DEFORMABLE MODELS USING DISPLACEMENT CONSTRAINTS

By

Larry Palazzi

B. Sc. (Computer Science) University of Windsor

B. Comm. University of Windsor

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

COMPUTER SCIENCE

We accept this thesis as conforming

to the required standard

.....

.....

THE UNIVERSITY OF BRITISH COLUMBIA

October 22, 1993

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Computer Science
The University of British Columbia
Department of Computer Science
2366 Main Mall
Vancouver, BC, Canada
V6T 1Z4

Date:

Abstract

This thesis explores a variation of a technique, called "displacement constraints" [19], used for creating and animating deformable objects. It is a constraint-based technique that uncouples the constraint forces from the external forces acting on an object, and solves a system of geometric constraints at each timestep using an iterative technique. Displacement constraints does not attempt to model real physics, but is a simple, efficient technique that offers interactive speeds for the modeling and animation of deformable objects.

The inability to respond globally to an external force is an inherent property of many discrete/nodal formulations. A global response is sometimes only obtained through the propagation of localized forces over multiple timesteps. Energy minimization techniques may also achieve certain global effects, but at a higher computational cost. We introduce a multilevel approach to force distribution to improve the global response of a deformable object. External forces acting on an object are propagated through the different levels of a hierarchical representation of the object. Animators can set the distribution of forces at each level, thus controlling an object's local and global behaviour.

Table of Contents

List of Figures

Acknowledgements

First and foremost, I am indebted to my supervisor, Dr. David Forsey, for the considerable advice, guidance and support he has provided me throughout this entire thesis. I would like to thank the second reader of this thesis, Dr. Uri Ascher, for his time and effort, and for his insightful comments. I would like to thank Gene Lee for being the student reader of this thesis. I would also like to thank Dr. Sebastian Reich for our many discussions on numerical analysis and Dr. George Baciu for his comments on the displacement constraints paper.

I would like to thank my family and friends back in Ontario for their continued love and encouragement, and for keeping in close touch during the past two years. I would also like to thank all of the close friends I have made since my move to Vancouver, for their help and support, and, of course, for all the good times.

Last but not least, I would like to thank my beautiful wife, Tammy, for her love, support and understanding during our time apart and during the "crunch" times of completing this thesis. Moving across the country, away from your family and friends to a new place where you know almost no one, is a very difficult experience, especially when your spouse is spending most of his/her life at school! For these reasons, I dedicate this thesis to her.

Chapter 1

Introduction

de-form
1: to spoil the form of
2a: to spoil the looks of: DISFIGURE <a face deformed by bitterness>
2b: to make hideous or monstrous
3: to alter the shape of by stress

- *Webster*

1.1 Historical Perspective

The quest for realism has been the major driving force in computer graphics research for many years. The desire for realistic computer images and natural-looking computer animations has stimulated great interest in computer graphics and has produced some very impressive results. However, the real world is a complex beast; we often take for granted its intricacies. When one considers the factors that are involved in even a simple action such as dropping a handkerchief, one soon realizes that "there are more things in heaven and earth...than are dreamt of in our philosophy."[1]. Needless to say, to simulate even a small subset of the real world on a computer is an arduous task. There are significantly many elements and parameters to consider when formulating a physical simulation. The result is a complex mathematical system that must be numerically integrated through time. Achieving real time interaction is difficult, if not impossible, with existing technologies. It is the speed of the system that is often crucial in the development of an animation sequence.

Despite these facts, dynamic modeling has become a popular topic of research in computer graphics. The main advantage of dynamic modeling is that animations can be created with

[1] Hamlet, Act I, Scene v

little or no user interaction. The animator sets up the initial configuration of the system and the dynamic equations govern the motions and behaviours of each object over time. The complex behaviours created by these methods are very difficult and time consuming to duplicate using other methods such as script-based or key-frame animation.

However, dynamic modeling does not hold all the answers. In particular, due to the complexity of the computation involved, implementations of such systems are very difficult and time consuming. Many methods are prone to numerical instability, making accurate solutions very expensive to attain. As a result, interactive modeling and animation is not usually possible. Although animations can be created with little or no user interaction, achieving a particular desired behaviour (i.e. the inverse dynamics problem) can be very difficult. Defining the equations and parameters to model a certain behaviour is nonintuitive and often unpredictable. These are the main reasons why these techniques have not been used extensively in commercial animation software. Animators want fast, easy to control methods for creating computer animations. If this is the case, then we can simplify our model in order to avoid complex calculations and provide more time efficient algorithms.

We will explore displacement constraints, a technique that achieves "realistic" behaviours without some of the overhead of complex dynamic systems. This technique, introduced by Gascuel and Gascuel [19], is useful when some semblance of realism is desired, rather than accurate physical behaviours. As an extension of their work, we will add the capability of controlling the local/global behaviour of an object using a multilevel approach. Very few deformable models support this feature. Usually, only a completely global behaviour or completely local behaviour (with global behaviour achieved through the propagation of local interactions) is possible. We will present a solution where local and global control are incorporated into a general unified model. The goal of our work is to develop a system that offers animators speed and control in creating complex computer animations.

1.2 Overview of Displacement Constraints

Displacement constraints is an approach that approximates the complex systems of equations normally encountered when modeling deformable or articulated objects. Objects are defined by

linking together primitive objects (such as points and line segments) by a system of geometric constraints (point-to-point, point-to-line, etc.). The constraint system is solved independently from the dynamic system. The dynamic phase treats each primitive object as a separate rigid body, thus simplifying the system of dynamic equations, while heuristics are used to determine how constraints imposed on an object are satisfied. It is a general algorithm that can be easily applied to systems with no corresponding physical model such as inverse kinematics and behavioral systems [19]. This method also provides an automatic mechanism for determining the initial configuration (initial solution of the constraints) of each object.

To create an object, primitive (rigid) objects are linked together using constraints. For example, the two line segments, s_1 and s_2 in figure 1.1(a), are joined together by a single point-to-point constraint. Several point-to-point constraints are used to define a chain-like object as shown in figure 1.1(b). Many such segments and constraints are combined to create complex objects such as meshes or lattices of arbitrary topology.

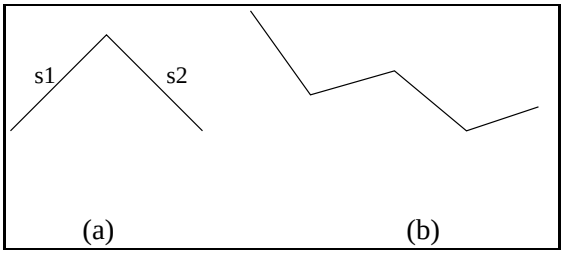

Figure 1.1: *Connecting segments using point-to-point constraints*

The idea behind displacement constraints is to uncouple the free movement of an object from the action of the constraints acting on an object's sub-components. Each sub-component is treated as an independent rigid object during the dynamic step. The external forces acting on each sub-component are calculated and the laws of motion are applied to each sub-component giving them new positions and orientations. Then, the sub-components are linked back together by enforcing the constraints acting on the object. The constraints are satisfied using iterative adjustments to the positions and orientations of each sub-component. This represents a simplified dynamic model that avoids complex computations that arise in other techniques,

such as solving a global system of coupled differential equations as in [9]. By using this simplified model, some physical credibility is sacrificed. However, Gascuel and Gascuel posit that sufficient physical credibility is maintained when first order momenta is conserved[2] [19].

One aspect of this technique, and of any physically accurate technique involving a discretized model, is the reliance on the local calculation of forces. With such a representation, the effect of an external force is local. Displacement constraints treats the sub-components of each object independent of the object as a whole. A force affects an object locally at a sub-component. Any overall global behaviour depends on the propagation of the local effect through its neighbours to the boundary of the object. There is no notion of assigning an object a global attribute, such as a particular angular velocity to make the object spin. This presents a problem if we want to animate an object that displays a more global behaviour, such as a stiff piece of paper falling to the ground, or being blown by the wind.

To solve this problem, we introduce a multilevel solution to the distribution of forces. Each object is represented as a composition of several levels of complexity. For example, in figure 1.2,

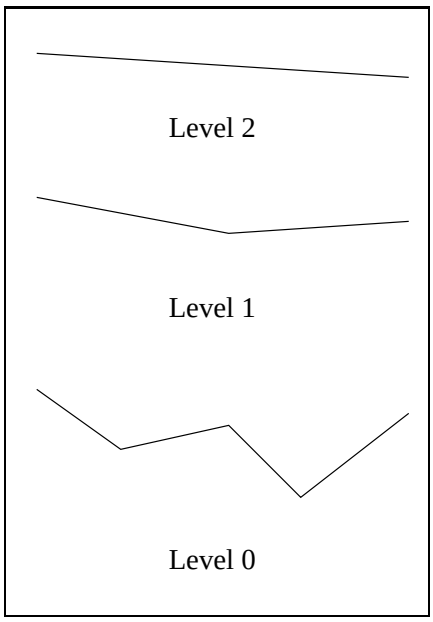

Figure 1.2: *Multilevel representation of a chain object*

[2]Displacement constraints does not actually preserve first order linear momenta as will be noted in section 3.5

Level 0 shows a chain-like object defined by four point-to-point constraints on five line segments. **Level 1** shows the next coarsest level, and **Level 2** shows the coarsest level as a rigid line segment. With such a representation of an object, the local and global behaviour of the object can be controlled by the animator. This is the main point of this thesis. In brief, if global behaviour is desired, then the force is applied at the coarsest grid (**Level 2** in figure 1.2). For a more local behaviour, forces are applied at the finer grids (**Level 0** in figure 1.2). By varying the proportion of the force applied at each level of an object, the animator can tune the local/global behaviour of that object. This gives animators greater control over the local/global behaviour of an object.

1.3 Thesis Layout

Previous research on deformable and flexible modeling will be presented in chapter 2. Displacement constraints will be discussed in detail in chapter 3, along with the implementation details and examples. The multilevel solution for local/global control will be presented in chapter 4. The conclusions and future work will be presented in chapter 5. The laws of motion are briefly outlined in Appendix A. Quaternions are discussed in Appendix B. The interface of the implementation is described in Appendix C.

Chapter 2

Related Work

As far as the laws of mathematics refer to reality, they are not certain, and as far as they are certain, they do not refer to reality.

- Albert Einstein

Deformable modeling for animation is a relatively new area of research and very interesting and promising results have already been achieved. This chapter serves to present a general survey of this body of research. Much of this research involves physically-based techniques which do not directly relate to the topic of this thesis. However, in the interest of completeness, we wish to clarify the role that displacement constraints plays in the arena of deformable model research.

Demetri Terzopoulos and his collaborators have published many results on this topic and will, therefore, have an entire section devoted to their work. Other significant results in this area of research will also be presented. This will be followed by a final discussion.

2.1 The Terzopoulos Regime

In 1987, Demetri Terzopoulos et al. [45] introduced elastically deformable models to the computer graphics community. They used simplified elasticity theory to develop deformable models of elastic curves, surfaces and solids. Physical properties, such as tension and rigidity, are incorporated into the model to accommodate static modeling of objects such as rubber, cloth, paper and flexible metals. Additional properties, such as mass and damping, allow for the dynamic simulation of these models. To model elastic objects, they use potential energies of deformation to define the internal elastic forces of an object as a function of the distance from the rest shape. When an object is at a rest position, its potential energy is zero. As the object

is deformed from its rest state (due to external forces), the potential energy grows larger. When external forces no longer act on the object, the internal energy forces return the object to its rest state. The dynamics of a deformable body are simulated by numerically integrating the ordinary differential equations through time. This system of PDE's is converted to an algebraic linear system by a numerical step-by-step procedure. These linear systems are solved using direct methods (Choleski decomposition) or relaxation methods (Gauss-Seidel). The algorithm handles collision detection by creating a field of potential energy around each object. Self-intersection is also addressed by surrounding the surface of each object with a self-repulsive collision force. The types of deformations possible with this model are mainly local in nature. Some global effects result from the minimization of the deformable energies.

This technique is based on nonlinear elasticity. Although this is in principle the most accurate way to characterize the behaviour of certain elastic phenomena, it poses some practical problems. First, the discrete equations become increasingly ill-conditioned as an object is made more rigid, or as the rest shape of an object is made more complex. Secondly, to integrate non-linear, time-varying partial differential equations and to deal with the probable nonuniqueness of their solutions requires relatively complex algorithms. Consequently, the computational cost of this technique is very high.

To solve these problems, Terzopoulos and Witkin [47] develop a hybrid formulation for deformable models. They decompose an object into a reference component and a displacement component. The reference component moves over time according to the laws of rigid body dynamics. Linear elasticity governs the behaviour of the displacement component (this obliterates any pretense to physical validity). By treating rigid body dynamics explicitly in the reference component, the resulting discrete equations remain well-conditioned as the model is made more rigid. However, this model does not behave well for highly elastic models, such as stretchy rubber sheets. This is due to the simple connection between the reference and displacement components through linear elastic forces. This model supports both local and global behaviours. The displacement component models local behaviour (deformations), while the reference component models global behaviour. Although this technique requires less computational effort than in [45], it is still computationally expensive and it is not interactive. Terzopoulos and

Fleischer [43] extend this work to model inelastic behaviour such as visco elasticity, plasticity and fracture.

In 1989, Terzopoulos et al. [46] further extend their previous work to include the simulation of thermal phenomena. The shape and dynamics of objects are not only governed by the Lagrange equations of nonrigid motion, as in [45, 47], but also by the partial differential heat equation. As in their previous work, objects behave elastically in their environment. But when an object comes into contact with a "hot" object, it conducts heat into its interior. The objects temperature rises and it becomes softer and more pliable. Eventually, when the temperature exceeds the melting point, the object melts into a simple, molecular liquid. They use molecular dynamics to model the liquid state. Fluid particles interact through long range attraction forces and short range repulsion forces between pairs of particles according to potentials of the Lennard-Jones type. They also incorporate friction into their model. This work could be extended to model inelastic behaviour (thermoplasticity), gaseous phenomena and radiative heat transfer.

In 1991, Terzopoulos and Metaxas [44] introduced deformable superquadrics as a tool for transforming geometric primitives and deformations into dynamic models. This is a hybrid model, combining parameterized superquadric ellipsoids (for global control) with free-form membrane splines (for local control). The global deformational degrees of freedom of the parameterized component control the global behaviour and attributes of the object. Local behaviour and surface detail are captured by the local deformation parameters of the underlying spline-based model. It is a dynamic deformable model whose behaviour is governed by the laws of rigid and nonrigid dynamics. Using simplified equations of motion, they achieved interactive rates on a graphics workstation. This model supports both local and global deformations. The global deformations, however, are restricted to the six degrees of freedom of the superellipsoids. Their main application of this technique is the approximation of surfaces from 2D image data and 3D range data. The superellipsoids capture the global shape features in an image, and the local spline-based model captures the local detail.

The work described thus far has concentrated on developing techniques for creating natural "looking" computer animations:

> We make no particular attempt to model specific materials accurately. Usually the general behaviour of a material will defy accurate mathematical description...Our goal is to develop physically-based models with associated procedures that can be utilized to create realistic animations...Hence, deformable models are convenient for computer graphics applications, where a keen concern with computational tractability motivates mathematical abstraction and expediency. [42]

Physical equations and parameters were incorporated into the models to achieve a desired level of realism. However, considerable simplifications of the dynamic equations were made to improve efficiency. Metaxas and Terzopoulos [34, 35] present an extension to [44], which they claim is a more physically valid approach. They use Lagrangian mechanics, nonlinear Kalman filtering theory to derive differential equations of motion that govern the dynamics of the models which are solved using the finite element method. This model supports both local and global deformations in a similar manner as in [44]. They extend this work to include parameterized global deformations such as tapers and bends. They also provide a mechanism for connecting rigid and/or nonrigid objects together via point-to-point constraints to create more complex models. Constraint forces are integrated into the equations of motion as in [9], but they add the capability of linking nonrigid parts. They also achieved interaction rates of several frames per second (on a SGI 4D-35TG).

This concludes the extensive body of research on deformable models by Terzopoulos et al. This work has been viewed as the "state-of-the-art" in computer graphics for physically-based deformable modeling and animation. They achieved very remarkable results and they continue to drive forward this area of research. Now we will present other results in the area of deformable model research.

2.2 Flexible Models

In 1988, Platt and Barr [38] use mathematical constraint methods based on physical constraint techniques and optimization theory to model and animate constrained flexible solids. They use reaction constraints to guide a flexible solid along a path and to prevent flexible solids from penetrating other polygonal objects. A reaction constraint is a procedure that determines the net external force acting at a point, and calculates the force required to satisfy a particular

constraint. This calculated reaction force cancels out any other forces that would otherwise violate a constraint. They require no additional differential equations but they are limited in scope. To handle complex, multiple constraints needed for flexible models, they use augmented Lagrangian constraints, a general technique for constrained optimization. They also allow for the inclusion of physical properties of flexible solids such as incompressibility (volume preservation) and moldability. This technique shows impressive results but is computationally expensive. As with most energy minimization methods, the model shows some global effects. However, mainly local behaviours are modeled with this technique.

More recently, Baraff and Witkin[7] present a dynamic model for flexible bodies where deformations take the form of global deformations of an object's rest shape. Objects are deformed by parametric "space warps" to all the points on the object. This limits the scope of possible deformations to the scope of the deformation function used. By restricting the possible deformations to those defined by a global parametric deformation, the complexity of the simulation is reduced. Also, because of the global nature of the shape parameters, the problems with stiffness, which are common in nodal formulations, do not arise. Non-penetration constraints are also incorporated into the model to accommodate collision detection/response and continuous contact. These constraints are maintained by analytically calculated contact forces. Multiple contact points are also supported. Using this technique, flexible objects may only exhibit global behaviours. Control of local behaviour is not addressed.

2.3 Articulated Figure Models

The following papers relate to articulated figure modeling and animation. There has been much research in this area [1, 2, 4, 5, 11, 12, 13, 14, 22, 23, 29, 30, 31, 36, 51, 52, 53, 54, 56, 57], and also in the area of animation and control of robotic structures [15, 16, 17, 18, 27, 32]. We include the following papers in this section since they directly relate to the displacement constraints technique, which was originally intended for articulated figure animation. These papers address similar problems, but present different solutions.

In 1988, Barzel and Barr [9] present a modeling system for constraint-based dynamics. Rigid primitive objects are linked together by various geometric constraint mechanisms such as

"point-to-point" and "point-to-nail" constraints. Using inverse dynamics, the constraint forces necessary to maintain these constraints during an animation are calculated and incorporated into the dynamic equations of motion. These coupled differential equations are numerically integrated through time. This allows the animated object to "assemble itself" while moving. Each rigid component follows the laws of rigid body dynamics and displays physically realistic behaviour during an animation. This approach is similar to displacement constraints in that it links together rigid primitive objects to form more complex objects that can then be animated dynamically. However, here the constraint forces are explicitly calculated and accounted for in the equations of motion. As with displacement constraints, forces act locally.

In 1990, van Overveld [49] presents a simple approach for 2D articulated figure animation, and later extends the model to 3D [50]. As with [9], and also with displacement constraints, rigid objects are connected together by geometric constraints to form a more complex object. This work differs from [9] in that it makes the assumption that the constraint forces and the external forces may be uncoupled. Displacement constraints also makes this assumption, but the manner in which the constraints are satisfied is quite different. With the method in [50], the individual rigid objects are first animated while neglecting the constraint forces. This eliminates the need for solving coupled systems of differential equations during the simulation. Then the constraint forces are calculated and used to satisfy the constraints during an iterative process. The purpose of this iterative process is to estimate the correction necessary to satisfy the constraints based on the calculated constraint forces. This process provides fast and stable convergence. It is not a physically correct model, but it provides an efficient method for creating complex animations. Also, behaviours modeled using this method are mainly local in nature.

2.4 Other Deformable Models

The following three papers present slightly different approaches to deformable models. Using particle systems and implicit surfaces, they develop unique methods that have produced very interesting results. We include these works to complete our survey on deformable models.

In 1992, Szeliski and Tonnesen [41] present an oriented particle system for modeling elastic surfaces. A surface is approximated by a set of particles, each of which is given a position as well

as an orientation. The inter-particle spacing is controlled by long range attraction forces and short range repulsion forces. New interaction potentials are devised to control the inter-particle orientation. The oriented particles tend to form locally planar or locally spherical arrangements. One its main advantages is that the topology of the surface is dynamic. Surfaces can be split, joined, or extended without the need for re-parameterization or manual intervention. The model can also be used to automatically approximate a surface from sparse 3D datasets, even when the topology of the surface is unknown. Both open and closed surfaces, either with or without holes, can be reconstructed. However, the fitted surface tends to appear "blobby" even if the original dataset possesses sharp features. The main drawback of this technique is the lack of precise control over the analytic (mathematical) representation of the surface. Particle-based surfaces also require more computation to simulate their dynamics than other parametric formulations such as spline-based surfaces. Global response of a surface to an external force is also not included in the model.

House et al. [28] also use particle systems to model flexible materials. They demonstrate their technique using a particle model of cloth. The constraints and interactions that occur at the thread level, such as thread collision, thread stretching, thread bending and thread trellising, are represented by energy functions. Their algorithm has three phases. First, the dynamic phase determines the forces acting on each particle and moves the particles according to the Newtonian equations of motion (for rigid bodies). Each particle is treated individually, ignoring all interparticle forces. Interparticle collisions and collisions between particles and other geometric models in the scene are accounted for during this phase. The second phase enforces the interparticle constraints by moving the particle system into a local energy minimum using a stochastic optimization technique called stochastic gradient descent. Collisions between particles and geometric objects are accounted for in this phase as well using high energy penalties when a particle penetrates an object. Finally, the velocities of each particle are adjusted to account for their new positions. This algorithm is very similar to the general algorithm of displacement constraints. The main differences are the geometric primitives used and the constraint solving algorithm (phase 2). As with displacement constraints, global behaviour is difficult to achieve. This is also a very computation intensive technique. One example simulation

required three CPU-days on an IBM RS6000 class workstation. Even though their claim that this is a "physically-based" model still remains to be shown, very convincing simulations of cloth were produced. However, other less computationally expensive cloth models exist that achieve even greater realism.

More recently, Gascuel [20] presents a model based on iso-surfaces of potential fields. An object is modeled as a skeleton (composed of rigid components that follow the laws of rigid dynamics) covered by a deformable skin layer. The skin layer is an isopotential implicit surface generated by the underlying skeleton. As the skeleton moves, the skin deforms with it. It is a continuous surface model that accommodates geometric and physical descriptions of objects. It also generates and maintains exact contact surfaces during collisions between flexible solids. This facilitates precise calculation of reaction forces. An object's deformation is determined from the contact surface generated during a collision. Deformations on the contact surface are used to compute the reaction forces. The model handles sudden collisions, lasting collisions and equilibrium situations. It also provides an elegant solution to the multiple collision problem that is independent of the order in which the collisions occur. Interactive modeling rates are possible, using a polygonal model of the implicit surface, and realistic animations can be created. However, this approach is not physically accurate, and it is restricted to locally deformable behaviours only. Global behaviour is achieved only through the propagation of local interactions.

2.5 Discussion

Many techniques for modeling deformable objects have been presented. Many use mathematical models based on physics to achieve certain levels of realism. Others sacrifice physical validity in return for more efficient algorithms. Displacement constraints belongs to the latter body of research. It is very similar to the geometric formulation of Barzel and Barr [9] and to the dynamic formulation of van Overveld [49, 50] and will be described in detail in the following chapter. Few of the techniques presented provide a means for controlling the local and global behaviour of a deformable body. Only one or the other is usually provided. The methods in [34, 35, 44] facilitate both local and global control, but the global deformations are limited by the parameters of the global formulation. We extend the displacement constraints technique to accommodate a more general solution for local/global control. This will be presented in chapter 4.

Chapter 3

Displacement Constraints

A thing may look specious in theory, and yet be ruinous in practice; a thing may look evil in theory, and yet be in practice excellent.

- Edmund Burke

Displacement constraints is a technique introduced by Gascuel and Gascuel [19] for constructing and animating articulated figures. According to the authors, it is a simplified dynamic modeling technique that takes physical parameters into account and preserves first order linear momentum (this point will be discussed later in section 3.5). Constraint forces are not explicitly calculated, but are formulated as geometric constraints that are solved using an iterative scheme. Avoiding the explicit calculation of constraint forces and solving for the constraints separately is more efficient than using coupled dynamic systems as in [9, 49, 50]. By coupling the constraint forces with the external forces, dynamic computations based on the simulation of Newtonian physics become both conceptually complex and computationally expensive.

Complex objects are defined by connecting primitive rigid objects together. Each primitive object has its own mass and tensor of inertia. These primitive objects are linked together by geometric constraints to form the main object. During the dynamic step of the animation, the primitive objects are treated independently and react to external forces, achieving new positions and orientations. Then, the constraints are enforced by adjusting the position and orientation of each primitive object and iterating until the constraints are satisfied.

Introduced originally for the modeling and animation of articulated figures, we apply the technique to the modeling of deformable surfaces as well. We also add control features, giving animators greater control over the local/global behaviour of an object. This is presented in chapter 4.

3.1 The General Algorithm

Given a set of objects, each of which is defined by several primitive objects, the new positions and orientations of each object are computed using the algorithm in figure 3.1. Step 1

For each time step:

1. Calculate all external forces.
2. Solve for the dynamics.
3. Solve the constraint system.
4. Update the linear and angular velocities.

Figure 3.1: *Original Displacement Constraints Algorithm*

determines all external forces (gravity, contact forces, user-applied forces, etc.) acting on each primitive object. Step 2 involves integrating through time the equations of motion for each primitive object (see Appendix A). Any external forces, other than constraint forces, are taken into consideration here. Each primitive object, acting independently from the other objects, reacts to these external forces to obtain a new position and orientation.

In step 3, the constraints are activated by iteratively displacing (rotating and translating) each primitive object until the constraints are met (within a set threshold), or until the maximum number of iterations is reached. This solution technique is purely geometric, rather than numerical and it is easy to implement and understand. Due to its geometric nature, it is also easy to visualize and debug. The algorithm will be described in greater detail in the following section.

Step 4 maintains the linear and angular velocities of each primitive object accounting for their new positions and orientations obtained in the previous step. The original positions and orientations prior to the dynamic step (step 2) are saved. After the constraints are solved, the original and final positions/orientations are used to update the effective linear/angular velocities achieved during the timestep.

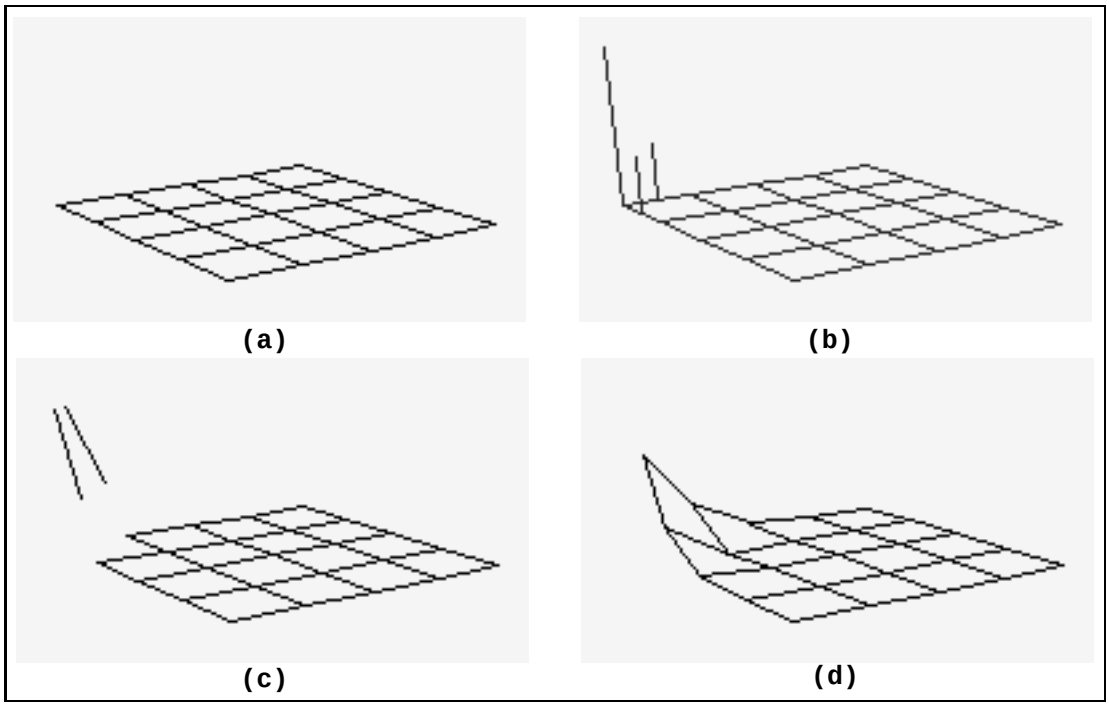

Figure 3.2: *Original Displacement Constraints Algorithm.*

The algorithm is also illustrated in figure 3.2. Figure 3.2(a) shows the initial configuration of a mesh created by linking line segments with point-to-point constraints. An upward force (shown as vertical lines) is applied to a corner of the mesh in figure 3.2(b). Notice that a portion of the force is also applied to the center of the segments at that mesh point. This is to give the segments linear acceleration, as well as angular acceleration. The percentage of the force applied to the center of a segment is under the control of the animator. Figure 3.2(c) shows the mesh after the dynamic step. Only the segments at that mesh point are affected. The effect of the force during the dynamic step is exaggerated to demonstrate the algorithm. Figure 3.2(d) shows the final mesh after the constraint solving step. These four figures represent one complete timestep of the animation.

3.2 Point-to-Point Constraints

A point-to-point constraint defines a geometric relationship between two solids, S_1 and S_2 with centers of mass $\vec{G_1}$ and $\vec{G_2}$, respectively (see figure 3.3(a)). It says that a particular point on each object ($\vec{P_1}$ on S_1 and $\vec{P_2}$ on S_2) must coincide at all times (here we assume all points are defined as three dimensional vectors in global coordinates). When solving the constraints, if a constraint is violated, we must compute a translation and rotation for each primitive object that, when applied to each object, will satisfy the constraint. Small translations and rotations

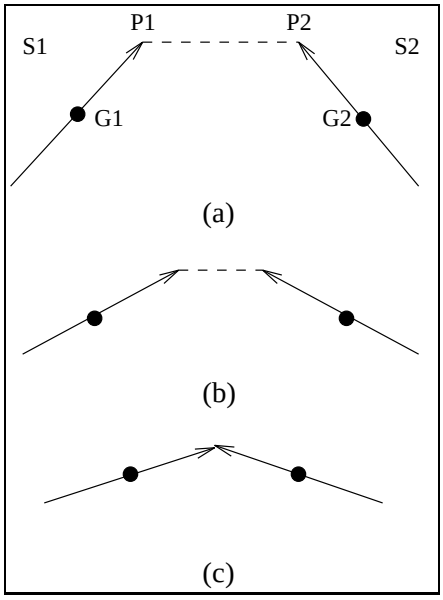

Figure 3.3: *Solving the constraints iteratively*

are applied to each primitive object iteratively until the constraints are met. As illustrated in figures 3.3(b) and 3.3(c), the objects (rigid line segments) are translated and rotated until the constraint is met.

3.2.1 Computing the Translations

When solving the constraints, Gascuel and Gascuel attempt to conserve first order linear momentum by using the equation:

$$m_1\vec{v_1} + m_2\vec{v_2} = constant \ ,$$

where the m_i's are the masses and the $\vec{v_i}$'s are the velocities of the primitive objects being constrained. Let $\overrightarrow{\Delta G_1}$ and $\overrightarrow{\Delta G_2}$ be the translation vectors we wish to solve for (the linear displacement of center of mass). To conserve linear momentum we must have:

$$m_1 \overrightarrow{\Delta G_1} + m_2 \overrightarrow{\Delta G_2} = 0 \ . \tag{3.1}$$

The point-to-point constraint is defined as:

$$P_1 + \overrightarrow{\Delta G_1} = P_2 + \overrightarrow{\Delta G_2} \ . \tag{3.2}$$

Solving equations 3.1 and 3.2, we obtain:

$$\overrightarrow{\Delta G_1} = \frac{m_2}{m_1 + m_2} \overrightarrow{P_1 P_2} \tag{3.3}$$

$$\overrightarrow{\Delta G_2} = \frac{m_1}{m_1 + m_2} \overrightarrow{P_2 P_1} \ . \tag{3.4}$$

So, the vectors $\overrightarrow{P_1 P_2}$ and $\overrightarrow{P_2 P_1}$ represent the total distance between points P_1 and P_2. The fractions, $\frac{m_2}{m_1+m_2}$, and $\frac{m_1}{m_1+m_2}$ are the amount that each object must translate along these distance vectors. The translations are being weighted by the masses of each primitive object. Intuitively, this means that a heavier object will move less than a lighter object.

3.2.2 Computing the Rotations

Solving a constraint with translations alone will result in very unrealistic behaviours. For a single constraint, a solution is always possible using only translations. However, for a more

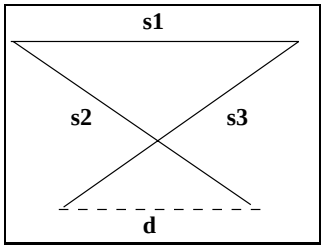

Figure 3.4: *Using only translations can result in an overconstrained system*

complex constraint system, rotations must also be applied. For example, in figure 3.4, segments s_1, s_2 and s_3 are all of equal length and are intended to form an equilateral triangle by three

point-to-point constraints. However, using translations only, all three constraints can never be satisfied. Meeting the constraint between segment s_2 and s_3 (bringing the distance d to zero) will violate the other two constraints. Therefore, we must also use rotations.

Let $\overrightarrow{\Delta\mho_1}$ and $\overrightarrow{\Delta\mho_2}$ be the small rotation vectors used to adjust the orientations of the primitive objects (segments s_1 and s_2 in figure 3.3, for example). We determine the ratio between rotation and translation by simulating the action of a piece of rubber of stiffness k pulling the two points together (points $\vec{P_1}$ and $\vec{P_2}$ in figure 3.3). The forces acting on each point would be:

$$\vec{F_1} = -\vec{F_2} = k\ \overrightarrow{P_1P_2}\ .$$

The translations produced by these forces during time interval Δt are:

$$\overrightarrow{\Delta G_1} = \frac{k\Delta t^2}{2m_1}\overrightarrow{P_1P_2} \tag{3.5}$$

$$\overrightarrow{\Delta G_2} = \frac{k\Delta t^2}{2m_2}\overrightarrow{P_2P_1}\ . \tag{3.6}$$

To obtain the same equations as in equation 3.3 and 3.4, we take:

$$k = \frac{m_1 m_2}{m_1 + m_2}\frac{2}{\Delta t^2}\ . \tag{3.7}$$

The rotations we need are just those produced by the same forces, $\vec{F_1}$ and $\vec{F_2}$ during Δt. Applying a force, $\vec{F}$ to a point $\vec{P}$ on an object with center of mass $\vec{G}$, produces the torque,

$$\vec{\tau} = \overrightarrow{GP} \times \vec{F}\ .$$

The torque, τ, is a rotation vector whose direction is the axis of rotation, and whose magnitude is the angle of rotation which we measure in radians (see figure 3.5).

So, the rotations are defined as:

$$\overrightarrow{\Delta\mho_1} = J_1^{-1}\frac{k\Delta t^2}{2}(\overrightarrow{G_1P_1} \times \overrightarrow{P_1P_2}) = \frac{m_1 m_2}{m_1 + m_2}J_1^{-1}(\overrightarrow{G_1P_1} \times \overrightarrow{P_1P_2}) \tag{3.8}$$

$$\overrightarrow{\Delta\mho_2} = J_2^{-1}\frac{k\Delta t^2}{2}(\overrightarrow{G_2P_2} \times \overrightarrow{P_2P_1}) = \frac{m_1 m_2}{m_1 + m_2}J_2^{-1}(\overrightarrow{G_2P_2} \times \overrightarrow{P_2P_1}) \tag{3.9}$$

where J is the Jacobian, or inertia tensor matrix that defines the spatial mass distribution of the object.

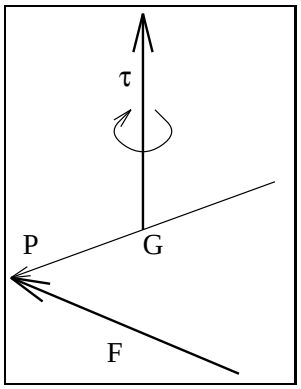

Figure 3.5: *The torque produced by a force at a distance*

To reduce complexity, and in so doing reduce physical accuracy even further, the values $\frac{m_1 m_2}{m_1+m_2} J_1^{-1}$ and $\frac{m_1 m_2}{m_1+m_2} J_2^{-1}$ can be replaced by parameters set by the user. These parameters control the amount of rotation applied to each primitive object during the constraint solving iterations. Another parameter γ, $0 <= \gamma <= 1$, can be set to control the amount of translation. The user can control these parameters to produce a different behaviour. When $\gamma = 1$ there is no rotation, only translation. This will produce a more rigid behaviour (figure 3.6(b)). When $\gamma = 0$ there is no translation, only rotation. This will produce a more flexible behaviour (figure 3.6(c)). This is because a low translation factor forces the primitive object to translate less during the iterations, giving it more time to rotate. However, in practice, γ must be greater than zero for the constraint solver to converge to a solution (i.e. in general, the constraints will not be solved using only rotations). When the constraints cannot be satisfied in the current number of iterations, then the animator is flagged.

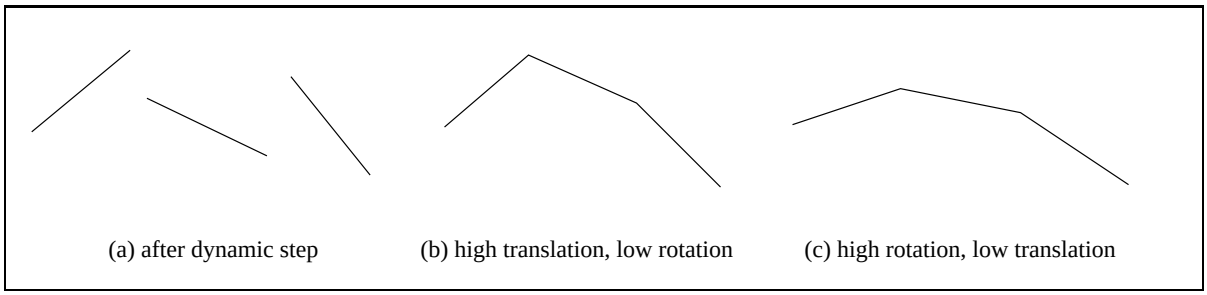

Figure 3.6: *Varying the ratio between translation and rotation*

3.3 Other Constraint Types

Gascuel and Gascuel [19] also describe how to incorporate other constraint types, such as point-to-segment constraints, angular constraints and twist constraints. Additional constraint types, such as point-to-curve, point-to-surface and point-in-sphere are other possible variations. Length constraints, where the length of a segment lies in a certain range, may also be useful. These constraints can be used to control the degrees of freedom at each link (or joint) in the solid, giving the animator even greater control over the behaviour of each object.

3.4 Initial Configuration Computation

Computing the initial configuration of a constrained system can be nontrivial. Fortunately, displacement constraints provides an automatic method of calculating an initial configuration of the constrained objects: simply iterate through the usual constraint loop (step 3 in figure 3.1) until the constraints are satisfied, except here we do not update the linear and angular velocities of the objects (step 4 is not executed). This gives us the initial position and orientation of each object in the system.

3.5 Analysis of Displacement Constraints

Gascuel and Gascuel [19] claim that displacement constraints possesses certain physical properties. Although they do not discuss the degree to which their technique models the real world, we will analyze this claim from a "real physics" point of view.

Equation 3.1 is not a proper representation of the law of conservation of momentum because the $\overrightarrow{\Delta G_i}$'s are not velocities! They are translation vectors. This induces an early linearization of the moment balance equations, which does not yield a correct formulation from a dynamics perspective. Also, equations 3.3 and 3.4 enforce the idea that heavier objects move less than lighter objects, which does not have any physical basis.

The idea that "the proportion between rotation and translation in the motion of a solid must depend on the ratio between tensor of inertia and mass" is not correct. First, such a ratio does not exist. The tensor of inertia is a spatial quantity measuring the distribution of mass in

a body, whereas the mass is a scalar quantity. Secondly, how this presumed ratio relates to the proportion between translation and rotation velocities is very unclear [3].

Gascuel and Gascuel also state that the process of obtaining the corrections iteratively is *equivalent* to adding constraint forces to meet the constraints. Their implied definition of "equivalent" is unclear. There is no reason to assume that, given two articulated solids subjected to external forces, these two approaches (uncoupled vs coupled) produce identical geometric models (position and orientations of the solids) at the next time step. In any case, the two approaches to solving the constraints are *not* equivalent. Discoupling the constraint forces from the free motion of the objects loses considerable physical credibility. The constraint forces are not explicitly calculated and are not incorporated into the dynamic equations. They are approximated by iterative tunings in the displacements on the constrained objects. This does not have any physical basis.

In short, this method is not physically valid from a physicist's or engineer's point of view and the claim that this method preserves first order linear momentum is a false one. Some physical parameters are taken into account, but the constraint equations derived do not accurately reflect a real dynamic process. This may seem like a harsh criticism, but it is not. We wish to eliminate any ambiguity regarding the physical validity of this technique and emphasize the point that even though this technique lacks physical accuracy, complex behaviours can be modeled. Displacement constraints serves as a means to creating computer animations that are *visually* comparable to more "physically-based" methods, but without all of the overhead.

3.6 Limitations and Benefits

An inherent limitation of displacement constraints is that objects only react locally. This stems from the local effect of external forces and the local interactions of the constraints. Only interactions between primitive objects are considered. A solid is not treated as a whole during the constraint processing, nor during the dynamic phase. The only global behaviour attained is due to the propagation of external forces throughout an object.

Another limitation is the lack of physical validity as mentioned in the previous section. This restricts displacement constraints to applications where speed, rather than accurate physics, is

required (i.e. applications such as commercial or artistic animation). A general problem with dynamic animation techniques is that animators have little or no control over the behaviour of an object over time. Once the initial conditions are specified, the behaviour of each object is governed by the dynamic equations. This is also true for displacement constraints.

In addition, displacement constraints can become time inefficient for more complex objects. By complex, we mean an object that is composed of many primitive objects whose connectivity is nontrivial. If many constraints are acting on a set of primitive objects, convergence to a solution deteriorates during the iterative constraint solving phase. As we restrict the motion of the primitive objects by adding more constraints to them, the solution space is reduced. Finding a global solution that satisfies all constraints becomes a more difficult task.

However, as mentioned earlier, displacement constraints has several benefits. It is a technique for solving geometrically constrained systems that is easy to understand and implement. Most importantly, especially for animators, interactive rates for modeling and animation are possible. Our purpose of using it here is to show how an algorithm with no explicit control over global deformation, can be augmented to provide such control with little change to the underlying formulation.

3.7 Implementation

Displacement constraints was originally designed for modeling and animating articulated figures. Using rigid line segments as our modeling primitive, we apply the method to the modeling of surfaces (2D meshes in 3D space). This is a natural extension in that it emphasizes the generality of displacement constraints as a technique for solving geometric constraints. Figure 3.7 shows a mesh created by linking together rigid line segments with point-to-point constraints. Each line segment is of equal length and the error is set to 0.4% of the length of a line segment. The error represents the maximum distance between any two constrained endpoints.

The algorithm outlined in figure 3.1 is used to animate the mesh through time. Each grid point of the mesh maintains a list of the segments and endpoints connected at that grid point. As each grid point is processed during the constraint solving phase, these segment/endpoint lists are traversed, processing each constraint sequentially. Figure 3.8 outlines the solution

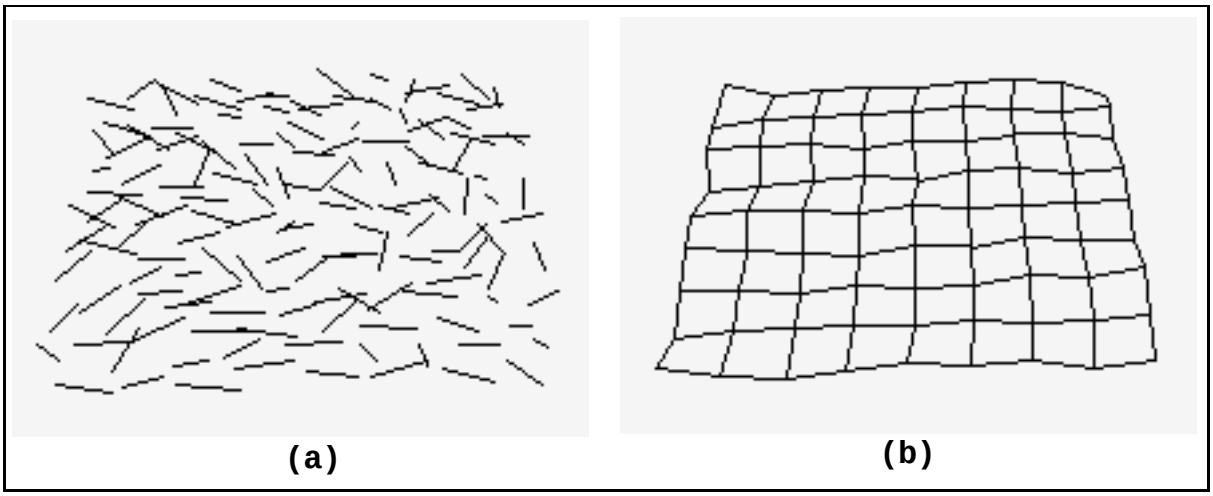

Figure 3.7: *Linking lines segments to form a mesh surface.*

algorithm (step 3 in figure 3.1) for a mesh object[1]. First, the initial error is calculated. The error returned from `find_error(k)` is the maximum distance between any two constrained endpoints. This is calculated initially to determine whether the constraint iterations need to be executed. The `while` loop represents one iteration of the constraint solving phase. The global parameters, `MIN_ERROR` and `MAX_ITERATIONS` are controlled by the animator (see figure 3.9(a)). At each iteration, the entire system of constraints is processed. For each constraint, a rotation and translation is applied to each constrained segment. The amount of the rotation and translation is also controlled by the animator (see figure 3.9(b)). The resulting error is then calculated. If the error is less than the minimum error (`MIN_ERROR`), or if the maximum number of iterations (`MAX_ITERATIONS`) is exceeded, then the iteration process halts and the resulting mesh is returned. To help suppress any artifacts that may be caused by the order in which the constraints are processed, a red/black ordering could be used when traversing the constraints. With such as ordering, every other mesh point is processed during one pass over the mesh, then the remaining mesh points are processed during the second pass. This still has an implied ordering, but it lessens the dependency imposed by the sequential ordering.

Figure 3.10 demonstrates the iterative solution using the mesh in figure 3.2. The original mesh is shown in figure 3.10(a) (corresponds to figure 3.2(b)). A force (shown as lines) is applied

[1] Here, `k`=0 since we are not using the multilevel algorithm.

```
void solve_constraints(int k) {
     /*k is the level number*/
  int num_itns ; /*to count the number of iterations*/
  double error ; /*the error in level k*/

  num_itns = 0 ;              /*initialize iteration counter*/
  error = find_error(k) ;     /*find initial error in level k*/
  while ((error > MIN_ERROR) && (num_itns <= MAX_ITERATIONS))
    { num_itns++ ;            /*increment iteration counter*/
      For each grid point in level k do
        {Process each pt-to-pt constraint at the current grid point.}

      error = find_error(k) ; /*find the error in level k */
    }
}
```

Figure 3.8: *Displacement Constraints Solution Algorithm*

to a corner of the mesh. Figure 3.10(b) shows the mesh after the dynamic step (corresponds to figure 3.2(c)). Figures 3.10(c) to 3.10(f) show the iterative solutions of the mesh. The final mesh in figure 3.10(f) corresponds to figure 3.2(d).

Figure 3.9 shows some of the various control panels available to the animator. In figure 3.9(a), the timestep (in seconds), the maximum number of iterations for the solution algorithm, and the minimum error are manipulated by the animator. The rotation factor in figure 3.9(b) represents the amount of rotation (in radians) applied to a segment during one constraint solving iteration. The translation factor is used to determine the amount of translation applied to each segment in a point-to-point constraint. It represents the percentage of the distance between two constrained points. Each constrained endpoint is translated by this amount during one constraint solving iteration.

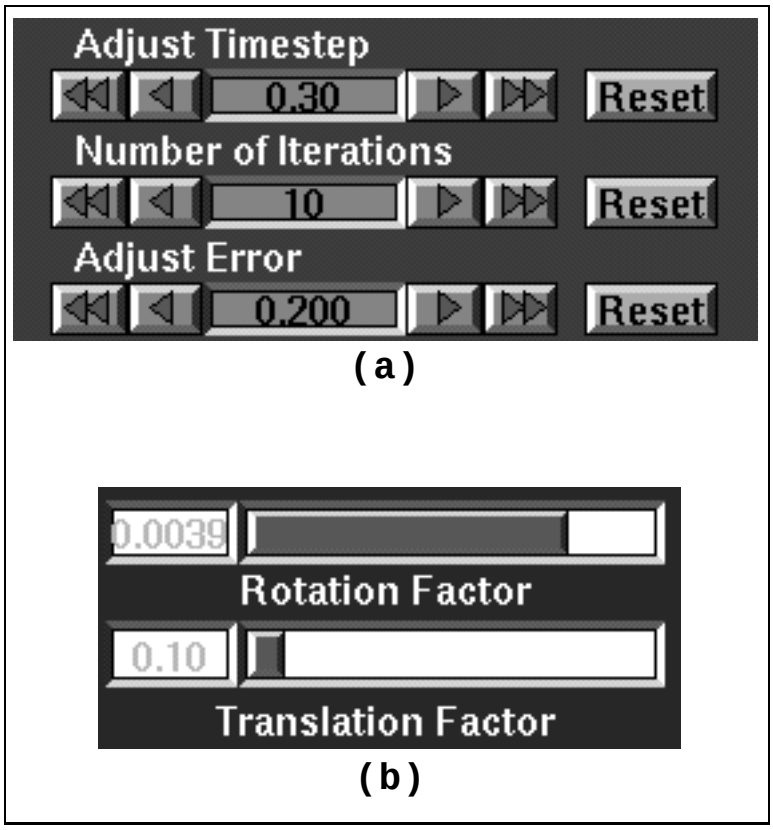

Figure 3.9: *User Interface Control Panels.*

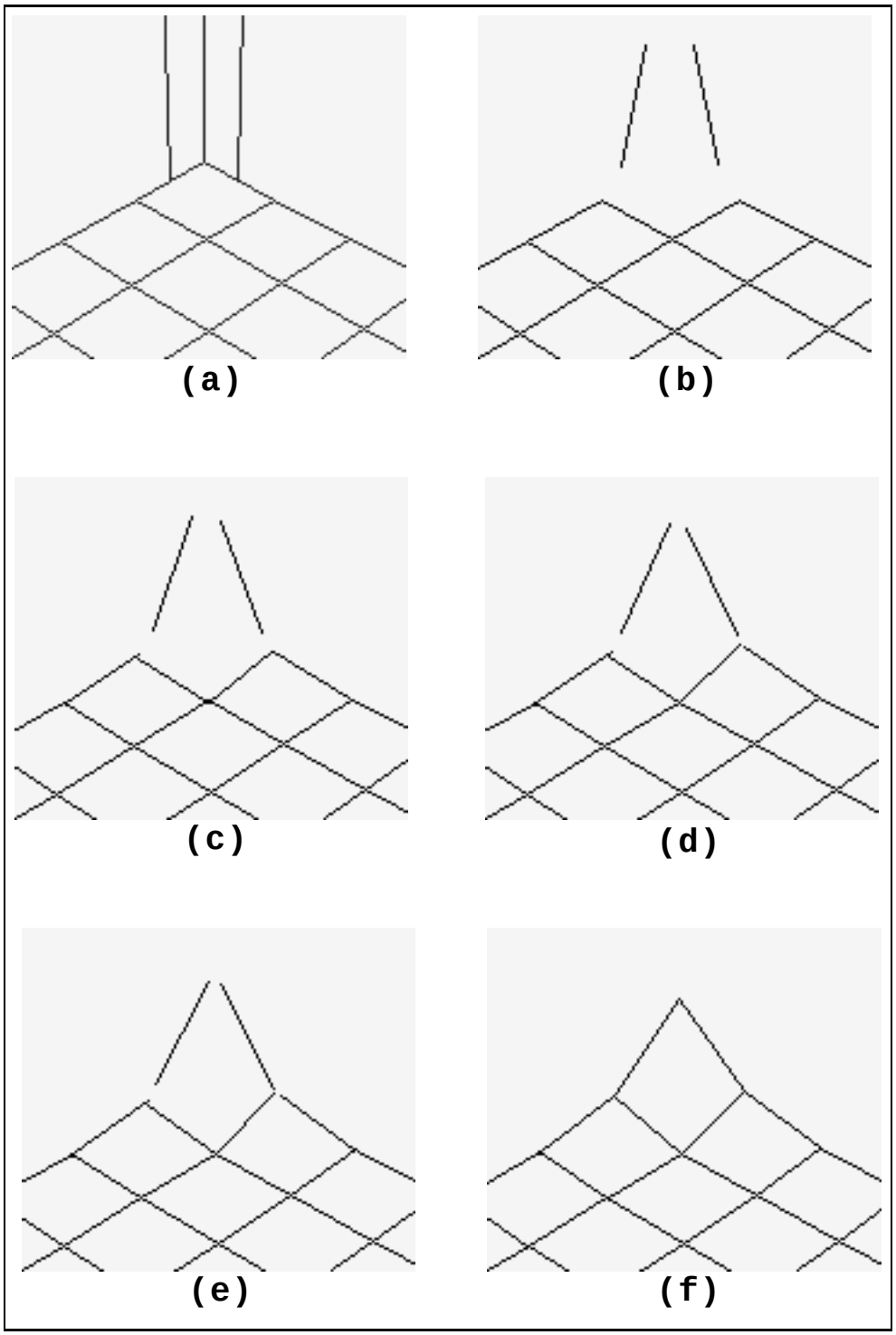

Figure 3.10: *One full cycle of Displacement Constraints, showing multiple iterations to satisfy constraints.*

3.8 Local Nature of Displacement Constraints

One important characteristic of displacement constraints is the locality of the effect of an external force. This is demonstrated in figures 3.11 and 3.12. An upward force is applied to a corner of the mesh in figure 3.11, and to the center of the mesh in figure 3.12. In both cases, the global effect of the forces is minimal. Only local responses are observed. Global responses are limited to those attained by the propagation of local interactions. The dynamics and constraints act on the sub-components of each primitive object, not on the object as a whole. Consequently, an object is restricted to local responses to external forces. Using displacement constraints alone, we cannot assign an object global properties. This does not pose a problem if the animator requires an object to exhibit only local deformations. However, if more global behaviours are desired, then simple positional constraints are not sufficient. As an alternative to an explicit global solution, we introduce a multilevel solution for global behaviour in the following chapter. An alternate solution would be to add more constraints to the objects. However, this would result in very a "stiff" numerical system that may become numerically unstable, without reducing the size of the timestep considerably.

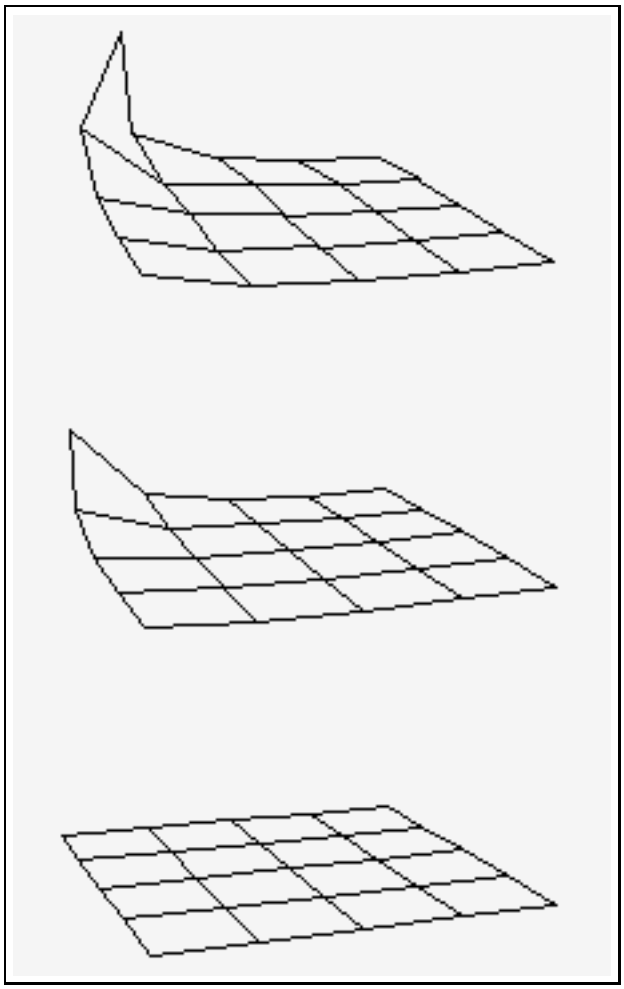

Figure 3.11: *Applying an upward force to a corner of a mesh.*

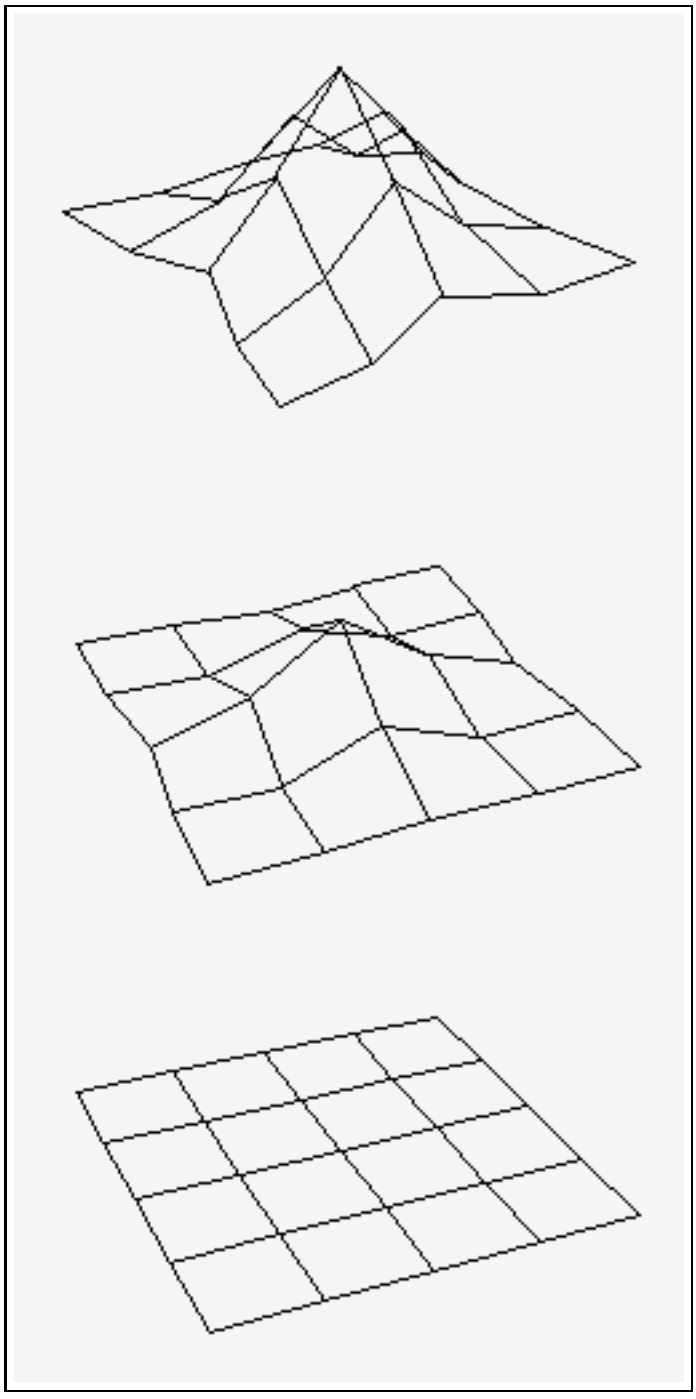

Figure 3.12: *Applying an upward force to the center of a mesh.*

Chapter 4

Achieving Global Behaviour

Physical concepts are free creations of the human mind, and are not, however it may seem, uniquely determined by the external world.

- Albert Einstein

4.1 Overview

For an object to respond globally to a single external force, there must exist some mechanism that allows an object composed of many elements to be treated as a whole, rather than a discretized collection of sub-parts. Otherwise, global behaviour will depend upon how the localized force is propagated to the surrounding regions. To solve this problem we present a solution for global behaviour that is inspired by the numerical multigrid method. For an introduction to multigrid methods see [10] and [33]. We will then demonstrate our application of this technique to our solution for global behaviour with several examples. To distinguish our solution from the multigrid method, we will refer to our solution as a *multilevel* solution.

4.2 Our Multilevel Solution

We adapt the multigrid concepts to formulate a solution for global behaviour. Previous techniques achieve global behaviour by using global deformation functions [7, 8, 37, 55]. These methods show realistic results for global responses, but local responses are not usually included. We use the geometric properties of multigrid to create a multilevel solution for force distribution. An object is decomposed into levels of decreasing resolution, until the coarsest (rigid) level is reached. External forces acting on the object are distributed to the different levels by the restriction operator. The nature and amount of force distributed at each level can

be set by the animator to control the local/global behaviour of an object. Figure 4.1 shows a 9×9 mesh (level 0 at lower left) and its decomposition into coarser levels.

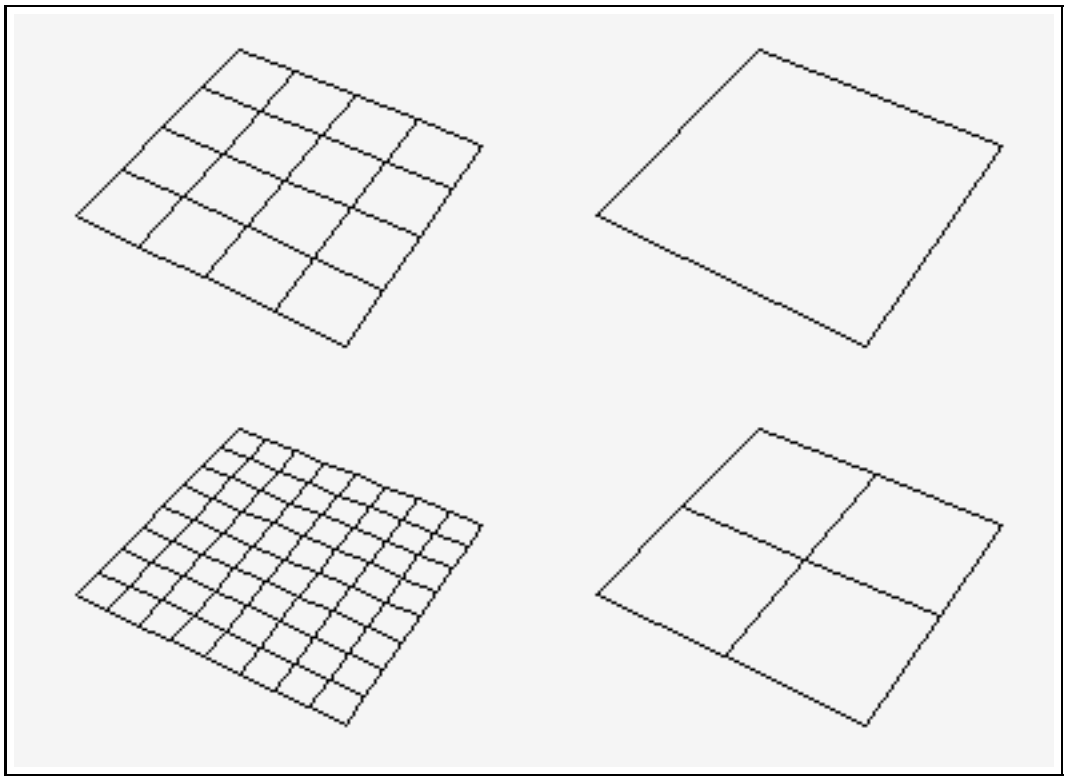

Figure 4.1: *Multilevel breakdown of a 9×9 mesh*

The general multilevel solution algorithm is shown in figure 4.2. Passing information from a finer grid to a coarser grid is called *restriction*. Passing information from a coarser grid to a finer grid is called *prolongation*. We assume that the spacing between mesh points in a particular level is half the spacing between mesh points in the next coarser level. In other words, we will be working with (square) meshes of size $2^n + 1$ where $n >= 0$. First, we calculate the external forces acting on each line segment in the original mesh. We define a restriction operator that passes positions, orientations, masses and forces from the finest level to all coarser levels. Then, starting from the coarsest mesh, the forces are applied to the segments in that mesh (dynamic phase), the constraints are solved (using the original displacement constraints method from figure 3.1) and the new positions and orientations are passed to the next finer level (prolongation). This process is repeated until the finest level is reached. The C version of

For each time step:

1. Determine forces acting on original mesh.
2. Restriction Phase - For all levels starting at finest:
   (a) Apply the restriction operator to transfer positions, orientations, masses and forces through to next coarsest level.
3. For all levels starting at coarsest:
   (a) Apply forces and solve using rigid body dynamics.
   (b) Satisfy the constraints using Gascuel's method to determine final position, orientation and linear and angular velocity at this level.
   (c) Prolongation Phase: use the prolongation operator to pass the solutions (positions and orientations) to the next finer level.

Figure 4.2: *DC Algorithm with Multilevel Solution*

this algorithm is provided in figure 4.3.

This algorithm is similar to the basic numerical multigrid scheme. The main differences are that we are using the displacement constraints iteration method to solve the constraints at each level, and the restriction/prolongation operators are passing geometric information between the levels, rather than numerical information (residuals and error approximations). Another important observation is that, unlike multigrid methods where the main goal is to accelerate convergence of the iterative solution method, here the multilevel approach serves to guide the solution to that intended by the animator (i.e. local/global). The actual convergence of the displacement constraints iterations at each level may or may not be affected by using the initial guess passed down from a coarser level.

4.2.1 The Restriction Operator

The restriction operator passes the positions, orientations, masses and forces from the finest level to the coarser levels, level by level. The linear and angular velocities are not passed between the levels. Instead, the segments in each level maintain their own velocities from the previous timestep. This method was shown to be more effective at maintaining accurate

```
void solve() {
  register int i ;

     /*LEVEL_NUM = number of levels*/
     /*level 0 = finest level, level LEVEL_NUM-1 = coarsest level*/

  forces(0) ; /*collect all forces acting on segments at finest level*/
  if (MULTI_LEVEL) /*if using the multilevel method*/
    {
      restriction() ; /*pass positions/forces to all coarser levels*/
      for (i=LEVEL_NUM-1; i>0; i--) /*for each level (coarser to finer)*/
        {
          save_old_positions(i) ; /*for adjusting the velocities later*/
          dynamics(i) ;           /*apply forces to each object*/
          solve_constraints(i) ;  /*solve constraints at level i*/
          adjust_velocities(i) ;  /*update new velocities at level i*/
          prolongation(i) ;       /*pass solutions from level i to i-1*/
        }
    }
/*for finest level*/
  save_old_positions(0) ;
  dynamics(0) ;
  solve_constraints(0) ;
  adjust_velocities(0) ;
}
```

Figure 4.3: *The C version of Displacement Constraints Algorithm with Multilevel Solution*

velocity information at each level. For each level, the initial positions/orientations before the dynamic phase, and the final positions after the constraint solving phase, are used to update each segment's linear/angular velocities. Here, we assume that, in the initial configuration of the mesh, all linear velocities and angular velocities are zero.

The positions and orientations for the segments in a level, k, are obtained directly from the next finer level, $k-1$. The endpoints of a segment in level k are assigned the values of the corresponding mesh points of level $k-1$. These two endpoints define the segment's orientation. This is illustrated in figure 4.6. The mesh points, $q1$ and $q2$, are used to define the position and orientation of the line segment connecting mesh points $q1'$ and $q2'$ at the next coarser level. If

r_{k-1} and c_{k-1} are the coordinates (row and column) of a mesh point in level $k-1$ (finer), then the corresponding coordinates, r_k and c_k, in level k (coarser) are defined by:

$$r_k = \lfloor \frac{r_{k-1}}{2} \rfloor \tag{4.1}$$

$$c_k = \lfloor \frac{c_{k-1}}{2} \rfloor \tag{4.2}$$

The coarsest level of the mesh can be made more rigid by the addition of two cross segments, $s1$ and $s2$, as shown in figure 4.4. This ensures that during the constraint processing, no bending of this level will occur.

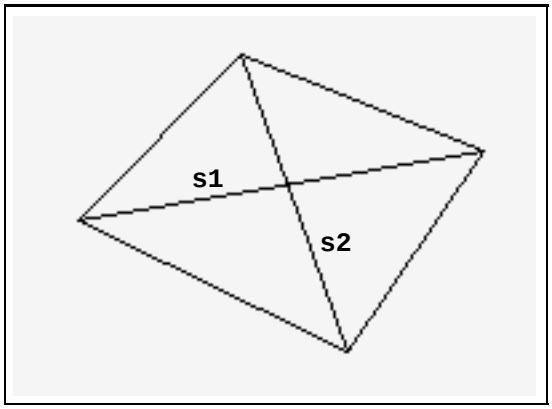

Figure 4.4: *Adding two cross segments at coarsest level to enforce rigidity*

The masses are distributed to each segment based on a weighted average scheme. This distribution maintains the total mass at each level (i.e. each level has equal total mass). It is assumed that each individual segment has an even mass distribution (the inertia tensor matrix for each line segment is the identity matrix). If all segments in the original mesh are of equal mass, then the total mass is evenly distributed to each segment in each coarser level.

A force is distributed to the coarser levels by assigning a portion of that force to the corresponding mesh points of the coarser levels. As mentioned previously, the amount of the force at each level is controlled by the animator. If the corresponding point of the finer mesh lies between two mesh points in the coarser mesh, then the force is evenly distributed among the neighbouring mesh points of the coarser level. In figure 4.5(a), a force is applied to a middle mesh point. Remember, from the previous chapter, that a portion of this force is also applied

to the center of the four line segments connected at that point (the forces are shown as straight lines). This mesh point does not have a corresponding point in the coarser level. Figure 4.5(b) shows the restriction to the next coarser level. The original force is evenly distributed to the segments connected to the four neighbouring mesh points, $p1$, $p2$, $p3$ and $p4$. A portion of these forces is also applied to the center of each segments. Figure 4.5(c) shows the state of the coarser

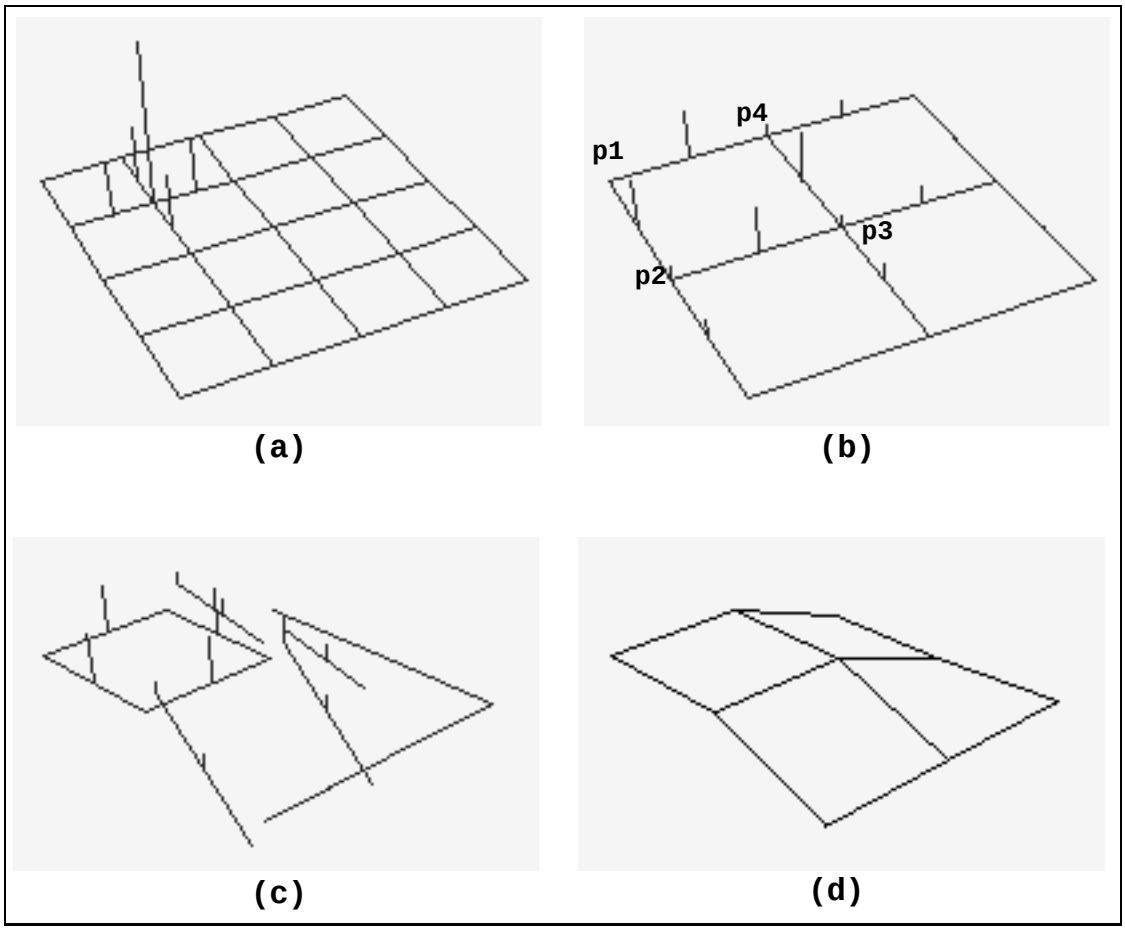

Figure 4.5: *Restriction of a center mesh point.*

level after the dynamic phase. Notice that the forces on the four segments connecting the mesh points $p1$, $p2$, $p3$ and $p4$, are only acting on the center of gravity of each segment. This is because equal forces are applied at both endpoints of each segments. The final configuration of the mesh after solving the constraints is shown in figure 4.5(d). In this example, the original force is divided equally between the three levels.

The animator also has the option of filtering the forces at each level in order to control the response of the mesh to an external force. This means that a portion of a force at a mesh point can be distributed to the neighbouring mesh points. For example, the force at mesh point $p1$ in figure 4.6(a), is restricted to $p1'$ in the next coarser level in figure 4.6(b). A portion of the force at the point , $p1'$, is then distributed to the neighbouring mesh points, $p2$ and $p3$. The same is done for the force at mesh point $p1''$ at the next coarser level in figure 4.6(c). To achieve different behaviours, the forces may be distributed to a wider portion of each level.

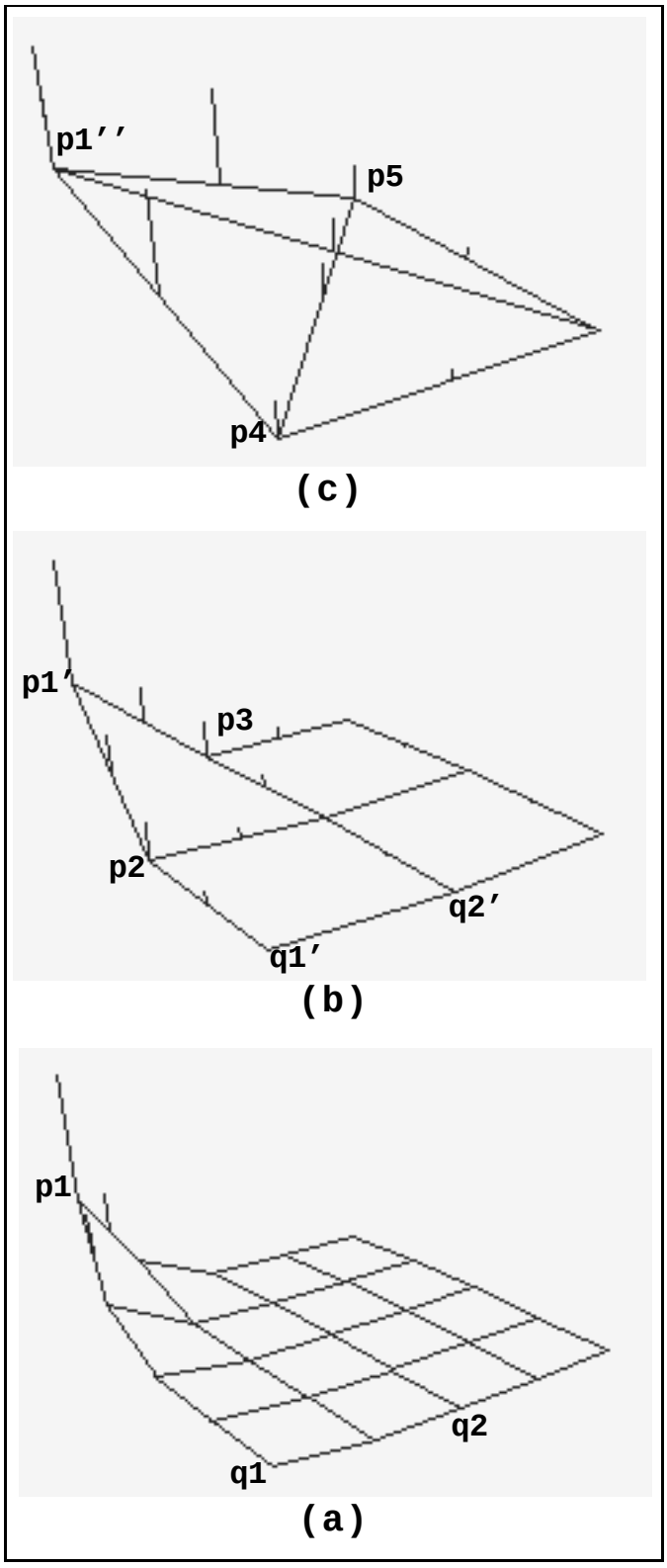

Figure 4.6: *Restriction Operator*

4.2.2 The Prolongation Operator

The prolongation operator passes the solutions (positions and orientations) from the level just solved to the next finer level. These solutions are used to obtain the initial configuration at the finer level. The segments at this level are positioned and oriented according to the solutions passed down from the coarser level. The mesh at this level is then solved, the new positions/orientations are passed to the next finer level, and so on.

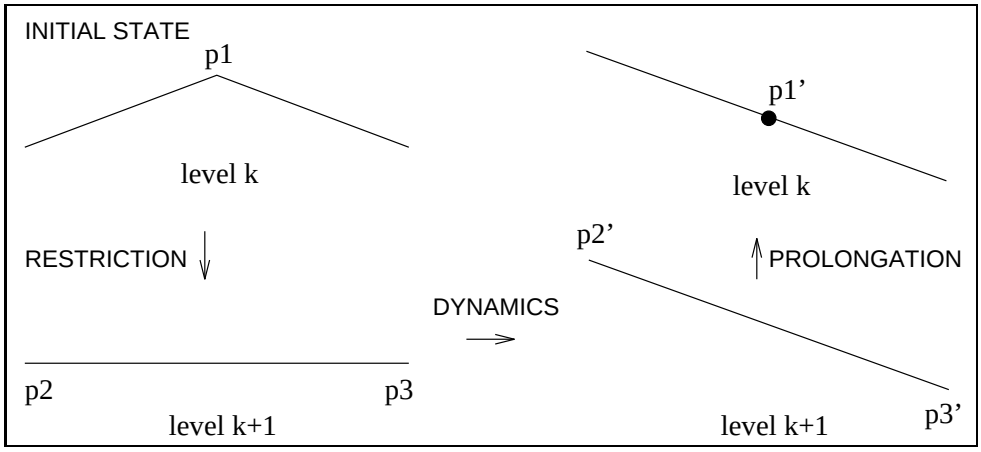

Figure 4.7: *Prolongation Operator using linear interpolation.*

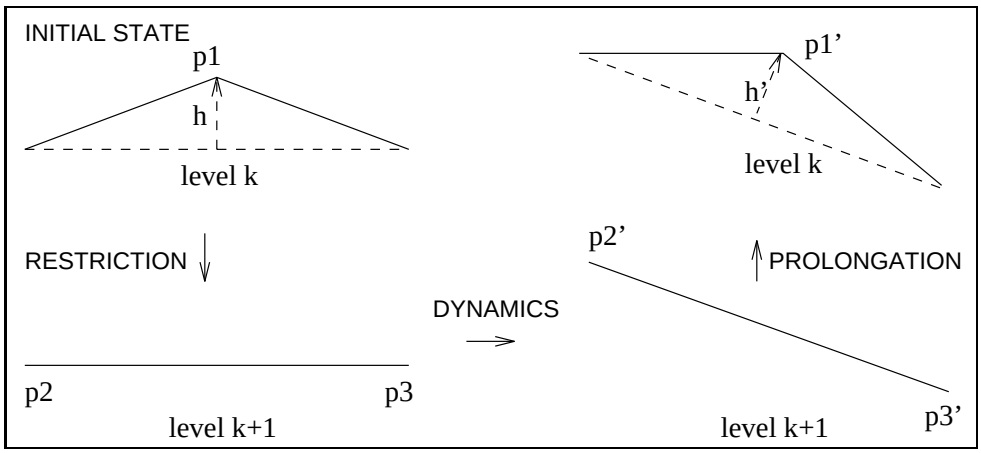

Figure 4.8: *Prolongation Operator using offset information.*

The prolongation operator used here is intended to be the exact adjoint of the restriction operator (i.e. if there are no changes in the coarser mesh, then the original finer mesh is

restored). However, difficulty arises when a mesh point in the finer mesh lies between two mesh points in the coarser mesh. Figure 4.7 illustrates this problem. Point $p1$ in level k lies between points $p2$ and $p3$ in level $k+1$. After the dynamic phase, the prolongation operator must determine the position for point $p1'$. Here, linear interpolation is used. The new point, $p1'$, is simply the midpoint between mesh points $p2'$ and $p3'$. The point $p1'$ is then used to determine the positions and orientations of the segments connected to $p1'$.

Simply using interpolation will eliminate the detailed features of the original mesh, as shown in figure 4.7. This will tend to smooth out the behaviour of the mesh, thus completely inhibiting local behaviour. The original position, $p1$, with respect to the mesh points, $p2$ and $p3$ should be used to determine the position of point $p1'$. Linear interpolation loses this offset information. Also, the interpolated segment, $\overline{p2p3}$, has a shorter length than the sum of the two original segments connected at $p1$. Since the lengths of the segments are fixed, the original lengths must be enforced when determining the new segments at the finer level (i.e. the two segments connected to $p1'$).

Figure 4.8 shows a different solution to these problems. The original offset vector, $\overline{h}$, from the point $p1$ to its corresponding midpoint in level $k+1$ (between $p2$ and $p3$) is calculated. This vector is used by the prolongation operator to calculate the position for the new mesh point, $p1'$. The offset vector, $\overline{h}$, is a local vector with respect to the line segment $\overline{p2p3}$. In our implementation, we use quaternions to update the orientation of the offset vectors during the prolongation phase. For example, the rotation between segments $\overline{p2p3}$ and $\overline{p2'p3'}$ is converted to a quaternion and applied to $\overline{h}$ to get $\overline{h}'$. The cross product of the two line segments defines the axis of rotation, $\overline{r}$, and the magnitude of $\overline{r}$ is set to the angle between the two segments. This solution retains the local details that exist in the finer mesh.

4.3 Example Animations

Figures 4.9, 4.10 and 4.11 contain frames taken from four separate animations. The error tolerance used when solving the constraints is set to 0.4% of the length of the smallest line segment; gravity is turned off and the animation is initiated by applying an upward impulse force (for one timestep) to one corner of the mesh. The initial surfaces are identical in terms of their physical properties, but the distribution of the forces between levels is different in each.

Figure 4.9 shows the behaviour of the surface when 100% of the force is applied to the finest mesh and 0% to all other levels. This behaviour is equivalent to the that calculated using Gascuel's original displacement constraints technique.

Figure 4.10 shows the behaviour of the surface when 100% of the force is applied to the coarsest level and 0% to all other levels. The surface acts as a stiff sheet and after the first time step has constant linear and angular velocity. No local response is observed.

In figure 4.11, the forces are distributed equally between all levels in the multilevel representation. The behaviour of the surface is intermediate between that of figure 4.9 and figure 4.10.

The next example in figure 4.12 also demonstrates achieving global behaviour using this multilevel approach. These are 9 frames taken from a 90 frame animation with a timestep size of 0.2 seconds per frame. The error is set to 0.4% of the length of a line segment. An upward impulse force is applied to a corner of the mesh in figure 4.12(a). This force is only in effect for one timestep. Figures 4.12(b) through (i) show the mesh reacting to that force, maintaining a global angular velocity. As before, no local response is observed.

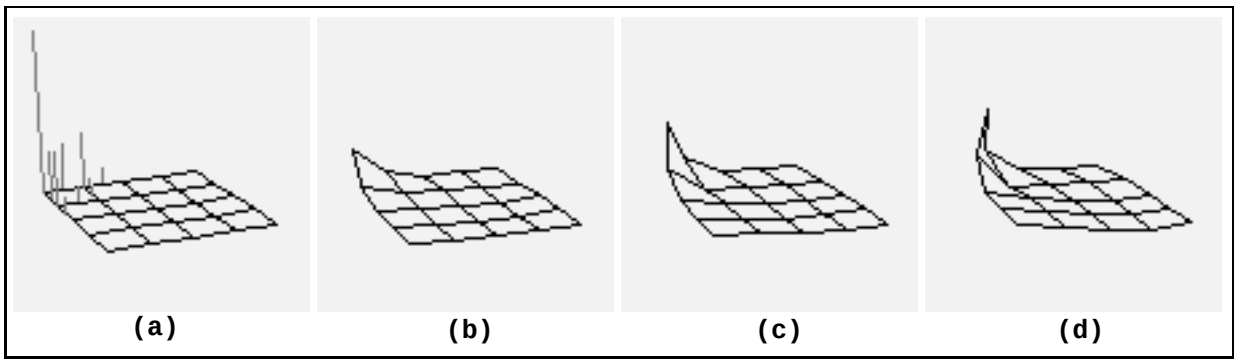

Figure 4.9: *Local Behaviour - 100% of force acts upon the finest level.*

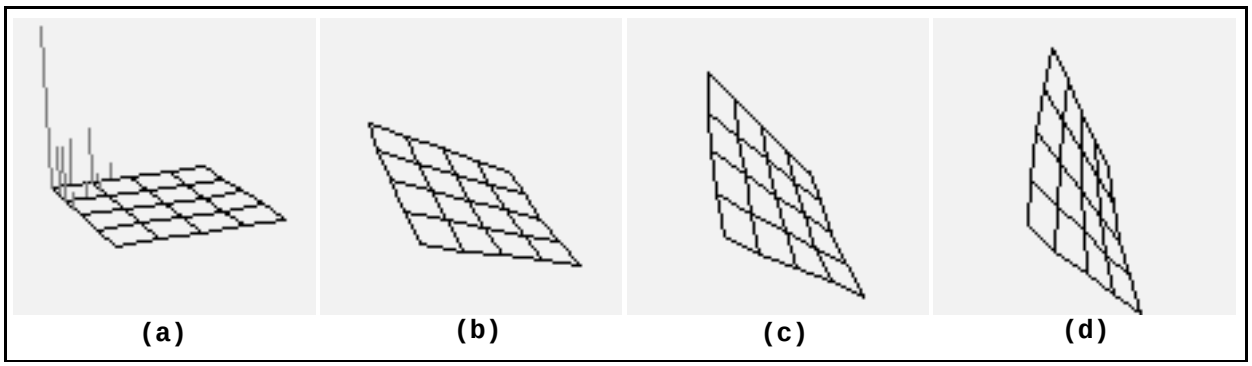

Figure 4.10: *Global Behaviour - 100% of force acts upon the coarsest level.*

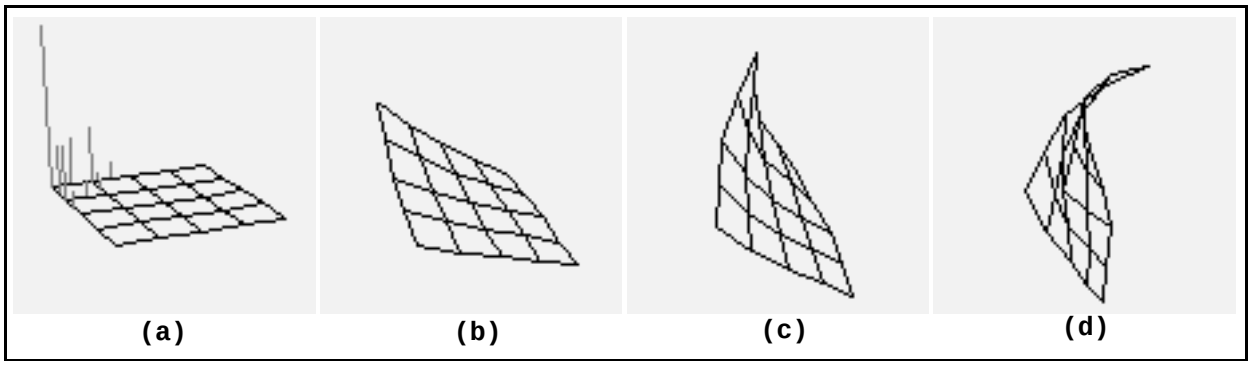

Figure 4.11: *Force distributed equally among levels.*

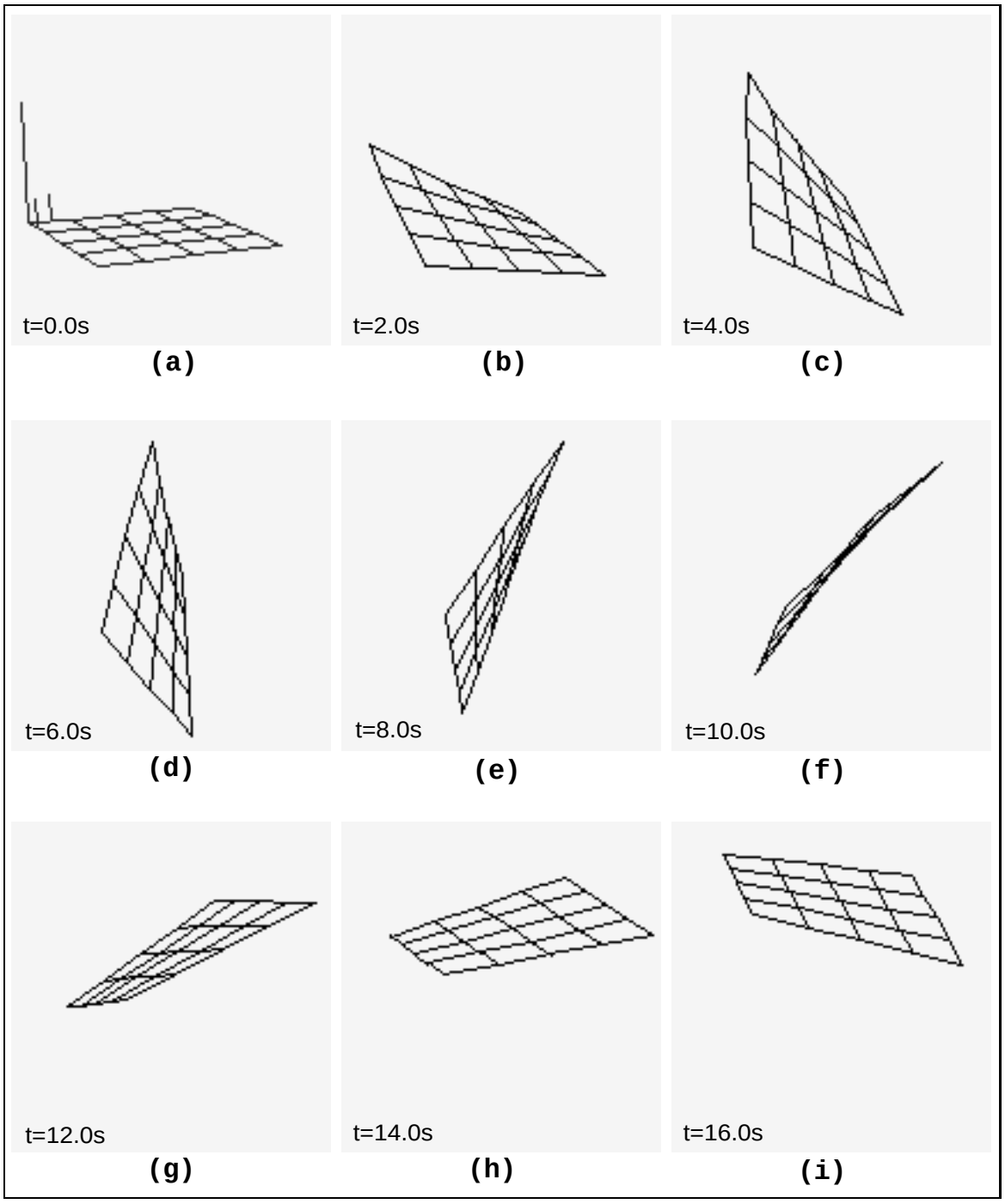

Figure 4.12: *Achieving global behaviour*

The final three examples (figures 4.13, 4.14 and 4.15) show a mesh bouncing off the floor using different force distributions. The error is set to 0.4% of the length of a line segment and the only external forces acting on the mesh are gravity and contact forces from the floor. Simulated shadows are included for clarity and are shown in white on the surface of the floor.

In figure 4.13, 100% of the forces are applied to the finest level and 0% to all others. This mesh is very flexible and its behaviour is equivalent to that modeled by the original displacement constraints algorithm.

In figure 4.14, 100% of the forces are distributed to the coarsest level to produce behaviour characteristic of a rigid surface. Figure 4.14(b) shows the mesh colliding with the floor and figures 4.14(c) through (f) show the new angular velocity (counter clockwise) resulting from the collision. In figure 4.14(f), another corner of the mesh collides with the floor and counteracts the angular velocity.

In the final example, the forces are distributed equally to all the levels of the mesh producing a behaviour that is intermediate between that of figure 4.13 and figure 4.14, with both local and global characteristics.

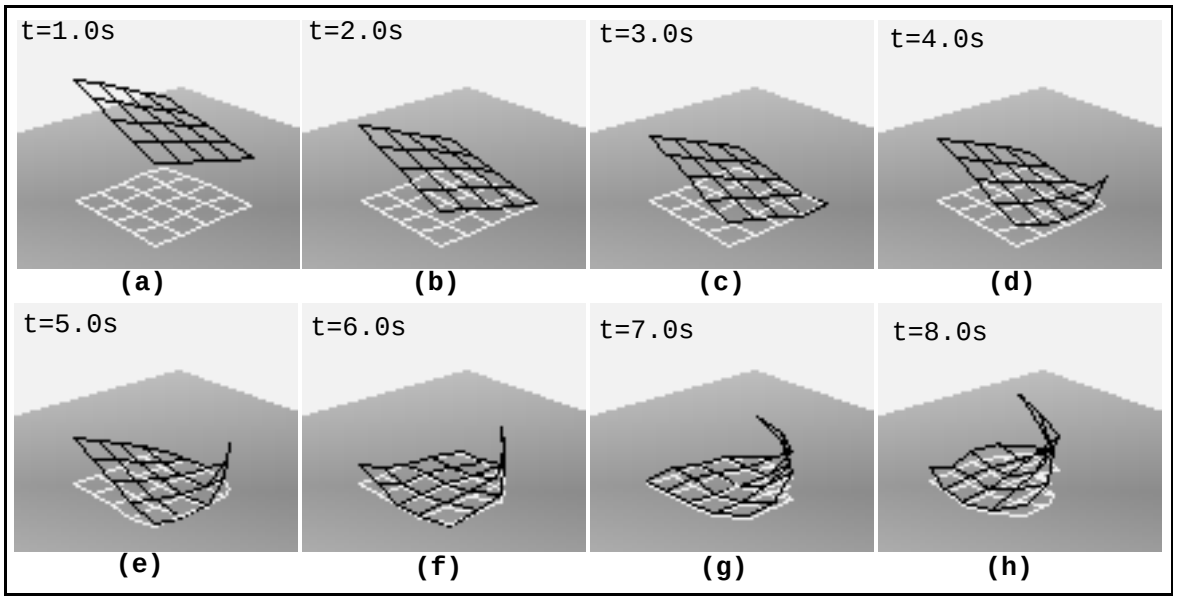

Figure 4.13: *Flexible mesh bouncing off floor. 100% of the force acts upon the finest level.*

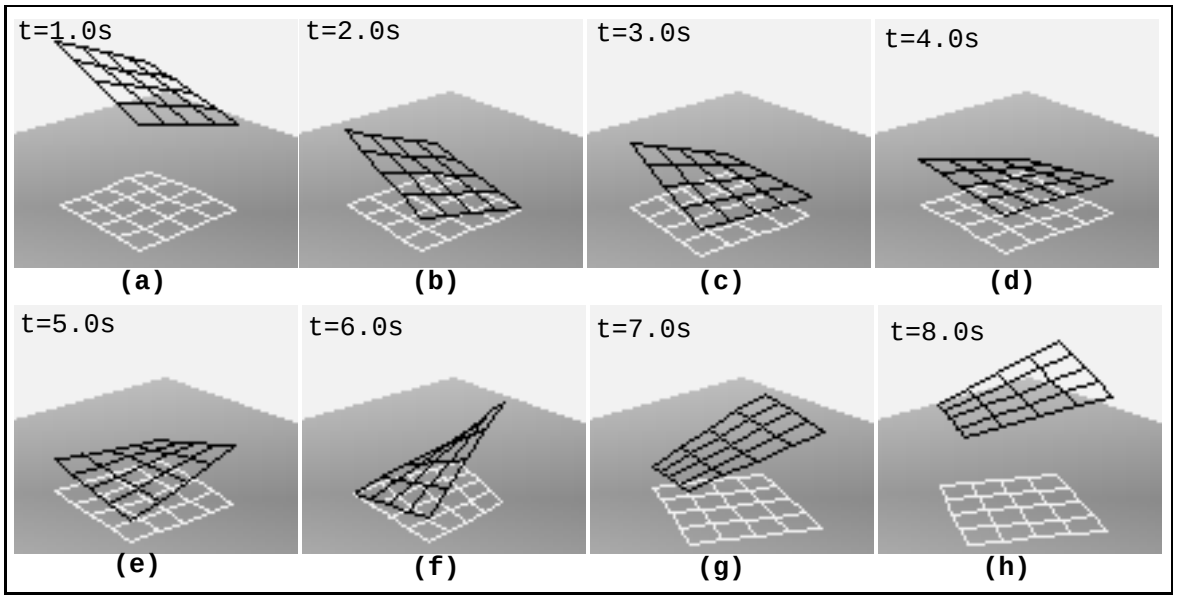

Figure 4.14: *Rigid mesh bouncing off floor. 100% of the force acts upon the coarsest level.*

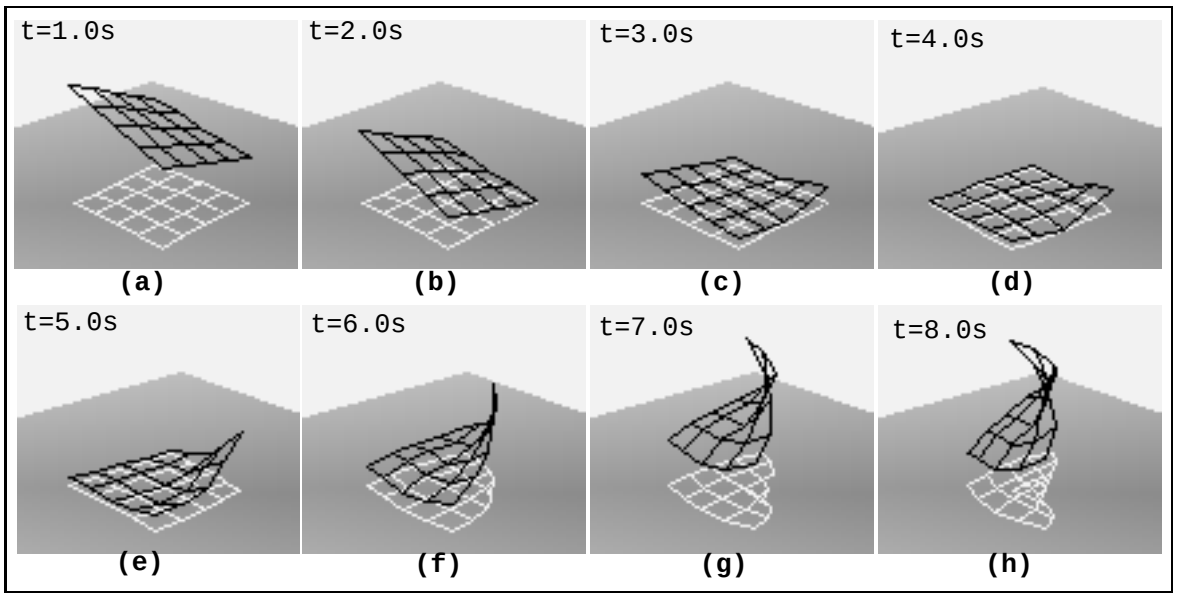

Figure 4.15: *An intermediate behaviour. The forces are distributed equally among levels.*

Chapter 5

Conclusions

Thank you on behalf of the group and I hope we pass the audition.

- The Beatles

We have implemented and analyzed the displacement constraints method for modeling and animating objects defined using geometric constraints. We have compared this method with other methods in our literature review. We have shown that displacement constraints is a technique that it easy to understand and implement, and it is useful for defining and animating deformable objects.

We introduced a multilevel approach for incorporating global behaviour into the generic displacement constraints model. With such a mechanism, an animator is able to control the local/global behaviour of an object. This feature is not present in most previous techniques. Using multilevel techniques for the distribution of forces on an object has shown to be very effective in controlling the local/global attributes of an inherently local deformable model. The entire animation system is easy to use, easy to understand and it offers tools that help animators model complex objects and behaviours with little effort.

5.1 Future Work

Our implementation currently supports only point-to-point constraints. The addition of various other constraint types, such as angular constraints, point-to-line constraints or length constraints, would enable us to model more complex objects and behaviours. Collision detection with other geometric models, and also self-intersection detection, could also be incorporated into the model.

We would like to extend the displacement constraints model to include the modeling of

elastic objects. Similar to [47], a reference shape that behaves as a rigid body could be defined for each object. This would represent the object's rest shape. As the object moves and deforms through time, forces are applied to each mesh point to restore the rest shape. These forces would be scaled by the distance from a mesh point to it's corresponding position in the reference shape. The greater the distance, the greater the force.

One important characteristic of the displacement constraints model is the independence of the primitive objects. The only connection between primitive objects is through the geometric constraints. This would facilitate the modeling of objects that can split apart, tear, or break into pieces.

Appendix A

Numerical Integration of the Equations of Motion for Rigid Body Dynamics

Each primitive object contains the following basic information:

- **m** - mass
- **x** - position vector
- **v** - linear velocity vector
- **f** - force accumulator vector
- **q** - orientation vector expressed as a quaternion
- ω - angular velocity vector
- **I** - tensor of inertia matrix
- **t** - torque accumulator vector
- **clr** - colour (rgb vector)

The equations of motion for rigid body dynamics [24] result in a second order system of differential equations. Euler's method is easy to implement and understand [39]. Instead of using the Euler method directly, we use the Euler-Cromer method [25]. This method takes the average of the old and new velocities computed by the Euler step. The resulting equations of motion are shown below [6, 48].

$$\mathbf{a}(t) = \mathbf{f}(t)/m \tag{A.1}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + (\Delta t)\mathbf{a}(t) \tag{A.2}$$

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + (\Delta t)\frac{\mathbf{v}(t+\Delta t) + \mathbf{v}(t)}{2} \tag{A.3}$$

$$\alpha(t) = I^{-1}\tau(t) \tag{A.4}$$

$$\omega(t+\Delta t) = \omega(t) + (\Delta t)\alpha(t) \tag{A.5}$$

$$\mathbf{r} = (\Delta t)\frac{\omega(t) + \omega(t+\Delta t)}{2} \tag{A.6}$$

$$\mathbf{q}_r \Leftarrow \mathbf{r} \tag{A.7}$$

$$\mathbf{q}(t+\Delta t) = \mathbf{q}(t)\mathbf{q}_r \tag{A.8}$$

where $\mathbf{f}$ is the external force, $\mathbf{a}$ is the linear acceleration, α is the angular acceleration, τ is the torque and $\mathbf{q}$ is the orientation vector, $\mathbf{r}$, expressed as a quaternion [40] (see also Appendix B).

We calculate the linear acceleration of an object, in equation A.1, given a force, $\mathbf{f}$, acting on the center of mass of the object. Equation A.2 determines the new linear velocity, $\mathbf{v}$, and equation A.3 determines the new position, $\mathbf{x}$.

We calculate the angular acceleration of an object, in equation A.4, given a torque, τ, acting at a certain distance from the center of mass of the object. Equation A.5 determines the new angular velocity, ω, and equation A.6 determines the new orientation vector, $\mathbf{r}$. This orientation vector, $\mathbf{r}$, is converted to a quaternion representation, $\mathbf{q_r}$, in equation A.7. We update the objects orientation, $\mathbf{q}$, represented as a quaternion, in equation A.8.

Appendix B

Quaternions

We use quaternions [40] to represent rotations. The matrix representation for rotations could be used instead, but using quaternions has several benefits. Quaternions require less memory and less computation due to their more compact notation. Quaternions are also less subject to roundoff error (numerical drifting) caused by successive rotations. This is because unit quaternions are closed under addition and multiplication.

Here we briefly describe the quaternion representation [48]. Given the unit vector, $\mathbf{a}$, representing the axis of rotation, and the angle of rotation, θ, the unit quaternion which defines this rotation is

$$\mathbf{q} = [s, \mathbf{w}] \tag{B.1}$$

where s is the scalar,

$$s = cos(\theta/2) \tag{B.2}$$

and $\mathbf{w}$ is the vector,

$$\mathbf{w} = \mathbf{a} sin(\theta/2). \tag{B.3}$$

Successive rotations are easily represented using quaternion multiplication. If $\mathbf{q_1}$ and $\mathbf{q_2}$ represent quaternion rotations, then the product $\mathbf{q_2 q_1}$ represents the composite rotation of $\mathbf{q_1}$ followed by $\mathbf{q_2}$. Multiplication between two quaternions, $\mathbf{q_2}$ and $\mathbf{q_1}$, is given by the following equation,

$$\mathbf{q_2 q_1} = [s_2, \mathbf{w_2}][s_1, \mathbf{w_1}] = [(s_1 s_2 - \mathbf{w_1} \cdot \mathbf{w_2}), (s_1 \mathbf{w_2} + s_2 \mathbf{w_1} + \mathbf{w_2} \times \mathbf{w_1})]. \tag{B.4}$$

To rotate a vector, $\mathbf{v} = (x, y, z)$, using quaternions, we first cast $\mathbf{v}$ as a quaternion, $\mathbf{p} = [0, (x, y, z)]$, where $s = 0$ is the scalar and $\mathbf{w} = (x, y, z)$ is the vector. To rotate $\mathbf{p}$ by a quaternion, $\mathbf{q}$, we multiply $\mathbf{p}$ on the left by $\mathbf{q}$, and on the right by the inverse $\mathbf{q}^{-1}$,

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}. \tag{B.5}$$

The rotated vector, (x', y', z'), is contained in the vector, $\mathbf{p}' = [0, (x', y', z')]$. The inverse of a quaternion is given by,

$$\mathbf{q}^{-1} = \frac{1}{(s^2 + \mathbf{w} \cdot \mathbf{w})}[s, -\mathbf{w}]. \tag{B.6}$$

From these definitions, it is a simple matter to convert from a quaternion to a rotation vector, and from a rotation vector to a quaternion.

Appendix C

Implementation Interface

The following interface was implemented in C using GL and the FORMS library (written by Mark Overmars) on SGI and IBM RS6000 computer graphics workstations. Figure C.1 shows the entire system. The main viewing window is displaying (in perspective) a 9×9 mesh, the

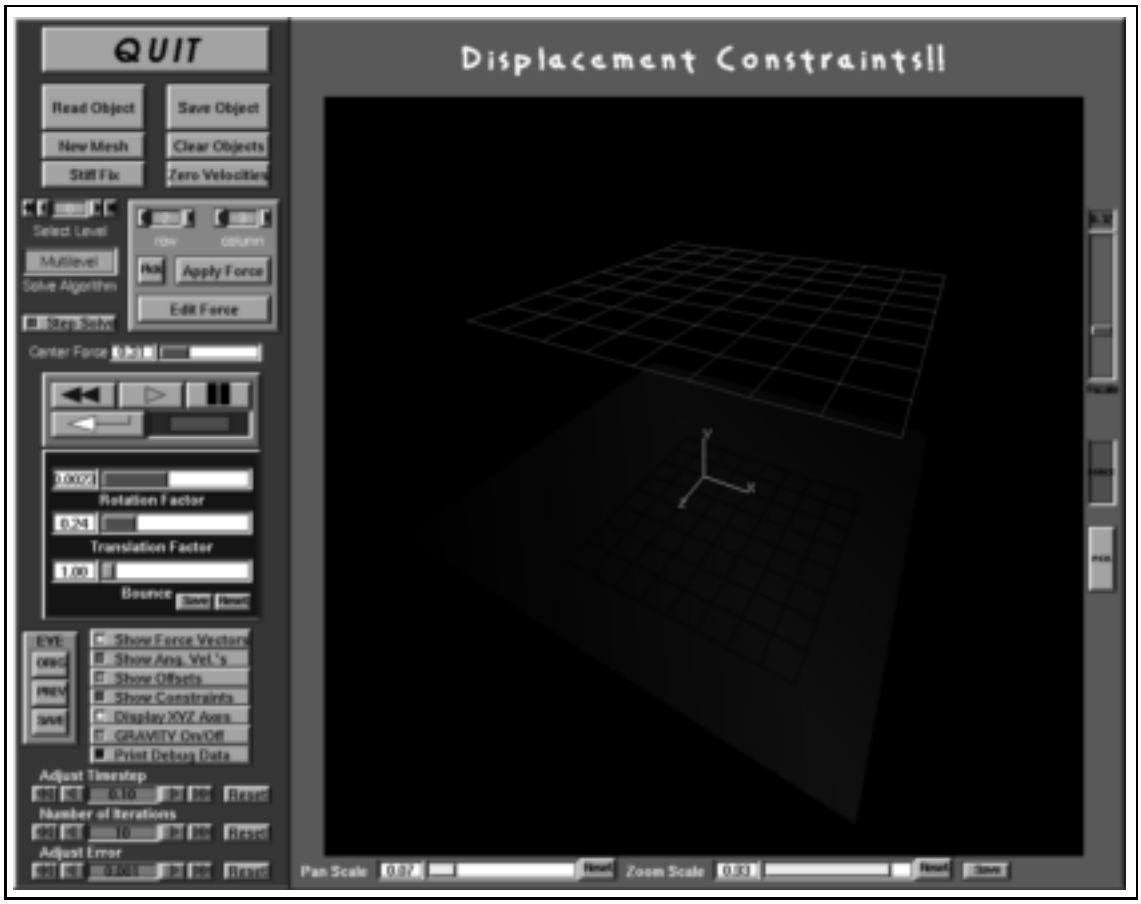

Figure C.1: *The Displacement Constraints Implementation Interface*

floor and the X, Y and Z axes. While the mouse tracker is in this window, the animator may adjust the view by zooming in or out (middle mouse button) or panning left and right or up

and down (left mouse button). The responsiveness of the changing view (pan and zoom) to the mouse tracker can be controlled by the sliders shown in figure C.2(a). The current settings of these sliders can be recorded by clicking on the "save" button. The right mouse button can be used to apply a force (whose direction is defined by the change in position of the mouse) to the mesh. For visual clarity, the length of the displayed force vectors can be scaled by the slider shown in figure C.2(c). When in "pick" mode (set by the button shown in figure C.2(b)), the right mouse may also used to pick a point on the mesh.

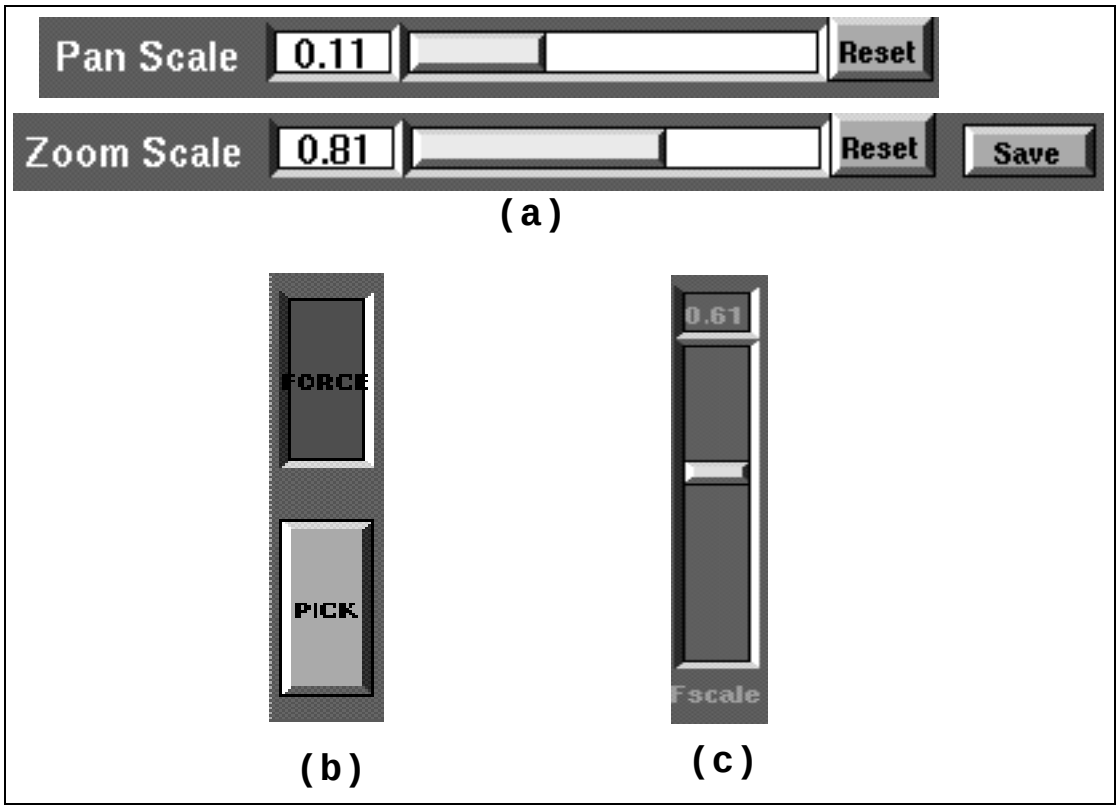

Figure C.2: *Miscellaneous Viewing Controls*

The panel shown in figure C.3 is used to read/save objects from/to a file, create a new objects (this pops up the panel shown in figure C.4), or clear the current objects. The "Stiff Fix" button adds two cross segments to the coarsest level of a mesh object. The "Zero Velocities" button initializes to zero the linear and angular velocities of all line segments in all levels of a mesh. The "Quit" button is specially designed to be self explanatory!

Figure C.3: *The Main Panel*

New mesh objects can be created using the input panel in figure C.4. The size of the (square) mesh is set by the "Mesh Size" counter. The length and mass of each line segment in the mesh are set by the respective counters. Each line segment can also be assigned an initial linear and angular velocity using their respective input fields.

The panel shown in figure C.5 is used to perform several tasks. The "Select Level" counter is used to select a certain level of the current mesh to be displayed. The "Solve Algorithm" menu can be used to select either the multilevel algorithm or the original displacement constraints algorithm. The "Step Solve" button activates the panel shown in figure C.10 which allows the animator to directly control each step of the solution algorithm. The "Center Force" slider sets the portion of a torque acting on a segment that is applied to the center of the segment to give it linear velocity. The "row" and "column" counter can be used to select a particular mesh point. The "Pick" button activates the picking functions. The "Apply Force" button activates the force on the selected mesh point for the next timestep. The "Edit Force" button activates the panel shown in figure C.11, which is used to edit the various force parameters.

The panel shown in figure C.6 provides various information display utilities. The "EYE" box shown on the left is used to "save" the current eye position, reset the eye to the previously saved position and reset the eye to its original position. The "Show Force Vectors", "Show

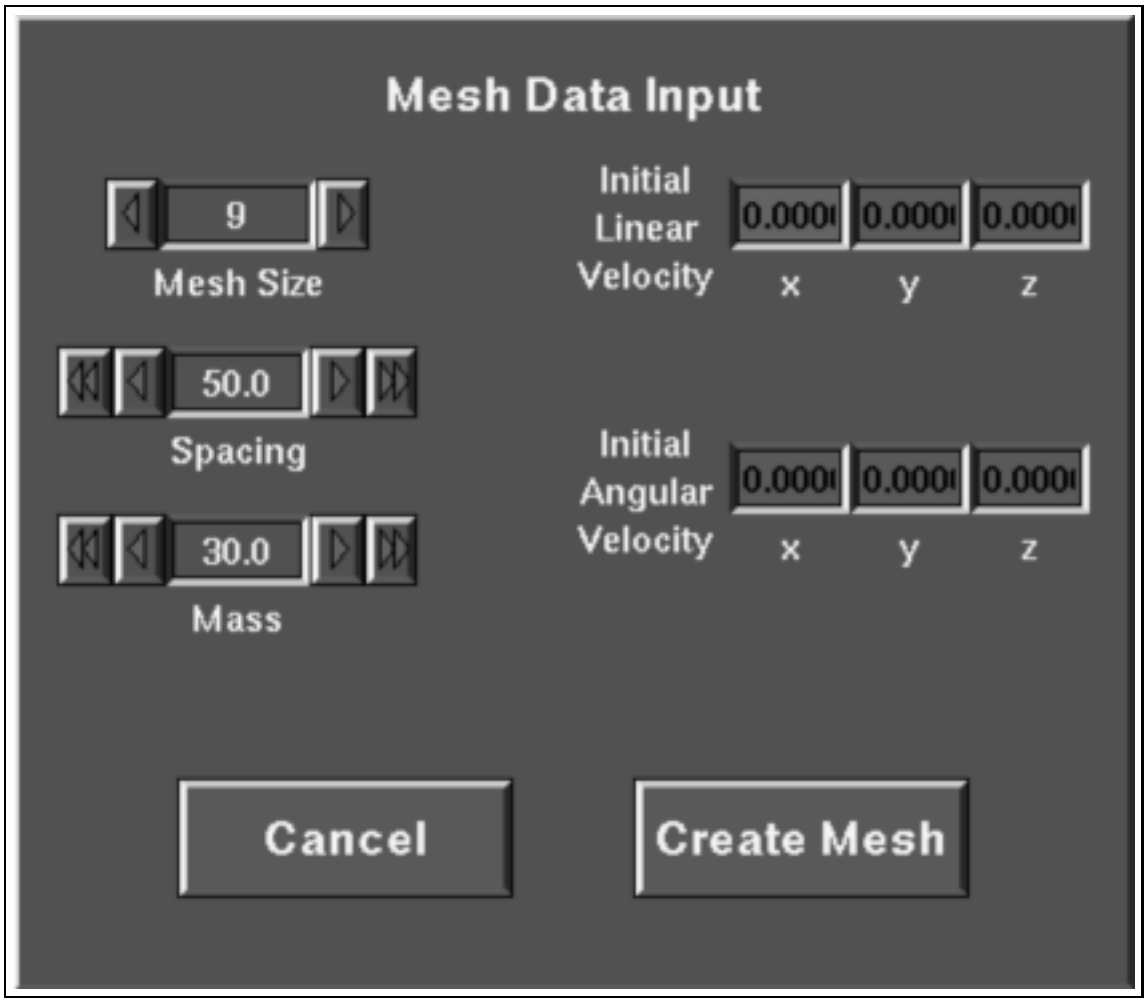

Figure C.4: *Create Mesh Panel*

Ang. Vel.'s" and "Show Offsets" buttons display/turn off the force vectors, angular velocity vectors and offset vectors, respectively, for each line segment displayed. When the "Show Constraints" button is activated, the screen is updated after each iteration of the constraint processing algorithm (rather than just once per timestep). The "Display XYZ Axes" button displays/turns off the coordinate axes. Gravity can be turned on or off using the "GRAVITY On/Off" button. When the "Print Debug Data" button is activated the error and number of iterations after each timestep are displayed.

Figure C.7 show the play panel. The play button (top row, middle button) initiates the animation. The pause button (top row, right button) and stop button (bottom row, right

Figure C.5: *Various Picking and Selection Controls*

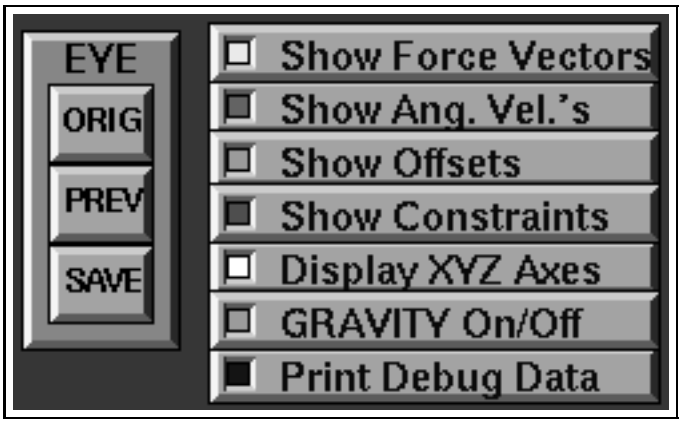

Figure C.6: *The Show Information Panel*

button) perform the obvious tasks. The restart button (bottom row, left button) restarts the animation from the initial position of the objects. The reverse button (top row, left button) causes the animation to run in reverse. The animation can be advanced frame by frame by activating both the play and pause buttons and clicking on the play button. Each click on the play button will advance the animation by one timestep.

The counters shown in figure C.8 were described in chapter 3. Figure C.9 show the rotation factor, translation factor and bounce sliders. The function of the rotation factor and translation factor sliders were described in chapter 3. The bounce slider scales the force applied to a segment

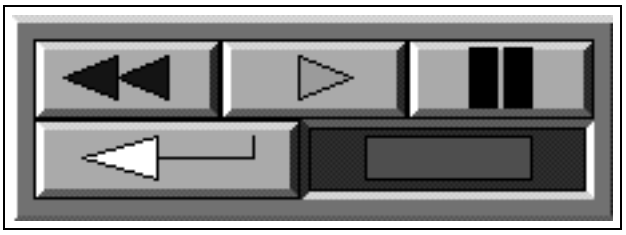

Figure C.7: *The Play Panel*

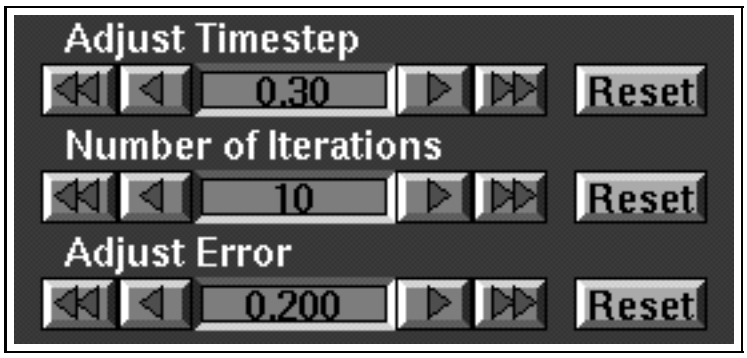

Figure C.8: *Timestep, Error and Iteration Counters*

when it hits the floor.

The panel shown in figure C.10 is used to execute each frame of the animation step by step. With such a utility, the animator can visualize the operations that are taking place during each step of the multilevel solution algorithm. The counter on the left is used to select a particular level of the mesh. The "Forces" button calculates all external forces that are acting on the mesh. The "Restriction" button restricts to the next coarser level, and displays the next coarser level. The "Dynamics" button executes the dynamics phase for one timestep. The "Solve Constraints" button initiates the original displacement constraints algorithm to solve the constraints at the current level. The "Pass Solutions" button performs the prolongation operation from the current level to the next finer level, and displays the next finer level.

The edit force panel shown in figure C.11 is used to set the various force parameters. The "Force Values" input fields define the orientation of the force vector. The magnitude of the force is set by the "Force Magnitude" input field. The "Save Force" button writes the current force values to a file. The "Reset Force" sets the force to the previously saved force values. The "Filter Fraction" input field sets the percentage of a force that is filtered to the neighbouring

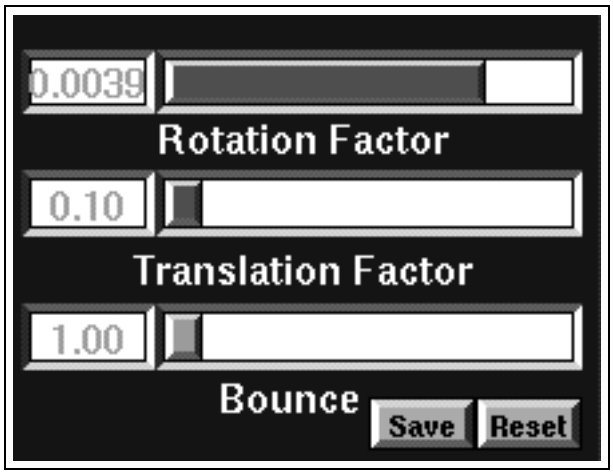

Figure C.9: *Rotation, Translation and Bounce Sliders*

mesh points. The percentage of the original force that is distributed at each level can be set by editing the respective level input fields. The sum of these fields is shown in the "SUM" box.

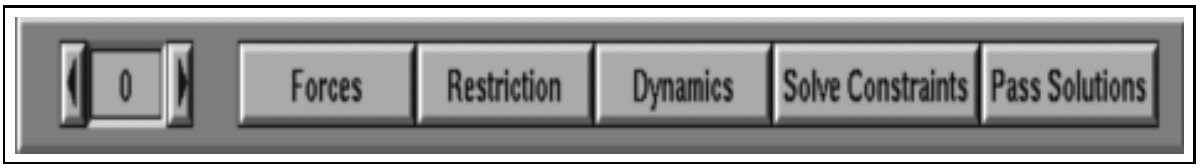

Figure C.10: *The Step Solve Panel*

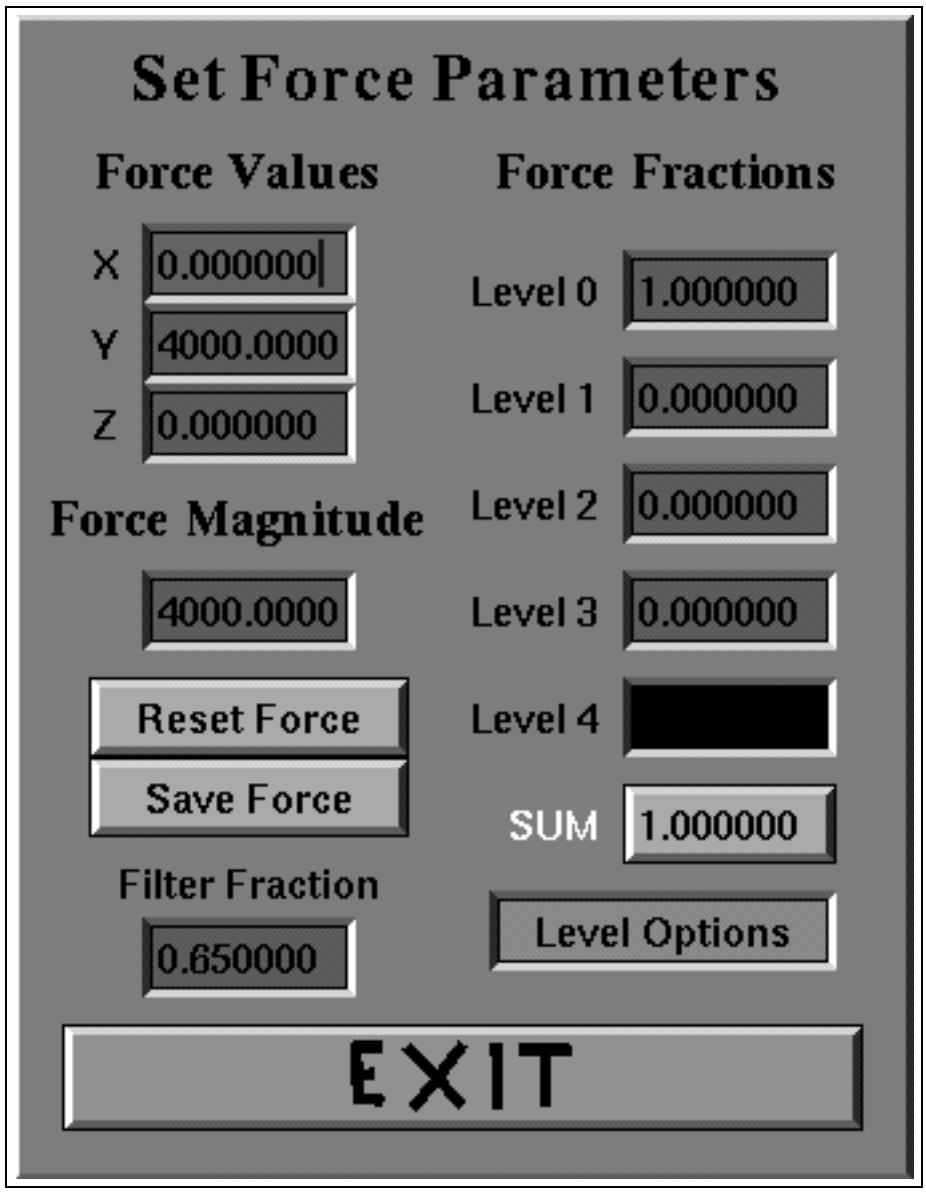

Figure C.11: *The Edit Force Panel*

Bibliography

[1] Armstrong, W., Green, M., and Lake, R. Near-Real-Time Control of Human Figure Models. *IEEE Computer Graphics and Applications*, pp. 52–61, June 1987.

[2] Armstrong, W. and Green, M. The Dynamics of Articulated Rigid Bodies for Purposes of Animation. *The Visual Computer*, pp. 231–240, 1985.

[3] Baciu, G. Personal Communication. Department of Systems Design, University of Waterloo, February, 1993.

[4] Badler, N. I., Barsky, B. A., and Zeltzer, D. *Making the Move: Mechanics, Control, and Animation of Articulated bodies.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[5] Badler, N. I., Manoochehri, K. H., and Walters, G. Articulated Figure Positioning by Multiple Constraints. *IEEE Computer Graphics and Applications*, pp. 28–38, June 1987.

[6] Baraff, D. Rigid Body Simulation. In *SIGGRAPH '92, Course 23 notes: An Introduction to Physically Based Modeling*, pp. H1 – H30, Chicago, Illinois, July 1992.

[7] Baraff, D. and Witkin, A. Dynamic Simulation of Non-penetrating Flexible Bodies. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2), pp. 303–308, July 1992.

[8] Barr, A. H. Global and Local Deformations of Solid Primitives. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3), pp. 21–30, July 1984.

[9] Barzel, R. and Barr, A. H. A Modeling System Based On Dynamic Constraints. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4), pp. 179–188, August 1988.

[10] Briggs, W. L. *A Multigrid Tutorial.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[11] Bruderlin, A. and Calvert, T. W. Goal-Directed, Dynamic Animation of Human Walking. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3), pp. 233–242, July 1989.

[12] Bruderlin, A. and Calvert, T. Interactive animation of personalized human locomotion. In *Graphics Interface '93*, pp. 17–23, May 1993.

[13] Cachola, D. G. and Schrack, G. F. Modeling and Animating Three-Dimensional Articulated Figures. In *Graphics Interface '86*, pp. 152–157, Vancouver, B.C., May 1986.

[14] Calvert, T. and Chapman, J. Aspects of the Kinematic Simulation of Human Movement. *IEEE Computer Graphics and Applications*, pp. 41–50, November 1982.

[15] Chang, P. H. A Closed-Form Solution for Inverse Kinematics of Robot Manipulators with Redundancy. *IEEE Journal of Robotics and Automation*, RA-3(5), pp. 393–403, October 1987.

[16] ElMaraghy, H. A. Kinematic and Geometric Modeling and Animation of Robots. In *Graphics Interface '86*, pp. 15–19, Vancouver, B.C., May 1986.

[17] Featherstone, R. The Calculation of Robot Using Articulated-Body Inertias. *International Journal of Robotics Research*, 2(1), pp. 13–30, 1983.

[18] Forrest-Barlach, M. and Babcock, S. M. Inverse Dynamics Position Control of a Compliant Manipulator. *IEEE Journal of Robotics and Automation*, RA-3(1), pp. 75–83, February 1987.

[19] Gascuel, J.-D. and Gascuel, M.-P. Displacement Constraints: A New Method for Interactive Dynamic Animation of Articulated Bodies. In *EUROGRAPHICS '92 Proceedings*, 1992.

[20] Gascuel, M.-P. An Implicit Formulation for Precise Contact Modeling between Flexible Solids. In *Proceedings of SIGGRAPH 93, Annual Conference Series, 1993*, pp. 313–320, August 1993.

[21] Gascuel, M., Verroust, A., and Puech, C. A Modeling System for Complex Deformable Bodies Suited to Animation and Collision Processing. *Journal of Visualization and Computer Animation*, 2(3), August 1991.

[22] Girard, M. Interactive Design of 3D Computer-Animated Legged Animal Motion. *IEEE Computer Graphics and Applications*, pp. 39–51, June 1987.

[23] Girard, M. and Maciejewski, A. Computational Modeling for the Computer Animation of Legged Figures. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3), pp. 263–270, July 1985.

[24] Goldstein, H. *Classical Mechanics.* Addison-Wesley, Reading, MA, second edition, 1983.

[25] Gould, H. and Tobochnik, J. *An Introduction to Computer Simulation Techniques.* Addison-Wesley, Reading, MA, 1988.

[26] Hendrix, Jimi, Modeling Simultaneous Actions and Continuous Processes. *Artificial Intelligence*, 4, pp. 145–180, 1973.

[27] Hoffmann, C. M. and Hopcroft, J. E. Simulation of Physical Systems from Geometric Models. *IEEE Journal of Robotics and Automation*, RA-3(3), pp. 194–206, June 1987.

[28] House, D. H., Breen, D. E., and Getto, P. H. On the Dynamic Simulation of Physically-Based Particle-System Models. In *EUROGRAPHICS '92 Proceedings*, 1992.

[29] Isaacs, P. and Cohen, M. Controlling Dynamic Simulation with Kinematic Constraints, Behaviour Functions and Inverse Dynamics. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4), pp. 215–224, July 1987.

[30] Isaacs, P. and Cohen, M. Mixed Method for Complex Kinematic Constraints in Dynamic Figure Animation. *The Visual Computer*, 2(4), pp. 296–305, December 1988.

[31] Korein, J. U. and Badler, N. I. Techniques for Generating the Goal-Directed Motion of Articulated Structures. *IEEE Computer Graphics and Applications*, pp. 71–81, November 1982.

[32] Lathrop, R. Constrained (closed-loop) Robot Simulation by Local Constraint Propagation. In *IEEE International Conference on Robotics and Automation*, pp. 689–694, San Francisco, 1986.

[33] McCormick, S. F. *Multilevel Adaptive Methods for Partial Differential Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[34] Metaxas, D. and Terzopoulos, D. Dynamic Deformation of Solid Primitives with Constraints. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2), pp. 309–312, July 1992.

[35] Metaxas, D. and Terzopoulos, D. Shape and Nonrigid Motion Estimation through Physics-Based Synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6), pp. 580–591, June 1993.

[36] O'Rourke, J. and Badler, N. I. Model-Based Image Analysis of Human Motion Using Constraint Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(6), pp. 522–535, November 1980.

[37] Pentland, A. and Williams, J. Good Vibrations: Modal Dynamics for Graphics and Animation. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3), pp. 215–222, July 1989.

[38] Platt, J. C. and Barr, A. H. Constraint Methods for Flexible Models. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4), pp. 279–288, August 1988.

[39] Press, W., Flannery, B., Teukolsky, S., and Vetterling, W. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1988.

[40] Shoemake, K. Animating Rotation With Quaternion Curves. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3), pp. 245–254, July 1985.

[41] Szeliski, R. and Tonnesen, D. Surface Modeling with Oriented Particle Systems. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2), pp. 185–194, July 1992.

[42] Terzopoulos, D. and Fleischer, K. Deformable Models. *The Visual Computer*, 4, pp. 306–331, December 1988.

[43] Terzopoulos, D. and Fleischer, K. Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4), pp. 269–278, August 1988.

[44] Terzopoulos, D. and Metaxas, D. Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7), pp. 703–714, July 1991.

[45] Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. Elastically Deformable Models. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4), pp. 205–214, July 1987.

[46] Terzopoulos, D., Platt, J., and Fleischer, K. Heating and Melting Deformable Models (from Goop to Glop). In *Graphics Interface '89*, pp. 219–226, June 1989.

[47] Terzopoulos, D. and Witkin, A. Physically-Based Models With Rigid And Deformable Components. In *Graphics Interface '88*, pp. 146–154, June 1988.

[48] Tonnesen, D. Spatially Coupled Particle Systems. In *SIGGRAPH '92, Course 16 notes: Particle System Modeling, Animation, and Physically Based Techniques*, pp. 4–2 – 4–21, Chicago, Illinois, July 1992.

[49] van Overveld, C. A Technique for Motion Specification in Computer Animation. *The Visual Computer*, 6, pp. 106–116, 1990.

[50] van Overveld, C. An Iterative Approach to Dynamic Simulation of 3-D Rigid-Body Motions for Real-Time Interactive Computer Animation. *The Visual Computer*, 7, pp. 29–38, 1991.

[51] Wilhelms, J. Virya - A Motion Control Editor for Kinematic and Dynamic Animation. In *Graphics Interface '86*, pp. 141–146, Vancouver, B.C., May 1986.

[52] Wilhelms, J. Toward Automatic Motion Control. *IEEE Computer Graphics and Applications*, pp. 11–22, April 1987.

[53] Wilhelms, J. Using Dynamic Analysis for Realistic Animation of Articulated Bodies. *IEEE Computer Graphics and Applications*, pp. 12–27, June 1987.

[54] Wilhems, J. and Barsky, B. Using Dynamic Analysis to Animate Articulated Bodies as Humans and Robots. In *Graphics Interface '85*, pp. 97–104, Montreal, Quebec, May 1985.

[55] Witkin, A. and Welch, W. Fast Animation and Control of Nonrigid Structures. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4), pp. 243–252, August 1990.

[56] Zeltzer, D. Motor Control Techniques for Figure Animation. *IEEE Computer Graphics and Applications*, pp. 53–59, November 1982.

[57] Zeltzer, D. Towards an Integrated View of 3-D Computer Animation. *The Visual Computer*, pp. 249–259, 1985.