissue as in reconstructive techniques.

The model might also be reformed to model growth, aging, injury, or the progression of disease.

3.1.4 Controllability

Since the simulation is a model of a living patient, it should be possible to activate the simulated muscles and compare the results with activation of real muscles in the patient.

If a reliable technique is available for generating a script of muscle actions for the patient, that script could be used to drive the model. This would serve as a powerful debugging tool for building the simulation. *Electromyograms*, which record electrical activity in muscles, could potentially provide this script of muscle activity, although accurate electromyographic recording may require insertion of needle electrodes rather than recording from the skin surface[46]. In the face there is the additional problem of distinguishing the action of individual muscles due to the large numbers of overlapping and interacting muscle groups. Another approach to controlling muscle actions is to develop standard scripts for muscle actions based on a logical organization of facial muscles. Such an organization is provided by the *Facial Action Coding System*, or FACS, which is a method of describing the contribution of muscles and muscle groups to the formation of expression. The basic composition and meaning of these actions have been found to be consistent between individuals and across cultures[11]. The FACS system has been used successfully in facial animation systems to control expressions[61][60][43][42], and the same techniques should extend well to facial animation based on physical simulation.

If either method proves successful at generating a useful script of muscle actions for input to the simulation, the door is open for a very powerful application of tissue simulation for surgical planning. By replaying muscle actions in the simulation which models the surgically altered face, it should be possible to observe the results of the surgery as they will appear in the patient's everyday life. The dynamic effect of changes in skin tension lines, the thickness of skin, muscle attachment points, or reshaping of the underlying hard tissue could then be evaluated and the surgical plan modified to generate the most satisfactory result. Being able to see how the surgery will affect the patient's ability to form facial expressions will address one of the most common complaints about face lift surgery. Often the results are acceptable until the patient smiles, at which point the results of poor surgical planning are especially visible.

3.1.5 Interactivity

Reforming the parameters of the simulation model should be reflected in deformation behavior in a timely manner; evolving ultimately the medical version of the Evans and Sutherland driving simulator described in section 2.2.2. Such a system would monitor the actions of the user (surgeon) and responds with the results of dynamic simulation in real-time. As noted in section 2.3.2, Thompson et. al.[55] emphasize the importance of a user-friendly interface to computer applications for clinical settings. A real-time simulation environment could provide the appropriate ease of use — especially if the reformation commands and their parameters are obtained by tracking the surgeon, who is going through the motions that would be used in the case of a real patient. Thus a surgeon might make an incision by moving the simulated scalpel along the simulated patient. The surgeon would then see the wound open. The surgeon could perform the appropriate surgical procedure, such as excising fat or preparing a skin flap, and then specify which wound edges are to be closed and the attachment points for the sutures.

Force-feedback interface devices may provide an important link for the surgeon's appreciation of the simulation results, just as the force-feedback automotive controls of a driving simulator give the driver a realistic sense of the car's response. While all the information required to control force-feedback devices will be available from the physical simulation, the simulation must be real-time in order to make use of them as input. It may be interesting to use them in training situations to guide the student through the appropriate motions for the operation.

To improve the realism of the surgical simulation, the system should go beyond the mechanical simulation and provide a model of the patient's physiological state during the course of the operation. Thus the interactive system should allow integration of the results of various simulations. Interesting types of simulations include the action of the patient's cardiovascular system, the response of the patient to anesthesia, and the action of complex surgical tools and instruments.

Another useful property of a surgical simulator would be the ability to provide user interaction with various data sources. Data from CT, MR, and ultrasound scans of the patient, as well as information from the medical literature, could be displayed in the same virtual environment as overlays or windows.

3.2 The Current Prototype

The ideal functional properties of a surgical simulator for plastic surgery as outlined in the last section can serve as goal by which to guide the development of prototype systems. What follows here is a functional description of the prototype system developed during the research work for this thesis. This description is in terms of the functional breakdown presented above. The *deformability* of the tissue model receives the most attention here both because it is the most highly developed aspect of the prototype, and because it is a central design issue for the implementation of a real system. Neither *formability* nor *reformability* can be addressed until the underlying deformable model has been developed, and the choice of a simulation technique for deformation is central to achieving *interactivity* in the system. The underlying deformable model is also important in considering the issue of *controllability*, since the muscles used for control must also be incorporated in the deformation model. The next sections describe the theory behind the implementation of the prototype system.

3.2.1 Deformability

Discretization

The problem of modeling deformation in soft tissue is approached with a discrete simulation model.[1] The soft tissue is represented as a set, P, of reference point samples, p_i, and a set, S, of force constraints between pairs of samples, s_j.[2] The force constraints represent the tissue which exists between the reference points. Since each constraint contains local information about the material properties of the tissue it represents, this technique can model the anatomy of inhomogeneous and anisotropic material. The individual constraints

[1]This approach is becoming common in computer graphics and animation applications requiring physical simulation[15][37][22][63]. It can be contrasted with the more matrix-intensive approach[54][53].

[2]In the following discussion, the subscript i is used in referring to point samples in the tissue model and the subscript j is used in referring to constraints. Temporary variables associated with a point sample or constraint are not listed in the definitions; they are only used for evaluating intermediate results and are not stored as part of the soft tissue structure. The $\vec{x}$ symbol indicates that variable x is a 3 component vector expressing a direction or a position in world space, and the $\hat{x}$ symbol indicates that x is a function. The notion is suggestive of the C language implementation of the model which uses dynamically allocated lists of reference points and constraints.

roughly simulate the behavior of collections of fibers of the tissue, and thus, as a whole, they model the volumetric effects of tissue as it is stretched or bunched.

Representation

Each point sample stores the point's mass, a flag telling if the point is mobile or fixed, and vectors representing the point's position, velocity, and the current total force acting on the point:

$$p_i = \{p_i^{mass}, p_i^{mobile}, \vec{p}_i^{\,position}, \vec{p}_i^{\,velocity}, \vec{p}_i^{\,force\_so\_far}\}$$

Each constraint represents a generalized spring and calculates forces which are added to the pair of total force accumulation vectors of the two point samples it connects. The constraint structure stores the geometric and mechanical data about the relationship between the point samples:

$$s_j = \{s_j^{side\_1}, s_j^{side\_2}, s_j^{original\_length}, \hat{s}_j^{elasticity\_function}, s_j^{material\_constant}, s_j^{viscosity}\}$$

When invoked, the constraint calculates the force vectors based on the amount the two points have been stretched relative to their original separation. The current length of the constraint is:

$$s_j^{current\_length} = length(\vec{p}_{s_j^{side\_2}}^{\,position} - \vec{p}_{s_j^{side\_1}}^{\,position})$$

where *length*() is the standard Euclidian distance function. The normalized vector from the point on one side of the constraint to the point on the other side — the line of force — is:

$$\vec{s}_j^{\,line\_of\_force} = \frac{\vec{p}_{s_j^{side\_2}}^{\,position} - \vec{p}_{s_j^{side\_1}}^{\,position}}{s_j^{current\_length}}$$

If the original length is greater than the current length, the force pushes the two points together. If it is less, the points are pushed apart.[3] The magnitude of the force is determined by a function which represents the material properties of the constraint. The following ratio is used as a measure of strain because it is zero when the points are in their rest state, and

[3] A current length of zero is an error condition which arises if the two points are exactly coincident due to deformation beyond the range representable by the model.

it is symmetric for extension and compression; so that, for example, doubling the distance between points would have the same strain value as halving it.

$$s_j^{strain} = \frac{max(s_j^{current\_length}, s_j^{original\_length})}{min(s_j^{current\_length}, s_j^{original\_length})} - 1$$

The strain is plugged into the elasticity function of the constraint. These functions are all zero when the points are in their rest configuration. They are also all monotonic. The currently implemented elasticity functions are of the form:

- $lin(strain) = strain$
- $exp(strain) = e^{strain} - 1$
- $log(strain) = ln(strain + 1)$
- $square(strain) = strain^2$

These functions have been used successfully in simulation experiments.

The strain rate is a scalar representing the velocity at which the points are coming together or moving apart along the line of force:

$$s_j^{strain\_rate} = (\vec{p}_{s_j^{side\_1}}^{velocity} \cdot \vec{s}_j^{line\_of\_force} - \vec{p}_{s_j^{side\_2}}^{velocity} \cdot \vec{s}_j^{line\_of\_force})$$

The strain rate and the viscosity constant of the constraint describes the force generated in opposition to velocity along the line of force. The force at the first point sample is increased by a generalized spring constraint s_j as follows:

$$\vec{p}_{s_j^{side\_1}}^{force\_so\_far_{new}} = \vec{p}_{s_j^{side\_1}}^{force\_so\_far_{old}}$$
$$+1/2\vec{s}_j^{line\_of\_force}(s_j^{material\_constant}\hat{s}_j^{elasticity\_function}(s_j^{strain}) + s_j^{viscosity}s_j^{strain\_rate})$$

The force at the other point sample is increased as follows:

$$\vec{p}_{s_j^{side\_2}}^{force\_so\_far_{new}} = \vec{p}_{s_j^{side\_2}}^{force\_so\_far_{old}}$$
$$-1/2\vec{s}_j^{line\_of\_force}(s_j^{material\_constant}\hat{s}_j^{elasticity\_function}(s_j^{strain}) + s_j^{viscosity}s_j^{strain\_rate})$$

If either of the point samples are marked as immobile, p_i^{mobile} is false, then the 1/2 scale factor is dropped, and the total force is added to the mobile side. If both are immobile, the forces are not evaluated.

These constraints are designed to model the behavior of collagen and elastin fibers in the tissue. The behavior of the fat cells and ground substance under compression is modeled by the arrangement of these generalized springs in a tetrahedral geometry which tends to mimic the lattice structure found in tissue and, through their resistance to compression, the poisson effect. Expansion in the plane perpendicular to muscle action can be seen in the example in figure 6-3.

Additional forces are generated to model the effect of other physical phenomena. Gravity is represented by a constraint which adds a force to each point sample which is proportional to its mass. The force function for the gravity field is:

$$\vec{p}_i^{\,force\_so\_far_{new}} = \vec{p}_i^{\,force\_so\_far_{old}} + G\, p_i^{mass}\, p_{earth}^{mass}\, \vec{p}_{earth}^{\,position}$$

where G is the gravitational constant and p_{earth} is a special point sample, representing the earth, whose position vector is used to store the vector pointing to the center of the earth, which is assumed to be constant for all p_i. For the purposes of tissue simulation, the inverse square relationship of gravitational force to distance from the center of the earth is neglected.

Forces are also generated in response to contact between soft tissue and underlying hard tissue. This is modeled via *drape* constraints on point samples.[4] A drape constraint detects when a point sample moves into a forbidden region and applies forces to push the point out of the region. An example of this type of constraint generates a force to keep a point sample out of the bounding sphere of an object. An instance of a drape constraint is:

$$d_j = \{p_i, \vec{d}_j^{\,position}, d_j^{radius}, d_j^{material\_constant}, \hat{d}_j^{\,elasticity\_function}\}$$

The line of force for a drape constraint is the direction which moves the point sample out of the region:

$$\vec{d}_j^{\,line\_of\_force} = \frac{\vec{p}_i^{\,position} - \vec{d}_j^{\,position}}{length(\vec{p}_i^{\,position} - \vec{d}_j^{\,position})}$$

The strain associated with deformation of the forbidden region in a drape constraint is:

$$d_j^{strain} = length(\vec{p}_i^{\,position} - \vec{d}^{\,position})$$

[4]The drape constraint is so named because it is used to drape tissue samples over obstructions.

The effect of the drape constraint on the force accumulation vector of the point sample is:

$$\vec{p}_i^{force\_so\_far_{new}} = \vec{p}_i^{force\_so\_far_{old}} + \begin{cases} \vec{0}, & \text{if } length(\vec{p}_i^{position} - \vec{d}^{position}) > d_j^{radius} \\ d_j^{material\_constant}\, \hat{d}_j^{elasticity\_function}(d_j^{strain})\, \vec{d}_j^{line\_of\_force}, & \textbf{otherwise} \end{cases}$$

Another force which can be added to the force accumulation vector is a general damping force. This force is a function of the velocity of a point sample and simulates the effect of the points moving in a viscous medium, such as the ground substance:

$$\vec{p}_i^{force\_so\_far_{new}} = \vec{p}_i^{force\_so\_far_{old}} - damping\vec{p}_i^{velocity}$$

where *damping* is a constant for all the point samples which is set in order to control the amount of oscillation in the medium.

Integration

The motion and deformation of soft tissue is simulated by stepping forward by discrete time steps. For each time step in the simulation, all the force-generating constraints are evaluated, which leaves a sum of the resulting forces in the accumulation vector ($\vec{p}^{force\_so\_far}$) in the point sample structure. The problem in time stepping is to integrate the resulting total force into a change in position for the time step. The key to this process is, of course,

$$F = ma$$

or for this problem,

$$\vec{p}_i^{acceleration} = \frac{\vec{p}_i^{force\_so\_far}}{p_i^{mass}}$$

This is Newton's Law which relates the acceleration at a point to the mass at that point and the total force acting on it.

The acceleration is the change in velocity with respect to time, so the acceleration term is multiplied by the time step to find the change in velocity within the time step:

$$\vec{p}_i^{velocity_{new}} = \vec{p}_i^{velocity_{old}} + \vec{p}_i^{acceleration}dt$$

where *dt* is the integration time step.

The velocity of the point sample is the change in its position with respect to time, so the velocity term is multiplied by the time step to find the change position for the time step:

$$\vec{p}_i^{\,position_{new}} = \vec{p}_i^{\,position_{old}} + \vec{p}_i^{\,velocity} dt$$

The force value for each point sample is an instantaneous value calculated at the beginning of the time step; the integration method just described, referred to as ***Euler integration***, assumes that the force remains constant during the time step at the value calculated for the beginning of the time step. This can be inaccurate if the force is actually changing significantly within the time step, which may of course happen since the point samples are moving within the time step.

A better estimation of the force to integrate over a time step can be obtained by evaluating the force constraints twice per time step — once at the beginning of the time step and once at the end — and averaging them. Since the positions of the point samples at the end of the time step are not known (they're the quantities we're trying to compute), the Euler method is used to estimate them. The force constraints are then reevaluated using the estimated positions, and the resulting force and velocity values are averaged with the force and velocity values at the beginning of the time step. These averaged values are then plugged into the integration equations above to make a more accurate prediction of the position and velocity at the end of the time step. This integration scheme is a second-order version of the *Runge-Kutta integration* technique[49].

Summary of Deformability

The deformable model chosen for the prototype system is a discrete simulation model which uses a network of force constraints and forward simulation to determine the dynamic behavior and rest configuration of a collection of point samples which represent a specimen of human facial tissue. The system can simulate the behavior of the forces of fibrous attachments between points, viscous damping of movement due to ground substance, collision forces due to interaction of soft tissue with underlying hard tissue, and gravity.

3.2.2 Formability

In the current system, networks of point samples and force constraints are created at run time, not compiled into the simulation code. There are two methods of creating these networks: "by hand" or algorithmically . The first method is a command line approach in which point samples are created, positioned, named, and their parameters set by typing in the appropriate values at the command line while the program is running (or in a control script which gets read in at run time). The same interface can be used to create various force constraints and set their parameters. This method is useful for testing the constraints and debugging the integration, and for using the dynamics for other simulated environment applications.

The second method creates simulation networks algorithmically from a source polyhedron. This method builds a set of point samples at the vertices of the polyhedron and at specified distances from the vertex along the surface normal of the polyhedron at that vertex. The algorithm also creates spring force constraints to connect the point samples in a lattice network. This approach provides a tool for experimenting with complex networks of constraints, the structure of which can be controlled using standard graphical modeling tools. This method has proven useful for generating test cases such as grids with missing polygons to model excisions of flesh.

The algorithm creates three layers of point samples in order to model the volumetric effects of skin as it is bunched and stretched. The top layer represents the epidermal surface of the skin and is rendered with filled polygons. The bottom layer is the bottom of the deep fascia or deep fat layer, which may be immobilized to signify connection to the underlying bone surface or collision detected with a bone surface via the drape constraint. The middle layer represents the superficial fascia and tendon layer — the SMAS layer — and serves as an attachment point for muscles, such as those of the face, which act directly on the skin to deform it (as opposed to the majority of muscles in the rest of the body which connect from deep fascia to deep fascia). The choice of three layers was made in order to minimize the amount of computation required to perform simulations and still allow this level of representation.

Muscles can be added to the network to connect arbitrary pairs of points in any of the layers. The muscle fibers are themselves instances of the spring constraints described above; these fibers can be collected into muscle units which act together. Each fiber of a muscle is specified by a single command line which defines the points of attachment, the physical properties of the spring constraint, and the name of the muscle unit to which the fiber is to be added. If the named muscle unit does not exist, a new one is created and given the specified name. Attachment points of the spring constraint can be specified either as a layer number and an index into the point sample list for that layer or as the nearest point sample to a specified point in world space.

3.2.3 Reformability

The prototype allows addition and deletion of force constraints via the command line interface. This is used in the prototype system to build and destroy small spring systems for experimentation. The prototype does not currently provide any high level tools for deleting force constraints from the algorithmically produced constraint networks. Such high-level tools can easily be built from the existing commands.

3.2.4 Controllability

The currently implemented control strategy for muscles allows definition of muscle lengths at key points in time. The length of a muscle unit is defined as a scale value relative to its original length. Between the specified time points, the scale value is interpolated using a cosine weighting curve. Cosine interpolation curves qualitatively match movement profiles recorded for human speech and have been used in several computer animation systems for movement of facial muscles[61][60]. More accurate descriptions of muscle stimulation over time did not appear to be available in the literature. The key point method allows groups of fibers to act collectively over the course of an animation in order to observe resulting tissue deformation.

For simulations of wound closures, sutures can be attached to the tissue. The sutures are actually modeled using the muscle structures and control procedures. They can be attached anywhere in the layers of tissue and can be scaled to near zero length over the course of the simulation. By connecting point samples on opposite sides of the wound, it is possible to pull the tissue to cover the spot left exposed by an excision.

3.2.5 Interactivity

The fact that the tissue simulation is embedded in an interactive simulation environment makes it possible to stop the simulation process, examine the current state (for example, rotate the simulated view) and possibly adjust some parameters before continuing. The simulation process is currently much too slow to allow real-time dynamic calculations on large tissue structures, however a major assumption of this thesis is that computer performance will continue to increase while the cost goes down. Faster performance can also be expected if the simulation is distributed over multiple machines. The simulation technique presented in section 3.2.1 is ideal for implementation on parallel machines due to the local nature of the computations.

Using current hardware, small constraint networks can be used to control the behavior of objects in a real-time application. Such an application allows experimentation with various combinations of elasticity functions, damping values, material constants, structural geometries, etc. The use of the DataGlove as an input device for these simulations allows very natural control of 3D motion to see how the dynamics will respond. The next chapter describes the interactive simulation system which allows this kind of experimentation with the tissue simulation. Because the tissue simulation is one of many interactive simulations supported, the system also serves as a prototype for surgical simulation environments which combine information from a variety of sources. For a surgical application, such a system could include simulations of various body systems, display of data from the medical literature, or views of the patient derived from various scanning techniques.

Chapter 4

Implementation of an Interactive Simulation System

The facial tissue simulation code is written as a module of a larger graphics system under development by the Computer Graphics and Animation Group at the MIT Media Laboratory[6][69][47]. The system, named bolio, allows diverse graphics applications (called *bolio tools*, see section 4.5) to share a common run-time environment and to display their results in a common 3D microworld. Through this system, the tools can also communicate with each other in a common format. As noted in sections 3.1.5 and 3.2.5, this will be an important feature of a surgical simulator, since the simulator must be able to mix the results of a number of independent simulations and possibly data from patient scanning and monitoring systems. A brief description of the internals of the bolio system will also clear up many important details for the discussion of the implementation of the facial tissue simulation in chapter 5.

This chapter reviews some of the internal data structures of bolio, its user interface and scripting mechanism, and its use of a constraint network architecture to pass information among the tools. Descriptions of several of the tools demonstrate how bolio's features allow a variety of simulation techniques to be used together in a real-time test application.

4.1 What bolio Is

Bolio is a system which serves as a common base for the development of simulations by providing an input/simulate/draw loop into which new applications can be incorporated. Bolio provides an object level interface to a graphics environment. Bolio is implemented in C on Hewlett-Packard 9000 workstations running the HP-UX operating system, a derivative of Berkeley Software Distribution UNIX 4.3 and AT&T System V UNIX.[1]

4.2 Objects and Viewports

The main data structure of interest in bolio is the bolio object, or `bOBJECT` structure illustrated in figure 4-1. This is the generic graphical object structure with fields to describe attributes (name, transformation matrix, bounding box...) common to all objects appearing in the microworld. The `description` field of the `bOBJECT` structure is a generic pointer to any of the specific object data types recognized by bolio. These data types (including `bPOLYHEDRON`, `bCAMERA`, `bLIGHT`, and `bDEPTHMAP`) all have specific data structures referenced by the `description` field of the `bOBJECT`, and which are used when that data type is read from or written to a file, compiled into displayable format for the rendering hardware, and other type-specific operations. The `bPOLYHEDRON` data type, for example, contains lists of the vertices, polygons, and edges which define the polyhedron. Most bolio commands operate on objects independent of the data type of their `description`.

Bolio uses a two stage file format for `bOBJECTs` which reflects the distinction between the `bOBJECT` level and the `description` level. Generic information about `bOBJECTs` is stored in files with a `.obj` extension. These are ascii text files which contain keyword/value pairs to define fields such as the object's name, its initial transformation matrix, and the data type and source file of its description. Since this file is generally very short, the entire text is kept in memory in a LIST pointed to by the `obj_file` field of the `bOBJECT`.[2] The `detail` keyword in the `.obj` file is followed by the name of a file from which to read the type-specific

[1]UNIX is a trademark of AT&T Bell Laboratories.

[2]The `LIST` data type is a header for an array of generic C pointers. `LISTs` are used extensively in bolio to group and order structures independent of data type.

```
typedef struct {
  char          *name;        /* bobj's unique name */
  char          *filename;    /* name of .obj file */
  LIST          *obj_file;    /* text from .obj file */
  LIST          *worlds;      /* bOBJ_WORLDs where posted */
  Generic       *description; /* description of object */
  bPOSITION     *position;    /* center, radius, bounding box */
  bORIENTATION *orientation; /* normal, up */
  bXFORM        *xform;       /* transformation matrix and local origin */
  struct limbs *constraints; /* for manus */
  bObjDirt      dirty;        /* bobj fields needing recompilation */
  short         visible;      /* is object visible or not? */
  LIST          *drobjs;      /* drOBJECTs: device-specific display info */
} bOBJECT;
```

Figure 4-1: The bOBJECT data structure.

description of the object. The description is read by routines written specifically for that type. These routines build the appropriate data structure for the type. A pointer to this data structure is stored in the description field. Example detail file types include .asc for an ascii representation of an object of type bPOLYHEDRON, .det for a binary description of an object of type bPOLYHEDRON, and .dm for an object of type bDEPTHMAP.

To make the rendering of device independent graphical objects more efficient, they are compiled into a format specific to the hardware platform. This device specific data is stored in structures called drobjs (drawing objects). A bOBJECT structure contains a list of pointers to the drobjs which define the hardware calls needed to display it on the screen. These drobjs are created by the compile routine specific to the bOBJECT description and are in a format dependent on the type of output device to be used for rendering. Currently, a set of data structures is used which is specific to the Starbase graphics system running on the Hewlett-Packard 9000 series of workstations[25].

Another important data structure in the bolio system is the VIEWPORT. This structure describes a portion of the screen and the objects to be rendered there. Each VIEWPORT contains structures defining its lower left and upper right corners, descriptions of the types and colors of its borders and title bar, and flags describing the type of rendering to be performed in this viewport (e.g., wireframe vs. shaded graphics). The VIEWPORT also has a pointer to the bOBJECT which serves as its current camera. The camera object contains a ViewStruct [48] and a pointer to a bOBJ_WORLD (a bOBJ_WORLD is a named list of bOBJECTS). Whenever a viewport needs to be redrawn, the viewing transformations and rendering equations are calculated using the information in the VIEWPORT and the bCAMERA and are applied to all objects pointed to by the bOBJ_WORLD of the current camera.

Many viewports can be displayed simultaneously at various (possibly overlapping) positions on the screen, allowing multiple views of a single bOBJ_WORLD or views of different worlds. For faster screen redrawing, and for making animations directly from the display screen, a borderless, full-screen viewport is available. A red/green stereo mode is also provided in the full screen display mode; in this mode the image is drawn twice for each frame, once in red and once in green, using bCAMERAs the viewpoints of which have been separated to approximate the interocular distance.

4.3 User Interface: Commands and Scripts

Bolio's user interface combines various modes of input. A mouse or graphics tablet can be used to select commands and object names from a menu system and provide 2D input and button events. Alternatively, these same commands, select operations, values, and button events can be generated as text by the keyboard or from script files. This mixed mode form of input is achieved because each selection on the menu is associated with a string, which is placed into an internal string buffer. This string buffer is where bolio commands look to find their input. Input from the keyboard is also placed in this buffer, along with strings from script files.

A stack of script file pointers is maintained to control where input comes from when a new string is required. When the stack is empty, the keyboard is polled for input, otherwise strings are read from the top file pointer on the input stack. The #include command, followed by a filename, is used to push a new file pointer onto the stack. This file is then used for input until it has been read completely, at which point it is popped from the stack. In this way, script files can be nested to an arbitrary depth.

Input can also come from a nine-knob box which is available on the Hewlett-Packard workstations; it is sampled by bolio and made available through a global structure to all commands. Commands are available which use the knobs to position objects and change their optical properties.

The essence of bolio's main loop consists of the following stages: sampling the state of input devices, executing a command (if any are pending), and redrawing the screen if it needs to be brought up to date. Bolio operations (bops) are the commands which are inserted in the main loop. Parameters to bops can be specified on the input line which invokes them in a manner similar to that used to pass arguments to C programs. bops are used to either set modes or directly control some internal variable according to the state of an input device. A bop may remain resident in the main loop, until it reaches a termination condition. Most bops are written so that if they are invoked with an incomplete set of parameters, they remain resident and prompt the user for additional input so they can complete. The `visible` command, for example, sets the visibility flag for a bOBJECT; if it is invoked without the name of an bOBJECT on the command line, it invokes a function which places a menu of selectable bOBJECTS on the screen and remains in the main loop until a bOBJECT is selected. Once it has the bOBJECT, it modifies the bOBJECT's visibility flag and removes itself from the main loop.

4.4 The *Manus* Constraint System

A constraint package was a key element of Sutherland's classic work, *Sketchpad*[51], and of Borning's *Thinglab*[5]. The *manus* constraint package is similar in spirit to both of these systems, although both of them were restricted to 2D graphics. All three systems incorporate rather general mechanisms for defining constraints and constraint satisfaction methods. However, the two earlier constraint systems incorporate an analysis stage, and Borning's work included two additional satisfaction techniques beyond one-pass solutions and relaxation.

Manus was developed initially to handle position and orientation constraints on the motion of rigid objects, and non-rigid motion of polygonal meshes. Thus, unlike the earlier *Sketchpad* and *Thinglab* systems, which were intended to satisfy multiple, interacting constraints encountered in geometric and mechanical design problems, bolio does not perform preliminary analysis of the constraint network. Since relaxation is time-consuming and may not converge, the purpose of this constraint planning step is to identify constraints that can be satisfied by simpler, direct means, so that relaxation is invoked only when necessary. However, since bolio supports an interactive, time-varying virtual environment, perhaps with active agents whose behavior may not be known *a priori*, constraint satisfaction has to proceed in parallel with forward simulation, so that a constraint pre-planning stage may not be feasible.

The *manus* constraint network is composed of bOBJECTs in the bolio world connected by instances of constraints. Each instance of a constraint contains information specific to the objects it is connected to and pointers to the code necessary to process the constraint. Thus, constraint instances share procedures but maintain private copies of relevant data structures.

Each time a constraint instance connects to a bOBJECT which should trigger it, it adds a pointer to itself into the bOBJECTs *who-cares* list (part of the **constraints** structure). Later, when a constraint instance modifies the bOBJECT, the bOBJECT notifies all constraint instances in its *who-cares* list. Those constraints instances then execute, modifying other bOBJECTs which trigger constraint instances in their *who-cares* list, etc. This process proceeds in an orderly manner managed by the *manus_renormalize* function.

When a bOBJECT triggers constraint instances in its *who-cares* list, it actually just puts a pointer to each instance on the end of a global *pending constraint instance list* (pending_queue). The *manus_renormalize* function goes sequentially through the queue (in effect performing a breadth-first search of the constraint network) invoking constraint instances as they are pulled from the list. As constraint instances execute, bOBJECTs they affect place new items at the end of the queue. This procedure continues until the queue is empty. We use several methods to ensure termination of this process:

- Allow a particular constraint instance to be placed on the queue only *once* per frame. This technique is used by the DataGlove constraints[3] which only sample the external device once per frame. Other user interface devices (mouse, knobs) also follow this

[3]See section 4.5.2.

convention.

- Program constraints such that their instances only modify their dependent bOBJECTs once per frame, treating all subsequent constraint invocations as interrupts of the forward simulation of the object's motion, which then instantaneously update the object's position and orientation. E.g., if the user catches a bouncing ball with the DataGlove, the glove constraint causes the simulation of the ball's motion to suspend; the ball is repositioned according to the glove constraint. Constraints are prioritized in the order in which they are invoked. Communication with the *roach* module[4] is also performed using this technique.

- If none of these methods prove flexible enough to handle the interactions of several constraints which all wish to control a single bOBJECT, the final resort is to express the constraints in terms of forces and let the solution evolve over time via forward simulation. This is used, for example, in the case of interpenetration prevention where several constraint instances may all influence the position of the same bOBJECT. This is similar to the technique (which has variously been called *energy constraints*[65], *dynamic constraints*[4], or *physically-based modeling*[54]) in which systems of force generating constraints are solved each frame to determine the positions of objects under the influence of multiple constraints.

- Carefully construct constraint networks keeping in mind the action of the constraint functions so as to avoid dangerous forms of loops.

The *manus_renormalization* function is invoked at three points within each iteration of bolio's main loop. Two special constraint structures exist for the start and end of each iteration, allowing events such as device sampling to be triggered by the start of a new frame or control of the video tape recorder to be signaled by the end of a frame. The renormalization operation is performed after each of these signals is sent. Renormalization is also performed after the command portion of the main loop, between the start and end of the frame.

[4]See section 4.5.3

An example *manus* operation (or **mop**) is the **link** constraint. This **mop** updates the size and position of a "link" object so that it appears to physically connect two other objects. The code which satisfies an instance of this constraint looks at the positions of the two objects and calculates an appropriate transformation matrix for the link object. When either of the two objects is moved, the **mop** is re-executed to properly transform the link object. The constraint instance contains a pointer to the code to calculate the appropriate transformation for the link object given the positions of the other two objects, and a structure containing pointers to the object to be used as a link, the two objects to be linked, and flag telling whether the linking transformation should be volume conserving. The **link** constraint allows the creation of graphical "rubberband lines" using any bOBJECT.

4.5 Bolio Tools

Application modules in the bolio system are called bolio tools. At the UNIX level, a configuration file called BOLIOTOOLS in the bolio directory contains a list of modules to be included in the version of bolio being compiled. The code for each tool is contained in a separate subdirectory of the bolio directory. The tool directory has a makefile[24] to build a library of that tool's code. For each included tool, bolio's compile script executes that makefile and reads three configuration files from the tool's directory: the **bopnames** file, which has a list of the main loop commands included for the tool; the **mopnames** file, which has a list of the constraint commands for this tool; and the **usrlibs** file, which contains a list of other system libraries which should be linked with the tool library into the executable bolio. The first two files are used to construct branch tables for the main command parser and to allow the constraint command parser to map input strings into function calls. The third file is used to add arguments to the command which links the executable bolio.

4.5.1 Core Tools

The bolio system includes a range of interactive graphics facilities on which to build applications. These include utility routines and interactive commands to manipulate the object and viewport structures described above (e.g. using the available input devices for moving and sizing viewports, changing camera parameters, transforming and changing colors of objects, positioning lights, etc.) Constraints to build object transformation hierarchies, substitute objects with a bounding box representation, and form links are also included in the core tools. These tools are of general use and are therefore included in all executable versions of bolio.

4.5.2 *Glove*

The DataGlove is a device which senses the position, orientation, and finger posture of a human hand (see figure 4-2)[70][50]. It transmits this information through a serial communication line to the Hewlett-Packard workstation, where the data can be sampled when needed. The glove is incorporated into bolio's main loop as follows. An instance of the `glovepoll` constraint is attached to the start-of-frame structure (so that glove code is given a chance to execute every time through the main loop). This constraint code updates an internal structure (the `struct glovepoll_data`) containing the current dataglove values; instances of other constraints depend on the values in the `glovepoll_data` structure and are triggered whenever those values are updated. One of the constraints which can be dependent on the `glovepoll_data` structure is the `dghand` constraint, which transforms a set of objects so that their screen position matches the position and orientation information supplied by the DataGlove, thereby providing a screen echo of the hand.

The `glovepoll` constraint also checks the current glove values against a posture table and sets a value in the `glovepoll_data` structure indicating the current hand posture. Constraint instances triggered by the `glovepoll_data` structure use the posture number as well as the position of the hand to perform actions. The `link_near` constraint, for example, stretches an object (draws a rubberband line) from one of the glove objects (usually the index finger tip) to the nearest in a set of other objects in the scene whenever the glove enters the appropriate posture. The `grabber` constraint acts similarly, except that when the correct posture is detected, it finds the nearest object (if it is within a threshold distance) and makes it track the location of the grabbing object (again, usually the index finger tip, whose position is being controlled by the DataGlove through the `glovepoll` constraint). Note that moving the object causes any constraint instances dependent on its positions to be signaled and executed. With this set of constraints, it is possible to use the DataGlove to interact with objects in the virtual world; those bolio tools which use position and orientation of objects as input are automatically able to make use of the glove as an input device.

The condition of matching a recorded posture can be used to trigger the execution of an arbitrary script using the posture_script constraint. A separate script may be associated with each of the following conditions: *entering* a posture, *during* a posture, and *leaving* a posture. The scripts can contain arbitrary commands to any of the bolio tools or can set up or delete instances of constraints. On entering the posture recorded as the *glove view posture*, for example, an instance of the vp_track_obj constraint[5] is established which continuously moves the bCAMERA eye point to the current glove position and keeps the bCAMERA view normal looking at the position of the cockroach. No scripts are executed by the posture_script constraint while the glove remains in the follow posture (the "during" phase). Upon leaving the follow posture, the posture_script constraint executes a script which turns off the vp_track_obj constraint.

Another glove constraint which depends on the sampled data in the glovepoll_data structure is the glove_cursor constraint. When this constraint detects the appropriate posture, it maps finger bends into cursor movement up and down the bolio menu. By making a hand movement, the glove wearer can cause this constraint to pass a button-press event to the menu code and thereby select the menu item. With this constraint, the entire system can be controlled through posture and positioning input from the glove rather than the traditional keyboard and mouse.

The glove tool includes bop commands which print the status of the glove, calibrate joint angles, set up posture conditions, and reset the glove hardware.

4.5.3 *Roach*

A control structure for hexapod walking has been implemented based on research into the neural mechanisms found in cockroaches[19]. The control system includes coupled oscillators to coordinate the action of the legs to form appropriate gait patterns for a given speed. The

[5]See section 4.5.8 below.

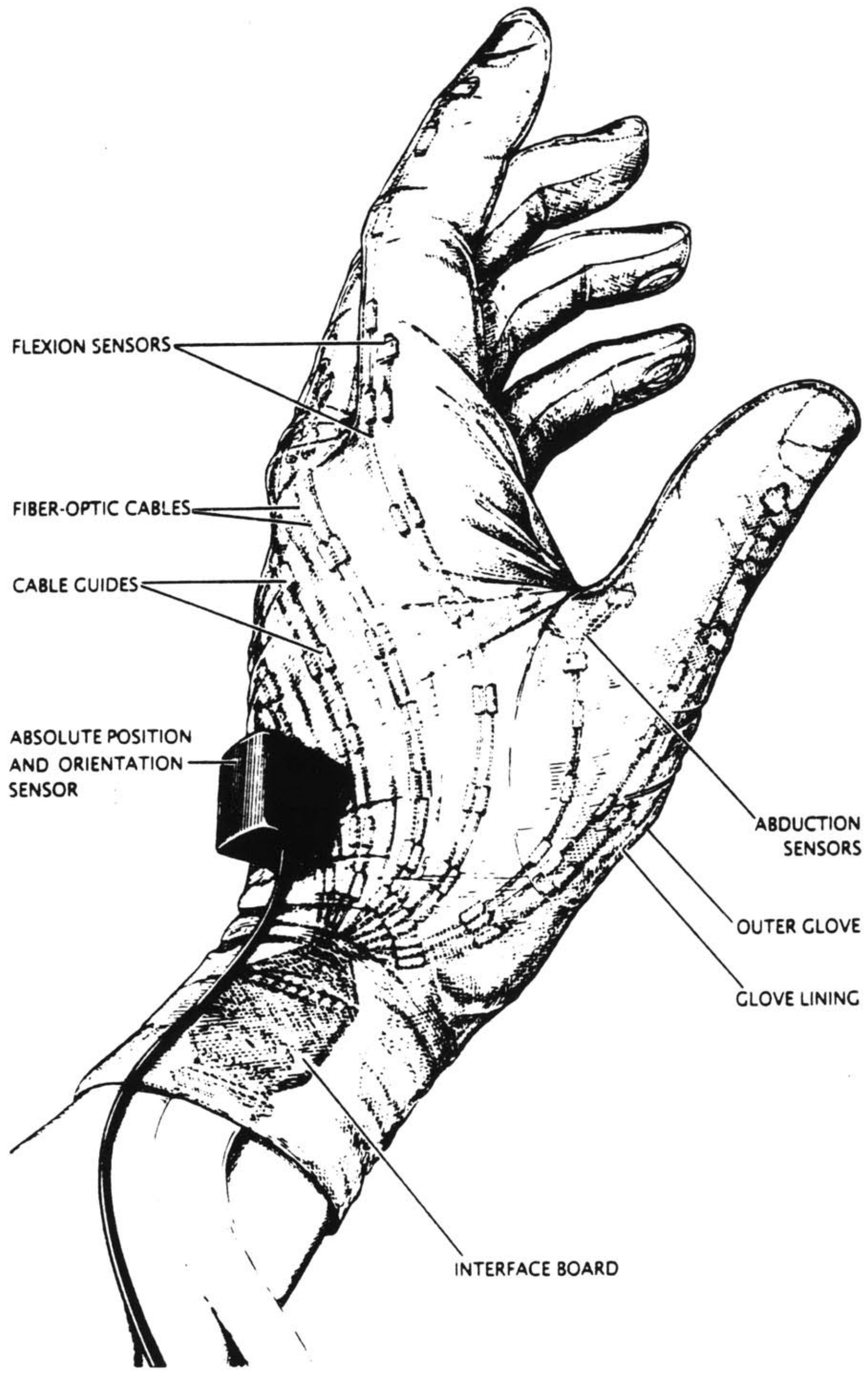

Figure 4-2: The DataGlove, developed by VPL Research Inc., provides real-time sampling of hand movement. The Polhemus sensor, manufactured by Polhemus Navigation Sciences division of McDonnell Douglas Corporation, detects the position and orientation of the wrist, while a system of modified fiber-optic cables stretched over the fingers detects the bend angles of the hand's joints[17].

implemented model uses inverse-kinematics to position the leg according to the current body position and the desired foot position[33][34].

Communication between the bolio environment and the *roach* module is accomplished through two constraints. The first constraint is **roachwalk**, an instance of which is attached to the start of the frame for each roach in the environment. It queries the *roach* module for the transformation matrices defining the positions of the roach's parts (each movable part is represented by a bOBJECT) for the current frame and triggers any constraint instances which depend on those bOBJECTs. The other constraint is the **roachorient** constraint. It provides a communication path from the rest of the bolio environment to the roach through the bOBJECT representing the roach's body. Any time the roach's body is moved, the **roachorient** constraint sends the information to the *roach* module, which updates its internal data structures accordingly. Thus, for example, if the DataGlove picks up the roach body, the *roach* module ensures that the roach's legs move too.

The results of simulated physical interactions are also propagated to the *roach* module through the **roachorient** constraint; the dynamic simulation tracks the position of the roach through a structure constrained to follow the roach's body object and updates the position of that object according to the forces that affect it such as gravity and ground forces or the collision force when the roach walks into the wall. It should be noted that through the constraint structure the dynamic simulation module can incorporate the action of the roach into the dynamics of the rest of the environment, for example allowing the roach to push on objects when there is a collision.

Commands coming through bolio's standard input which begin with the word "roach" are passed to the *roach* module's input parser. With this it is easy to mix *roach* initialization and control commands into the scripts which set up and modify bolio environments.

4.5.4 *Sa*

Sa is a figure animation system which includes routines for describing and manipulating jointed figures, an event-driven simulation mechanism, and an animation language for controlling jointed figure motion[68][67]. Like many of the bolio tools, *sa* was developed as a standalone program generating its own graphical output. By putting it into bolio, figures *sa* controls can interact in a graphical environment which includes, for example, hexapods and the DataGlove. *Sa*'s simulation mechanisms take into account the articulated structure of a walking figure, the current gait, and the support requirements to place the figure's limbs in space. The transformation matrices which define the positions of the figure's parts are copied to the bOBJECT which represents that part. Future work will use *manus* to further integrate *sa* into the simulation environment for control legged locomotion with inverse-kinematics and dynamics.

4.5.5 *Pathplan*

The *pathplanning* problem is one of finding a collision-free path through a cluttered environment for a moving agent.[6] A map of the free space in the environment is constructed by projecting the bounding box of each of the objects onto the ground plane of the environment. The generalized bounding boxes are grown by the radius of the agent, so that in subsequent operations the agent may be thought of as a single point. The map defining the regions through which the agent may not travel is represented by the *pathplan* module as a visibility graph (vgraph). To find a collision free path from current position of the agent to any other point in the environment, the straight path is first checked for collisions with objects in the environment, and is used if there are no collisions. If the direct path can't be used, the vgraph is searched for the shortest alternate route which does not pass through any objects.

As a preprocessing step, the *pathplan* module in bolio creates the vgraph from the bOBJECTs in the current bOBJ_WORLD (with the exception of any bOBJECTs, for example, the agent's body parts and the object representing the floor, entered in a special pathforbid list). A data structure representing the vgraph is maintained internally by the *pathplan* module and is optionally displayed as a bPOLYHEDRON consisting of polylines. When requesting paths, the start and end points may be specified either as numbers or as the names of bOBJECTs whose world space centers are to be used. The resulting path is output as a bPOLYHEDRON, the vertices of which are the points along the path. In this way, the path can be displayed in the world, and modules which use the path to control movement

[6]A detailed description of the implementation of pathplanning in bolio may be found in [47].

(currently only the *roach*) can access the resulting bOBJECT without knowledge of the source of the path.

4.5.6 *Face*

The facial tissue simulation code described in this thesis is implemented as a boliotool. The next chapter describes its implementation in detail.

4.5.7 *Cam_moves* and *Moves*

The *cam_moves* and *moves* bolio tools provide bops and mops for defining the movements of cameras and objects over a sequence of frames and for defining their motion with respect to the motion of other objects in the scene. The mv_obj_path constraint and the mv_vp_path constraint cause a bOBJECT or a bCAMERA respectively to move along a piecewise linear path defined by the vertices of a bPOLYHEDRON. The vp_track_obj constraint is used to control the position and orientation of a bCAMERA using bOBJECTs to define one or both of the following parameters. The *eye* bOBJECT defines the viewpoint of the object. The *lookat* bOBJECT defines the direction for the view normal of the bCAMERA. If the eye bOBJECT is not specified, the bCAMERA view point remains stationary and the view normal turns to track the current position of the lookat bOBJECT. If only the eye bOBJECT is specified, then the view normal is defined by the orientation of the eye bOBJECT. If both the eye and the lookat are defined, then the bCAMERA position is determined from the eye bOBJECT, and the view normal is defined by the lookat bOBJECT.

4.5.8 *Robot*

A set of routines to perform inverse-kinematics on jointed figures are provided by the robot tool. The routines allow the specification of jointed figures using the Denavit-Hartenberg (DH) description conventions, which define articulated links in terms of their rotation axes and lengths[45]. This information is embodied in the Jacobian matrix, which can provide the joint angle velocities needed to achieve a given end effector velocities through the calculation of its pseudo-inverse[41][14].

The inverse-kinematics constraint in bolio (the `ik` constraint command) takes as arguments a list of bOBJECTS which will serve as joints for an articulated figure. The DH description of the figure is calculated based on the current locations of the objects in the list. The first item on the list is the base and the last item is the end effector. The inverse-kinematic code to calculate new joint angles is constrained to run any time either the base object or the end effector object is moved. In this way, the pose of the figure can be manipulated by any bolio tool (the DataGlove or dynamic simulation) while maintaining the kinematic relationships of the joints.

4.6 The Roach 'n' Glove Microworld Demo

A more complete example may clarify how these constraints are used, if not details of their implementation. The roach and glove demo is a set of bolio scripts and menus which make use of the most advanced features of the system. The next sections describe the simulation in terms of the commands with which it can be manipulated. The following set of commands is available at the keyboard or via of pop-up menus (items from the menus can be picked with the DataGlove or the mouse).

While the simulation is running, it is possible to switch between two displayed scenes. The first scene, or world, is the *roach world* which consists of the cockroach and various objects (cubes and soccer balls) on a grid floor. [7] The second world is the *spring world*, which consists of only four of the objects from the roach world (three soccer balls and a cube) which can be connected together via springs. In both worlds, the hand of the user (who is wearing the DataGlove) moves in space among the objects. By forming the "grab" posture, the `link_near` constraint is triggered and a red line is drawn from the current glove location to the nearest of the objects in the scene. If the user moves so that the glove touches an object (moves within a specified distance of the bounding sphere), the glove's `grabber` constraint causes the object to track the motion of the glove. This tracking continues until the user leaves the grab posture.

In the roach world, the cockroach walks around on a ground plane randomly, in response

[7] Note that the word objects here refers to the cubes and soccer balls, not the floor, roach legs, and dghand parts, all of which are bOBJECTs.

to a "follow" posture, or according to a path generated by the pathplanning module. For random walking, the roach code merely picks a new point on the plane at random and walks to it; when that point is reached, a new random point is picked and so on. Random walking continues until some other *roach* command is selected. When the user's hand is in the follow posture, a red line is drawn from the center of the roach to the projection of the glove on the ground plane and the roach is told to walk along the path of the red line. The red line and the roach's endpoint are updated every frame as long as the user is in the follow posture. The last two methods for controlling the roach do not take into account obstacles in the path. To create a path which avoids obstacles, the pathplanning tool can be invoked. The first step in using the pathplanner is to build a `vgraph` (which must be recalculated if the objects in the scene move) and then to request that a path be found. The roach is used as the start point of the path, and the projection of the glove on the ground plane is the end point of the path; the roach is then told to follow the path. It is also possible to pick up and reposition the roach by moving close enough while in the grab posture.

Appendix A contains a list of the commands available to the user and describes how they affect the constraint network and the data structures.

Chapter 5

Implementation of Dynamic Tissue Simulation

This chapter presents the implementation details of the deformation model presented in section 3.2. Just as the bolio animation system described in the last chapter is designed to serve as a framework for testing interactive simulations, the soft tissue simulation package described in this chapter is designed to serve as a framework for testing non-rigid dynamic simulation techniques. A modular approach to this problem is crucial since the development of the ideal system described in section 3.1 will require continued research.

5.1 Overview

The code for the facial tissue animation system is modularized as follows. The state information for the dynamic simulation is stored in a network of structures. Each point sample structure (a `struct blot`, see figure 5-1) contains state information for that point and a pointer to the constraints structure (the same structure as that pointed to by `bOBJECT`s as described in chapter 4), which describes its relationship to the other point samples and structures in the model. The relationships are represented by instances of constraints, each of which calculates a contribution to the force at the point sample. The most common constraint operator is the `shock`, which represents a generalized spring between two blots. The constraint operator has a local data structure which encodes state data about the spring

```
struct blot
{       char            *name; /* for command line interactions */
        float           mass; /* mass represented by this point sample */
        WorldPoint      position; /* pull vectors are relative to here */
        WorldPoint      velocity; /* movement per unit time */
        WorldPoint      force_so_far; /* sum of forces */
        int             mobile; /* if TRUE, blot position can be updated */
        struct limbs    *constraints; /* constraints acting on this blot */
};
```

Figure 5-1: The `struct blot`.

(see figure 5-2) and has pointers to the two blots which make up the endpoints of the spring.

The simulation time is recorded in a global structure called `Time`, of type `struct timer`, as illustrated in figure 5-3. This structure is used by the numerical integration routines in the `simulate` command[1] to trigger the execution of any force generating constraints acting on blots.

The `FACE` structure (figure 5-4) is a set of lists which collect the blots and shocks into layers. Currently, the `FACE` structure contains lists for three layers of blots and five layers of shocks. The `FACE` structure also contains pointers to bolio polyhedral objects (`bOBJECTS` of type `bPOLYHEDRON`), which are used to render the results of the simulation. The `FACE` structure also contains a list of muscle_units (see figure 5-5), each of which is a named collection of shocks attached to blots in the face.

The `MUSCLE` structure is a named collection of springs which can be controlled as a unit.

5.2 Force Constraints

The following constraint operators each define contributions to the force at blots given the current state of the blots. Each instance of a force generating constraint is made dependent on the start of a time step so that all instances of these constraints are executed once per time step to calculate the total force acting on each blot. Their mathematical definition

[1]See section 5.8.

```
struct shock_data
{       int     elasticity; /* index into ela_funct array of pointers
                                to elasticity functions */
        float   constant; /* spring constant for this shock */
        float   viscosity; /* damping constant for this shock */
        float   orig_len; /* original length of this connection */
        float   current_len; /* last length calculated */
        float   cutoff; /* where the shock cuts off
                                (only if positive) */
        struct blot *side_1, *side_2; /* the two sides of the spring */
};
```

Figure 5-2: The struct shock_data.

```
struct timer
{       float   now;    /* current time value */
        float   then;   /* last time; dt = now - then */
        struct limbs    *constraints;   /* all force constraints are
                                           triggered by this */
};
extern struct timer     Time;
```

Figure 5-3: The struct timer and the variable Time.

```
typedef struct
{
     char       *name;              /* the name of this face */
     bOBJECT    *source;            /* the source geometry of the face */
     bOBJECT    *bones;             /* the bone drawing */
     bOBJECT    *deep_fascia;       /* the deep fat drawing */
     bOBJECT    *smas;              /* the the smas drawing */
     bOBJECT    *sup_fascia;        /* the upper fat drawing */
     bOBJECT    *skin;              /* the the skin */
     bOBJECT    *muscles;           /* the muscle drawing */

      /* all the point samples of the face, lists of pointers to
              struct blot */
     LIST  *bone_blots;             /* immobile blots of the bottom layer */
     LIST  *smas_blots;             /* the superficial musculo-aponeurotic
                                       system layer */
     LIST  *skin_blots;             /* the dermal layer */

      /* the next four lists are lists of face_shock_entries */
      /* springs representing horizontal layers */
     LIST    *bone_shocks;   /* bottom layer of tissue */
     LIST    *smas_shocks;   /* aponeurotic tissue */
     LIST    *skin_shocks;   /* dermal tissue */
      /* springs representing vertical and diagonal layers */
     LIST    *deep_fascia_shocks;     /* deep fascia */
     LIST    *sup_fascia_shocks;      /* superficial fascia */

      /* muscles */
     LIST    *muscle_units;  /* list of muscle structures */
} FACE;
```

Figure 5-4: The FACE structure.

```
typedef struct
{       char    *name;      /* the name of this muscle */
        LIST    *fibers;    /* the springs making up the muscle pointers
                               to face_shock_entries */
        struct limbs   *arms;   /* for hanging constraints (i.e. twitch) */
} MUSCLE;

struct face_shock_entry
{       int       from_blot_layer;   /* which layer list to use */
        int       from_blot_number;  /* which entry in the layer list */
        int       to_blot_layer;
        int       to_blot_number;
        struct manus    *shock;    /* the constraint data structure */
};
```

Figure 5-5: The `MUSCLE` and the `struct face_shock_entry`.

was given in section 3.2.1. This section describes their implementation and data structures.

5.2.1 Shock

The shock constraint, as already mentioned, acts as a generalized spring. Its parameters are two blots, a spring constant, and an elasticity function. On initialization, it calculates the distance between the two blots and stores that value as the `orig_len`. When invoked at a time step, it compares the current distance (calculated for the time step in `current_len`) between the two blots to the original distance. A force is applied to each blot in either the direction of the other blot (if the current length is greater than the original) or away from the other blot (if the current length is less). The magnitude of the force is the constant times the value returned by the elasticity function for the shock.

5.2.2 Gravity

The gravity constraint is very similar to the shock constraint, except that the force it generates is dependent on the masses of the blots it acts between, rather than the distance between them. A blot named "earth" is automatically created and serves as the default connection point for gravity constraints acting on blots in the FACE structure.

5.2.3 Drape

The drape constraint is basically a collision detection constraint used to keep a blot from entering through a region. Unlike shock and gravity constraints, this constraint depends on a blot's position and on a region defined by a bOBJECT. Forces are applied to the blot to keep it out of a region defined by the bOBJECT and the block_type field of the struct drape_data structure (see figure 5-6). Two values of block_type are valid for any type of bOBJECT; they are the block_plane which defines the region above the plane defined by the world-space z value of the bOBJECT, and the block_sphere which defines the region inside the world-space bounding sphere of the bOBJECT. The block_depthmap is valid only for bOBJECTs of type bDEPTHMAP. In this type of drape constraint, the region into which the blot cannot pass is defined by the values in the array of depth samples in the bDEPTHMAP and the local-space to world-space transformation matrix of the bDEPTHMAP. To determine the force on a blot, the inverse of the bDEPTHMAP's transformation matrix is applied to the blot's position, which puts it into the bDEPTHMAP's local-space. The blot position's x and y components in this space are used as indices into the bDEPTHMAP array. A weighted average of the z values at the points surrounding the transformed blot is calculated and compared with the transformed z value of the blot; if the blot's value is less than the depthmap sample, a force is calculated to move it toward the surface of the depthmap. This force is then transformed to world space and scaled by the constant and the weighting returned by the elasticity function.

5.2.4 Cling

The cling constraint is exactly like the drape constraint, except that it always applies forces to attract the blot to the surface of a region, independent of whether the blot is inside or outside the region. The regions are defined just as they are in the case of the drape constraint.

```
struct drape_data
{       int     elasticity; /* index into table of elasticity functions */
        float   constant; /* material constant for the collision */
        float   threshold; /* how deep before cutting off */
        float   distance;  /* how far away to start */
        int     block_type; /* 0 => bobj bounding sphere
                               1 => bobj world center z value */
};
```

Figure 5-6: The struct drape_data.

5.3 Elasticity Functions

All of the force generating constraints described above (except gravity) have as a parameter an index into a table of elasticity functions. These functions describe how much force will be generated by a constraint relative to how much the constraint is violated. Each elasticity function receives as arguments the rest state and the current state of the constraint and returns a positive real number weighting value. Currently implemented functions were presented in section 3.2.1.

5.4 Muscle Control

5.4.1 twitch

The twitch constraint implements the key-framed control of muscle actions. Muscle actions are defined as a list of time/scale pairs. The scale value tells how much each of the muscle fibers is to be scaled relative to its original length at the specified time. At each time step, the lengths of the muscle fibers are updated to an interpolated value between the nearest time samples before and after the current time. Cosine interpolation is used to calculate the intermediate value. Figure 5-7 shows the struct twitch_data which stores the information associated with a twitch constraint.

```
struct twitch_data
{   float     basetime;   /* all times entries are relative to this */
    float     times, scales;  /* time/scale pairs for interpolation */
    int       nosamples;  /* number of time/scale pairs */
    int       last_sample; /* makes the current entry easier to find */
    MUSCLE    *muscle;    /* which muscle to effect */
    float     *original_lengths;
                          /* original lengths of each muscle fiber */
    int       no_fibers;  /* only apply to original number of fibers,
                            also tells size of original_lengths array */
    struct limbs  *constraints; /* in case of future constraints which
                             might relate these parameters to other
                             variables or to a model of higher control */
};
```

Figure 5-7: The struct twitch_data.

5.4.2 muscle_set

The rest lengths of the springs which serve as the fibers of the muscle units can be instantaneously set using the muscle_set command. The fibers of a specified muscle can either be set to a specified length or can be scaled by a specified amount relative to their current lengths. This command has no effect if the muscle is currently under the influence of a twitch constraint.

5.5 Drawing and Interaction Constraints

In keeping with the bolio philosophy of modularity, the interaction between the facial tissue simulation code and other simulation modules is performed through intermediate bOBJECT structures. This can happen at two different levels. The first level associates a bOBJECT with a single point sample in the tissue simulation. The second level associates a bOBJECT with the network of point samples and springs in the FACE structure.

5.5.1 bobblot and blotbob

Two symmetric constraints are used to associate the position of point sample with the position of a bOBJECT. An instance of the bobblot constraint applies a translation to the matrix of a bOBJECT such that the center of its bounding box in world space is equal to the position field of the struct blot which it is constrained to follow. The instance of this constraint is made dependent on the struct blot so that whenever it moves, the corresponding bOBJECT is updated.

The complementary constraint to the bobblot is the blotbob. An instance of this constraint sets the position of a struct blot equal to the world space center of the bounding box of the bOBJECT which it is constrained to follow. The instance of this constraint is made dependent on the bOBJECT so that whenever it moves, the corresponding struct blot is updated. This constraint allows the physical simulation to be influenced by manipulation of bOBJECTs by other simulation or input modules (such as the DataGlove).

bOBJECT/struct blot pairs can be connected by one instance of each of these constraints. For example, a bOBJECT ball shape and a struct blot which stores the dynamic information about the ball can be connected using the combination of the two constraints. Normally, the position of the ball is controlled by the evolving dynamic simulation — it may fall under gravity, bounce off the walls, floor, or other objects, or be connected via springs to other objects in the world. If the position of the ball bOBJECT is changed by something other than the physical simulation (say the DataGlove), then that new position is taken as input to the simulation and influences the further evolution of the dynamic system.

5.5.2 update_all_face_objects

The shapes of surfaces and solids represented in the FACE structures are displayed by making a correspondence between the position fields of the struct blots and the positions of vertices in the set of bPOLYHEDRON bOBJECTs maintained in the FACE structure for the layers of the network. This operation is invoked whenever a new drawing of the tissue simulation is required. It updates the positions of the vertices and recalculates the surface normals for the polygons representing the skin surface so that an accurate shaded picture can be drawn.

There is currently no high-level way to manipulate the struct blots in the FACE structure through the results of other object positioning modules (e.g., grabbing a handful of tissue with the DataGlove), because the current simulation and rendering interface is not fast enough to support that level of interaction.

5.6 Building FACE Structures

The flesh_out command is used to build a FACE structure from a *source* bPOLYHEDRON. The command creates of the FACE structure such that the initial bone layer is the exact shape and position of the source bPOLYHEDRON surface. The vertices of the source bPOLYHEDRON define the positions of the point mass samples, and the edges of the polygons on the source bPOLYHEDRON surface provide the local connectivity information to define the location of the spring constraints which define the tissue layer.

Two types of connections are formed between the point samples within a layer. A *membrane* connection is created between each pair of point samples which correspond to vertices connected by an edge in the original bPOLYHEDRON description. The membrane connections are meant to maintain the original distance between points, and thus are resistant to stretching and compression. *Plate* connections are meant to resist bending of the material. For each pair of membrane springs which share a common point sample, an additional spring is added between the other two point samples. This forms a triangular shape which resists bending.

The flesh_out command constructs the three tissue layers by building out from the vertices of the source bPOLYHEDRON along their *vertex normals.*[2] The parameters of the command include the *deep_fascia_thickness* and the *superficial_fascia_thickness*, which define the distances between the bone layer and the smas layer and the smas layer and the skin layer, respectively, as measured along the vertex normal. The bone layer is coincident with the surface of the source bPOLYHEDRON. Each of these layers has the same arrangement of point samples and springs. Two spring connections are made between each pair of adjacent layers for each spring connection made within the smas layer. This forms diagonal cross-bars which resist vertical compression and torsion of the tissue.

[2]The normal at a vertex is defined as the average of the outward facing normals of the polygons of which the vertex is a member, normalized to unit length.

5.7 Numerical Integration

The numerical integration scheme (as described in section 3.2.1) is executed using the `simulate` command. This command accepts as parameters the time step to use (*dt*) and the number of iterations to perform (it will continue until interrupted with a button or keyboard event if the number is negative). The command also accepts a number of optional arguments. The `-until time` flag causes simulation to continue until the `now` field of the `Time` structure reaches the specified time. Normally, the `update_all_face_objects` function is called every time step, however, the `-skip count` flag can be used to tell the `simulate` command to perform count steps of integration before changing the shape of the `FACE bPOLYHEDRONs`. This also inhibits redrawing of the graphics screen and thus saves time when simulating the tissue at higher time resolution.

The `-runge_kutta` flag causes the `simulate` command to use a second-order version of Runge-Kutta numerical integration. In this case, the command saves the current states of all the blots in the simulation, and then forward simulates using the standard Euler integration technique to get an approximation of the blot positions at the end of the time step. At this point, the force constraints, which are dependent on the `Time` structure, are triggered again to calculate the total force which would be acting on the blots if they were actually moved to those positions. The force and velocity values calculated at the end of the time step are then averaged with the values saved from the beginning of the time step. These new averaged values are use to make the final calculation of the positions and velocities at the end of the time step.

Chapter 6

Results

6.1 Experimental Results

Figures 6-1 through 6-3 show the behavior of the simulation technique under a variety of initial and boundary conditions.

6.1.1 Wound Closure

Figure 6-1 shows a sequence of states from a simulation of wound closure in a tissue sample. The force constraints within the lower two tissue layers are shown as lines, and the skin layer is shown as a polygon mesh. The force constraints between the layers are not shown. The initial configuration of the tissue model contains a hole which models a section of excised tissue. A set of force constraints representing sutures are connected to the reference points on the sides of the wound. The rest length of these force constraints is decreased over the course of the simulation in order to effect wound closure. The boundary of the tissue sample is unconstrained.

The simulation results show the deformation caused in the surrounding tissue as a result of the wound closure procedure. The resulting deformation causes not only deformation in the plane of the skin, but also shows changes in the thickness of the skin at various regions around the wound closure. The bulges at the near and far sides of the closure match observed "dog's ears" behavior of real tissue in response to surgical procedures[64].

The following is a bolio command script which controls simulation of the closure of an elliptical excision. Semicolon is the comment character in bolio. The source bPOLYHEDRON, fusiform, used by the flesh_out algorithm has missing polygons in the region which models the excision. It was created using a standard 3D graphics object modeler[52].

```
;; wound closure test script
;;
; setup graphics
#include lookat_wound
; read the source data object (this object has polygons missing to
; approximate a fusiform wound shape)
instance_object /u/pieper/data/fusiform
; Build flesh layers out from the object.  Both fascia layers are 0.07
; units thick (the fusiform object is 1 x 1 square).  Each point sample
; has a mass of one unit (the -me flag sets the mass each).  The force
; constraints built by the flesh_out command have a default material
; constant of 1.
flesh_out fusiform -dfth 0.07 -sfth 0.07 -me 1
; Make triangles out of the polygons in the top surface
triangulate skin.face.fusiform
; Make original object invisible
visible fusiform -off
; use muscle structures to form sutures across the opening
; first two numbers are indices into the point sample list indicating the
; points to connect, the second two numbers indicate which layers to
; connect.  The -c flag sets the material constant, the -m flag sets the
; material type to linear.
; put a set of sutures across the skin
add_muscle skinsuture 102 103 2 2 -c 5 -m 4
add_muscle skinsuture 91 92 2 2 -c 5 -m 4
add_muscle skinsuture 81 82 2 2 -c 5 -m 4
add_muscle skinsuture 70 71 2 2 -c 5 -m 4
; put a set of sutures across the smas
add_muscle smassuture 102 103 1 1 -c 5 -m 4
add_muscle smassuture 91 92 1 1 -c 5 -m 4
add_muscle smassuture 81 82 1 1 -c 5 -m 4
add_muscle smassuture 70 71 1 1 -c 5 -m 4
; put a set of sutures across the bone
add_muscle bonesuture 102 103 0 0 -c 5 -m 4
add_muscle bonesuture 91 92 0 0 -c 5 -m 4
add_muscle bonesuture 81 82 0 0 -c 5 -m 4
add_muscle bonesuture 70 71 0 0 -c 5 -m 4
; set up key-framed lengths for the sutures
setup twitch skinsuture 0 1 5 0.1 10 0.01 40 0.01 50 .5 -start 0.5
setup twitch smassuture 0 1 5 0.1 10 0.01 40 0.01 50 .5 -start 0.5
setup twitch bonesuture 0 1 5 0.1 10 0.01 40 0.01 50 .5 -start 0.5
; set damping value
damping 0.2
; simulate with Runge-Kutta integration, with timesteps of 0.01,
; redrawing every 5 frames, stop when time reaches 100.
simulate -dt 0.01 -skip 5 -runge_kutta -until 100
```

6.1.2 Draping Over Hard Tissue

Figure 6-2 shows deformation of a tissue sample as it travels over underlying hard tissue. The bottom layer of the material is collision detected with the hard tissue via the drape constraint operating on the sphere object (using the **drape_sphere block_type**) and the floor (using the **drape_plane block_type**). The interaction of soft tissue with underlying hard tissue is important for modeling the interaction of the lips and cheeks with the teeth, and the general movement of the skin under the influence of muscle actions.

The following is a bolio command script which controls simulation of the tissue sample draping over hard tissue.

```
;; balldrape
;;
;; animation test for skin dynamics, soft tissue draping over hard
;;
;
; setup view
#include lookat_balldrape
; read base object for building tissue
instance_object data/grid.10.10 sourcegrid
; read the ball
instance_object data/sphere.obj ball
; read the plane
instance_object data/grid.5.5 floor
instance_object data/wiregrid.5.5 gridfloor
edit_xform floor -t 0 0 -0.2501 -s 4 4 1
edit_xform gridfloor -t 0 0 -0.25 -s 4 4 1
; change colors for the floor
colors floor -faceted -diff .98 .98 .98 -high 0.8 0.8 0.8 -spec 214
colors gridfloor -faceted -wire .05 .05 .05
colors ball -smooth -wire .05 .05 .05
;
```

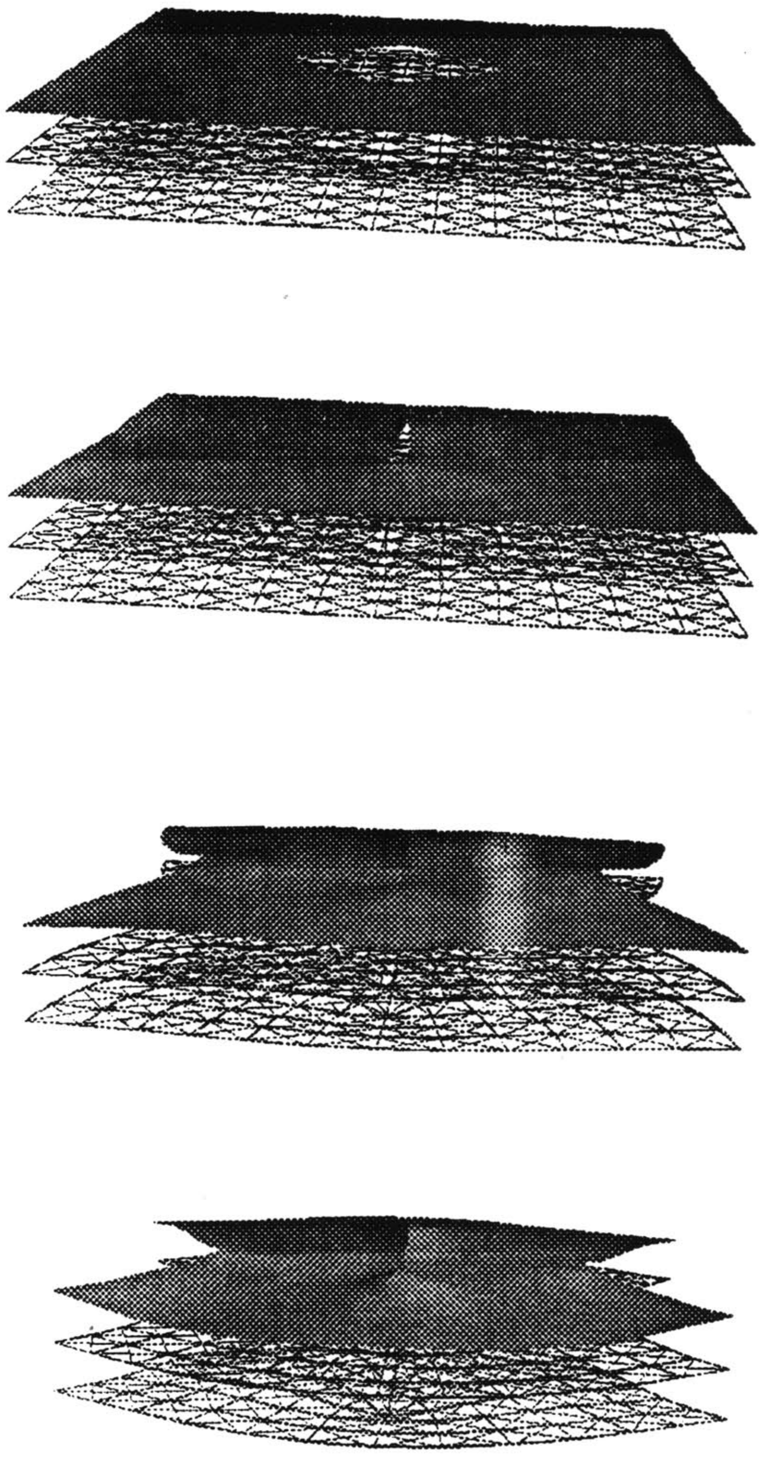

Figure 6-1: (a) Progressive states of wound closure simulation. The simulation model contains 522 sample points.

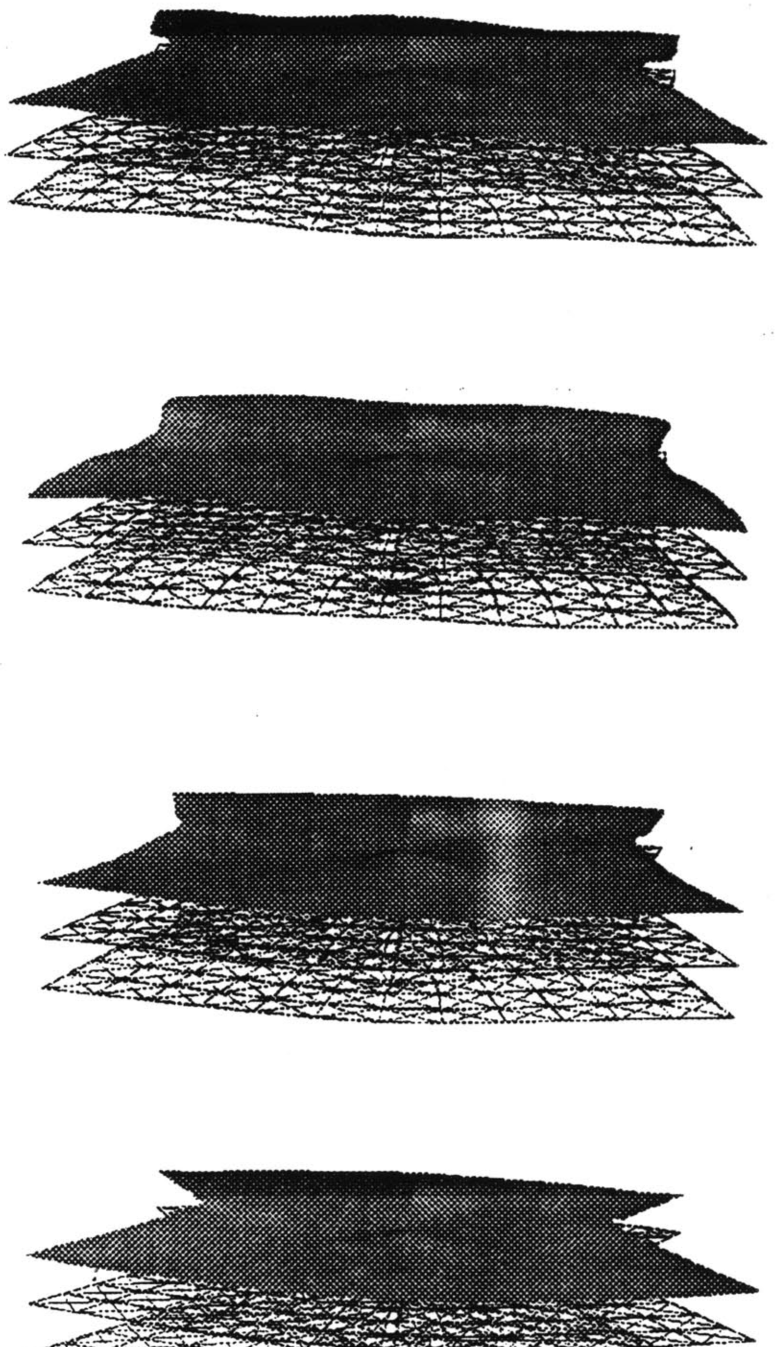

Figure 6-1: (b) Progressive states of wound closure simulation.

```
; build flesh from source object.  Fascia layers are 0.1 units thick
; (source grid is 1 x 1).  Each point sample has a mass of one unit (the
; -me flag sets the mass each)
;
flesh_out sourcegrid -dfth 0.1 -sfth 0.1 -me 1
; make skin surface renderable
triangulate skin.face.sourcegrid
; make the source invisible
visible sourcegrid
; add gravity to all the face blots
gravity_face -g 0.1
; drape over ball's bounding sphere -c sets the material constant
; of the drape
drape -c 1000 -s ball
; drape over floor plane
drape -c 1000 -p floor
;add_muscle muscle 35 65 1 1 4 20
; set damping to quell oscillation
damping 0.2
;
simulate -dt 0.05 -skip 2 -runge_kutta -until 50
```

6.1.3 Muscles Acting Within Tissue

Figure 6-3 shows a tissue sample falling under the influence of gravity (the **gravity** constraint) and colliding with a solid floor (the **block_plane** type **drape** constraint). A set of five muscle fibers are connected across the tissue sample which contract to deform the material. This simulation shows the combination of the following important aspects of facial tissue modeling: the multiple layer of tissue interact and deform in a realist volumetric bulging in opposition to the line of contraction (the *poisson effect*); the soft tissue interacts with underlying solid surface in a manner analogous to the interaction of soft tissue with underlying hard tissue; and force constraints representing muscle actions are attached at arbitrary points with in the tissue and controlled to obtain deformation behavior in the tissue.

The following is a bolio command script which controls simulation of the tissue sample with muscle action.

```
;; twitch
```

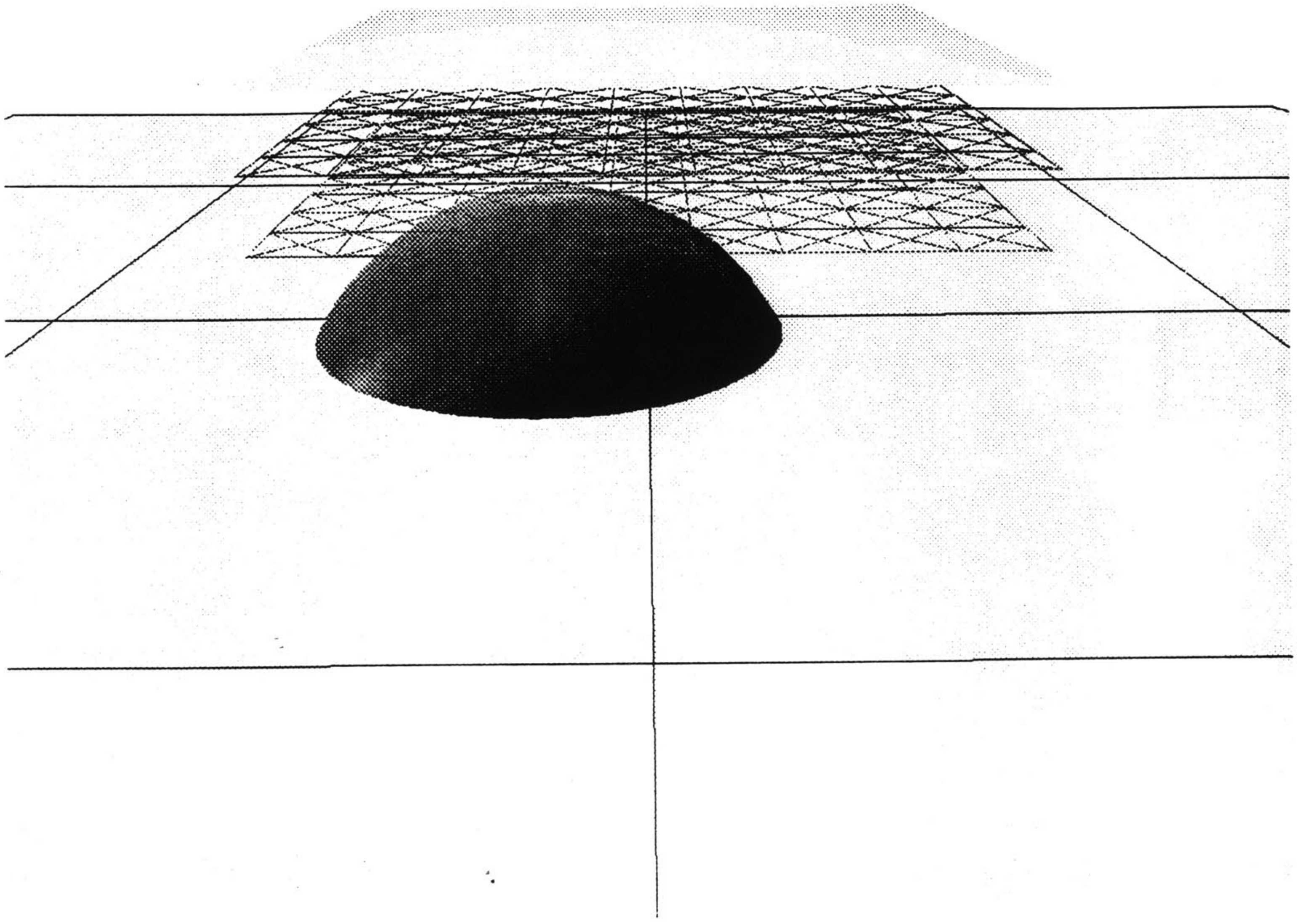

Figure 6-2: (a) Progressive states of hard tissue interaction simulation. The simulation model contains 300 sample points.

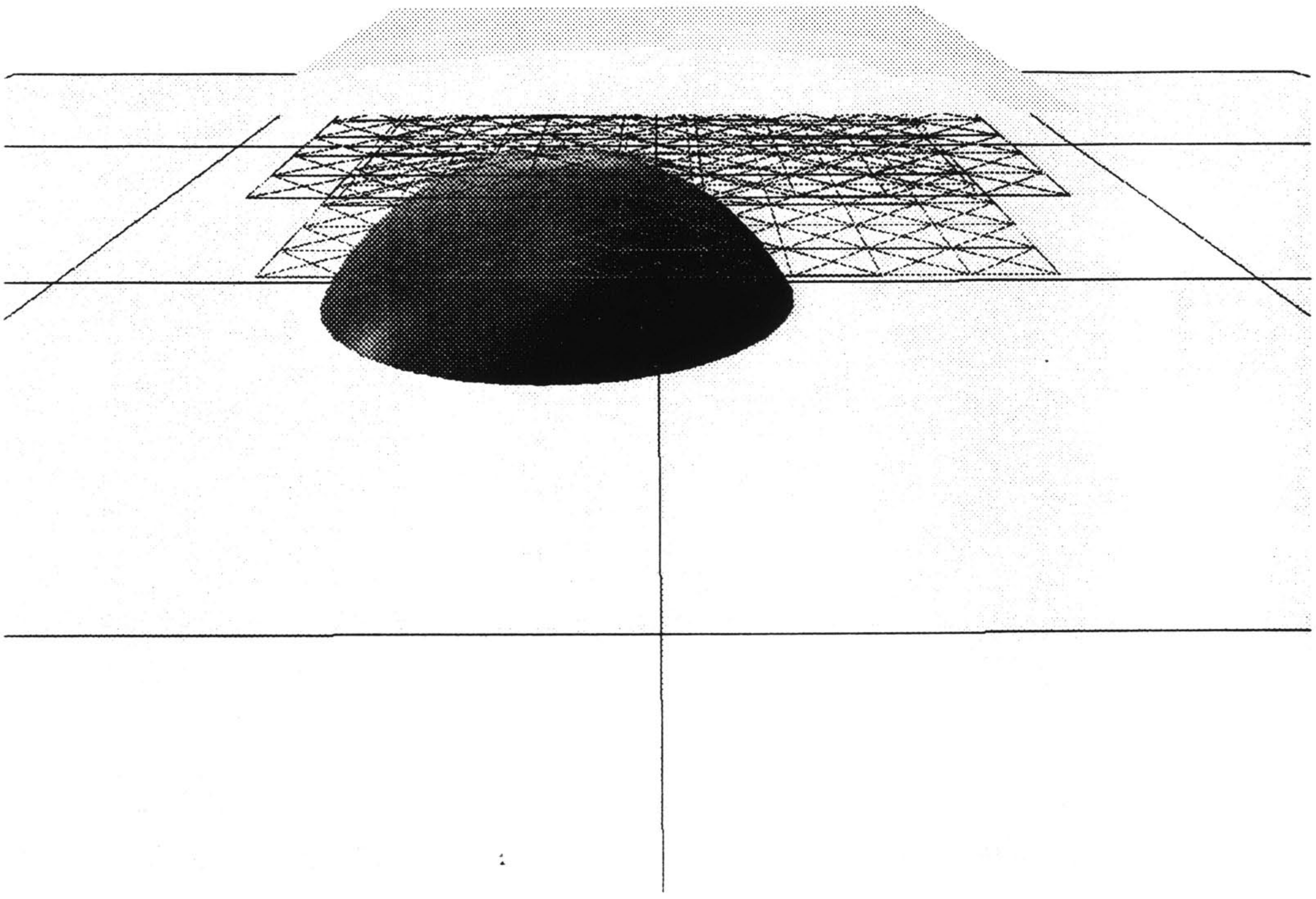

Figure 6-2: (b) Progressive states of hard tissue interaction simulation.

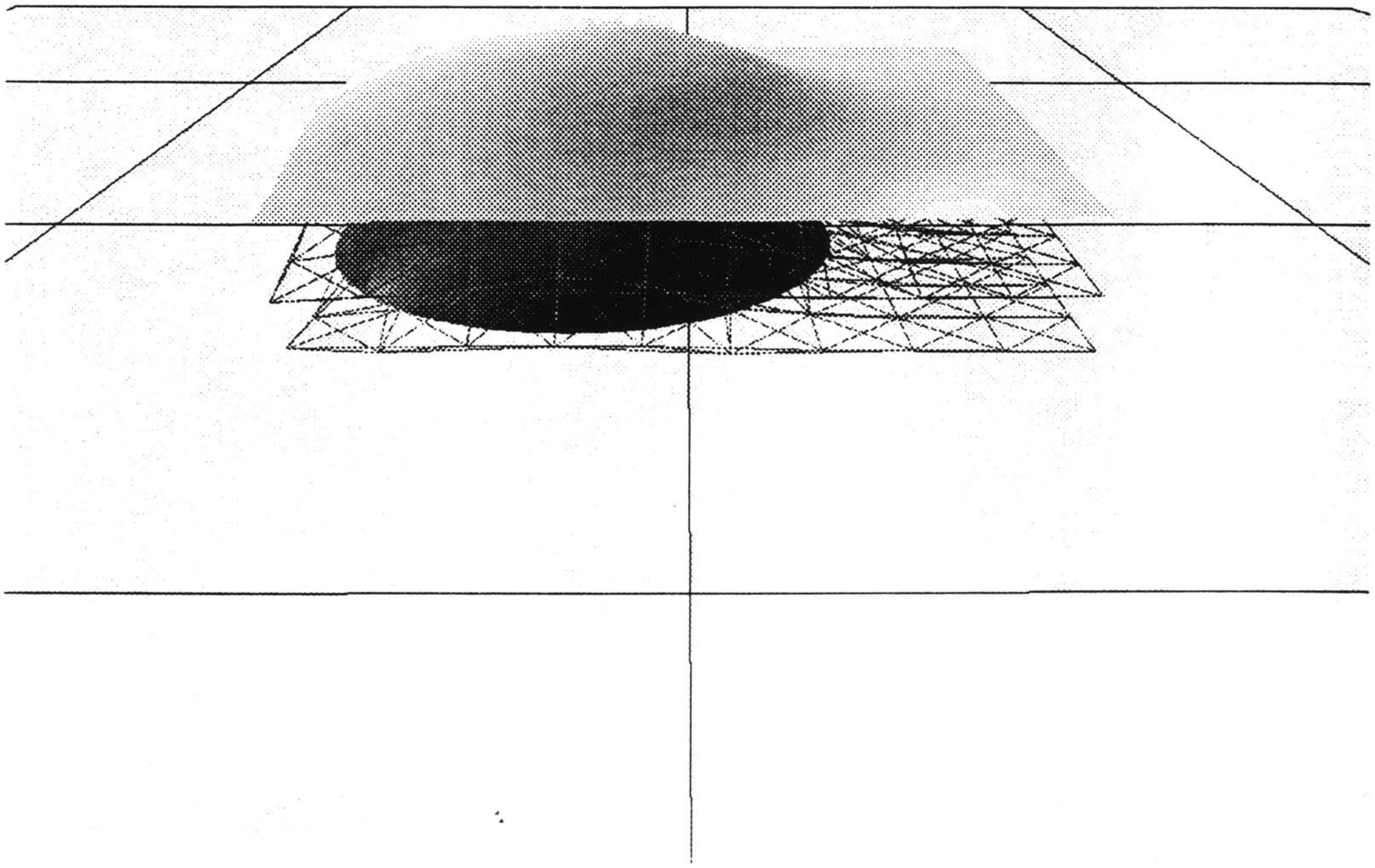

Figure 6-2: (c) Progressive states of hard tissue interaction simulation.

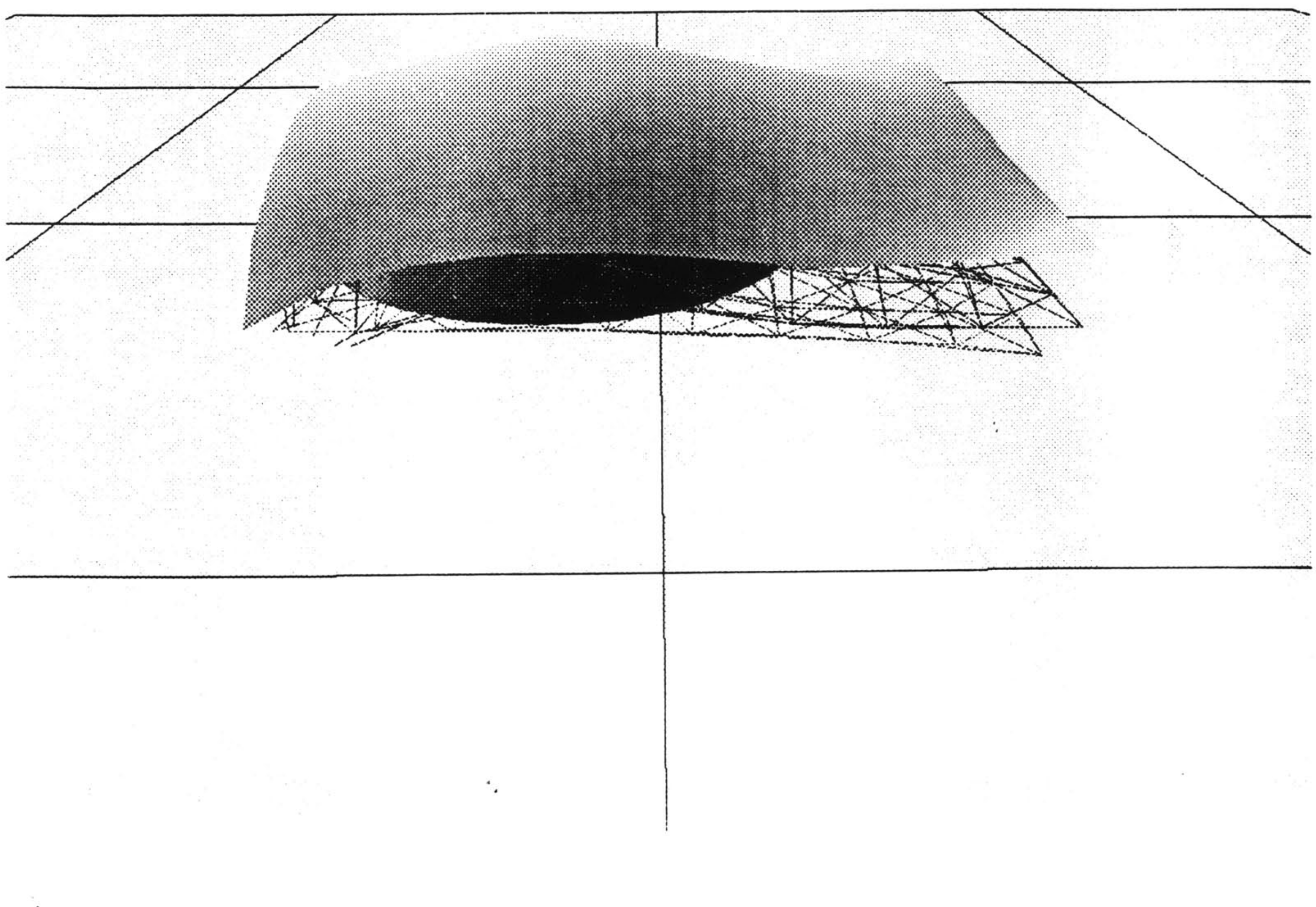

Figure 6-2: (d) Progressive states of hard tissue interaction simulation.

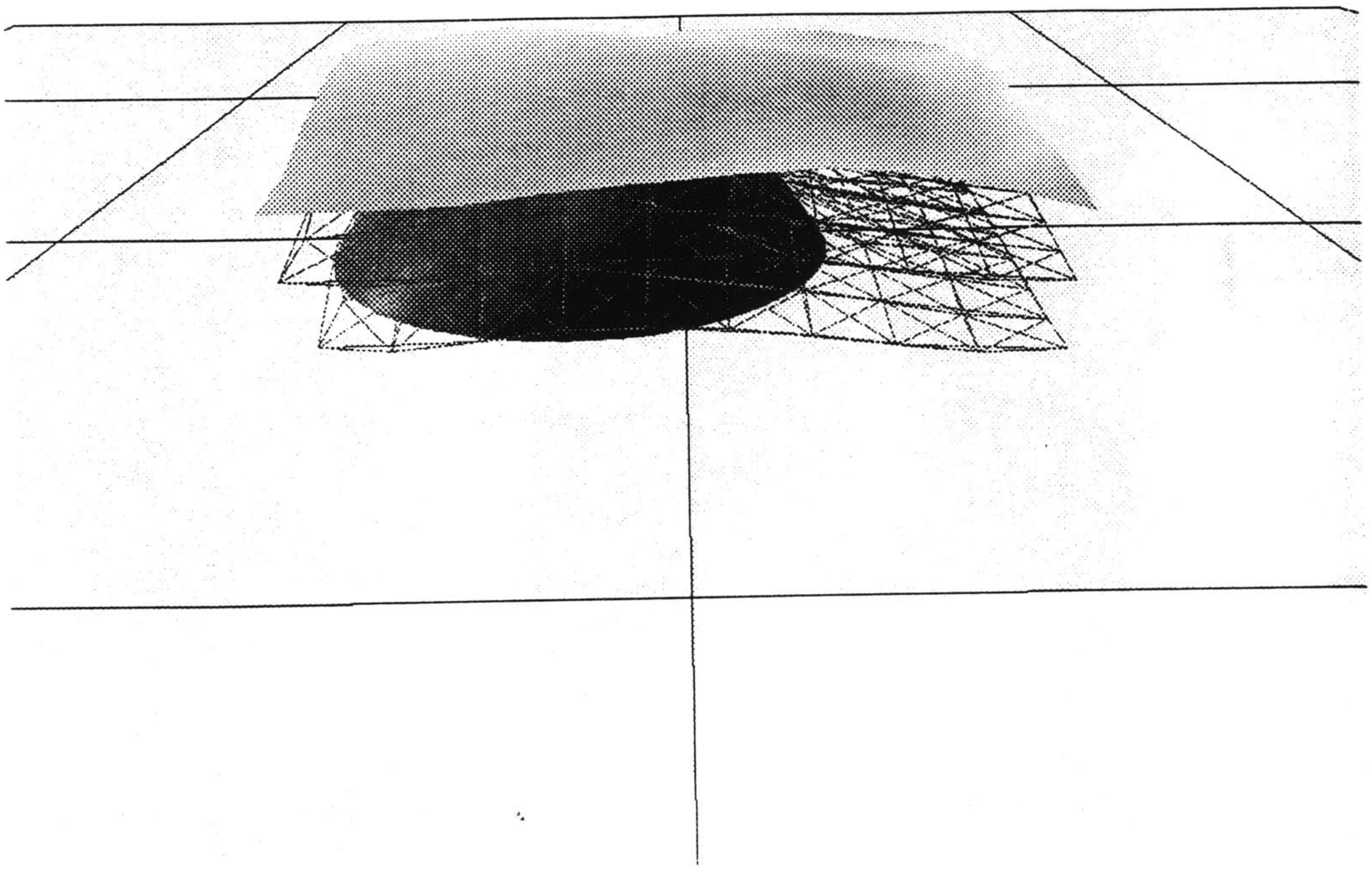

Figure 6-2: (e) Progressive states of hard tissue interaction simulation.

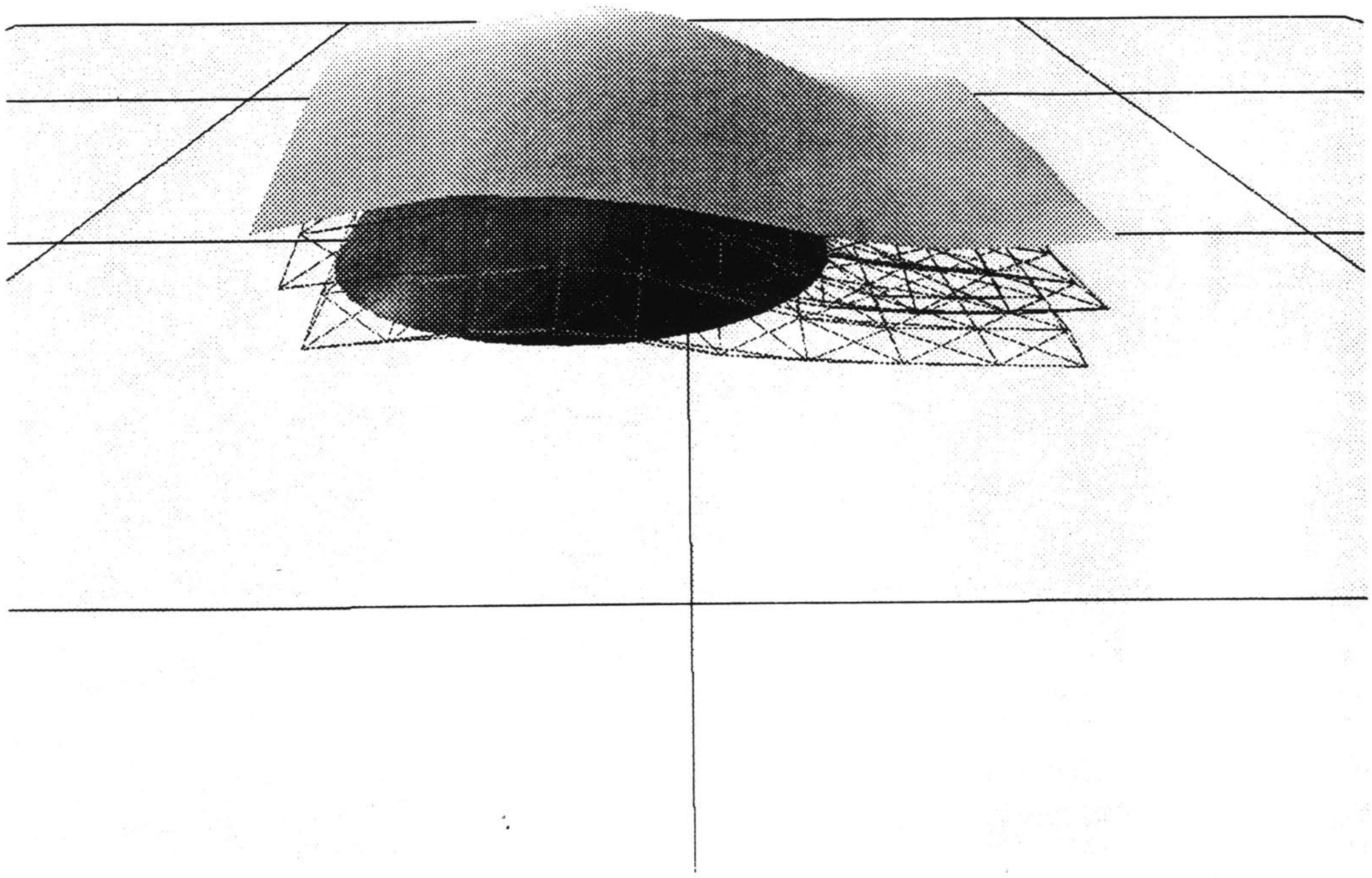

Figure 6-2: (f) Progressive states of hard tissue interaction simulation.

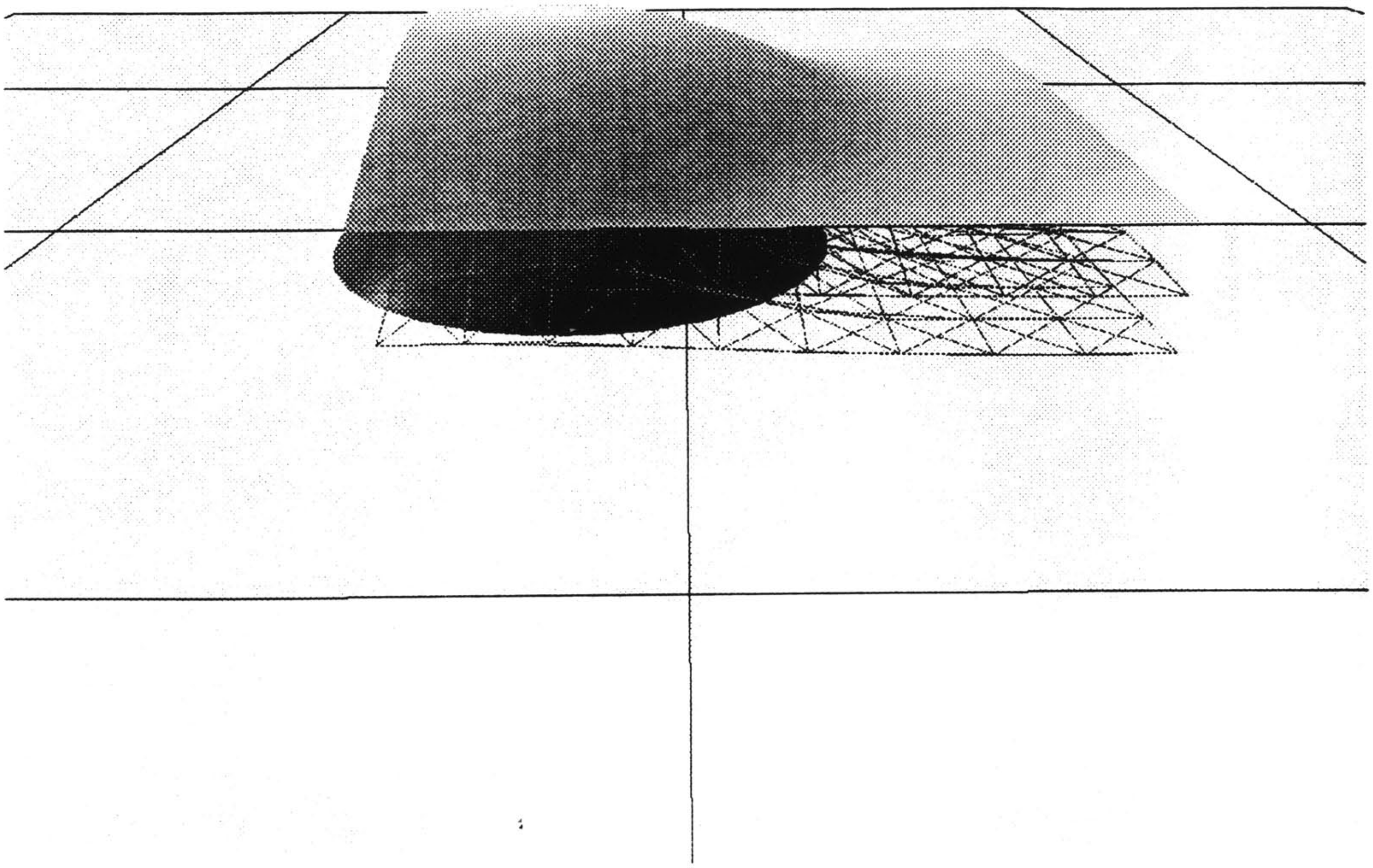

Figure 6-2: (g) Progressive states of hard tissue interaction simulation.

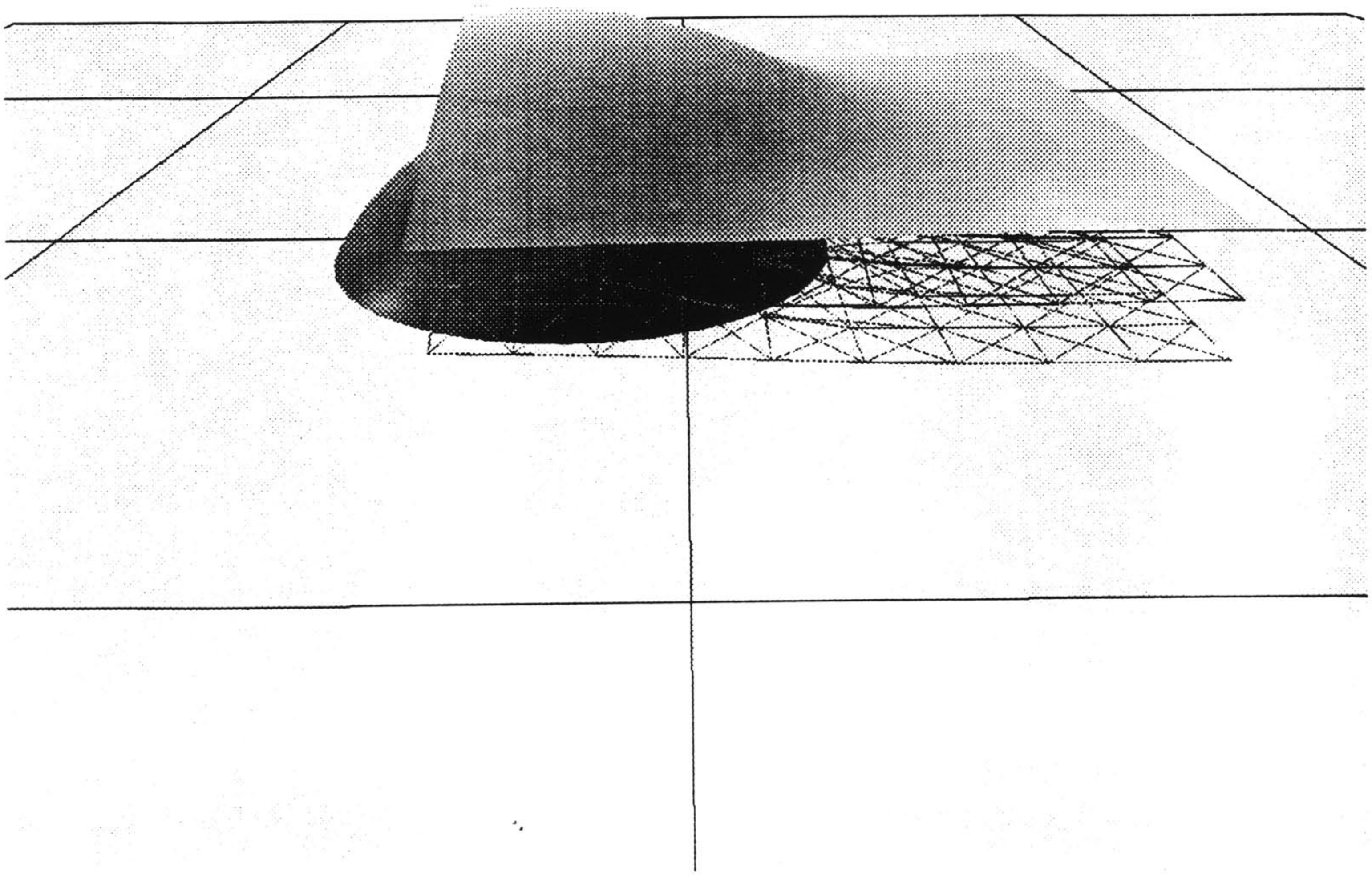

Figure 6-2: (h) Progressive states of hard tissue interaction simulation.

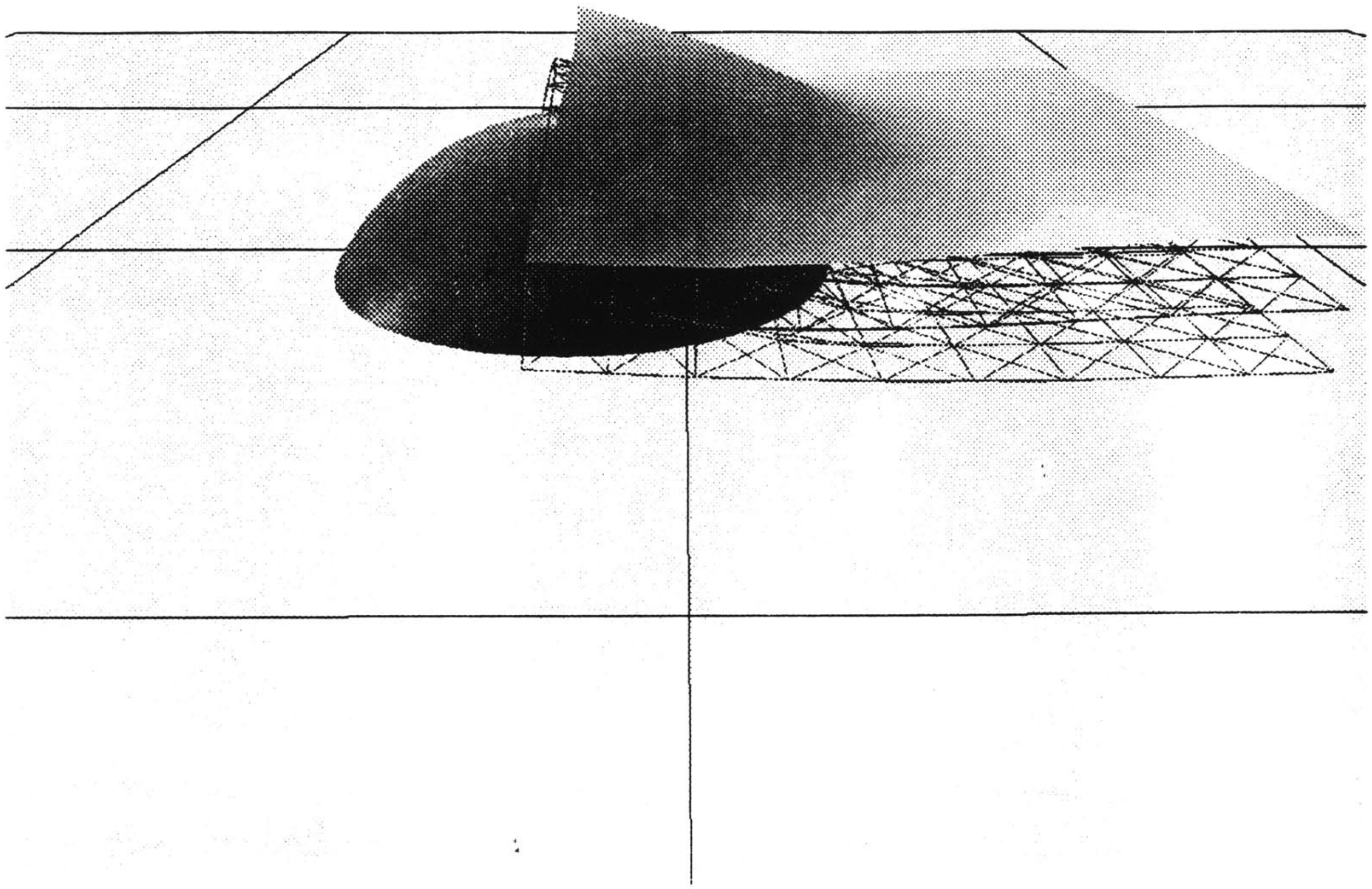

Figure 6-2: (i) Progressive states of hard tissue interaction simulation.

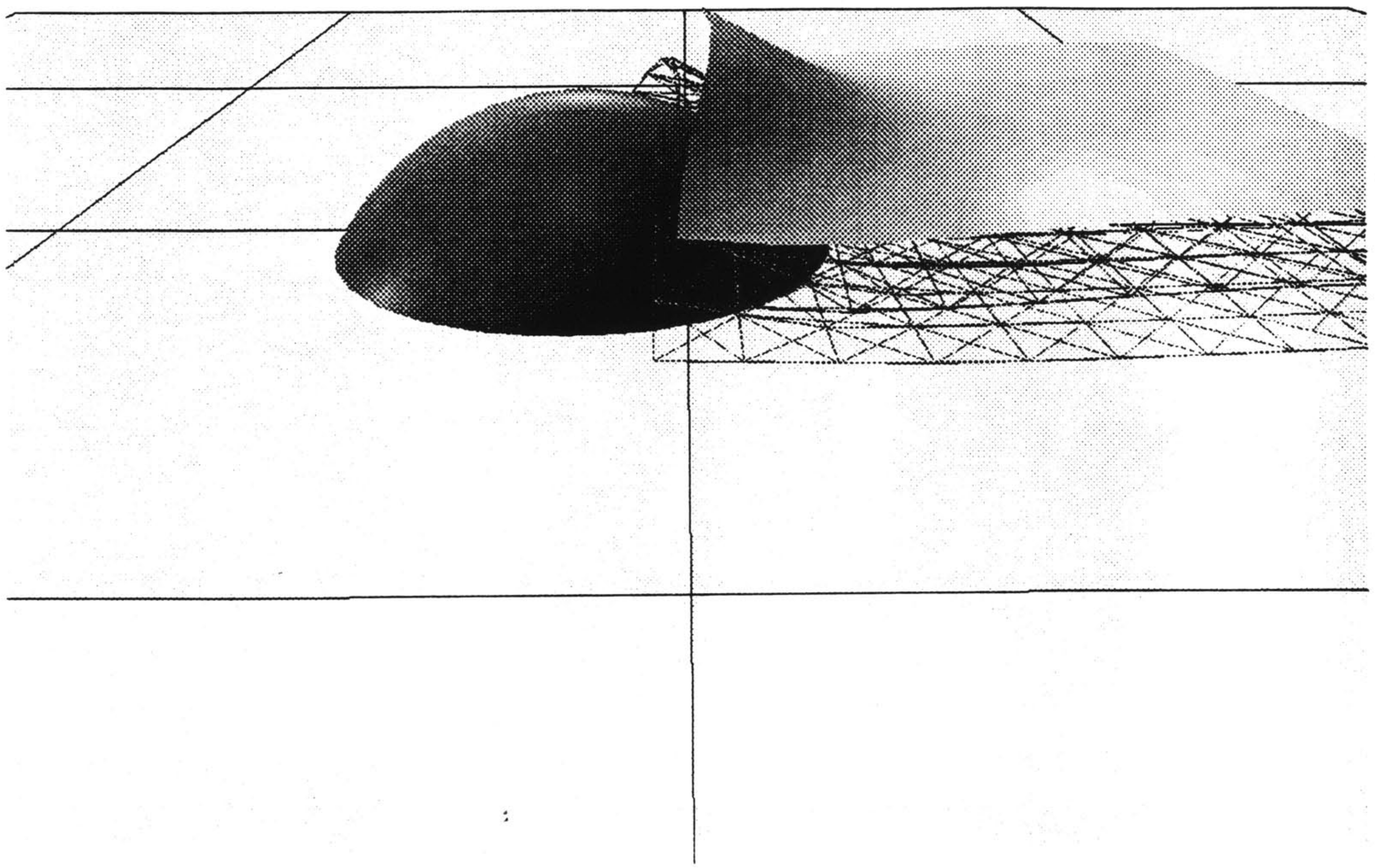

Figure 6-2: (j) Progressive states of hard tissue interaction simulation.

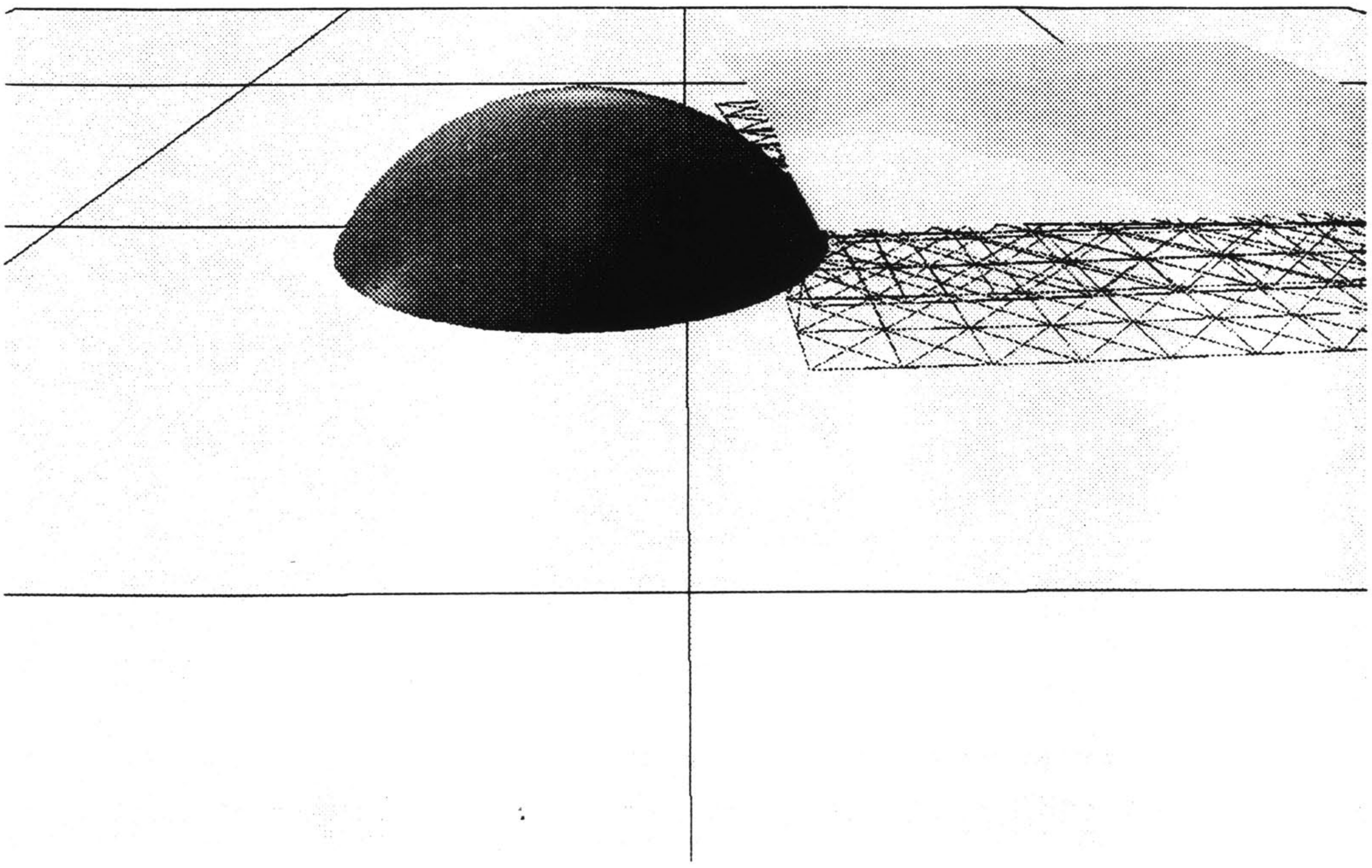

Figure 6-2: (k) Progressive states of hard tissue interaction simulation.

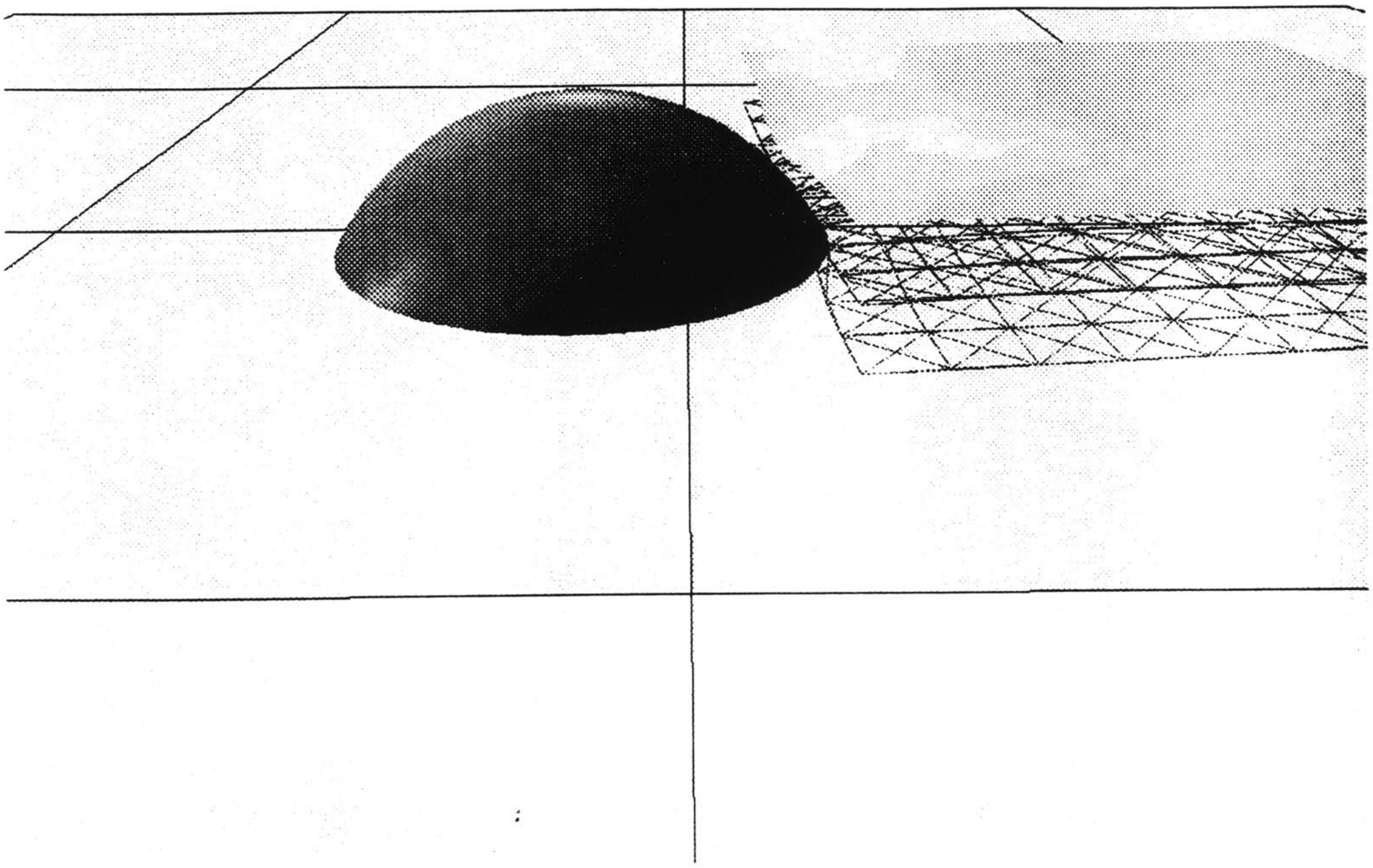

Figure 6-2: (1) Progressive states of hard tissue interaction simulation.

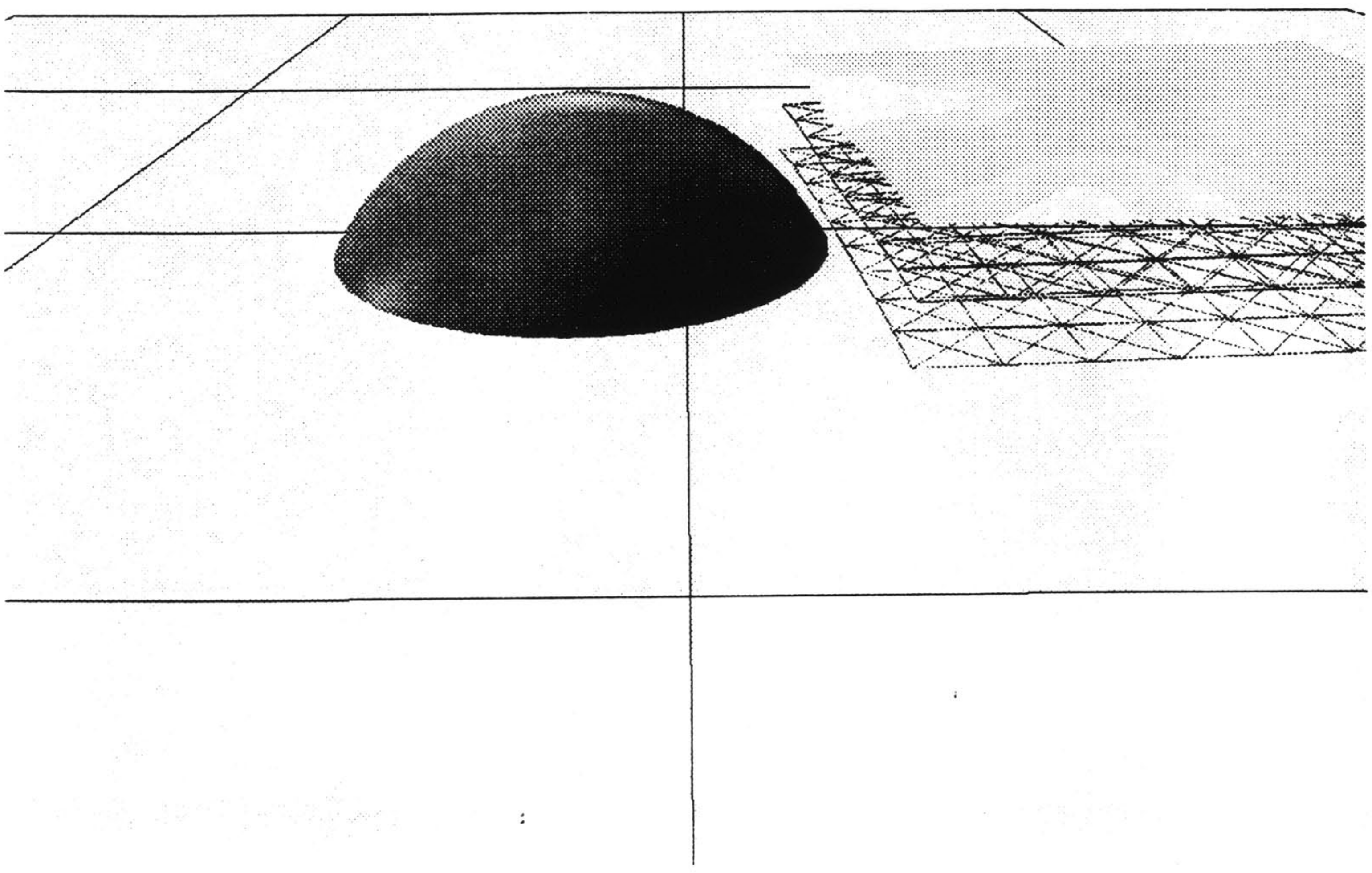

Figure 6-2: (m) Progressive states of hard tissue interaction simulation.

```
;;
;; animation test for skin dynamics, muscle acting in tissue
;;
; initialize graphics
#include lookat_twitch
; read base object
instance_object data/grid.10.10 sourcegrid
;instance_object data/grid.2.2 sourcegrid
; rotate object source object
edit_xform sourcegrid -r 0 10 0
; apply transformation matrix to source vertices
harden sourcegrid
; read the plane
instance_object data/grid.5.5 floor
instance_object data/wiregrid.5.5 gridfloor
edit_xform floor -t 0 0 -0.4501 -s 4 4 1
edit_xform gridfloor -t 0 0 -0.45 -s 4 4 1
; change colors for the floor
colors floor -faceted -diff .98 .98 .98 -high 0.8 0.8 0.8 -spec 1
colors gridfloor -faceted -wire .05 .05 .05
;
; build flesh from source.
flesh_out sourcegrid -sc 2 -dfth 0.1 -sfth 0.1 -me 1
; make skin surface renderable
triangulate skin.face.sourcegrid
; turn off source
visible sourcegrid
; add gravity to all the face blots
gravity_face -g 0.1
; drape over floor plane
drape -c 1000 -p floor
;
; put a bunch of fibers together in this muscle.
; first two numbers are indices into the point sample list indicating the
; points to connect, the second two numbers indicate which layers to
; connect.  The -c flag sets the material constant, the -m flag sets the
; material type to linear.
add_muscle muscle 13 83 1 1 -m 4 -c 20
add_muscle muscle 33 63 1 1 -m 4 -c 20
add_muscle muscle 14 84 1 1 -m 4 -c 20
add_muscle muscle 34 64 1 1 -m 4 -c 20
add_muscle muscle 15 85 1 1 -m 4 -c 20
add_muscle muscle 35 65 1 1 -m 4 -c 20
setup twitch muscle 0 1 10 0.3 30 0.3 40 1 -start 2
; set damping to quell oscillation
```

```
damping 0.3
simulate -dt 0.05 -skip 5 -runge_kutta -until 50
```

6.2 Limitations of the Current Model and Directions for Improvement

While section 3.1 looked at the long term goals for the development of a soft tissue simulator, this section looks at near term goals for the prototype system. The improvements can be installed in the near term to increase the range of possible simulations. The current implementation has several shortcomings which will need to be refined as the system is developed, but because the underlying system (bolio) has been written in a flexible manner, changes in the physical model can often be accomplished with minor code revisions. (Many such revisions have been performed in developing the current model.) The following sections look at some of the issues which should be addressed as the model is revised.

6.2.1 Deformability

The model of deformability, as currently implemented, includes only point to point force constraints (springs). This is a good technique, in general, since three dimensional structures can be built using springs as building blocks. However, some behavior of soft tissue is best described in terms of constraints involving more than two points. One such behavior is the volume-preserving deformation of fluids, as exhibited by the fluids of the ground substance. Another behavior which occurs at a higher level is the problem of collections of springs inverting. This occurs because the springs don't accurately model the fact that points of a solid element are not able to pass through each other. A solution to both of these problems is to define elements which are collections of points. The volume of the collection can be calculated, and after deformation, forces can be applied to points to move them in a direction which would tend to return the collection to its original volume. Inversion of a tetrahedron of points can be determined by looking at the position of a point with respect to the plane defined by the other three points — if the point moves on to the wrong side of that plane, a force can be applied to move it back.

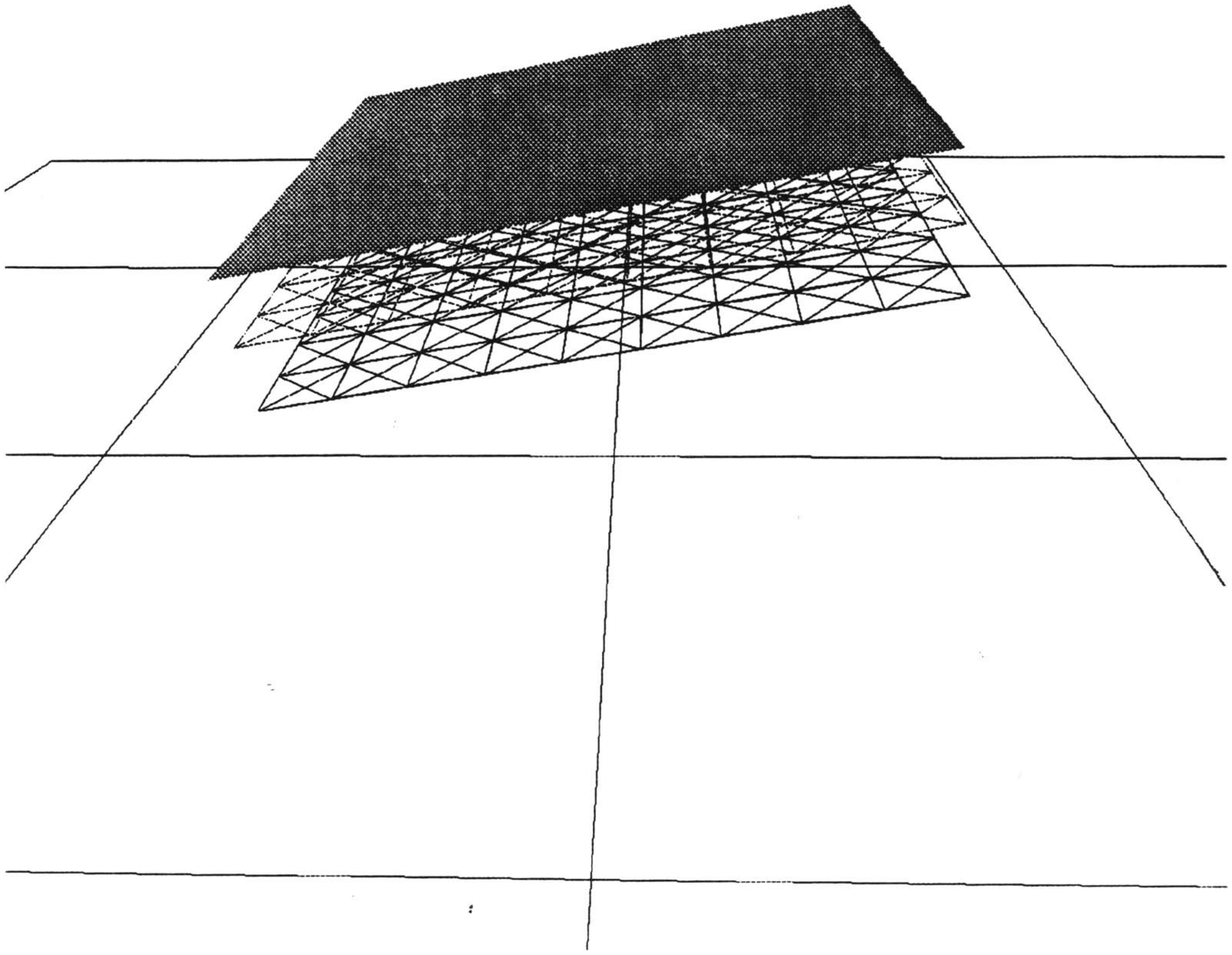

Figure 6-3: (a) Progressive states muscle action and hard tissue interaction simulation. The simulation model contains 300 sample points.

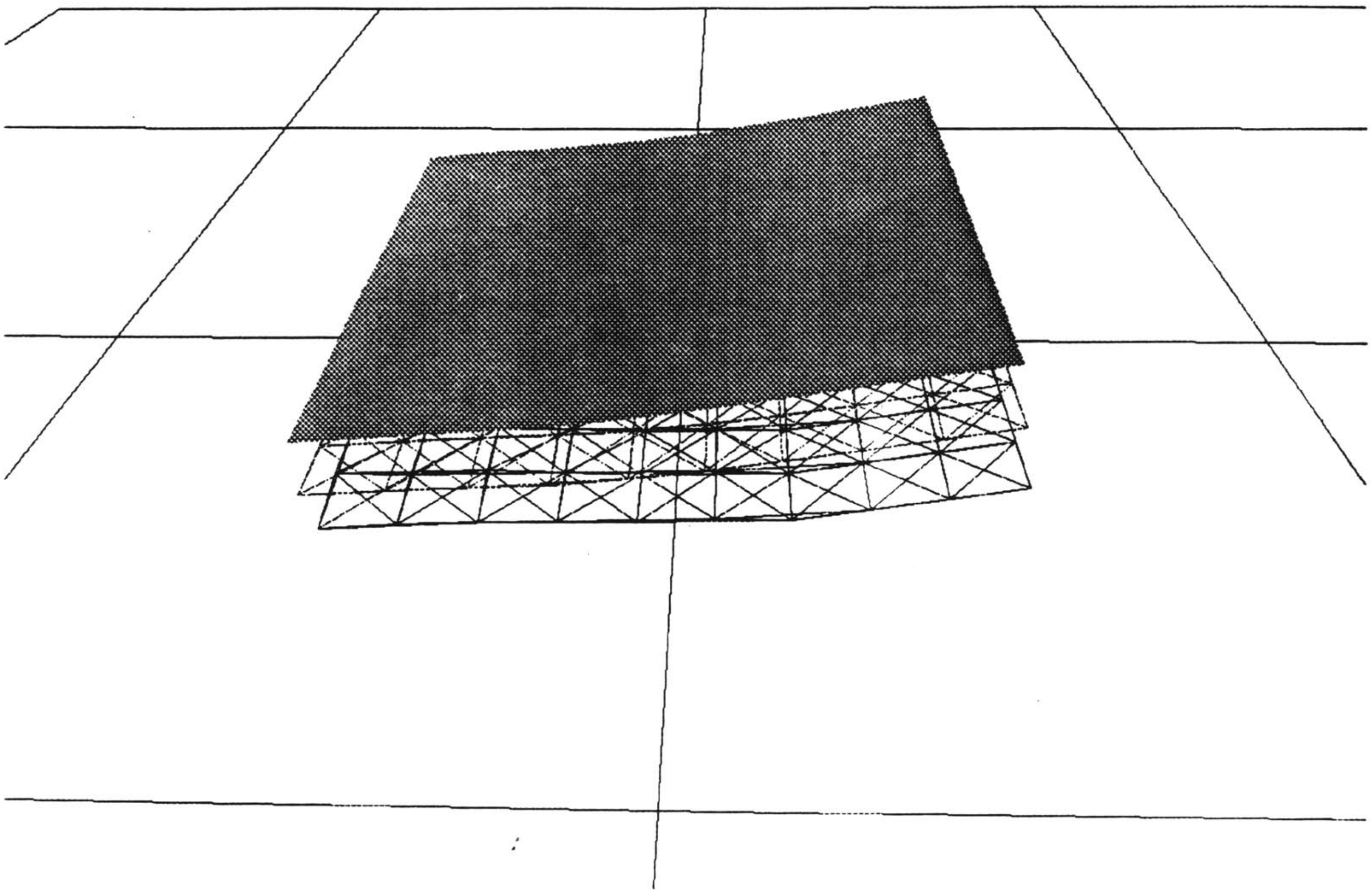

Figure 6-3: (b) Progressive states muscle action and hard tissue interaction simulation.

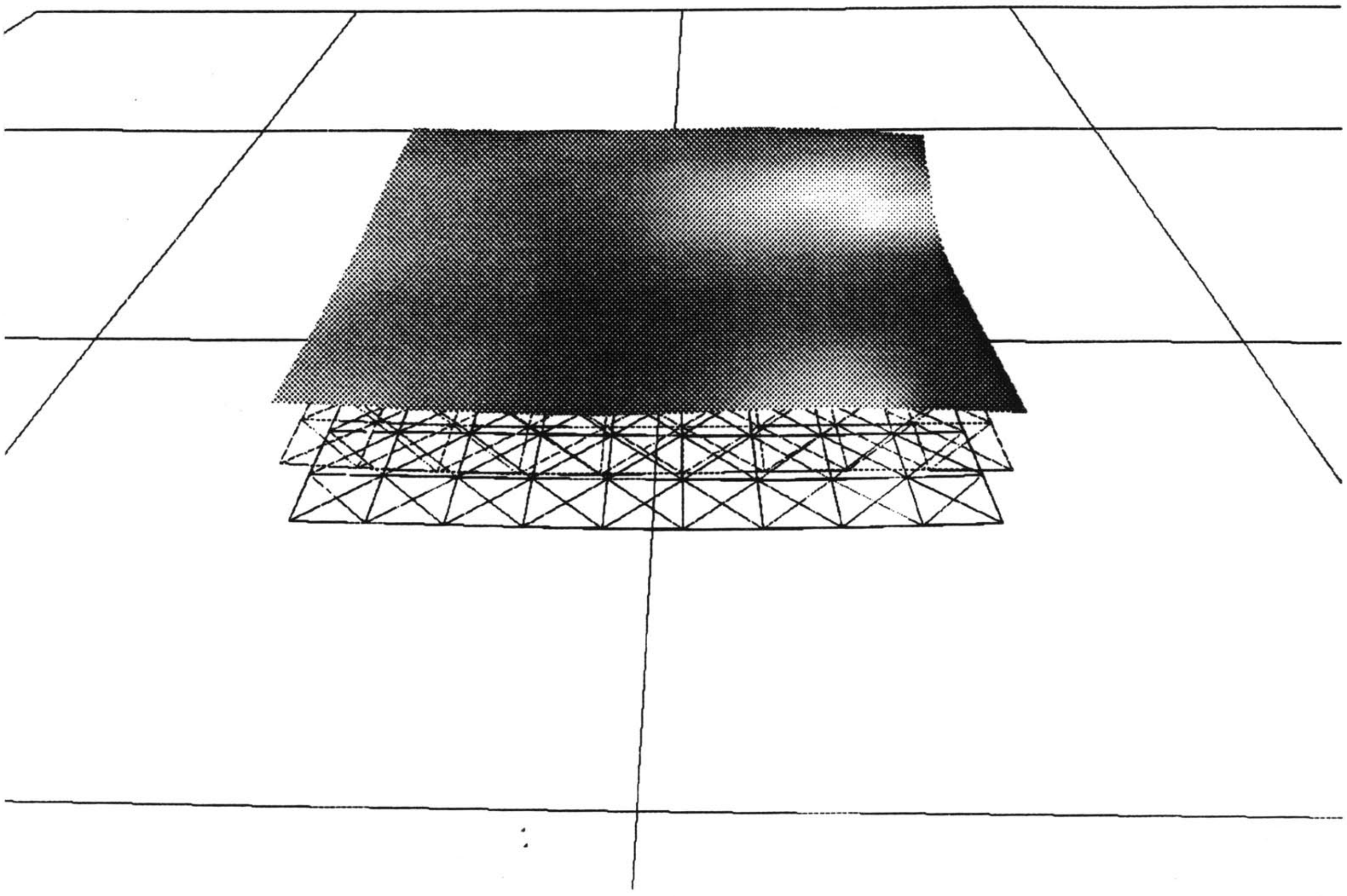

Figure 6-3: (c) Progressive states muscle action and hard tissue interaction simulation.

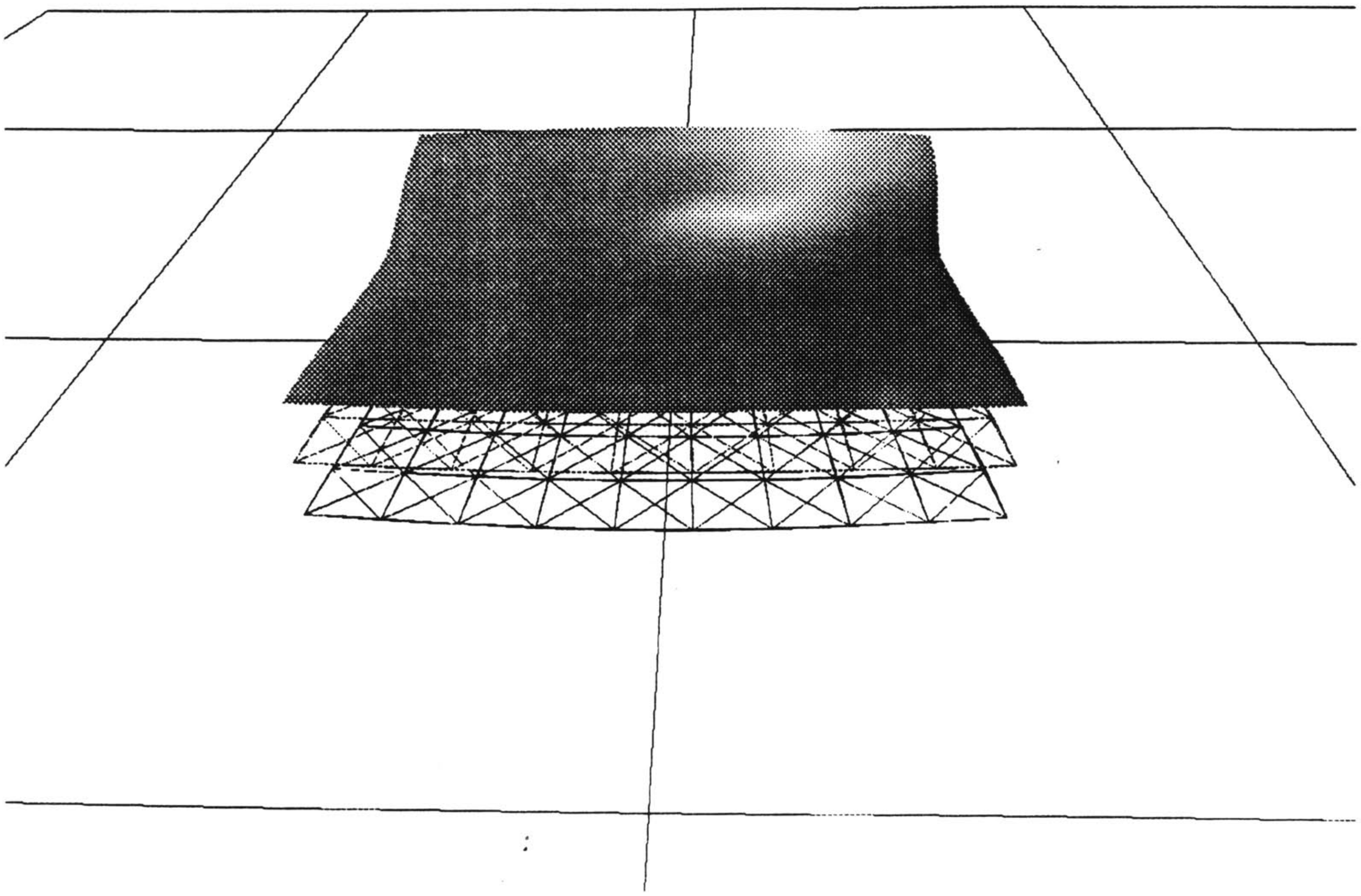

Figure 6-3: (d) Progressive states muscle action and hard tissue interaction simulation.

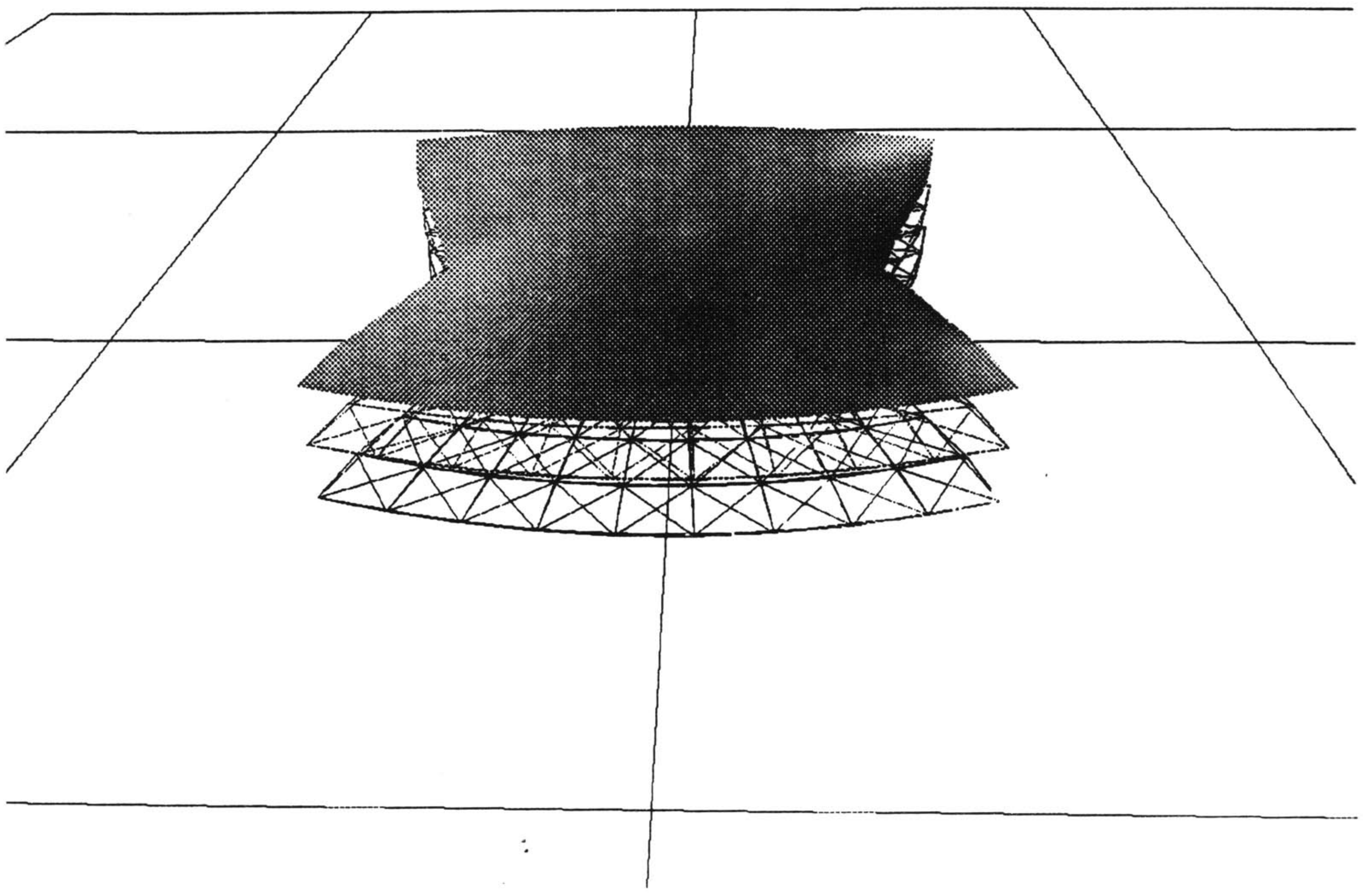

Figure 6-3: (e) Progressive states muscle action and hard tissue interaction simulation.

Another technique which could be applied to improve the efficiency and accuracy of the simulation in situations of large and irregular deformation would be to automatically adjust the resolution of the sampling depending on the amount of deformation; that is, to create new point samples and springs in areas where the model is experiencing large amounts of deformation and leaving the lower resolution in the areas of lower deformation. This would allow for high resolution sampling of regions of complex deformation (for example, where the skin is wrinkling) while maintaining a simpler representation in regions of less complex deformation. Of course it would also be desirable to adaptively reduce the resolution of the sampling if the deformation returns to a less complex state. The *multigrid* method of relaxation is similar to this approach, except that it changes the resolution of the sampling throughout the entire simulated material. It is used to evaluate a gross measure of the deformation at low resolution which serves as a first approximation from which to develop a more accurate deformation at high resolution. Feynman successfully applied the multigrid technique for simulating the deformation of cloth[15].

A related problem occurs in the time domain. When the state of forces acting on a point sample is changing rapidly with respect to time, the fixed time step of integration may not accurately sample that change, and thus, an incorrect force may be used. A solution to this problem is to take a measure of the change in the force during the time step, perhaps by comparing the forces calculated at the start and the end of the time step in the version of Runge-Kutta integration described in section 5.8. The numerical analysis literature contains this and other schemes for automatically adjusting integration time steps[49][44]. An interesting idea which seems not to have been addressed is the possibility of using small time steps in one part of the simulated material (perhaps at a collision point, where the forces are changing rapidly) while using a large time step for the samples in the rest of the material where the forces are not changing as rapidly. A simulation using multiple time steps could potentially be much faster than one which maintains a global time step, although the added work of keeping track of local time steps and the problem of integration at the boundaries between regions may make this approach infeasible.

6.2.2 Formability

The tools currently available to build simulation models operate at either too gross or too fine a detail to be practical for building accurate models. The `flesh_out` command, for example, has only a single thickness parameter for each of the fascia layers, which doesn't allow for thickness variations across the face. The `flesh_out` command and the `FACE` structures are currently hard-coded to support the three-layered model in use at this time. The command line method of building constraint networks is unworkable for defining large simulation models. Some form of editor should be designed to provide higher level tools for forming the simulation models. A *grow/shrink* operator might be built, for example, to change the rest lengths of springs within a selected region of material. Similar operators could change the material properties within regions or increase/decrease the resolution of the sampling. While this type of editor would be useful for generating various models to test the simulation, a better approach for generating complex models in the long term will be tools which automatically extract physical models from scanned patient data (as discussed in section 3.1.2).

6.2.3 Reformability

There are currently no high level tools for removing springs from the `FACE` structures. An appropriate tool would allow selection of a region of the tissue and update the `FACE` and `bOBJECT` data structures appropriately.

6.2.4 Controllability

The current use of cosine interpolation between key values to control muscle lengths is awkward for defining coordinated muscle actions. A facial muscle model based on FACS is being developed by Waite[60]; this model should be incorporated into the simulator.

6.2.5 Interactivity

Speed

The main interactivity improvement to be made to the prototype is to increase the execution speed. To some extent, this will be achieved by porting the code to faster machines — newer versions of graphics workstations. However, since this will only provide a speed improvement of a few times at best, a more promising approach is to look at distributing the simulation over multiple processors. This could be implemented by using a special purpose parallel computer, such as a Connection Machine, or by using several workstations connected via a local network. Because it is a discrete simulation model, the simulation is well suited to parallel computation no matter what the architecture, and either approach would require special purpose coding changes to support the parallelism.

Rendering

The use of triangulated mesh as the output primitive introduces some rendering problems since it is not symmetric. While the input `bPOLYHEDRON` for the **flesh_out** command need not be triangular, the output polygons in the **skin** `bPOLYHEDRON` mesh must be triangular in order to ensure that the resulting polygons are planar in spite of the movements of the vertices. A vertex in the mesh is in both of the two triangles of the grid section to the lower left and the upper right, but is in only one of the triangles of the upper left and lower right grid sections. When this point moves, the change in the rendered surface is not distributed evenly, having a much larger impact in the directions where it affects two triangles than in the directions where only one triangle is affected. This problem is localized to the rendering of the surface and does not represent an asymmetry in the underlying physical model. This problem would be eliminated if a higher order interpolation were used to render the surface rather than the linear interpolation of the triangulated mesh. A hermite spline surface patch, for instance, would eliminate this asymmetry.

Other rendering issues include specifying optical properties for the skin surface. Currently, a shiny, sickly green is chosen by default, and rendering is performed without texture mapping or shadows. The optical properties of the skin should change from point to point on the skin surface and should also be able to change over time to simulate effects like blushing or going pale. Obviously, a wide range of skin color choices should be available. Some effort should be made to find rendering techniques which better represent the fine detailed texture of skin texture, the appearance of facial hair, and the play of shadows on the face.

Chapter 7

Conclusion

This thesis has presented a mathematical model which can be used to simulate many of the aspects of the mechanical behavior of facial tissue which are relevant to plastic surgery. A set of criteria were established by which to judge the development of surgical simulators: deformability, formability, reformability, controllability, and interactivity. The model can simulate the action of layers of soft tissue as they interact with underlying hard tissue, internal muscle forces, and external forces such as gravity. Models can be built which approximate patient anatomy including features such as excised skin. thesis has also described directions for improvement of both the model and the system. This thesis has also presented an interactive simulation platform which has many features that will be critical in future surgical simulators.

The main conclusion to draw from the work presented in this thesis is that the technology for computer simulation of human facial tissue is not yet completely understood, but that the potential application of such technology would be very valuable. The problem is *difficult* because human facial tissue is complex in behavior (and thus it is hard to predict its response to applied forces) and complex in anatomy (and thus it is difficult to build models which represent the reality of a living subject). Computer modeling of soft tissue will be very ***valuable*** because understanding the nature of soft tissue is critical for the successful application of plastic and reconstructive surgery.

The second conclusion to draw is that in order to be useful in a clinical setting, the simulation must be embedded in an interactive system. The graphical output should allow the user to watch the changing shape of the material, since this helps develop an understanding the mechanical deformation process. The input devices should allow interaction with the simulation using the same gestures and motions as would be used when dealing with the real physical objects. Another important requirement of an interactive system is high-performance computation. This is particularly true in this case due to the complexity of simulating soft tissue. Historic trends indicate that the price of computer hardware will go down and that performance will increase. This suggests that while today's hardware cannot support a simulator of plastic surgery, future systems, possibly employing parallel computation, will be powerful enough and will be available at a low enough price for a breakthrough in clinical application.

Acknowledgments

"It takes a whole lot of help to make it on your own." *Steve Forbert*

Many thanks are due to many people, here are a few of each. My wife Carol, for agreeing to come on an educational adventure (it ain't over yet!), and then for making it the best time of my life. Mom and Dad, Dave and Jim, Grandma and the rest of my Family (Piepers and Mishlers) for love and support. To the playpen kids, for a great place to live and work, the folks in cga who've made the snakepit, and everyone in the building for making the Media Lab a great place to study. To David Zeltzer for guidance and technology. To Joe Rosen for inspiration, background, and vision. To Bubu for the bball idea. To Cliffie for the "b" word. To Hewlett-Packard Corporation for the great machines to work on. To NHK, Gould, Apple, and all the other far-sighted companies who've made Computer Graphics and Animation at the Media Lab possible.

The bolio system has been a group effort in every sense. Cliff Brett wrote the original version of bolio and much of the internal graphics code in the current version. David Sturman wrote the DataGlove interface and helped developed many demo scripts including the Roach 'n' Glove demo. Mike McKenna wrote the *roach* module. David Zeltzer wrote the skeleton animation system. Paul Dworkin helped design the functionality of bolio. Peter Schroeder wrote the *pathplan* module and the *moves* and *cam_moves* modules. Dave Chen's graphics code in rendermatic have been an invaluable example for much of bolio's internal structure. Dave also wrote the inverse-kinematics module. The contributors to bolio have shown great patience with my naming conventions and my programming style.

Bibliography

[1] Rohan Abeyaratne. MIT Professor of Mechanical Engineering, personal communication.

[2] Rohan Abeyaratne. Constitutive law for an elastic material. In *Course Notes for Applied Elastcity*, chapter 4. MIT, Fall 1987. unpublished.

[3] R. J. Atkin and N. Fox. *An introduction to the theory of Elasticity.* Longman Group Limited, London, 1980.

[4] Ronen Barzel and Alan H. Barr. A modeling system based on dynamic constraints. In *Computer Graphics*, volume 22, August 1988.

[5] A. Borning. Thinglab — a constraint oriented simulation laboratory. In *Technical Report No. SSL-79-3*, Xerox PARC, Palo Alto, California, July 1979.

[6] Cliff Brett, Steve Pieper, and David Zeltzer. Putting it all together: An integrated package for viewing and editing 3D microworlds. In *Proc. 4th Usenix Computer Graphics Workshop*, Cambridge, MA, October 1987.

[7] William J. Cromie. Computer-aided surgery. *MIT Report*, XVI(9), November 1988.

[8] Robert A. Debrin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *Computer Graphics*, volume 22, August 1988.

[9] Xiao Qi Deng. *A Finite Element Analysis of Surgery of the Human Facial Tissues.* PhD thesis, Columbia University, 1988.

[10] Rod Deyo, John A. Briggs, and Pete Doenges. Getting dynamics in gear: Graphics and dynamics in driving simulation. In *Computer Graphics*, volume 22, August 1988.

[11] Paul Ekman. Cross cultural studies of facial expression. In Paul Ekman, editor, *Darwin and Facial Expressions*, chapter 4. Academic Press, New York San Francisco London, 1973.

[12] Harry R. Elden. Biophysical analysis of aging skin. In Harry R. Elden, editor, *Biophysical Properties of Skin*, chapter 1. Wiley-Interscience, New York, 1977.

[13] Elliott H. Rose, M.D., Lars M. Vistnes, M.D., and George A. Ksander, M.S. A microarchitectural model of regional variations in hypodermal mobility in porcine and human skin. *Annals of Plastic Surgery*, 1(3), May 1978.

[14] Alejandro J. Ferdman. *Robotics Techniques for Conrolling Computer Animated Figures.* Thesis Submitted to the Department of Architecture, Massachusetts Institute of Technology, 1986.

[15] Carl Feynman. *Modeling the Appearance of Cloth.* Thesis Submitted to the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1986.

[16] S. S. Fisher, M. McGreevy, J. Humphries, and W. Robinett. Virtual environment display system. In *ACM 1986 Workshop on Interactive 3D Graphics*, October 1986.

[17] James D. Foley. Interfaces for advanced computing. *Scientific American*, pages 127–134, October 1987.

[18] G. Jost, M.D. and Y. Levet, M.D. Parotid fascia and face lifting: A critical evaluation of the SMAS concept. *Plastic and Reconstructive Surgery*, 74(1), July 1984.

[19] C. R. Gallistel. *The Organization of Action: A New Synthesis.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1980.

[20] M. J. V. Gordon, B. Strauch, and S. A. Blau. Abstract of paper in progress and personal communication, 1987.

[21] R. D. Harkness. Mechanical properties of skin in relation to its biological function and its chemical components. In Harry R. Elden, editor, *Biophysical Properties of Skin*, chapter 11. Wiley-Interscience, New York, 1977.

[22] David Haumann. Modeling the physical behavior of flexible objects. In *SIGGRAPH '87 Course Notes on Topics in Physically Based Modeling*, August 1987.

[23] Henry Gray, F.R.S. *Anatomy, Descriptive and Surgical.* Stein and Day, New York, 1977.

[24] Hewlett-Packard Company. HP-UX programmers manual, 1988.

[25] Hewlett-Packard Company. Starbase reference, 1988.

[26] J. M. Pensler, M.D., J. W. Ward, M.D., and S. W. Parry, M.D. The superficial musculoaponeurotic system in the upper lip: An anatomic study in cadavers. *Plastic and Reconstructive Surgery*, 75(4), April 1985.

[27] Jeffrey L. Marsh, M.D., Michael W. Vannier, M.D., W. Grant Stevens, M.D., James O. Warren, B.S.A.E, Donald Gayou, Ph.D., and Daniel M. Dye, Ph.D. Computerized imaging for soft tissue and osseous reconstruction in the head and neck. *Clinics in Plastic Surgery*, 12(2), April 1985.

[28] R. M. Kenedi, T. Gibson, J. H. Evans, and J. C. Barbenel. Tissue mechanics. *Physics in Medicine and Biology*, 20(5):699–717, February 1975.

[29] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, May 1988.

[30] N. Magnenat-Thalmann, E Primeau, and D. Thalmann. Abstract muscle action procedures for human face animation. *Visual Computer*, May 1988.

[31] Mario Gonzalez-Ulloa, M.D., F.A.C.S., F.I.C.S., F.R.C.M. and Eduardo Stevens Flores, M.D. Senility of the face—basic study to understand its causes and effects. *Plastic and Reconstructive Surgery*, 36(2):239–246, 1965.

[32] Mark L. Lemmon, M.D. Superficial fascia rhytidectomy. *Clinics in Plastic Surgery*, 10(3), July 1983.

[33] Michael McKenna and David Zeltzer. Gait control and dynamic simulation for legged locomotion. MIT Media Laboratory Computer Graphics and Animation Group Work in Progress, National Science Foundation Grant Number IRI-8712772: Modeling Motor Behavior and Virtual Environments for Three Dimensional Computer Animation.

[34] Michael McKenna and David L. Zeltzer. Dynamic simulation of autonomous legged locomotion. 1989. Submitted for publication.

[35] Thomas A. McMahon. *Muscles, Reflexes, and Locomotion.* Princeton University Press, Princeton, New Jersey, 1984.

[36] Michael W. Vannier, M.D., Jeffrey L. Marsh, M.D., and James O. Warren, M.D. Three dimensional computer graphics for craniofacial surgical planning and evaluation. In *Computer Graphics*, volume 17, July 1983.

[37] Gavin S. P. Miller. Motion dynamics of snakes and worms. In *Computer Graphics*, volume 22, August 1988.

[38] Debi P. Mukherjee and Allan S Hoffman. Physical and mechanical properties of elastin. In Harry R. Elden, editor, *Biophysical Properties of Skin*, chapter 6. Wiley-Interscience, New York, 1977.

[39] Fredric I. Parke. Computer generated animation of faces. *Proceedings of the ACM Annual Conference*, 1, 1972.

[40] Fredric I. Parke. Parameterized models for facial animation. *Computer Graphics and Applications*, November 1982.

[41] Richard P. Paul. *Robot Manipulators: Mathematics, Programming, and Control.* MIT Press, Cambridge, Massachusetts, 1981.

[42] Stephen M. Platt. *A Structural Model of the Human Face.* PhD thesis, University of Pennsylvania, 1985.

[43] Stephen M. Platt and Norman I. Badler. Animating facial expressions. In *Computer Graphics*, volume 15, August 1981.

[44] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipice in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, 1988.

[45] Eric Ribble. *Synthesis of Human Skeletal Motion and the Design of a Special-Purpose Processor for Real-Time Animation of Human and Animal Figure Motion.* Thesis Presented to the Department of Electrical Engineering, Ohio State University, 1982.

[46] Mark Rosenzweig and Arnold Leiman. *Physiological Psychology.* D. C. Heath and Company, 1982.

[47] Peter Schroeder and David Zeltzer. Pathplanning inside BOLIO. In *SIGGRAPH '88 Course Notes on Synthetic Actors*, August 1988.

[48] Alvy Ray Smith. *The Viewing Transformation.* Computer Graphics Project, Computer Division, Lucasfilm Ltd., Marin, Califoria, May, 1984.

[49] Gilbert Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, 1986.

[50] David J. Sturman, David L. Zeltzer, and Steve Pieper. Hands-on interaction with virtual environments. 1989. Submitted for publication.

[51] Ivan E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the AFIPS Spring Joint Computer Conference.*, volume 23, pages 329–346, Spring 1963.

[52] Symbolics, Inc. S-geometry manual, 1988.

[53] Dimetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics*, volume 22, August 1988.

[54] Dimetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Computer Graphics*, volume 21, July 1987.

[55] David Thompson, William Buford, Loyd Myers, David Giurintano, and John Brewer. A hand biomechanics workstation. In *Computer Graphics*, volume 22, August 1988.

[56] V. Wright, M.D., F.R.C.P. Elasticity and deformation of skin. In Harry R. Elden, editor, *Biophysical Properties of Skin*, chapter 12. Wiley-Interscience, New York, 1977.

[57] Arthur J. Vander, James H. Sherman, and Dorothy S. Luciano. *Human Physiology: The Mechanisms of Body Function.* McGraw Hill, Inc., New York, 1975.

[58] Jouko Viljanto. Tensile strength of healing wounds. In Harry R. Elden, editor, *Biophysical Properties of Skin*, chapter 13. Wiley-Interscience, New York, 1977.

[59] Vladimir Mitz M.D. and Martine Peyronie M.D. The superficial musculo-aponeurotic system (SMAS) in the partoid and cheek area. *Plastic and Reconstructive Surgery*, 58(1):80–88, July 1976.

[60] Clea Waite. *The Facial Action Control Editor, face: A Parametric Facial Expression Editor for Computer Generated Animation.* Thesis Submitted to the Media Arts and Sciences Section, Massachusetts Institute of Technology, 1988.

[61] Keith Waters. A muscle model for animating three-dimensional facial expression. In *Computer Graphics*, volume 21, July 1987.

[62] Wayne F. Larrabee, Jr., M.D. A finite element model of skin deformation. *Laryngoscope*, pages 399–419, April 1986.

[63] Jerry Weil, 1987. unpublished manuscript.

[64] William C. Grabb, M.D. and James W. Smith, M.D. *Plastic Surgery.* Little, Brown and Company, Boston, 1979.

[65] Andrew Witkin, Kurt Fleischer, and Alan Barr. Energy constraints on parameterized models. In *Computer Graphics*, volume 21, July 1987.

[66] F. E. Zajec, E. L. Topp, and P. J. Stevenson. A dimensionless musculotendon model. *Proceedings of the 8th Annual Conference of the IEEE Engineering in Medicine and Biology Society*, 1986.

[67] David L. Zeltzer. *Representation and Control of Three Dimensional Computer Animated Figures.* PhD thesis, Ohio State University, August 1984.

[68] David L. Zeltzer. Motor control techniques for figure animation. *Computer Graphics and Applications*, 2(9), November 1982.

[69] David L. Zeltzer, Steven D. Pieper, and David J. Sturman. An integrated graphical simulation platform. *Proceedings of Graphics Interface 89*, 1989. Submitted for publication.

[70] T. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A hand gesture interface device. In *Proceedings of Human Factors in Computing Systems and Graphics Interface*, pages 189–192, 1987.

Appendix A

Roach 'n' Glove Microworld Commands

In the lists below, the mnemonic alias for the command is given, followed by the command name as it appears on the menu. Most of the commands actually invoke scripts to perform the specified actions. Each of the commands is preceded by a back-quote (') when typing the keyboard alias.

A.1 Physical Simulation Commands

- p+: physics on. This turns on the simulation clock. When the simulation clock is turned on, the system calculates the effect of force generating constraints (springs, gravity, collision detection) and integrates the forces and velocities of the objects to calculate new positions in each frame.
- p-: physics off. This turns off the simulation clock.
- op: objects pyramid. This command connects four objects together with springs in a tetrahedral shape. Any of the objects in the resulting structure may be moved (e.g., picked up by the glove) and the effect will be transferred to the other objects via the springs.
- ot: objects triangle. Three objects are connected in a triangle with springs.

- ob: objects bola. Four objects are connected with springs to form a bola shape. One object in the middle connects to the three other objects, each of which only connects to the middle object.
- of: objects free. All spring constraints are deleted.
- f-: floor off. The collision detection constraint which prevents the objects from passing through the floor grid and the walls is turned off. The floor is on by default.
- f+: floor on. The collision detection constraint for the floor and walls is turned back on.

A.2 Roach Commands

- r+: roachwalk on. This command activates the `roachwalk` constraint. This constraint is dependent on the start of each frame, and causes the roach to update the position of its `bOBJECTS` to reflect the state of the behavioral simulation.
- r-: roachwalk off. This command suspends execution of the `roachwalk` constraint.
- rc: roach center. This commands the roach module to move the roach back to the center of the floor grid. The roach's `bOBJECTS` are updated accordingly.
- rrw: roach random walk. This command puts the roach into random walking mode. In this mode, the roach picks a random point on the floor grid and walks to it. When it reaches the point, it picks a new one randomly.
- rv+: roach view on. This command attaches the `bCAMERA` to the roach's position and orientation, providing a view as if you were "riding the cockroach". The DataGlove is also re-oriented to move relative to the roach.
- rv-: roach view off. This command makes the camera stop following the roach.

A.3 Glove Commands

- gc: glove calibrate. This command records a set of key postures to set the calibration tables in a **struct glovepoll_data**. The calibration records the flex values for the user's joints for a series of poses. First, with all joints fully extended; second, with the thumb closed as far as possible and the other fingers fully relaxed; and third, with the thumb relaxed and the other fingers closed. Twenty samples of the flex values are averaged for each posture in order to determine the range of motion for each of the joints.

- gg: glove grab. This command sets the **struct glovepoll_data** posture table entry for the grab command. To record the posture, the glove module performs 30 consecutive reads of the DataGlove, and stores the minimum and maximum values recorded for each joint. Subsequently, whenever the **glovepoll** constraint detects that all of the joint positions are within the range of those recorded for the grab posture, the number for that posture is set in the **struct glovepoll_data**. Both the **link_near** and the **grabber** constraints are dependent on the grab posture.

- gm: glove menu. This command sets the **struct glovepoll_data** posture table entry for the glove menu command. When the menu posture is entered, the menu appropriate to the current world is displayed, and the joints of the index and middle fingers control the cursor's movement up and down the menu. When the desired menu item is highlighted, it can be selected by a twist of the hand. This is implemented via the **glove_cursor** constraint, which is dependent on the menu posture.

- gf: glove follow. This command sets the **struct glovepoll_data** posture table entry for the glove follow command. When the follow posture is entered, a line is drawn from the current position of the roach to the current projection of the glove position on the ground plane. The roach is told to walk to the projection of glove position. As long as the glove remains in this posture, the path will be continuously updated and the roach will follow the changing position of the glove.

- gp: glove path follow. This command acts the same as the glove follow command except that rather than following a straight line from its current position to the glove shadow, the roach follows a path calculated by the *pathplan* module. The shortest, collision-free path from the roach to the glove shadow is continuously calculated while the glove is in the path follow posture.
- gv: glove view. This command sets the `struct glovepoll_data` posture table entry for the glove view command. When the view posture is entered, the view camera is constrained so that the eye point tracks the position of the glove and is always looking at the center of the roach's body.

A.4 View Commands

- vr: view reset. This command resets the view parameters to their default values.
- ve: view exaggerated. This command loads a set of view parameters which correspond to a very wide angle lens. The exaggerated perspective provides a somewhat more convincing sense of depth which can help the user correlate their hand movements with the movements of the glove in the simulated world.
- 3+: red/green 3D display on. This command puts the screen into red/green stereo mode for viewing with special glasses which have one red lens and one green lens. A separate image is rendered for each eye; one in shades of red for the eye with the red lens and one in shades of green for the eye with the green lens. The two images of the scene are blended on the screen.
- 3-: red/green 3D display off. This command turns the red/green stereo mode off.

A.5 Robot Arm Commands

- ra+: robot arm on. This command turns on a four joint inverse-kinematic robot arm. The base object of the robot arm is fixed in space above the surface of the floor grid, and the end effector is pulled down to the floor by gravity. While either the base or the end effector can be manipulated by the glove, the kinematic relationships of the arm are maintained.

- ra-: robot arm off. This command turns the robot arm off.

A.6 Pathplanning Commands

- vm: vgraph make. This command creates a visibility graph for the current positions of the objects in the world. A `bPOLYLINE` `bOBJECT`, which represents the vgraph, is displayed to show the possible paths for the roach in the current world.
- v+: vgraph visible. This command turns the vgraph `bOBJECT` visible.
- v-: vgraph visible. This command turns the vgraph `bOBJECT` invisible.