PHYSICS-BASED ANIMATION AND CONTROL OF FLEXIBLE CHARACTERS

by

Petros Faloutsos

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Physics-Based Animation and Control of Flexible Characters
Master of Science degree, 1995
Petros Faloutsos
Graduate Department of Computer Science
University of Toronto

Abstract

This work deals with the animation and control of flexible and active characters. These are characters whose rigidity and shape can vary in accordance with the desired aesthetic result and goal of the motion. In our approach characters change shape and learn to move using a set of user-defined deformation models, implemented using free-form deformations. Restricting the possible deformations to those which can be constructed by the set of pre-defined deformation models allows for both efficient simulation and predictable results. The interaction with the environment is physics-based and it is implemented using Lagrangian dynamics. Lagrangian dynamics and the use of parameterized deformations lead to a compact formulation of the equations of motion. Using this physical framework, the control problem can be addressed using methods that have been developed for controlling the motion of simulated articulated figures. In general, our work combines key-framing, physics-based animation techniques, control and motion synthesis for flexible characters.

Acknowledgements

First I would like to thank my parents for their support and for teaching me things that no educational institute could ever teach me. The other members of my family Michalis, Christos and Maria provided support, guidance and love which is gratefully appreciated. Special thanks to Michalis for useful comments and help with debugging.

This work could not have been completed without the encouragement, support, enthousiasm and expert guidance of my supervisors Michiel van de Panne and Demetri Terzopoulos. Working with both of them is a priviledge and a valuable educational experience. Thanks also to the people of the DGP lab of the University of Toronto for their cooperation and good will.

Many thanks to my roomate, Karen Block, whose presence fills the house with energy and joy.

I would like to thank the volley-ball compagny for all the fun times we had together, Michalis, Piotrek, Joanna, Danka, David and Richard "the bear". Special thanks to Piotrek for all those "one beer in Victory Cafe" nights.

I would also like to thank my friends Dimitra Vista, Spiros, George, Vassilis, Dimitris, Fattaneh, Maria-Claudia, Wayne, Dimitra Marangkozis, Minas, Laurie Harris and Laurie Paquette; their friendship makes life better for me. Special thanks to Dimitra Marangkozis, Minas Spetsakis, Dimitra Vista, Spiros Mancoridis for their help and support during my first days in Canada.

Many thanks to my old friends in Greece for all the good moments we shared together, Giannis, Aleksis, Thanos, Kostas Tsimidakis, Kostas Kotsifakis, Sofia, Dimitra, Giwrgos, Sokratis, Andreas, Romina, Peny Anesti, Peny Strapatsaki and Gwgw. Special thanks to Aspasia for all those letters and to my cousin Aleksis for all the moments we shared together and for all his driving to the airport of Athens.

Kathy Yen, the graduate administrator, deserves a great THANK YOU for her friendship, help, support and efficiency. Life would have been much more difficult without her. Kathy can deal with bureaucracy like nobody else can.

I would also like to thank the DCS librarians for their efficiency and their help.

Lastly, many thanks to the University of Toronto for its amazing international educational environment.

Contents

List of Figures

Chapter 1

Introduction

Animation has been very popular, long before the existence of computer graphics. Walt Disney has turned cartoon animation into a wonderful art form and has produced many timeless classics. In the earliest form of cartoon animation all aspects of the work are under the direct control of the animator. Motion is created by drawing objects in different positions over successive frames such that when played in sequence, they elicit the perception of motion. The qualities of Walt Disney animation are such that the motion is smooth, continuous, and characterized by "squash-and-stretch" effects [26] which serve to give the character life.

The subsequent development of computer graphics has provided tools to assist in the creation of character animations. We will distinguish between the two main animation techniques: *key-framing* and *physics-based animation*.

Key-framing is a traditional animation technique well-adapted for use with computers. The animator produces a small number of *key-frames* and an interpolation between them is generated to produce a perception of continuous motion. A variety of interpolation methods can be employed. Typically cubic spline interpolation can be used. Reference [15] presents a nice overview of the key-framing technique along with pointers to the relative literature.

The physics-based approach uses the laws of physics in order to simulate the character's motion and interaction with the environment. Recent advances in computer hardware permit animators to use computationally expensive techniques for physics-based simulation and realistic animation of living characters.

Character animation can be divided into two categories with respect to the nature of the animated characters: cartoon animation and human-animal animation. The latter is concerned with trying to simulate as accurately as possible the motion and biomechanics of an animal. Much of the previous work here deals with articulated creatures, whose biomechanics are based on the presence of a skeleton [7, 52, 55, 54]. Actuators are attached on the skeleton and they are responsible for the creation of a desired motion. Researchers have addressed the simulation of animals with different levels of physical accuracy. An extensive biomechanical fish model, capable of producing rich and realistic motion, is developed in [51]. It should be noted some animals such as snakes lack the problem of balance. The animation and control of humans and many animals require the additional problem of balance to be solved. Small errors in the control parameters accumulate and eventually result in the loss of the animal's balance. In general we can point out two main difficulties in the animation of animals: a) developing a suitable biomechanical model and b) producing realistic motion using the actuators of the model.

Cartoon animation is concerned with the simulation of realistic and unrealistic characters who normally behave in a physically correct way, but they are sometimes required to perform non-physical motions and tasks. Here, we must distinguish between a number of different approaches to cartoon animation. First, there are animators who try to make their characters as realistic as possible. For example the well-known school of Japanese animation focuses mostly on the accurate representation of the real-world, accurate in terms of motion and interaction with the environment. This kind of cartoon animation mimics that of movies with real actors. Second, some animators use cartoons not for visual pleasure but as a means to express their ideas. Consider for example the "Simpsons" series which focuses more on what the characters say and do than how they are graphically presented. Lastly there is the approach where animators try to create an appealing visual result incorporating appealing but not realistic visual effects. Typical examples here are the Walt Disney and Loony-toon animations. In both we often see inanimate objects coming to life or animals animated in a human-like way. In Loony-toon animation, funny violations of the physical laws, like the ones presented in Appendix B, are often presented. The work presented in this document deals mainly with the kind of animation presented in the last approach. We are interested in producing appealing and smooth motion in an approximation of Walt Disney's world. For the rest of the document the term cartoon or flexible character animation is used in this context.

1.1 Motivation

Cartoon animation is a fascinating task mostly because it is the product of an animator's imagination. Appealing exaggerations of the physical world such as giving life to non-living characters (for example, when animating teapots or tables), assigning animals human qualities and changing the physical laws that govern the materials and motions of the characters can all be desirable visual effects. Computer graphics can provide tools that can significantly assist animators in this task. In recent years a significant body of work has been developed for dealing with the physics-based simulation of rigid and deformable models. Active flexible models of snakes and worms are dealt with in [30] and realistic fish models are dealt with in the recent work of [51]. The problem of controlling active objects has been extensively addressed ([7, 30, 51, 52, 54, 55, 7]). This problem focuses on designing controllers that make the characters perform specific motions. Techniques to automate the design of the controllers have been investigated [52, 54, 55].

This work explores a new method for the physics-based animation of flexible characters. There are a number of different formulations for modeling flexible characters, animating them and for automatically discovering suitable motions. This works aims to combine techniques from all the above tasks in a unified and efficient framework suited for cartoon animation. We believe that a complete and useful system for cartoon animation should integrate key-framing and physics-based techniques. Key-framing for allowing cartoon characters to perform unnatural tasks (see Appendix B). Physics-based simulation models the character's interaction with the environment in a physical way, thus ensuring a realistic result. Physics-based simulation has the disadvantage that the animator looses some of the control over the animation. The advantage is that the animator, using physics-based simulation, does not have to key-frame significant parts of an animation. In conclusion both techniques have advantages and disadvantages thus we believe that a complete system should provide both.

In order to deal with active characters, a notion of *active control* needs to be incorporated. A pure physics-based simulation consists only of a *passive dynamic simulation*. The latter involves the simulation of inanimate objects which respond only to *external* forces. *Active dynamics* implies the simulation of creatures that can move using internal muscles and actuators. *Active control* implies that a way to control the internal functionality of

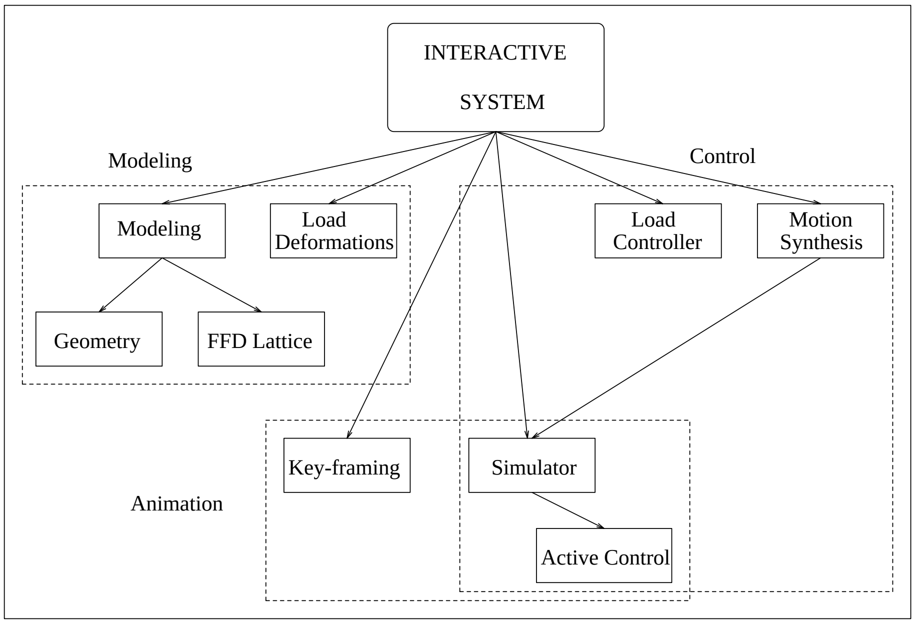

Figure 1.1: Overview of the system

the object is incorporated in the physics-based simulation. Active control can be of great help with respect to animating living characters, given the fact that living characters exhibit autonomy of movement. Furthermore, the control can be enhanced by incorporating techniques for automatic discovery of motion or *motion synthesis*. The latter refers to techniques like those presented in [19, 55], to automatically explore different ways of locomotion for living characters.

Our system implements techniques for all of the above different tasks (key-framing, passive dynamic simulation, active control and motion synthesis), providing a complete animation framework for flexible characters. Figure 1.1 presents a graphical scheme of the concepts implemented in our system.

It is worth noting that the three components of Figure 1.1 show the independence of the modeling, animation and control in the system. The animator can interactively revise any of these parts without affecting the others.

The techniques used in this work are briefly presented in the next section. More details are given in the chapters that follow (see overview section).

1.2 Techniques Used in This Work

A brief presentation of the techniques used in the various parts of our system (Figure 1.1) is presented. In this section we assume that the reader has some knowledge of the techniques presented. A detailed description of all the concepts involved follows in subsequent sections of the document.

- **Modeling.** For the modeling of the deformations we use *free-from deformation* [39]. It is an implicit technique which deforms the space around an object. The object follows the deformation of the space in a predefined way using a spline formulation. Objects are thus deformed implicitly.

- **Animation.** For key-framing we choose to implement linear interpolation between frames. For the physics-based simulation we use Lagrangian dynamics. The Lagrangian approach allows for compact and efficient formulation. To represent the orientation of an object in space we use rigid body parameters (translation, rotation). To represent the shape of the flexible object we use deformation parameters, one for each user-defined deformation mode. The deformations are applied within a local coordinate frame. The interaction with the environment involves collision detection and resolution. Currently we detect collisions only between one object and the ground. The collision model is based on a *penalty method* (§4.9) and friction is implemented using a Coulomb model.

- **Control.** We employ two levels of control. The low-level control consists of a set of PD-controllers that produce the internal forces. There is one PD-controller for each mode of deformation. The high-level control is based on the *pose control* method [55, 56] (see Chapters 3 and 5). A pose graph drives the PD-controllers in order to produce desired modes of locomotion. For motion synthesis, this control scheme is used in conjunction with a simulated annealing parameter search procedure. The latter refines a controller iteratively using forward simulation and evaluation of the motion produced using cost functions (§3.2.5 and §5.1).

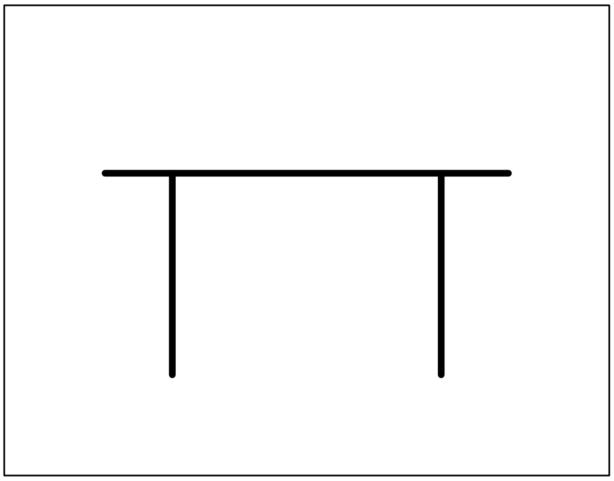

Figure 1.2: The table

1.3 An Example

To illustrate the kind of characters and the kind of motions within the scope of this work, the example of an animate table is used. In Figure 1.2, the initial shape (geometric model) of the character is shown. It is a 2D model consisting of three straight line segments. This character is going to be used as an example through out this document.

Having designed the model, it is the animator's responsibility to design a set of deformation modes that give life to the naturally lifeless character. An example of what kind of motion an animator can give to a character of this kind is shown in Figure 1.3. The motion begins using key-framing run towards the edge of the cliff, followed by a passive simulation. The key-framed motion uses two deformations, a bend and a shear. The simulated motion uses the same bend deformation and a vertical squash deformation.

We suggest that the reader keep this example in mind while reading the rest of this document.

1.4 Contributions

In this work the special nature of cartoon characters is exploited in such a way that techniques for the animation and control of articulated figures are applied to our flexible models. The formulation presented defines both the motion and the nature (elasticity) of the material of the character in a unified way. For the same character model, different deformation modes can be used for different tasks of the character, giving more power to the animator.

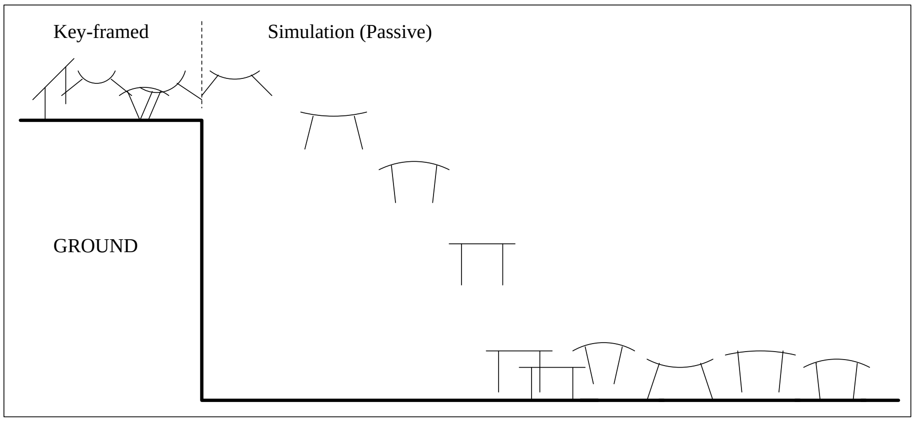

Figure 1.3: Table jumping off a cliff

In other words the model is not bound to certain deformation modes; the latter can be chosen at will.

Physics-based simulation needs a *dynamic model* of the character. By that term we mean a geometric model capable of reacting to forces. In most previous works, the geometric model defines the functionality of the character. In our work, the different ways a character can move (deformation modes) are defined independently of the character's geometric model. As shown later, this is achieved using parameterized free-form deformations. In particular an object is capable of deforming in a number of user-defined modes. The deformations are implicitly applied to the object using the free-form deformation technique. The implicit nature of the technique makes the undeformed geometric model independent of the physics-based formulation. This is motivated by the fact that a cartoon character can change rigidity in time according to the situations that arise and second because characters such as teapots that come to life do not have well-defined actuators. When animating a lifeless character like a teapot, there are different ways with which the object can locomote since any part of the object can form an actuator. In such cases the animator should be able to change the motion configuration without altering the model of the character. This is what the present system allows for. The set of the character's deformations is not bound to any actuator topology.

The main contribution of this work is a method for the active, physics-based animation of flexible characters using user-designed deformation modes. We can summarize the main features and contributions of our work as follows:

- **Independence between object model and motion-deformations.** The deformation modes that a character is capable of following are defined independently of the associated geometric model. Different objects can use the same deformations, or equivalently an object can be equipped with different deformations throughout a simulation.

- **Use of the same formulation for motion and material deformation.** The motion and the flexibility of the character are both defined in a unified way. This can be both bad and good depending on the specific case. For example, when the character should exploit its elastic properties in order to move, this unified formulation is convenient. In addition, the free-form deformation formulation allows the animator to use an arbitrary number of free-from deformation components (lattices) according to the desired level of detail. This is explained more in subsequent sections.

- **Use of existing articulated figure techniques for dynamics and control.** Using parameterized free-form deformations we derive a formulation that in principle is equivalent to the articulated figure case. Thus, existing techniques, like the ones described in [32, 54, 55], can be used for our flexible models.

- **Efficiency.** Restricting the allowable deformations to those which can be constructed from the user-defined modes of deformation, allows both for efficient and predictable results.

- **Combination of key-framing, passive dynamics and active control.** Key-framing remains an essential technique for cartoon animation since cartoon characters are capable of doing non-physical tasks. Passive dynamics can reduce the work required from the animator and at the same time help the animator to produce realistic results when desired. Active control can help the animator to easily experiment with different modes of a character's motion or even automatically discover different ways of locomotion.

1.5 Overview

The remainder of the document is organized as follows. **Chapter 2** presents related-previous work on flexible dynamic models. **Chapter 3** discusses previous work on control and motion synthesis issues. **Chapter 4** presents our dynamic-flexible formulation. **Chapter 5** analyses the way this formulation can be combined with existing rigid body control and motion synthesis techniques. **Chapter 6** provides general discussion, conclusions and possible extensions to the current work. The chapters that deal with background and related work focus on the techniques used in the implemented system.

Appendix A presents some detailed calculations involving the dynamic model in a simple case. Appendix B presents the *Cartoon Laws of Physics*, and Appendix C illustrates some typical input used in the interaction between the user and the system. Mathematical formulations are presented in their most general form.

Chapter 2

Dynamic Simulation of Flexible Models

Many approaches have been proposed for the physical simulation of systems of bodies. Both the case of rigid bodies and the case of flexible bodies have been addressed. Before we analyze the concepts involved in a physics-based simulation, some terms need to be defined. The *geometric model* refers to a model that captures the shape of an object. The *dynamic model* refers to a model that approximates the physical properties of an object. It integrates the object's geometry with its physical properties such as mass distribution.

The physical simulation of rigid and flexible objects involves three major problems.

1. Formulation of the equations of motion: The dynamic model of the objects is the basis of a physics-based approach. It is desirable to come up with an efficient but accurate formulation. The dynamic model defines the parameters that control the state of an object and consequently it is a part of the system of equations that govern the motion of the object. The number of parameters usually defines the dimension of the linear system that must be solved in order to compute the position of the simulated objects in time. A model with a small set of state parameters is preferable. Included in the problem of dynamic modeling is that of resolving constraints on the various parts of a compound object in order to keep them together.

2. Collision detection: The interaction with the environment must be modeled in a consistent and physically correct way. The simulation must be able to detect when the various bodies are in contact or collide with each other or with obstacles in the

environment. Non-interpenetration constraints should be enforced.

3. Collision resolution: Once a collision occurs the simulator must resolve it. The problem is complex for flexible bodies since collisions result in deformations. Deformations change the shape of the object and as a result the physical parameters of the object change (center of mass, moments of inertia). The most difficult problem is to accurately calculate contact surfaces when flexible models collide. The surfaces of the objects change continuously and the flexible nature of the objects introduce long (with respect to the time step) contact periods between colliding objects.

The rest of this chapter presents an overview of existing solutions for simulating the motion of flexible models. The next section presents background information concerning Lagrangian dynamics.

2.1 Lagrangian Dynamics

The Lagrangian dynamics formulation [18, 28, 61] is a formulation of the physical laws of motion. It is a very general formulation and in many cases it results in compact equations. It generalizes the well known Newtonian law $\mathbf{F} = m\ddot{\mathbf{x}}$ into $\mathbf{Q} = \mathbf{M}\ddot{\mathbf{q}}$ where $\mathbf{Q}$ are the generalized forces, $\mathbf{M}$ is the generalized mass matrix and $\ddot{\mathbf{q}}$ is the vector of the generalized accelerations.

The Lagrangian formulation can be derived from *Hamilton's Principle*, a variational principle which can be summarized as follows [28]:

> *Of all the possible paths along which a dynamical system may move from one point to another in configuration space within a specified time interval, the actual path followed is that which minimizes the time integral of the Lagrangian function of the system.*

Configuration space implies the generalized space formed by the parameters that define the state of the system and the Lagrangian function of the system is the difference between its *kinetic* and its *potential* energy. Using variational calculus this problem results in a set of n equations of motion, where n is the number of the state parameters.

Assume that the position of an object in space can be completely defined by n parameters, represented by the vector $\mathbf{q}$. Assume that the object has a volume V and a mass density $\rho(\mathbf{p})$ where $\mathbf{p}$ represents the position of a point in a local coordinate system. The total mass is

$$M = \int_V \rho(\mathbf{p}) d\mathbf{p}$$

The total kinetic energy of the object is

$$E_k = \frac{1}{2} \int_V \rho(\mathbf{p}) \dot{\mathbf{x}}^T(\mathbf{p}) \dot{\mathbf{x}}(\mathbf{p}) d\mathbf{p}$$

The expression for the kinetic energy can be simplified in most cases involving rigid objects, resulting in a formulation that eliminates the triple integral involving total quantities such as the total mass, angular velocity and moments of inertia. This is not possible in the case of flexible objects. The Lagrangian is defined as $L = E_k - E_v$ where E_v is the total potential energy of the object. The Lagrangian formulation consists of the following equations of motion:

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} - Q_k = 0, \qquad k = 1, \ldots, n \tag{2.1}$$

where L is the Lagrangian, q_k is the k-th component of the state vector ($k = 1..n$), Q_k is the total generalized force acting on the object along the q_k generalized coordinate, and the dot indicates a time derivative. In general the above is a linear system of n coupled partial differential equations of the second order with respect to the generalized accelerations $\ddot{\mathbf{q}}$. Its complexity depends on the corresponding object. Since the system is of second order, initial conditions on the generalized positions $\mathbf{q}$ and the generalized velocities $\dot{\mathbf{q}}$ are needed in order to be solved. If the above system has an analytical solution then we can produce analytical expressions for the generalized accelerations, essentially a function $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}(t, \mathbf{q}_{t_0}, \dot{\mathbf{q}}_{t_0})$ where $\mathbf{q}_{t_0}, \dot{\mathbf{q}}_{t_0}$ are the initial conditions. Assuming that this function can be integrated twice, the generalized velocities and positions are calculated as follows:

$$\dot{\mathbf{q}}(t) = \int_{t_0}^{t} \ddot{\mathbf{q}}(t', \mathbf{q}_{t_0}, \dot{\mathbf{q}}_{t_0}) dt' + \dot{\mathbf{q}}_{t_0} \tag{2.2}$$

$$\mathbf{q}(t) = \int_{t_0}^{t} \dot{\mathbf{q}}(t') dt' + \mathbf{q}_{t_0} \tag{2.3}$$

In general the system may not have an analytical solution. This is usually the case for flexible objects. In this case the system is discretized in time and solved numerically with a linear solver with respect to the generalized accelerations $\ddot{\mathbf{q}}$. The latter can be numerically integrated to yield the generalized velocities $\dot{\mathbf{q}}$ and the generalized positions $\mathbf{q}$. For example, using finite differences and explicit Euler integration we arrive at the following expressions:

$$\dot{\mathbf{q}}_n = \frac{\mathbf{q}_n - \mathbf{q}_{n-1}}{\Delta t} \tag{2.4}$$

$$\ddot{\mathbf{q}}_n = \frac{\mathbf{q}_n - 2\mathbf{q}_{n-1} + \mathbf{q}_{n-2}}{\Delta t^2} \tag{2.5}$$

where the indexes $n-2, n-1, n$ indicate successive time steps.

An external force distribution $\mathbf{f}(\mathbf{s})$ in Cartesian coordinates, acting on the surface of the object, can be transformed in generalized coordinates as follows:

$$\mathbf{Q} = \int_{\mathbf{S}} \mathbf{J}^T \mathbf{f}(\mathbf{s}) d\mathbf{s}$$

where T denotes the transpose, $\mathbf{S}$ the surface of the object and $\mathbf{J}$ is the Jacobian matrix defined as

$$J_{ij} = \partial x_i / \partial q_j \quad i = 1 \ldots 3, \;\; j = 1 \ldots n$$

In the above expression $\mathbf{x} = (x_1, x_2, x_3)$ is the position of the infinitesimal surface element $d\mathbf{s}$ of the object, in Cartesian coordinates.

2.2 Model-Based Dynamic Formulations

The geometric modeling of deformable objects can be done in many ways. Models based on polygons, parametric surfaces, generalized cylinders or superquadrics (see [45]), can be used. The requirement is that the model must have a representation that can cooperate with the deformation properties of the object and that can efficiently perform collision detection. A dynamic model must incorporate physical properties along with the geometry of an object. A mass density should thus be defined on top of the object's geometry.

There are two main approaches towards the modeling of deformable objects, the *implicit* and the *explicit* one which are discussed in the following sections.

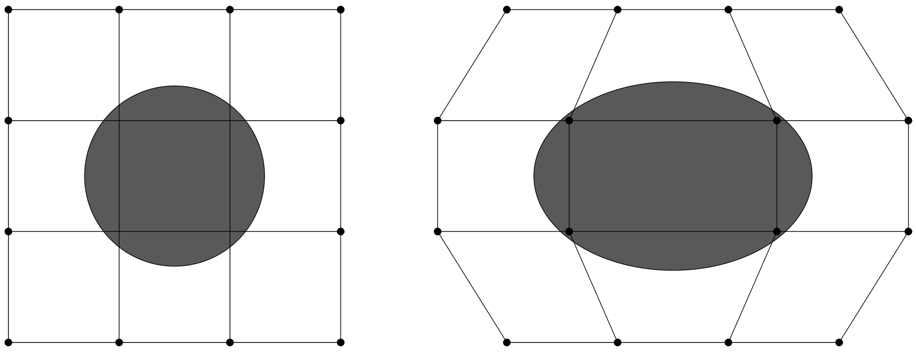

Figure 2.1: 2D free-form deformation of a circle

2.2.1 Implicit Deformable Models

The implicit deformation concept is based on the following idea: the deformation of a set of objects is achieved by deforming the space that contains the objects. The objects follow the deformation of the space in a predefined way, that is a deformation function is applied to them. This function maps a point of the undeformed space onto a point of the deformed one in a well-defined way. In the case of implicit deformations, the specification of the mass densities is generally independent of the way the deformations are specified.

In the next section we describe in detail the free-form deformation technique since it is the one that we use in our work.

2.2.2 Free-form Deformations

Free-form deformations (FFDs) is a technique proposed for use in graphics in [39]. It belongs to the category of implicit deformations. In the case of free-form deformations an implicit deformation is achieved as follows. The space around the objects is defined by a lattice. The objects are embedded in that orthogonal lattice and the deformation is achieved by moving the lattice points. This is illustrated in Figure 1. An object (circle) is embedded in a 2D lattice in Figure 1 (a); the displacement of the lattice points shown in Figure 1 (b) changes the shape of the circle into an ellipse.

The technique requires a function that relates the position of the lattice points with the position of the object points. In mathematical terms, the deformation function performs a mapping $\mathcal{R}^2 \rightarrow \mathcal{R}^2$ for the 2D case. We will explain the deformation mechanism for the

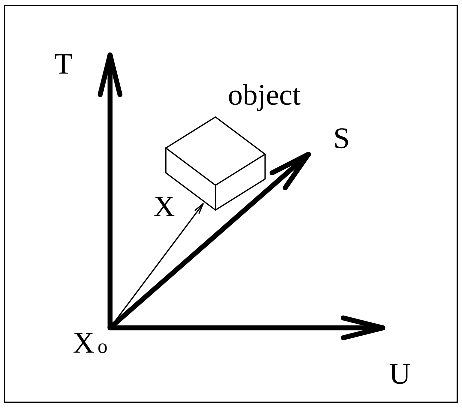

Figure 2.2: Local coordinate system

2D case, considering the lattice as defining a surface on which the object points lie. This surface is defined using a parametric formulation where the points of the lattice constitute the control points of the surface. Manipulation of the lattice points results in a different parametric surface thus a different mapping. The surface can be defined using any kind of basis function, depending on the desired properties of the surface (mapping). For example we can use quadratic or cubic formulations. A common choice is that of bicubic formulations using Bernstein polynomials as the basis functions. If the lattice consists of more than 16 points then a possible choice for the basis function would be the B-spline blending functions. B-spline surfaces have intrinsic C^2 continuity and better local control than Bezier surfaces. As we will see in Chapter 4 we have chosen to use a Bezier formulation.

The mathematical formulation for the general 3D case is the following. Given a lattice $\mathbf{L}$ with dimensions $l \times m \times n$ and a point with local coordinates (s,t,u) the world coordinates of the deformed point are given by the following vector-valued trivariate Bernstein polynomial:

$$\mathbf{X}_{\mathrm{ffd}} = \sum_{\mathrm{i}=0}^{\mathrm{l}} \begin{pmatrix} l \\ i \end{pmatrix} (1-\mathrm{s})^{\mathrm{l-i}}\mathrm{s}^{\mathrm{i}} \left[\sum_{\mathrm{j}=0}^{\mathrm{m}} \begin{pmatrix} m \\ j \end{pmatrix} (1-\mathrm{t})^{\mathrm{m-j}}\mathrm{t}^{\mathrm{j}} \left[\sum_{\mathrm{k}=0}^{\mathrm{n}} \begin{pmatrix} n \\ k \end{pmatrix} (1-\mathrm{u})^{\mathrm{n-k}}\mathrm{u}^{\mathrm{k}} \mathbf{L}_{\mathrm{ijk}} \right]\right] \tag{2.6}$$

where $\mathbf{L}_{ijk} = \mathbf{X}_0 + (i/l)\mathbf{S} + (j/m)\mathbf{T} + (k/n)\mathbf{U}$ are the world coordinates of the lattice points and $(\mathbf{X}_0, \mathbf{S}, \mathbf{T}, \mathbf{U})$ is the local (lattice) coordinate system (Figure 2.2). The local coordinates (s,t,u) of a world space point $\mathbf{X}$ are calculated as follows [39]:

$$s = \frac{(\mathbf{T}\times\mathbf{U})\cdot(\mathbf{X}-\mathbf{X}_0)}{(\mathbf{T}\times\mathbf{U})\cdot\mathbf{S}}, \quad t = \frac{(\mathbf{S}\times\mathbf{U})\cdot(\mathbf{X}-\mathbf{X}_0)}{(\mathbf{S}\times\mathbf{U})\cdot\mathbf{T}}, \quad u = \frac{(\mathbf{S}\times\mathbf{T})\cdot(\mathbf{X}-\mathbf{X}_0)}{(\mathbf{S}\times\mathbf{T})\cdot\mathbf{U}}$$

Free-form deformations have been used extensively as a sculpturing tool [9, 39]. Complex shapes can be achieved by imposing a number of lattices on initial simple shapes and deforming parts of them. For example in [39] a telephone handset is sculpted starting from a smooth roundish bar.

Free-form deformations have several limitations. The manipulation of an object using a lattice is indirect. One must know how the displacement of the lattice points will affect the actual object. The deformations of the object are also restricted by the complexity of the lattice. The more points the lattice has the more degrees of freedom the deformations have. Indirect manipulation also introduces the following problem. If the complexity of the deformation is close to the object's complexity then the deformed object may not be consistent with the undeformed. For example, if a cube is twisted, it may fold onto itself, or a convex polygon may become a concave one.

Fortunately free-from deformations also have features which are useful for our purposes. They can be applied independently of the geometric representation of the object. The space that corresponds to the free-form deformation lattice may contain any number of objects (which may have different geometric representations). The deformations maintain the relative positions of the objects as long as the relative positions of the lattice points are maintained [1]. If two objects initially do not intersect, they do not intersect after an arbitrary deformation as well. By simply choosing an appropriate basis function we can have the desired properties (local control, continuity) and by using a finer lattice we can add more degrees of freedom to our deformations.

Extensions to FFDs are described in [9], [39]. More complex formulations have been explored and are currently used in graphics. Reference [9] introduced the *Extended Free-form Deformations* where the initial lattice of the FFD is not cubic but follows the shape of the object. With this method it is easier to see how the displacement of the lattice points would affect the actual object.

Ways of using implicit deformable models in animation are discussed in the next section.

2.2.3 Animating Implicit Deformable Models

An animation technique for free-from deformations is described in [10]. The objects are embedded in a 3D-lattice that can change over time. The object can be animated, change

[1] By that we mean that the spatial ordering of the lattice points is maintained after the deformation.

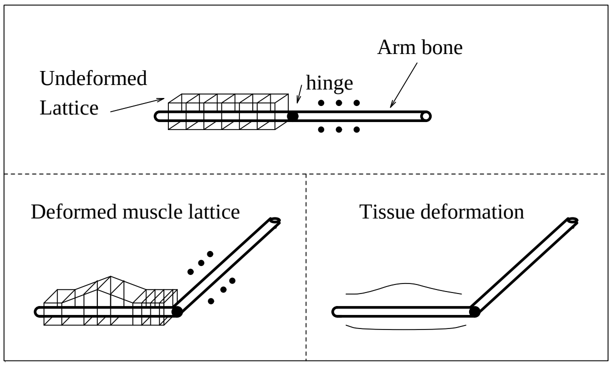

Figure 2.3: Muscle-tissue deformation: The black dots indicate that the lattice continues and covers the whole bone structure.

shape or morph [2] into an other object. The technique is straightforward and based on key-framing. There is no physics-based animation of any kind and the animator is responsible for designing all the deformations to be applied to the lattice during animation.

Physics-based free-form deformations were suggested in [48] but not implemented. In [60], linear implicit deformations are used in a physics-based model. In both [48] and [60] the proposed model is passive; no actuators exist that can produce autonomous motion.

In [7], free-form deformations are used as a model for the dynamic simulation of muscle and fatty tissue deformation. The muscle model consists of a geometry embedded in FFD-lattices whose control points have mass and are interconnected with springs. The lattices are attached to the skeleton in an approximation of the real biological structure, such that skeletal motion deforms the lattices and thus the muscles. The physics-based part is in this case applied to the lattice instead of the actual tissue around the muscle. Figure 2.3 illustrates the free-from deformation muscle-tissue model.

A second approach towards the physics-based animation of flexible objects using implicit deformations is presented in [3]. Objects are modeled using polygonal and parametric formulations. This work is limited to deformations that are linear with respect to the state of the object. In general the function that maps a point in body space to world space is as follows:

$$D_{R(t)}(p) = R(t)Z(p)$$

[2]The term *morphing* in computer graphics is used when an object transforms into an other object, by a smooth and visually continuous change.

where the elements of p are the parametric coordinates of the points, D are the world coordinates of the object's point, R is the state matrix and $Z(p)$ a vector valued function that does not depend either on time or on R. Z can be non-linear with respect to p. These kinds of deformations greatly simplifies the equations of motion of the body. In [60] it is shown that using Lagrangian dynamics, the resulting equations of motion are

$$\ddot{R} = Q(t)M^{-1} \tag{2.7}$$

where M is a constant square matrix that can be precomputed. Q is the generalized force matrix.

A method for implementing attachment constraints (point to point) is described in [58] and used in [60]. A constraint force is calculated which cancels the applied force acting to separate attached parts. Using the previous linear formulation the constraint matrix is constant (except when constraints are added or deleted), thus its inverse can be precalculated for efficiency.

Our formulation is most closely related to that presented in [3]. However, we define the deformations in a different way which allows for more complex deformations.

2.2.4 Explicit Deformable Models

Explicit formulations define deformations directly on flexible models. Models are discretized with respect to the material coordinates by using finite differences or finite elements. The most straightforward model consists of a network of springs and point masses.

The use of visco-elastic models is first presented in the context of graphics in [44], [46]. The models have visco-elastic properties and can incorporate inelastic behavior and fracture. These models can accurately simulate the physics of elastic and inelastic deformations. In the discretized version of this formulation, the model consists of a set of nodal points. The deformation is achieved by a relative displacement of the nodal points from the rest (undeformed) positions. This displacement is governed by parameters which define the elastic behavior of the object. Displacement of nodal points may introduce restoring forces which realize the desired elastic behavior of the object. The equations of motion for this formulation are:

$$\mu\frac{\partial^2\mathbf{x}}{\partial t^2} + \gamma\frac{\partial\mathbf{x}}{\partial t} + \delta_{\mathbf{x}}\mathcal{E} = \mathbf{f} \tag{2.8}$$

where

$\mathbf{x}$	are the positions of the mass points
μ	is the mass density
γ	is a damping factor
$\delta_{\mathbf{x}}\mathcal{E}$	is the internal force that resists deformation
$\mathbf{f}$	is the total external force on $\mathbf{x}$

The discrete form of the above equations is

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}(\mathbf{q})\mathbf{q} = \mathbf{f} \tag{2.9}$$

where $\mathbf{M}, \mathbf{C}, \mathbf{K}(\mathbf{q})$ are the mass, damping and stiffness matrices respectively and $\mathbf{f}$ is the net external force.

In [47] the heating and melting of deformable models is simulated. The formulation uses the equations of motion (2.8), in addition to the heat equation from thermal physics. The model is discretized using finite element analysis (finite differences can also be used).

Another formulation is presented in [45]. The objects here are formed by superquadrics that support global and local deformations. The formulation incorporates the global shape parameters of a conventional superquadric with the local degrees of freedom of a spline. The equations of motion are as follows:

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} &= \mathbf{g}_q + \mathbf{f}_q, \\ \mathbf{g}_q &= -\dot{\mathbf{M}}\dot{\mathbf{q}} + \frac{1}{2}\left[\frac{\partial}{\partial \mathbf{q}}(\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}})\right]^T \end{aligned} \tag{2.10}$$

where:

$\mathbf{M}, \mathbf{C}, \mathbf{K}$ is the mass, damping and stiffness matrices

$\mathbf{g}_q$ is the centrifugal and coriolis force

$\mathbf{f}_q$ is the net external force

The discretization of the model in material coordinates is done using finite element analysis. In particular bilinear quadrilateral and linear triangular elements are used. Comparing (2.9) to (2.10) we see that in the latter, the stiffness matrix $\mathbf{K}$ does not depend on $\mathbf{q}$ as in (2.9). However, the term $\mathbf{g}_q$ is introduced in (2.10).

Recent work with deformable models and surfaces employs a technique based on particle systems [43, 50]. The interaction between particles can be modeled in different ways. In [50] concepts from molecular dynamics are used in order to model fluids using thermal particles. Potential functions between pairs of particles are defined as a function of the temperature in order to establish the required level of rigidity of the simulated fluid. In [43] surface particles are used to model deformable surfaces. The particles have an orientation which is dynamically constrained according to the desired smoothness of the surface.

Our formulation with respect to dynamics is most closely related to the one presented in [45]. Our equations of motion before discretization are very similar to the ones in [45] since both approaches distinguish between a global rigid body motion and a local (within a body coordinate system) deformation motion.

2.3 Collisions: Detection and Resolution

Collision detection and resolution is an important problem in physics-based simulation, since it models the interaction between objects and their environment. The formulation of this problem is generally independent of the model of the objects. However, it is more difficult to arrive at efficient collision detection and resolution methods for flexible models. The following sections separately address the two parts of the collision problem.

2.3.1 Collision Detection

Collision detection depends on the model representation used for the simulated objects. The rigid body case is simpler than the flexible one because the objects do not change shape. This allows for some quantities such as bounding boxes, to be precomputed.

Collision detection requires determining when objects interpenetrate. To detect interpenetration, a common approach is to use an implicit function for every object in the simulated space [3]. Given a point in space, an implicit function specifies if the point lies in, out or on the associated object. Variations of this technique have been extensively used in graphics.

The problem of collision detection between polyhedral objects was studied in [31]. In [2] and [3] an efficient method for detecting collisions when implicit and parametric models are used is presented. This method is restricted to linear deformations and to contact with finite number of points. Reference [16] introduces an efficient method to calculate precise contact surfaces between colliding implicit models. The method is based on an implicit

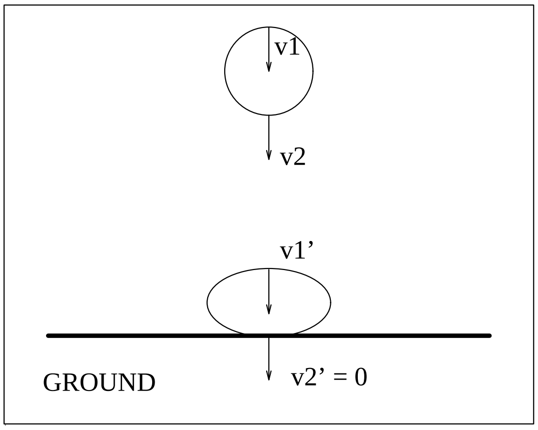

Figure 2.4: Collision between a flexible ball and the ground

formulation that generates models around skeletons, using isopotential implicit surfaces. In [41] a more general method is presented, dealing with any kind of implicit or parametric surfaces as long as inclusion functions can be defined for them. Collision detection is based on interval methods for constrained optimization.

Following collision detection the interaction between the objects must be resolved. The next section discusses that problem.

2.3.2 Collision Resolution

Two collision resolution methods are commonly used [3, 16, 31, 44, 46], *Impulsive forces* and *Penalty methods*. We begin by describing the use of *Impulsive forces*. Impulsive forces result in instantaneous changes in the velocity of the colliding objects [61]. With respect to a formulation that uses generalized coordinates, the effect of (generalized) impulses is to produce instantaneous change in the generalized momenta.

The formulation of the equations incorporating impulsive forces is straightforward for ideal rigid bodies. For flexible bodies the situation can be more complex. When a collision takes place the contact points are subject to an instantaneous (discontinuous) change in velocity, while other points are only subject to non impulsive forces that stem from the body's ability to resist deformation. For example in Figure 2.4 while the flexible ball is falling $\mathbf{v}1 = \mathbf{v}2 \neq 0$, but the moment it hits the ground $\mathbf{v}2$ becomes instantaneously zero while $\mathbf{v}1$ gradually reduces to zero with a rate that depends on the elasticity of the ball.

In [3] collision resolution for flexible objects is handled by simulating collisions using a two phase model as follows: consider a collision between an elastic body and a fixed obstacle.

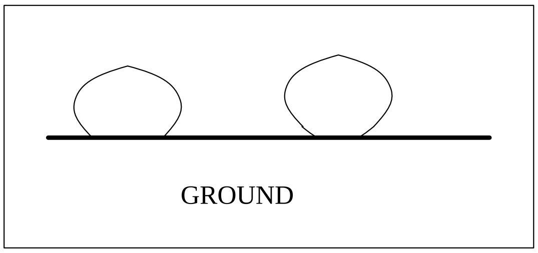

Figure 2.5: Collision of an object and the ground using different attenuation functions

In the first phase an impulse prevents interpenetration. The second non-impulsive phase follows during which the body, being in continuous contact with the obstacle, deforms due to its own momentum, building up internal strain energy. This energy makes the object rebound as we would expect from an elastic body.

Penalty methods have been widely used because of their ease of implementation. Objects interpenetrate and the amount of interpenetration is used to introduce restoring forces that push the objects apart. In [31] spring forces are used to prevent bodies in contact from interpenetrating. In [46], [44] to prevent the deformable objects from interpenetrating immobile objects, a potential energy of the form "$c\exp(-f(\mathbf{r})/\epsilon)$" is created around each object, where f is the object's in-out function. Constants c, ϵ are chosen such that the energy becomes prohibitive if the model attempts to penetrate the object. The resulting collision force is:

$$\mathbf{f} = -\left(c\frac{\nabla f(\mathbf{r})}{\epsilon} \exp\left(-\frac{f(\mathbf{r})}{\epsilon}\right) \cdot \mathbf{n} \right) \mathbf{n} \tag{2.11}$$

where $\mathbf{n}$ is the unit normal of the deformable body's surface. Penalty methods are effective and easy to use but they suffer from the problem of interpenetration and they produce stiff systems of equations [35].

Recently [16] proposed a method for calculating contact surfaces for an implicit deformable model also introduced in the same work. Given the contact surfaces the deformation forces are accurately integrated over the surface. Deformation propagates from the contact surface to the rest of the body according to an attenuation function (Figure 2.5). On the left part of the figure the attenuation function allows the deformation from the collision to propagate deeper into the object.

2.4 Comparison with our Model

The dynamic models presented in the previous sections are capable of representing many types of flexible objects. However, they are mostly oriented towards articulated figures [7, 52, 54, 55], realistic creatures [51] or non-active (non-living objects) [3, 35, 16, 45, 48]. In general most of them are too complex for flexible characters like teapots that come to life. In this thesis we introduce an approach specially suited for characters of this kind. Our approach, based on a dynamic free-form deformation model, produces an efficient dynamic formulation.

Chapter 3

Control

The dynamic models described in the previous section incorporate physics into a geometric model, and encompass both passive and active characters. For active characters it is desirable to simulate them realistically, incorporating in the model the ability to use and control actuators in order to originate autonomous motion. This leads to the *control problem* which is one of the most important issues in the animation of living creatures, especially in a physics-based approach. The task of controlling an animated character means making it follow desired motion paths, take desired poses, or having it perform specific modes of locomotion. For an example of animating a horse, it would be convenient to be able to design controllers that would make the horse walk,trot, or run. The difficulty of the control problem increases with the complexity of the character and the complexity of the desired task. It is presumably easier to design a controller to make a horse walk slowly, than to design one to make it jump over an obstacle. The latter case involves the orientation of the body changing significantly during the course of the motion, making the motion sensitive to perturbations. Controllers should be able to maintain the balance of the figure, a characteristic that introduces restrictions in the design of the controller.

We can distinguish between two different approaches towards the control problem in animation. The first involves *constraint* methods. These methods make use of constraints in order to make the character follow specified motion paths. The constraints are included implicitly into the equations of motion [5, 35, 57, 59]. Typical constraints include fixing a point in space, keeping parts of a composite object together and forbidding interpenetration. The second approach derives from *automatic control.* This involves the design of a controller that is capable of driving an object to perform specified motions. To achieve desired motions, controllers are synthesized and govern the motion of the objects without explicitly calculating the trajectory. In this approach sensory feedback can be used when necessary [52]. The controller approach is a better model of the natural behavior of a living character, where motion must always arise from control actions.

The control problem is most often formulated as an *optimization problem*. The object is required to perform a task while minimizing or maximizing a cost or objective function. For example, we might want to formulate the problem such that the object follows the desired path while minimizing its energy or the time that it takes for the task to be completed. Using existing optimization techniques, controllers can be optimized with respect to the desired motion. It has been shown that optimized control can be used to automatically synthesize controllers capable of making living characters do specific modes of locomotion [19, 32, 49, 52, 54].

Automated control has been studied extensively in robotics [27]. Many control methods exist in order to make robots perform complex tasks. However, controlling a robot is different from controlling an animated character. This is explained further shortly.

We begin by reviewing constraint methods. Methods based on controller-synthesis are then presented. There is a great amount of previous work on control methods. We will present only a part of it, focusing on the most representative and the most relevant to our work methods.

3.1 Constraint Methods

One of the original approaches applied to controlling animated characters uses *constraints* on the motion of a character. This involves integrating constraints to be maintained into the equations of motion.

In [5] *inverse dynamics* [1] is used to produce critically damped forces to enforce constraints. A descriptive diagram of constraint methods is shown in Figure 3.1 [35]. Dynamic constraints, based on inverse dynamics, are not suited for flexible models because flexible models have relatively more degrees of freedom. Reaction constraints supply reaction forces

[1] Inverse dynamics is technique which given the desired motion path, calculates the forces that make the object follow that path.

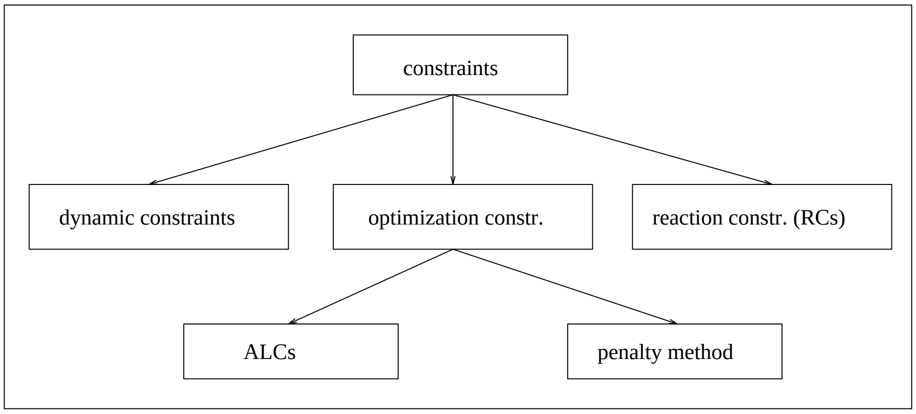

Figure 3.1: A hierarchy of constraint methods

to prevent interpenetration and to allow objects to be dragged and pulled, but are limited in scope. Optimization constraints, based on optimization theory, allow more sophisticated control. The penalty method, also used in [46, 57], is analogous to adding a rubber band to attract the object to the constraints. The Augmented Lagrange multipliers technique adds differential equations to the system and calculates Lagrange multipliers allowing multiple constraints to be fulfilled by the object. This method along with reaction constraints is used in [5]. The previous work done on this kind of control is very successful in the case of passive objects.

A similar approach is presented in [59], and extended in [8, 32]. This approach combines space and time constraints. The animator specifies *what* the object should do and *how* this should be done. For example "jump from here to there, with minimal energy". The problem is addressed as a constrained optimization problem, where the solution satisfies the *what* constraints, optimizing the *how* criteria. The work in [59] is one of the very first approaches towards the animation of active objects, but it is limited to articulated figures. In [8] an extended implementation based on *spacetime* constraints is presented. The system implements inequality and conditional constraints, while introducing additional optimizations of the algorithm for efficiency. Constraint methods have animated plastic, elastic, moldable materials [35], a simple figure [59] and a simple arm that throws a ball in a basket [8].

In [58, 60], flexible models are addressed. Using linear global deformations a compact representation for flexible models is formulated. The formulation supports constrained dynamics in a fashion similar to that used for rigid bodies.

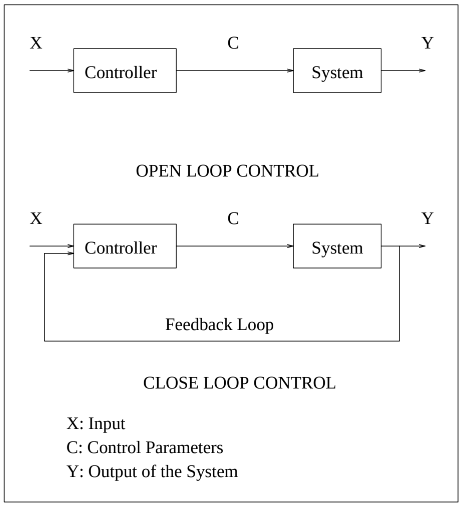

Figure 3.2: Open and closed loop control

The above methods cannot be used for physically-real characters like robots, mainly because they deal with trajectories and not control functions. As well, penalty methods require interpenetration which cannot be achieved in reality. Some constraint methods have been used in robotics [27].

The above methods produce limited but satisfactory results. However, the rich motions that actuators are able to produce are not represented efficiently with current formulations based on constraint methods. Creatures with a high number of degrees of freedom, like four-legged animals, are capable of performing complex motions which require a lot of constraints. The next section presents an alternative formulation that better models the actual control behavior of living characters.

3.2 Methods based on Controllers

A controller is a mechanism that determines individual control functions for actuators in order to perform specific motion tasks. Control strategies can be classified with respect to their use of feedback as being *Open Loop* or *Closed Loop* as shown in Figure 3.2. The first one assumes no sensory feedback. The controller has no knowledge of the outcome of its actions. This is the simplest method and has been used extensively in graphics. The second method involves a feedback mechanism. The controller has access to sensory or state information and thus can evaluate its operation and respond accordingly. The input of the controller is usually the system's state.

It is also useful to distinguish between *high-level* and *low-level* control. This distinction usually applies for complex characters. High-level control refers to the part of a controller that governs the overall motion. It can be associated with intentions or behavior [51]. For our purposes, it implements the intention of the character to take a specific posture. Low-level control implements the motion by taking a restricted set of control parameters, for example joint angles, and accordingly drive the actuators of the object. A very popular low-level control mechanism is the PD-controller [13].

The next section presents popular techniques for controlling animated characters.

3.2.1 Finite State Machines

Finite state machine (FSM) structures have been used extensively in animation. In [42], they are used as a means to add and remove constraints in the animation of a biped. In [62] a hierarchy of finite-state machines is used to provide kinematic control over a human skeleton. In [21] an FSM structure is used to control juggling, pumping a swing, and riding a see-saw. Typically these controllers discretize a periodic cycle into several states. An alternative is to use sinusoidal functions (oscillations) as done in [29, 30, 53]. The work presented in [37] on controlling hopping and running motions presents an important innovation. It shows how an effective decomposition can be used for these class of motions, to yield a set of simpler and solvable control problems.

3.2.2 Other techniques

A simple but effective technique for articulated figures is proposed in [56]. This technique is based on *cyclic pose control graphs*. A pose is defined as the specification of desired positions of all the actuated degrees of freedom of a character. For articulated figures it consists usually of the character's joint angles. Each state (pose) remains active for a finite amount of time. The controller repeatedly cycles through the graph using the parameters of the currently active pose as input to the PD-controllers. As stated in [55], the transition between states could be based on sensory data. The use of cyclic pose graphs results in controllers that produce periodic motions. The pose control technique is extended in [55] to acyclic graphs in order to achieve non-periodic motions.

There is some existing work on sensor-based controllers for animation. A controller is synthesized using weighted connections of sensors and actuators, possibly through in-

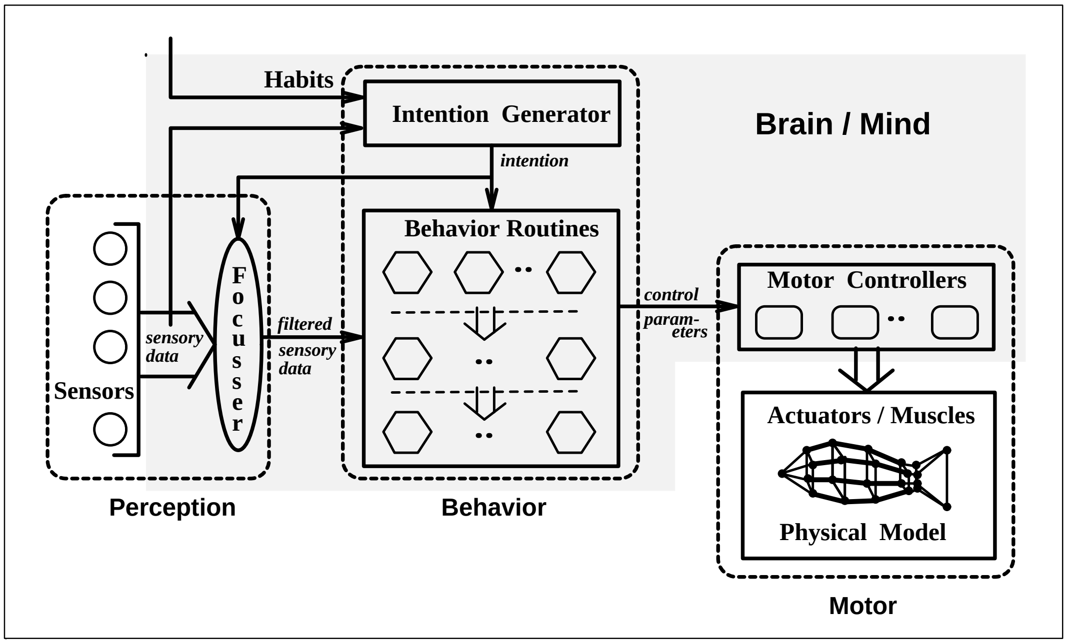

Figure 3.3: Control mechanism of artificial fish

termediate nodes. In [52], a non-linear network of weighted connections between a small number of binary sensors and the actuators (muscles) of the animated (articulated) creature is used. The network has internal delays, making it capable of dynamic behavior. A similar approach is presented in [32] using a stimulus-response paradigm.

Detailed models showing control of living flexible characters have been investigated in [19, 49, 51]. This work simulates fishes using a dynamic and highly autonomous fish model. The resulting artificial fishes behave as autonomous agents. The control scheme for the artificial fishes is shown in Figure 3.3 [2]. High-level control uses sensory feedback and intentions in order to produce behaviors for the fish. Behaviors are passed to the second level where the appropriate motor reactions are resolved. In particular, motor controllers transform physical parameters such as the desired turn angle into specific muscle actions. Actuators are driven by low level controllers, for example PD-controllers, which form the lowest control level.

Controllers can be synthesized automatically using optimization techniques. This process which is called motion synthesis, tries to simulate the way animals learn to locomote. Techniques addressing this problem are analyzed in the next section.

[2] Courtesy of Demetri Terzopoulos and Xiaoyan Tu.

3.2.3 Automatic Discovery of Motion - Optimization

Given a physical, system an interesting optimization problem is to ask how this system can perform certain tasks while minimizing a "cost". More formally, given a set of parameters P and an objective function π, we would like to find the configuration P that minimizes the objective function $\pi(P)$. Optimization problems have been studied extensively and there is a vast literature on optimal control, combinatorial optimization, optimization problems where the reader can seek more information [1, 6, 11, 12, 33]. In general we can distinguish between *deterministic* and *probabilistic* approaches to the optimization problem. Automatic discovery of motion or *motion synthesis* (MS) can be formulated as an optimization problem and it is often addressed using probabilistic methods. Probabilistic methods are popular because they are easy to implement, they are suitable for searching large spaces and they can employ techniques to avoid local optima. They can be based on a generate-and-test strategy which eliminates the need to incorporate directly the equations of motion into the objective function [19]. Controllers are synthesized and tested using forward simulation.

Generate-and-test algorithms are used in [32, 52]. Both methods use stimulus-response type representations which can often be mapped to finite-state machines if the resulting motion is periodic. The motion synthesis method in [52] implements a two phase search which distinguishes between global and local search of the parameter space. In the first phase, random controllers are synthesized and evaluated. In the second phase the best controllers of the first phase are refined and improved. The two phases define a coarse-to-fine strategy in searching for the best controllers. In [32] the parameter space is sampled in a global way after an initial random population of controllers has been generated. Local searches are then performed using the mutation and crossover operations of genetic algorithms.

In [19, 49] the automatic synthesis of motions for deformable models is addressed. Deformable models have a large number of parameters that must be controlled. In [19, 49] the high-dimensionality of the state/actuator space of the animated model is reduced through control abstraction. The state space is reduced into task-specific state spaces. More specifically, the controllers are projected on a set of basis functions (Fourier basis functions) and the significant parts of the projection are kept. This results in a compact representation which reduces the dimensionality of the controller and hence the space of possible solutions. After the projection, the space of possible solutions is searched with a simulated annealing

1. Choose an initial configuration P
2. Choose an initial temperature $T > 0$
3. While $T > T_{frozen}$
   (a) Pick a random neighbor P' of P
   (b) Let $\Delta = Cost(P') - Cost(P)$
   (c) If $\Delta \leq 0$ set $P = P'$
   (d) If $\Delta > 0$ set $P = P'$ with probability $e^{-\Delta/T}$
   (e) Set $T = rT$
4. Return P

where r is the *cooling rate* and $Cost$ is a cost function [24].

Figure 3.4: Simulated annealing for minimization

technique [52].

Searching the space of possible controllers is an important step of the motion synthesis problem. Two popular search methods for the stochastic motion synthesis problem are described in the next sections.

3.2.4 Genetic Algorithms

Genetic algorithms were first developed by Holland in [22]. They have been used widely in applications as a way to find optima in large search spaces. Recently they have been introduced in computer graphics applications. In [40] such an algorithm is used to synthesize abstract textures and images. In [32] a genetic algorithm is used to find control solutions. Most genetic algorithms follow the same general scheme [17, 38]. Starting from an initial population of generic objects consisting of functions, images, parameters, etc, new objects are generated (evolve) using the operations of mutation and crossover. If the offspring performs well, they are used as the parents for the next iteration of the algorithm. Instances that perform badly are also kept as a way to avoid local optima. When a generation has evolved that satisfies certain criteria, the algorithm stops.

3.2.5 Simulated Annealing

Kirkpatrick, Gelatt, and Vecchi, proposed in 1983 a class of stochastic algorithms based on probabilistic *hill-climbing* [25]. Hill climbing algorithms implement a stochastic way to avoid local optima. Among them, simulated annealing has found widespread use [14, 19, 52, 55].

The algorithm is shown in Figure 3.4. The cooling rate controls the hill-climbing part of the algorithm, reducing the probability of an uphill move as the number of iterations increases.

The algorithm is sensitive to the choice of initial configuration P and the annealing schedule. A good initial guess can make the algorithm find a global optimum quickly. A bad guess may lead in getting stuck in local optima. Theoretically, simulated annealing will find the global optimum if the cooling ratio is infinitely low.

3.3 Conclusion

Controlling animated characters is a complicated task. Among the methods that solve this problem, controller methods best approximate the way a real animal controls itself. When using controllers in physics-based simulation, the control scheme must be incorporated in the dynamic formulation. The next chapter discusses our dynamic model and Chapter 5 presents our dynamic control scheme.

Chapter 4

Dynamic Free-form Deformations

4.1 Modeling Flexible Characters

This work addresses the problem of flexible and active character animation, using physics-based simulation. In particular, we are focusing on lifelike animation of inanimate objects. We will call these characters cartoon or active-flexible characters, although they do not represent a typical TV cartoon character. An active-flexible character simulation differs in several ways from the animation of a flexible, passive object.

1. The rigidity of the character may vary in time. The character can change properties and shape according to the desired result, and has no consistent skeleton. The character may present elastic properties at some times while at other times it may act much like an articulated object.

2. A cartoon character is *active*. It can move or deform itself voluntarily using its internal actuators, as well as moving and deforming in response to external forces.

We have chosen to use the free-form deformation technique to model our flexible and active characters for the following reasons:

- The animator can specify the desired modes of deformation during animations with a conventional modeling tool.

- By changing the basis functions we can alter the properties of the deformation, including its smoothness and continuity.

- The technique is independent of the object model,
- If more degrees of freedom are required for the deformations, the control lattice can be expanded or subdivided.
- Free-form deformations cooperate efficiently with physics-based simulation and control techniques.

We believe that FFDs can be the main tool of an interactive flexible character animation system in which animation and object modeling are independent. The animator can add lattices according to the desired level of detail without altering the object model. In general, the object need only be modeled once, followed by experiments to determine desired deformation modes and levels of detail. This way the animator can experiment with different kinds of functionality and properties for the same object.

In our implementation we address the 2D case. It could be extended to 3D without significant changes. Our system permits any number of deformation modes to be used. We have experimented with up to four deformation modes.

4.2 Geometric Model

The animator can model the object to be animated using their technique of preference. The connection between the object model and the deformation mechanism is established by a set of object *control points*. These points can be sample points of the actual object, or the control parameters of a parametric representation. It is convenient to use a small number of object points during the simulation for efficiency. The final rendering of the object can be done using more expensive techniques. This idea is also used in [3]. We have experiment with polygonal and parametric curves.

Next section analyzes how we use the free-form deformation technique to develop an efficient dynamic flexible model.

4.3 Deformations-Motion

As mentioned earlier the free-form deformation scheme considers the object embedded in a parametric space which is initially an orthogonal lattice. This space is defined by the lattice control points. Each dimension of the lattice can be subdivided independently of the others. We have chosen to use a 4×4 lattice for simplicity. The basis functions for the deformations are the Bernstein polynomials. Bernstein polynomials are simple and they provide the convex hull property for the deformations. However, if the lattice is larger than 4x4, then cubic B-spline basis functions may be used in order to guarantee C^2-continuity. In that case, proper care will be taken to enforce the convex hull property [4].

Given a point (s, t) in body (lattice) coordinates, its world coordinates are given by

$$\mathbf{P} = \sum_{i=0}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) (1-s)^{n-i} s^i \left[\sum_{j=0}^{m} \left(\begin{array}{c} m \\ j \end{array} \right) (1-t)^{m-j} t^j \mathbf{L}_{ij} \right], \quad s, t \in [0,1] \qquad (4.1)$$

where $\mathbf{L}$ is the lattice. In the case of a 4x4 lattice $n = m = 3$.

In order to conveniently represent the different ways that an object can deform, a set of deformation modes of the object is defined and parameterized. The notion of deformation modes has been previously used in [34]. This work focuses on the principal modal deformations of solid objects. We allow the deformation modes to be defined in way that allows for complex deformations to be easily synthesized from simpler ones. In addition, we implement sophisticated control techniques which are better suited for complex and active characters than inverse kinematics.

Each deformation mode has a corresponding amplitude $\gamma_i \in [-1, 1]$. Amplitudes γ_i are restricted to $[-1, 1]$ for our implementation. We elaborate on this later on. To activate a deformation, the user needs to specify the deformation mode and the associated amplitude. Figure 4.1 presents some examples of how the lattice and an object deform using two modes of deformation: a bend and a shear. Results produced by different values of the associated amplitudes are shown.

In addition to being able to deform, the lattice and hence the object is free to translate and rotate. A local coordinate system is initially imposed on the object; this system is initially parallel to the world coordinate system and its origin coincides with the bottom left point of the object's lattice. The object deforms within the local coordinate system. Rotation is performed around the origin of the local coordinate system. Assuming two deformation modes the state vector $\mathbf{q}$ is

$$\mathbf{q}^T = [\ t_x \quad t_y \quad \theta \quad \gamma_1 \quad \gamma_2\ ]$$

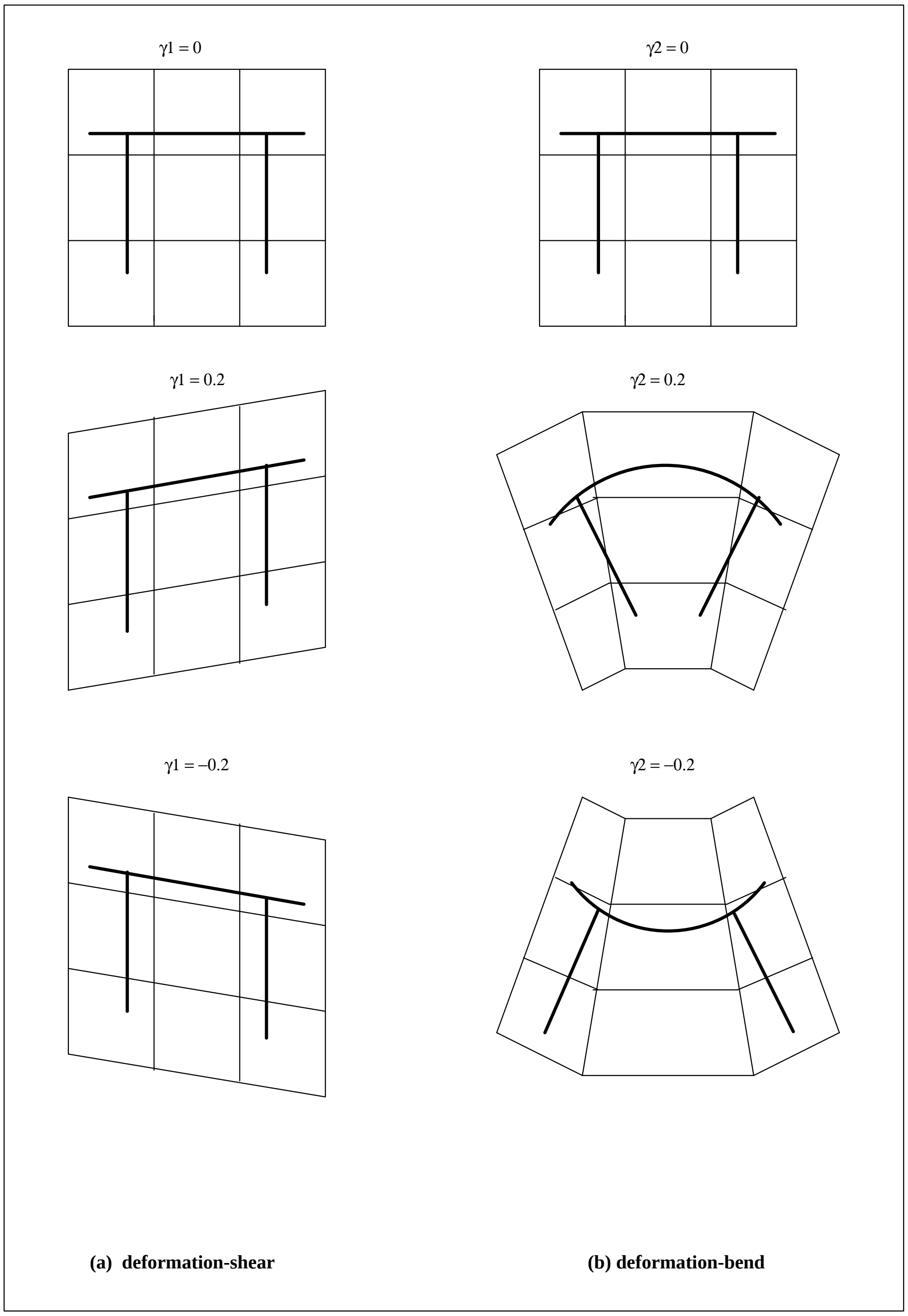

Figure 4.1: Different lattice deformations for different values of the modal amplitude γ

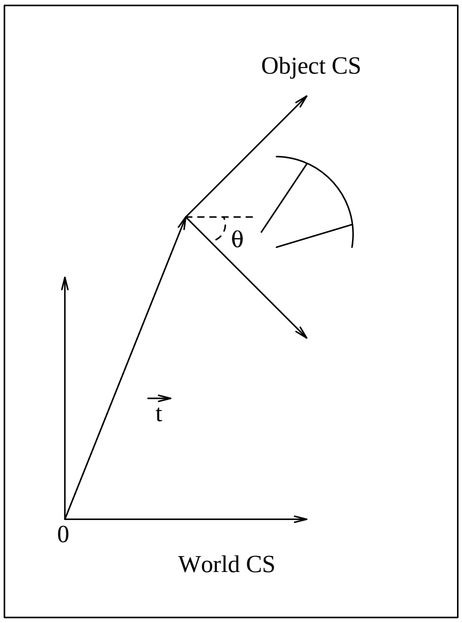

Figure 4.2: The table for some value of the state vector

where t_x, t_y are the translation parameters, θ is the rotation parameter and γ_1, γ_2 are the deformation amplitudes. Figure 4.2 shows the table with a state vector:

$$\mathbf{q}^T = [\ 100 \quad 300 \quad -60^\circ \quad 0.0 \quad 0.2\ ]$$

where the first deformation mode is a shear and the second a bend.

In equation 4.1 the world coordinates of the lattice points are used, i.e. after the translation and rotation have been performed. We now need to represent the dependencies between the state variables and the position of an object point. This formulation is shown below

$$\mathbf{P} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \mathbf{R} \sum_{i=0}^{n} \sum_{j=0}^{m} \left(\sum_{k=1}^{D} \gamma_k \begin{bmatrix} d_{ij,x}^k \\ d_{ij,y}^k \end{bmatrix} + \begin{bmatrix} L_{ij}^x \\ L_{ij}^y \end{bmatrix} \right) B(s,i)B(t,j), \tag{4.2}$$

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4.3}$$

where

$\mathbf{P}$	is an object point in world coordinates
(s,t)	are its lattice coordinates
γ_k	is the amplitude of the k-th deformation mode
d^k_{ij}	are constants which define the k-th deformation mode of the i,j-point of the lattice
$B(s(t),i(j))$	is the Bernstein polynomial $(1-s)^{3-i}s^i$
$\mathbf{L}_{ij}$	are the undeformed lattice coordinates

Equation 4.2 shows how we define and use the various motions of the object. The lattice (and implicitly the object) can deform in D different ways. Each deformation is associated with an amplitude γ_k which is a state variable. In the example mentioned above where only two deformations are used $D = 2$. The deformations are done with respect to the object coordinate system. We assume the following order in the application of the motions of the object: deformations, rotation, translation. Equation 4.2 is consistent with this assumption.

Each deformation mode is defined by a matrix $\mathbf{d}$. These matrices are added to the lattice points within the local coordinate system, after they have been weighted with the associated amplitude γ_k. The amplitudes of the deformation modes in our implementation take values in the interval $[-1,1]$. A value of 1 implies maximum deformation in one sense and a value of -1 means maximum deformation in the opposite sense. This concept implements the parameterization of the deformation modes. See Appendix A for a simple and detailed example.

Equation 4.2 is the starting point for the simulation of the dynamics of the object, to be presented in the next section. We define the term *dynamic free-form deformation* as the free motion of a set of points constrained to the set of motions expressible using FFDs.

4.4 Dynamic Free-Form Deformations

To incorporate dynamics into our flexible models we use Lagrangian dynamics [18, 28]. As a first step, a set of generalized coordinates is required, that is a set of variables that can completely define the state of the object in time. The most appropriate one for our case is the state vector $\mathbf{q}$ which is given by the expression

$$\mathbf{q}^T = [\ t_x \quad t_y \quad \theta \quad \gamma_1 \quad \gamma_2 \quad \ldots \quad \gamma_D] \tag{4.4}$$

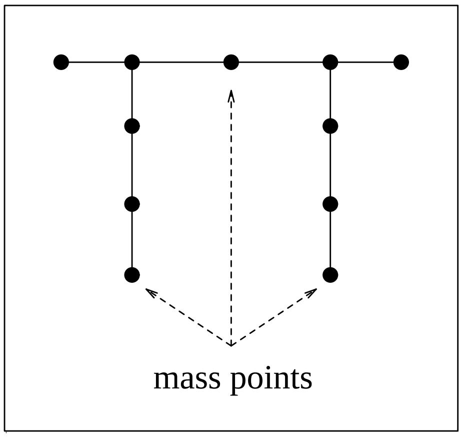

Figure 4.3: Discretization of the table

Each component of the state vector is a generalized coordinate. Lagrange's formulation is given by

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q_k}} - \frac{\partial L}{\partial q_k} - Q_k = 0 \tag{4.5}$$

where L is the Lagrangian, q_k is the k-th component of the state vector ($k = 1, ..., D$), Q_k is the total generalized force acting on the object along the q_k generalized coordinate, and the dot indicates a time derivative. The Lagrangian L is the difference between the kinetic (E) and the potential (V) energy of the object. However, we choose to use only the kinetic energy because potential energies can be associated with forces included in $\mathbf{Q}$. The following sections explain in detail how we calculate each of the quantities of (4.5). In the following sections we use the discrete version of the Lagrangian dynamics described in Chapter 2.

4.5 Kinetic Energy-Derivatives of the Lagrangian

In this section the terms of (4.5) are derived. Starting from the Lagrangian and considering only the kinetic energy we obtain $L = E$. The object is discretized in material coordinates using point masses. For example, the table in Figure 4.3 is discretized using 11 mass points. The kinetic energy of the k-th point mass is given by $E_k = \frac{1}{2}m_k\dot{\mathbf{x}}_k^T\dot{\mathbf{x}}_k$, where m_k is the point mass and $\mathbf{x}_k$ is the position of the point in world coordinates. Using (4.2) we can write E_k with respect to the generalized coordinates. To do that we first note that $\dot{x}_i = \sum_j \frac{\partial x_i}{\partial q_j}\dot{q}_j$. Defining the Jacobian matrix $\mathbf{J}$ as $J_{ij} = \partial x_i/\partial q_j$, we can write $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$. Using (4.2) and $\mathbf{q}$

as represented in (4.4), the expression for the Jacobian matrix is

$$\mathbf{J}^T = \begin{bmatrix} 1 & 0 & \mathbf{a}(s,t)\,||_x & \mathbf{b}(s,t,1)\,||_x & \dots & \mathbf{b}(s,t,D)\,||_x \\ 0 & 1 & \mathbf{a}(s,t)\,||_y & \mathbf{b}(s,t,1)\,||_y & \dots & \mathbf{b}(s,t,D)\,||_y \end{bmatrix} \tag{4.6}$$

where

$$\mathbf{a}(s,t) = \frac{\partial \mathbf{R}}{\partial \theta} \sum_{i=0}^{n} \sum_{j=0}^{m} \left(\sum_{k=1}^{D} \gamma_k \mathbf{d}_{ij}^k + \mathbf{L}_{ij} \right) B(s,i)B(t,j) \tag{4.7}$$

$$\mathbf{b}(s,t,k) = \mathbf{R} \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{d}_{ij}^k B(s,i)B(t,j) \tag{4.8}$$

$n = m = 3$ in the case of the 4×4 lattice and "$||_{x(y)}$" indicates the $x(y)$ component of a vector.

The kinetic energy with respect to the generalized coordinates is

$$E_k = \frac{1}{2} m_k \dot{\mathbf{x}}_k^T \dot{\mathbf{x}}_k = \frac{1}{2} m_k (\mathbf{J}_k \dot{\mathbf{q}})^T \mathbf{J}_k \dot{\mathbf{q}} = \frac{1}{2} m_k \dot{\mathbf{q}}^T \mathbf{J}_k^T \mathbf{J}_k \dot{\mathbf{q}}$$

The total kinetic energy of the object is $E = \sum_k E_k$ and thus the Lagrangian is

$$L = \sum_k E_k = \sum_k \frac{1}{2} m_k \dot{\mathbf{q}}^T \mathbf{J}_k^T \mathbf{J}_k \dot{\mathbf{q}} = \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{M} \dot{\mathbf{q}} \tag{4.9}$$

where

$$\mathbf{M} = \sum_k m_k \mathbf{J}_k^T \mathbf{J}_k \tag{4.10}$$

is a symmetric generalized mass matrix. Having an expression for the Lagrangian, we can calculate the required derivatives for (4.5). In the following expressions, terms of the form $M_{ij} q_j$ imply summation over j.

$$\frac{\partial L}{\partial \dot{q}_k} = \frac{1}{2} \left(\delta_{ki} M_{ij} \dot{q}_j + \dot{q}_i^T M_{ij} \delta_{kj} \right) = \frac{1}{2} \left(M_{kj} \dot{q}_j + \dot{q}_i^T M_{ik} \right) = \frac{1}{2} \left(M_{kj} \dot{q}_j + \dot{q}_j^T M_{jk} \right)$$

where δ_{ki} is the Kronecker δ-function. Since $\mathbf{M}$ is symmetric, $M_{kj} = M_{jk}$ thus $M_{kj} \dot{q}_j = \dot{q}_i^T M_{ik}$ and

$$\frac{\partial L}{\partial \dot{q}_k} = M_{kj} \dot{q}_j \Rightarrow \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} = M_{kj} \ddot{q}_j + \left(\frac{d}{dt} M_{kj} \right) \dot{q}_j = M_{kj} \ddot{q}_j + \sum_l \left(\frac{\partial M_{kj}}{\partial q_l} \dot{q}_l \right) \dot{q}_j \tag{4.11}$$

The second term that involves the Lagrangian in (4.5), is calculated as follows:

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}\dot{q_k}^T \frac{\partial \mathbf{M}}{\partial q_k}\dot{q_k} \tag{4.12}$$

The derivatives of $\mathbf{M}$ required in equations 4.11, 4.12 can be calculated as follows:

$$\frac{\partial \mathbf{M}}{\partial q_j} = \frac{\partial}{\partial q_j}\sum_k m_k \mathbf{J}_k^T \mathbf{J}_k = \sum_k m_k \left[\left(\mathbf{J}_k^T \frac{\partial \mathbf{J}_k}{\partial q_j}\right)^T + \mathbf{J}_k^T \frac{\partial \mathbf{J}_k}{\partial q_j}\right] \tag{4.13}$$

Lastly,we need expressions for the partial derivatives of $\mathbf{J}$ with respect to the generalized coordinates $\mathbf{q}$ (equation 4.4). We derive these expressions using (4.6):

$$\frac{\partial \mathbf{J}}{\partial t_x} = \mathbf{0} \tag{4.14}$$

$$\frac{\partial \mathbf{J}}{\partial t_y} = \mathbf{0} \tag{4.15}$$

$$\frac{\partial \mathbf{J}}{\partial \theta} = \begin{bmatrix} 0 & 0 & \mathbf{c}(s,t)\,||_x & \mathbf{f}(s,t,1)\,||_x & \dots & \mathbf{f}(s,t,D)\,||_x \\ 0 & 0 & \mathbf{c}(s,t)\,||_y & \mathbf{f}(s,t,1)\,||_y & \dots & \mathbf{f}(s,t,D)\,||_y \end{bmatrix} \tag{4.16}$$

$$\frac{\partial \mathbf{J}}{\partial \gamma_k} = \begin{bmatrix} 0 & 0 & \frac{\partial \mathbf{R}}{\partial \theta}\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{d}_{i,j}^k\,||_x & 0 & \dots & 0 \\ 0 & 0 & \frac{\partial \mathbf{R}}{\partial \theta}\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{d}_{i,j}^k\,||_y & 0 & \dots & 0 \end{bmatrix} \tag{4.17}$$

where $n = m = 3$ for a 4×4 lattice and

$$\mathbf{c}(s,t) = \frac{\partial^2 \mathbf{R}}{\partial \theta^2}\sum_{i=0}^{n}\sum_{j=0}^{m}\left(\sum_{k=1}^{D}\gamma_k \mathbf{d}_{ij}^k + \mathbf{L}_{ij}\right)B(s,i)B(t,j) \tag{4.18}$$

$$\mathbf{f}(s,t,k) = \frac{\partial \mathbf{R}}{\partial \theta}\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{d}_{ij}^k B(s,i)B(t,j) \tag{4.19}$$

4.6 External Forces

All external forces acting on the simulated object must be transformed into generalized forces, that is forces acting along the generalized coordinates (directions). If a force $\mathbf{f}$ acts on the point $\mathbf{x}$ of the object then the associated generalized force is given by the formula

$$Q_j = \sum_i \frac{\partial x_i}{\partial q_j} f_i = \sum_i J_{ij} f_i \Rightarrow \mathbf{Q} = \mathbf{J}^T \mathbf{f} \tag{4.20}$$

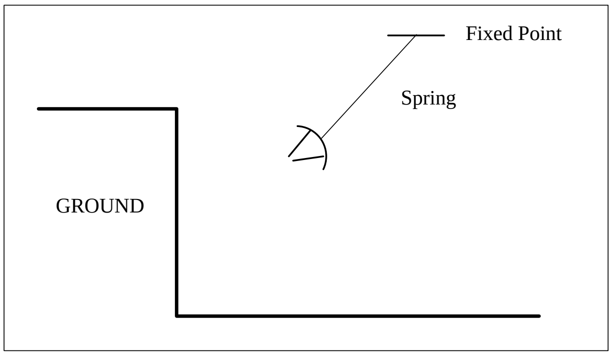

Figure 4.4: Table and spring

The above expression derives from the principle of virtual work [18]. A force $\mathbf{f}$ in cartesian coordinates that undergoes a virtual displacement $\delta\mathbf{x}$ produces virtual work $\delta W = \mathbf{f} \cdot \delta\mathbf{x}$. Based on that we can write

$$\delta W = \mathbf{f} \cdot \delta\mathbf{x} = \sum_j \mathbf{f} \cdot \frac{\partial \mathbf{x}}{\partial \mathbf{q_j}} \delta q_j = \mathbf{Q}\delta\mathbf{q}$$

We have implemented gravity forces, collision forces and a spring force that can be applied to objects. As an example animation, Figure 4.4 shows a table jumping from a cliff and a spring catching the table.

4.7 Internal Forces

Our Lagrangian includes only the kinetic energy E. Any kind of deformation potential energy is included as a generalized force. This is done as follows. The gravitational force for each point mass is $(m_k g \mathbf{i}_y)$ where g is the gravitational acceleration (typically $9.81 m/sec$) and $\mathbf{i}_y$ is the normal of the $y-$direction. It is transformed into a generalized force using (4.20). The potential energies of deformations are defined by the user and are a function of the deformation amplitudes γ_i. For example, deformation one (state parameter γ_1) may be associated with a potential energy of the form $V_1 = K(\gamma_1)^2$, where K indicates a constant. Assuming that the potential energy V corresponds to a conservative field, the associated force is given by $\mathbf{F} = -\nabla V$. In the case of the previous example the force is

$$\mathbf{Q} = -\nabla V_1 \Rightarrow Q_{\gamma_1} = -\frac{\partial V_1}{\partial \gamma_1} = 2K\gamma_1$$

where the gradient operator ∇ is defined as

$$\nabla := \left(\frac{\partial}{\partial t_x} \quad \frac{\partial}{\partial t_y} \quad \frac{\partial}{\partial \theta} \quad \frac{\partial}{\partial \gamma_1} \quad \frac{\partial}{\partial \gamma_2} \quad \cdots \quad \frac{\partial}{\partial \gamma_D} \right)$$

and D is the number of the deformation modes of the object.

4.8 Integration

Substituting (4.11) and (4.12) in (4.5) and assuming that $\mathbf{Q}$ includes the forces due to deformations and gravity, the equations of motion of the object would take the form:

$$M_{kj}\ddot{q}_j = Q_k + \frac{1}{2}\dot{q}_i{}^T \frac{\partial M_{ij}}{\partial q_k}\dot{q}_j - \left(\sum_l \frac{\partial M_{kj}}{\partial q_l}\dot{q}_l\right)\dot{q}_j \tag{4.21}$$

The above equations form a system of the form $\mathbf{Mx} = \mathbf{b}$, of $|\mathbf{q}|$ coupled equations with $|\mathbf{q}|$ unknowns. The system is of the second order with respect to $\mathbf{q}$ and needs to be numerically integrated forward through time.

To do that we work as follows. At each time step of the simulation the linear system (4.21) is solved for $\ddot{\mathbf{q}}$ using Gaussian-elimination with LU-factorization [36]. Once solved, using Euler integration and finite differences we obtain the new values for $\mathbf{q}, \dot{\mathbf{q}}$ as follows:

$$\dot{\mathbf{q}}_n = \frac{\mathbf{q}_n - \mathbf{q}_{n-1}}{\Delta t} \tag{4.22}$$

$$\ddot{\mathbf{q}}_n = \frac{\mathbf{q}_n - 2\mathbf{q}_{n-1} + \mathbf{q}_{n-2}}{\Delta t^2} \tag{4.23}$$

Other integration methods can also be used. A very popular one is the *implicit Euler integration* which is more computationally expensive than the explicit one but allows for a larger time step. The explicit Euler method proved to be sufficient in our experiments.

4.9 Interpenetration

For the purposes of this work, we only deal with collisions between one object and the horizontal ground. The ground is modeled using a penalty method which simulates the ground force using a spring and damper unit as shown in Figure 4.5 (a). When a point on the object goes below the ground, a spring and damper unit is attached between the point

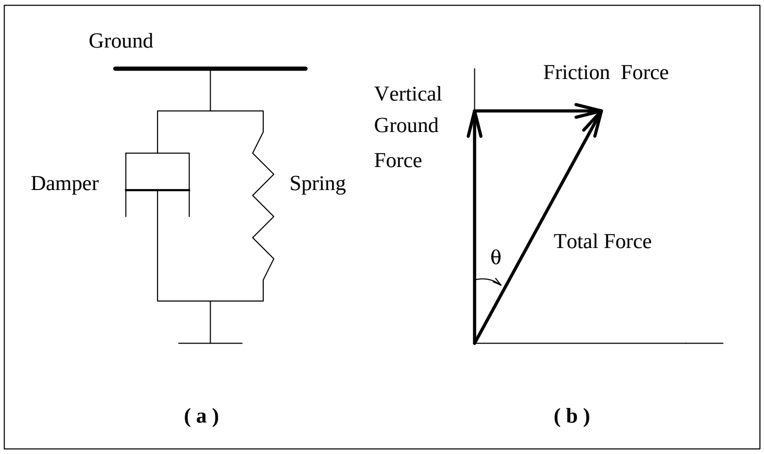

Figure 4.5: Ground force model

of entry and the object point. The same unit thus implements friction and the vertical ground force. However, to incorporate more flexibility in the simulation, we use different damping coefficients for the friction and the vertical part of the force. Moreover, to avoid undesirable effects when the time step used is large, the absolute value of the ratio between the friction force and the vertical force is not allowed to exceed $\tan(\theta)$.

The constants defining the stiffness of the unit have been chosen such that the interpenetrations caused by collisions are sufficiently small. Using this formulation the ground force given an interpenetration $\Delta\mathbf{l} = (\Delta x, \Delta y)^T$ is

$$f_x = -K_d \Delta x - FR_{dam} v_x \tag{4.24}$$

$$f_y = -K_d \Delta y - C_{dam} v_y \tag{4.25}$$

where K_d is the spring constant, C_{dam} is the damping constant for the vertical force, FR_{dam} is the friction damping constant and $\mathbf{v} = (v_x, v_y)$ is the velocity of the point that is under the ground. The velocity of a point in cartesian coordinates is calculated as

$$\mathbf{v} = \sum_j \frac{\partial \mathbf{x}}{\partial q_j} \dot{q}_j = \mathbf{J}\dot{\mathbf{q}}$$

where $\mathbf{x}$ is the position of the point in world coordinates.

In the previous formulation, the ground is assumed to be horizontal. If the ground is of arbitrary slope, then the tangent on the point of entry is calculated, and the coordinate system is rotated accordingly.

Addressing the problem of collisions for flexible models in a complete way, is a difficult task. There are a number of methods that could be used, including the method described in [3]. This method is limited only to linear deformations.

4.10 Animation - Implementation

The implemented system works as follows. The user supplies a discrete representation of an object such as the point masses shown in Figure 4.3. The object can consist of different parts which currently do not move independently from each other. Using a command line interface the user specifies a deformation file to be loaded, which holds the matrices defining the deformation modes. This operation can be done at any time so that a different set of deformation modes replaces the current ones. With the deformation modes loaded, the user can produce key-framed or physics-based motion.

4.10.1 Key-Framing

The user can specify a deformation mode and the desired amplitude to interactively make the object deform. For example given a deformation defined as

```
x: 150 150 -150 -150
   50 50 -50 -50
   -50 -50 50 50
   -150 -150 150 150

y: 0 100 100 0
   0 100 100 0
   0 100 100 0
   0 100 100 0
```

invoking the deformation with an amplitude of 0.2 will make the table of Figure 1.2 deform as shown in Figure 4.6. The numbers in the previous deformation matrix correspond to pixels with respect to drawing the object and to meters with respect to the physics-based simulation. The user can also constrain a point of the object such that when the object

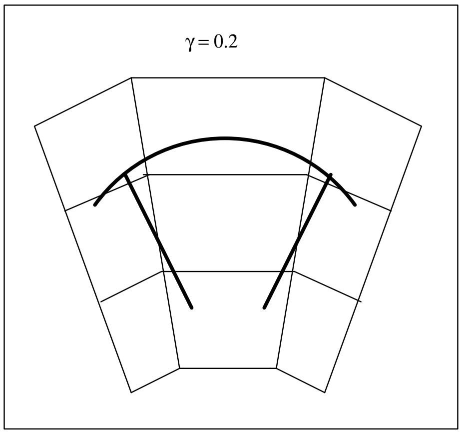

Figure 4.6: Bent table

deforms this point remains in place. This function was implemented in order to prevent the object from losing contact with the ground when deforming during key-framing.

A key-framed motion is produced by specifying a sequence of deformations. In Figure 4.12 the first part of the motion is key-framed.

4.10.2 Physics-Based Simulation

The simulator implements a numerical integrator for the Lagrange equations of motion (4.21). A number of parameters can be used to customize the simulation. The user-interface at present is a simple command line interpreter. The parameters that can be modified are the constants used in the ground model, flags that turn on/off features such as gravity, etc. An example of typical interpreter input is presented in Appendix C.

For simulation purposes it is enough to discretize the object into a small number of point masses. For example, the table is discretized into 11 points. During rendering, more points could be used for an improved visual result.

The simulation runs at interactive speeds. Time steps used vary from 0.01 to 0.1. The complexity of the system is $O(n)$ with respect to the number of point masses. It is also $O(n)$ with respect to the number of lattice points. The number of deformation modes defines the dimension of the linear system that is solved in each simulation iteration, thus the complexity of the implemented system is $O(n^3)$ with respect to the number of deformation modes.

The stability of the system depends on four major issues analyzed in the next section.

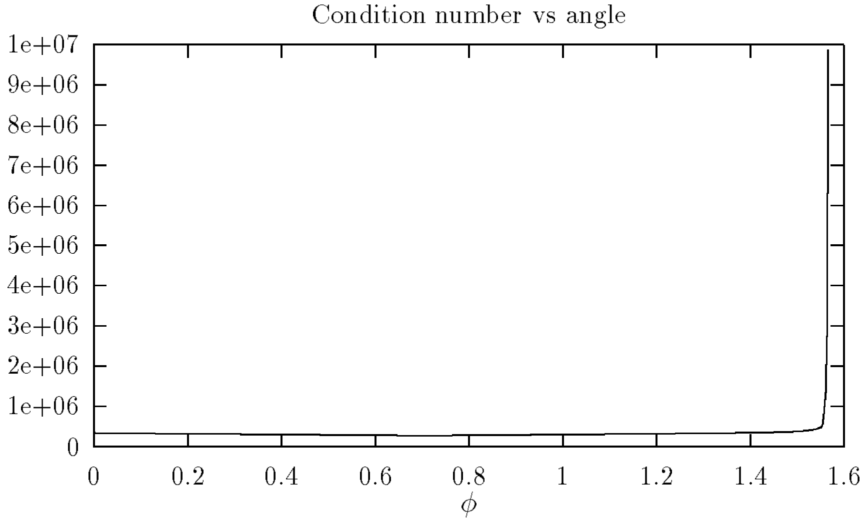

Figure 4.7: Condition number vs angle

4.10.3 Stability

The main factors that affect the stability of the system are:

1. Time step
2. Stiffness introduced by the ground model
3. Mass distribution
4. Set of deformations

Time Step: Since the simulation is discretized in time intervals, a numerical error is made which increases with the time step. The smaller the time step is, the better the simulation approximates the real-continuous motion of the object.

Stiffness: The ground model is based on a penalty method. Penalty methods introduce stiffness into the equations as the interpenetration energy becomes large [35]. To solve this problem a small time step must be used.

Mass distribution: The set of deformations must have a natural connection with the mass distribution of the object. That means that they must correspond to valid deformations for the particular mass distribution of the animated object. It is important to note that the deformations are defined with respect to the lattice points. This permits situations like the following to arise: a bend deformation defined along the x-axis for an object such as a

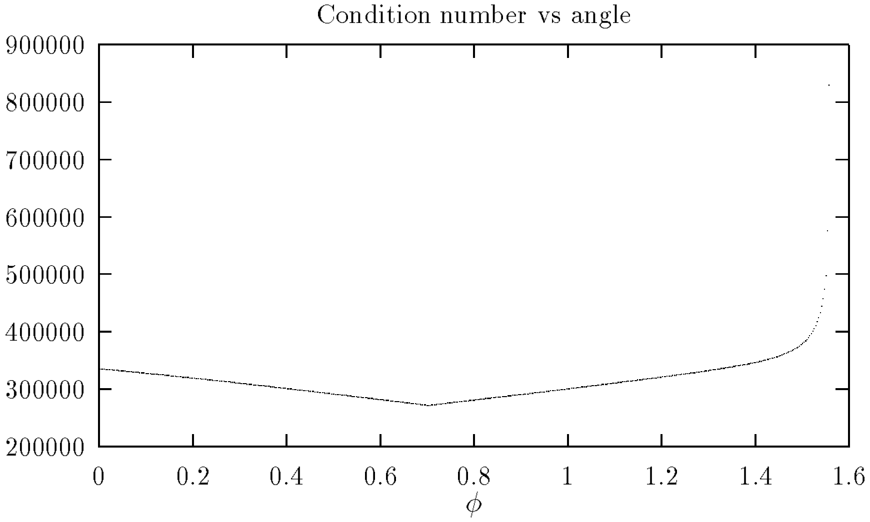

Figure 4.8: Condition number vs angle

one-dimensional vertical rod. Such a deformation has no meaning for this kind of object. There are no mass points to bend horizontally. In this case, the system is unstable.

Set of deformations: The set of deformation modes along with the rigid body parameters, defines the degrees of freedom of the object. Assuming a set of D deformation modes, the object is "moving" in a $(D+3)$-dimensional space (the "3" stands for the number of rigid body degrees of freedom). In accordance with the orthogonal Cartesian space, we can introduce the concept of orthogonality to our $(D+3)$-dimensional deformable space. The deformation modes in the set should have "orthogonal" components in order to have a well-defined linear system of equations. For example, imagine the case where the same deformation mode exists twice in the set. This results in the system having an equation appearing twice, which means that the system is overdetermined and the determinant of the system matrix is zero. Such cases were tested experimentally and the expected ill-conditioning was observed.

The convergence of a linear system depends on the *condition number* of the system matrix. The smaller the condition number the more stable the system is [20]. The condition number of a matrix is defined as

$$cond(M) = \|M\|\|M^{-1}\|$$

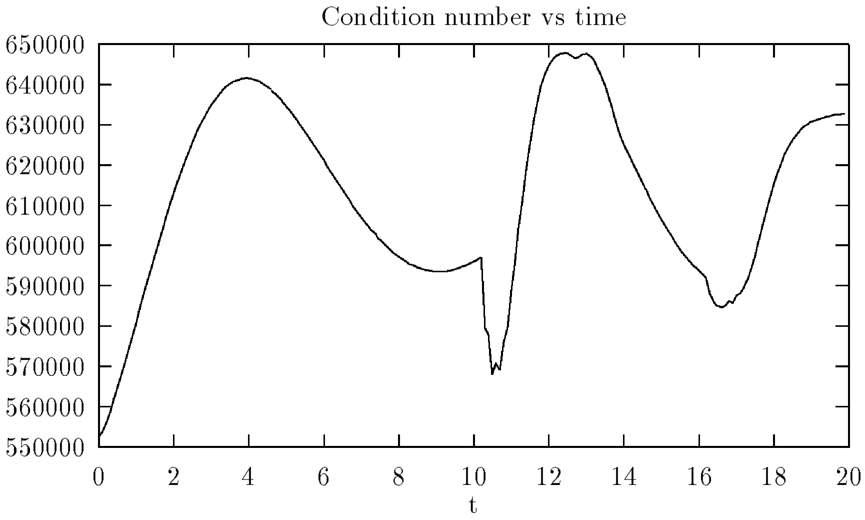

Figure 4.9: Condition number vs time

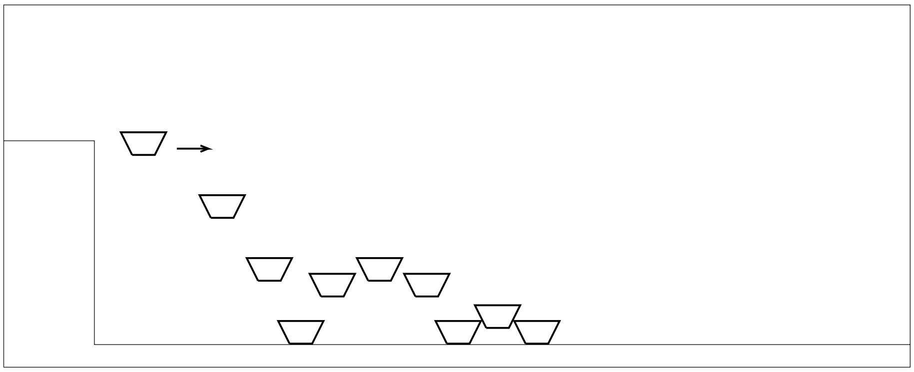

Figure 4.10: Falling pot, friction disabled

In Figure 4.7 the condition number is plotted with respect to the angle between two deformations. The deformations are vertical squash and horizontal squash. The latter is gradually rotated such that when $\phi = \pi/2$ the two deformations are both vertical squash.

The condition number indicates instability when ϕ approaches $\pi/2$. In Figure 4.8 the same data points are plotted except from the last three ones, in order to reduce the range in y and better reveal the dependence of the condition number on ϕ. Also the condition number is plotted without connecting the points, in order to show the placement of the ϕ samples.

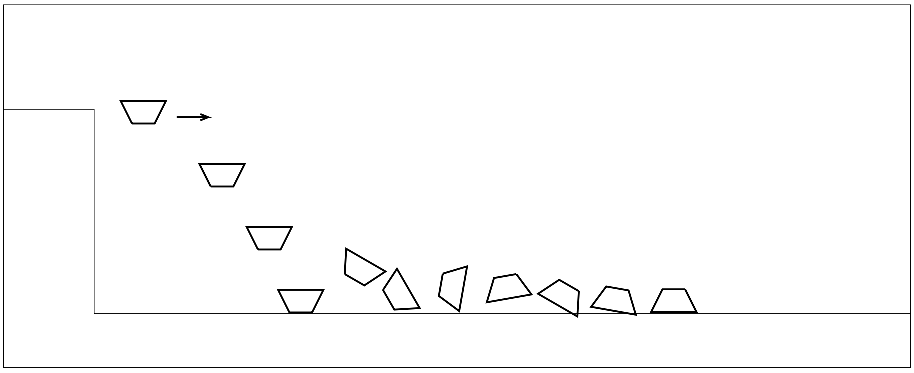

Figure 4.11: Falling pot, friction enabled

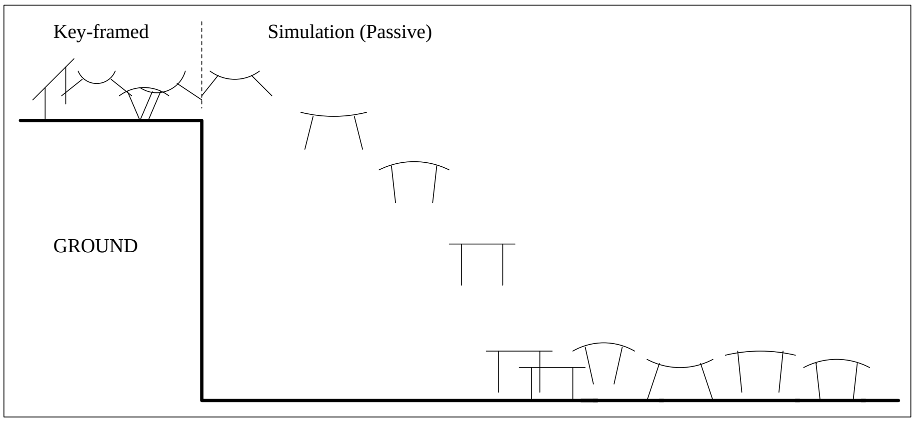

Figure 4.12: Table jumping off a cliff

In Figure 4.9 the condition number of the system matrix is plotted versus time, during a simulation. The object used is the table and the deformation modes consisted of a horizontal bend, a horizontal squash and a vertical squash. The minima represent collisions with the ground. We must note that the forces introduced by the collisions do not affect the system matrix. Thus, the instability due to the penalty method used for the collision model cannot be observed on the condition number.

In conclusion, similar deformation modes can create stability problems. The animator should be aware of this during the design of the deformation set. We plan to search for a systematic way to check the "closeness" of deformation modes.

4.11 Results

In Figure 4.10 snapshots from the animation of a pot are presented. The pot has been defined as being rigid and friction is disabled. In Figure 4.11 the same simulation was carried out with friction enabled. In Figure 4.12 the table is key-framed to jump off a cliff, then the simulator takes over. The two deformations of Figure 4.1 (bend-shear) are used in the key-framed part. In the simulated part the previous bend and a squash deformation are used.

Chapter 5

Control

A general control strategy for complex active characters is to distinguish between a high control level and a low control level. The first implements the intention of the character to perform desired tasks while the latter, driven by the high control level, drives the actuators of the character. The high control level can usually be mapped onto a finite state machine (Chapter 3).

The control of our character is based on pose controllers, which operate as a high level control system. The pose control structure operates as a finite state machine, where each state is mapped onto a pose. A pose is defined in terms of the amplitudes of the deformations modes. In other words a pose is a vector $\mathbf{p} = (\gamma_1, \ldots, \gamma_D)$, where D is the number of the deformation modes. As mentioned in Chapter 4, the modal amplitudes constitute the generalized degrees of freedom $\mathbf{q}$, we can thus equivalently say that a pose is a value of vector $\mathbf{q}$. Each state (pose) has an associated *transition* time. The latter specifies the time period for which the associated pose remains active. The control mechanism cycles continuously through the pose graph, starting with the pose defined as the first one. Currently, we have implemented only *cyclic* pose graphs which result in periodic motions. The control structure of our system is shown in Figure 5.1.

The pose control structure drives a set of PD-controllers [13]. Assuming that x is a degree of freedom of the system, a simplified form of a PD-controller is defined by:

$$f = k(x - x_0) + d\dot{x} \tag{5.1}$$

where x is the controlled variable, x_0 is the desired value for x, $\dot{x}$ is the time derivative of x

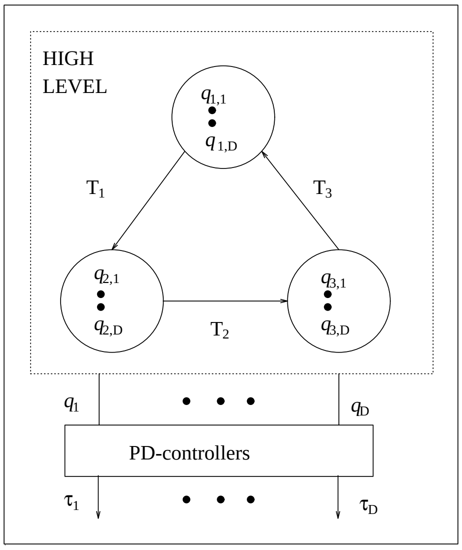

Figure 5.1: The control structure

and f is the output of the controller. If x represents a joint angle of an articulated robot, the effect over time of f is a torque that attempts to make the joint assume the position x_0. k and d are coefficients which define the stiffness and the damping of the control mechanism, respectively. The active pose drives the PD-controllers which then make the object deform into the desired shape. Typically the desired shape is never achieved exactly because of the presence of external forces. The deformation of the object into the desired pose can be done either by using potential energy terms in the Lagrangian or by directly adding deformation forces to the vector of generalized forces. We have chosen to use the second method. There is a PD-controller associated with each deformation mode, which produces a generalized control force for each deformation mode. This is expressed by

$$\mathbf{Qr} = -(\mathbf{C}(\mathbf{q} - \mathbf{q}_0) + \mathbf{D}\dot{\mathbf{q}})$$

where $\mathbf{Qr}$ are the generalized control forces, $\mathbf{C}$ is a diagonal stiffness matrix, $\mathbf{D}$ is a diagonal damping matrix and $\mathbf{q}_0$ are the desired pose parameters.

The dynamics formulation allows for any deformation mode to be designated as active or passive. An active deformation mode has associated control forces in addition to responding to external forces. The set of active deformation modes defines the motions of the character while the passive one defines additional ways in which the character reacts to external forces. In general, the active modes of deformation should be stiffer and more damped than

1. Specify an initial configuration P
2. Specify an initial temperature $T > 0$
3. Do forward simulation for time t_f and calculate the cost $Cost(P)$
4. While $T > T_{frozen}$
   (a) Pick a random neighbor P' of P
   (b) Do forward simulation for time t_f and calculate the cost $Cost(P')$
   (c) Let $\Delta = Cost(P') - Cost(P)$
   (d) If $\Delta \leq 0$ set $P = P'$
   (e) If $\Delta > 0$ set $P = P'$ with probability $e^{-\Delta/T}$
   (f) Set $T = rT$
5. Return P

Figure 5.2: Simulated annealing

the passive ones. The former should not produce oscillations since physical active motions do not often exhibit this, while the latter should produce oscillations for a better visual effect. The control of the above issues falls on the animator who can define the deformation parameters according to the desired result.

5.1 Motion Synthesis

Given an object and a set of deformation modes, it is convenient (and for complex characters necessary) to have a procedure that automatically produces controllers capable of making the object perform interesting modes of locomotion. Previous work in this area addresses the case of articulated figures (see §3.2.3) with the exception of [19, 49]. We use these methods for deformable characters. We have chosen to use *simulated annealing* to solve the motion synthesis problem. This optimization technique is reviewed in §3.2.5 and a generic algorithm is shown in Figure 3.4. In general, other optimization methods can also be used. It has been shown [19] that gradient-descent methods are often sufficient to produce good results.

The simulated annealing algorithm used is shown in Figure 5.2. The initial parameter set P specifies a pose graph chosen by the user. It consists of a cyclic pose graph and the

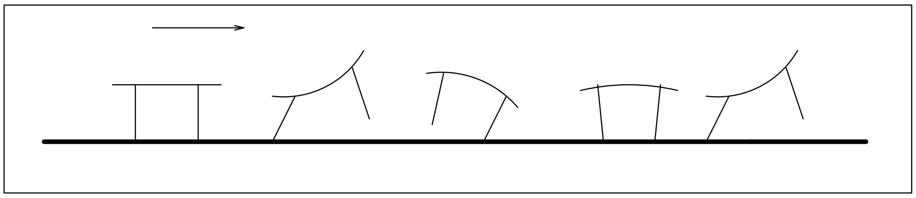

Figure 5.3: A table performing a bounding motion

associated transition times. The poses are snapshots of the internal deformations during a motion that the animator would like the character to perform. The initial state of the simulation is also important because it can affect significantly the produced motion. For example, it can delay the establishment of a steady cycle. It is advisable to use a relatively intuitive pose. Forward simulation should allow for a steady cycle to be established. The most important choice in defining an optimization process, is the choice of the cost or *optimization function*. This function is minimized or maximized by the annealing process. The optimization function characterizes the produced motions as desired or undesired. There are many parameters which serve to characterize a motion. Typical parameters are the speed, acceleration, work, orientation of the object during or at the end of the motion and the desired trajectory that the object must follow. Depending on the desired result the optimization function may involve any of the above parameters. Typically, realistic motions must consume a reasonable amount of control energy. If the latter is not included in the optimization function, the character will likely show unrealistic amounts of energy while performing the synthesized motion. At present, the design of the optimization function relies on the animator's intuition, experience and understanding of the dynamics and of the functionality of the character. The difficulty of designing the optimization function, depends on the complexity of the character and the complexity of the desired motion.

We formulate our experiments as minimization problems. The optimization functions involve combinations of a number of parameters. In all cases, the optimization function used depends linearly on the control energy used, in order to prevent the synthesis of unrealistic high-energy motions. Specific functions are presented in the next section, where we present the results of the motion synthesis experiments.

5.2 Results

Cost function	Comments
$1/q_x$	Hoping motion with unrealistic jumps
$consumed\ \ energy/q_x$	Hoping motion, normal jumps
$consumed\ \ energy/(10000000.0\sqrt{q_x\dot{q}_x})$	Bounding motion
$consumed\ \ energy/(50000000.0\sqrt{q_x\dot{q}_x})$	Faster bounding motion (Figure 5.3)
$consumed\ \ energy/(50000000.0\sqrt{q_x})$	Worm-like motion (Figure 5.6)

Table 5.1: Cost functions

The optimization can be performed with respect to different subsets of the control parameters. Some experiments with the table optimize the transition times between different poses, some the poses and some both poses and transition times. The more parameters are used in the optimization process the larger the space of admissible control functions and consequently the likelier it is to arrive at an efficient solution. This agrees with our experimental results for the table. The best motion was produced when both transition times and poses where optimized, Figure 5.3.

For most of the experiments the goal is to produce a controller that would make the table perform a bounding motion. The table is equipped with a set of four deformations modes: vertical shear, horizontal bend, vertical squash and horizontal squash. The first two are active deformation modes and the remaining two allow for additional passive motion. Our most successful experiment performed optimization on all the parameters defining the pose graph. After 60 simulation trials the table was able to perform a stable periodic mode of locomotion, shown in Figure 5.3. The cost function used, rewards the distance traveled and penalizes the energy used. Table 5.1 presents the cost functions and the associated result; q_x is the distance traveled along the x-direction and $\dot{q}_x$ is the velocity along the x-direction. If $q_x\dot{q}_x < 0$, the cost is set to a high value.

Figures 5.4 and 5.5 present the decrease of the cost as the number of forward trials increases. In these particular examples, there has been no up-hill movement during the simulated annealing process. The cost function in both cases was $consumed\ \ energy/(const * \sqrt{q_x * \dot{q}_x})$, where $consumed\ \ energy$ is the total energy consumed by the deformation forces though out the forward simulation (including the forces associated with passive deformations modes), $const$ is a constant, q_x is the total x-distance traveled by the object during the forward simulation and $\dot{q}_x$ is the velocity along the x direction at the end of the for-

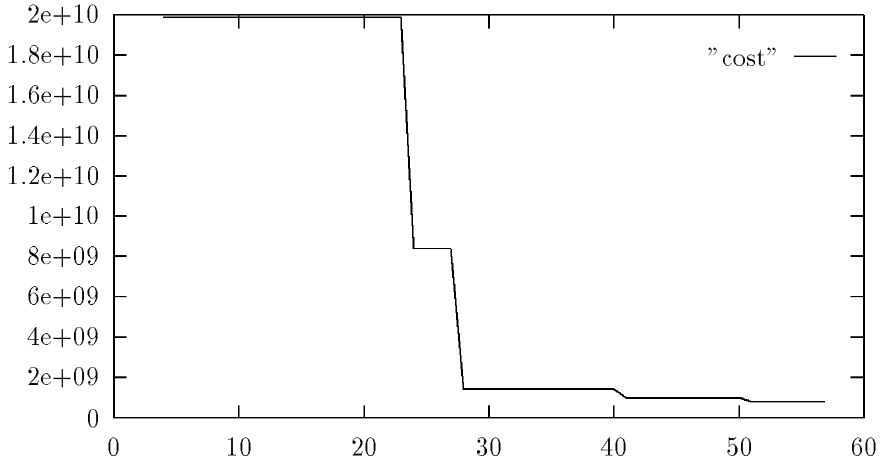

Figure 5.4: Cost function vs number of iterations (57 iterations total)

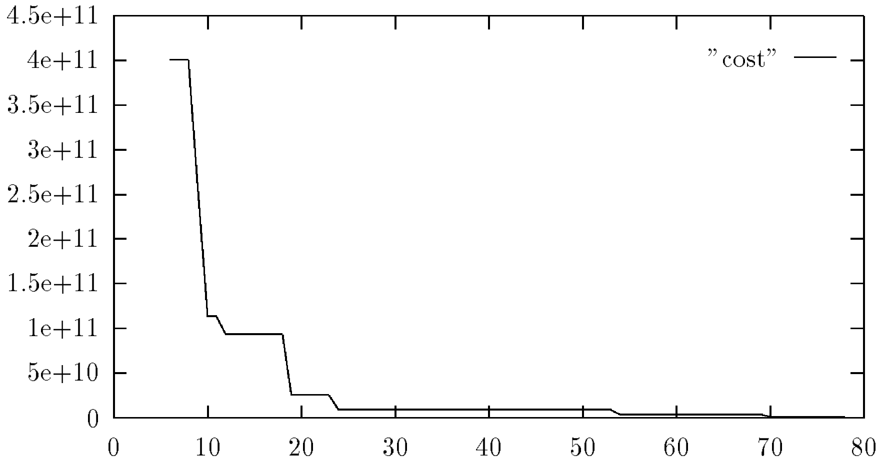

Figure 5.5: Cost function vs number of iterations (70 iterations total)

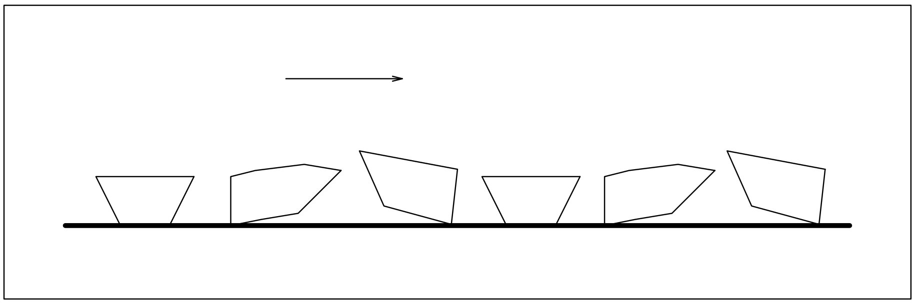

Figure 5.6: Pot performing a worm-like motion

ward simulation. The experiment associated with Figure 5.5 resulted in the synthesis of a controller that produced the motion shown in Figure 5.3.

The experiment shown in Figure 5.3 takes less than an hour on a modern graphics workstation. The simulation used a time step of 0.01 and each forward trial simulation took 1200 time steps. For a table discretized using 11 points and 4 deformation modes, each simulation trial takes 12 seconds of simulation time and 0.12 seconds of actual CPU time.

5.3 Conclusion

We have implemented a pose control technique for controlling dynamic free-form deformation models. Pose controllers allow for a wide variety of motions to be produced. Using optimization techniques, controllers can be refined to produce interesting variations of the initial motions. The experimental results allow us to be optimistic about the potential of this technique. There remain several directions to expand our work. These issues are presented in the next chapter, along with a summary of our work.

Chapter 6

Conclusion

6.1 Summary

We have presented a framework and an implementation for deformable character animation. Our approach uses parameterized deformation modes to constrain the dynamics of flexible characters. The resulting formulation is compact, efficient and resembles the formulations used for articulated figures in the sense that it is based on a restricted number of degrees of freedom. It allows control methods, previously applied to the animation of articulated figures, to be implemented for deformable models. User-defined deformation modes provide an effective means of simplifying the animation and control of flexible characters. Different objects can make use of existing sets of deformation modes.

The technique presented is particularly well suited for non articulated, traditionally passive characters, such as teapots. Such characters do not have real actuators. An intuitive way for such characters to locomote is by deforming using a limited set of deformation modes, such as stretching and squashing.

Our implementation combines key-framing, physics-based simulation, dynamic control and motion synthesis for flexible character animation. We believe the flexibility offered by a mix of techniques to be important for cartoon animation. However, the real effectiveness of our system can only be tested once placed in the hands of animators, something that has rarely been done for physics-based animation systems.

In general, our formulation could be the basis of a new system for flexible cartoon character animation, or could be incorporated in an existing system.

6.2 Future Work

The work presented in this thesis can be extended in various ways. The current formulation at present is two-dimensional. We plan to develop a three-dimensional version. The extension is straightforward since the free-from deformation formulation is easily extended to any number of dimensions. Some care must be taken for expressing rigid body rotations for which quaternions may be used.

With respect to modeling, more complex formulations can be used, for example *extended free-from deformations* [9]. A more straightforward extension would be the use of multiple lattices for different parts of an object.

Dynamic constraints [35, 57] could be implemented in order to simplify the construction of objects from existing parts. The combination of this with the use of multiple lattices will allow our formulation to be used on articulated figures. Each part of the character can have its own lattice. To complete the dynamics, a complete and efficient method for detecting and resolving collisions can be implemented. This includes collisions with both stationary and moving, rigid or deformable objects. Some of the methods described in Chapter 2 can be used.

Libraries of objects and deformations could be designed, such that the user could interactively construct new objects from existing ones and choose or blend existing deformations.

Special features can be added to the system. For example, morphing operations could be incorporated both in the key-framing and the physics-based simulation. Similarly, cartoon effects could also be taken into account. Appendix B presents the *cartoon laws of physics* [23]. As an example, one of them is quoted below:

> *All principles of gravity are negated by fear.*

These laws describe cartoon effects that have not yet been dealt with in animation systems. We intend to investigate ways of incorporating some of them in our system.

Appendix A

A simple example

This appendix contains the detailed calculations of the matrices for a simple case, a rod consisting of two point masses with equal mass, Figure A.1. The rod can translate, rotate and has a simple deformation mode. The deformation is defined by matrix **d** in equation A.1 and is associated with an amplitude a. The rod behaves like a pair of point masses connected by an ideal spring.

For this example, 4.2 has the following form:

$$\mathbf{P} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \mathbf{R} \sum_{i=0}^{3} \sum_{j=0}^{3} \left(a \begin{bmatrix} d_{ij,x} \\ d_{ij,y} \end{bmatrix} + \begin{bmatrix} L^x_{ij,loc} \\ L^y_{ij,loc} \end{bmatrix} \right) B(l_x, i) B(l_y, j), \tag{A.1}$$

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

where

$\mathbf{P}$	is an object point in world coordinates
(l_x, l_y)	are its lattice coordinates
a	is the parameter of deformation
$d_{ij,x(y)}$	is a constant which defines how much parameter a will affect the $x(y)$ coordinate of the i,j-point of the lattice
$B(l_x(l_y), i(j))$	is the Bernstein polynomial (see equation (2))
$L^{x(y)}_{ij,loc}$	is the initial $x(y)$-coordinate of the i,j-lattice point

Matrix **d** for our particular implementation is

$$[d_{ij,x}] = \mathbf{0}$$

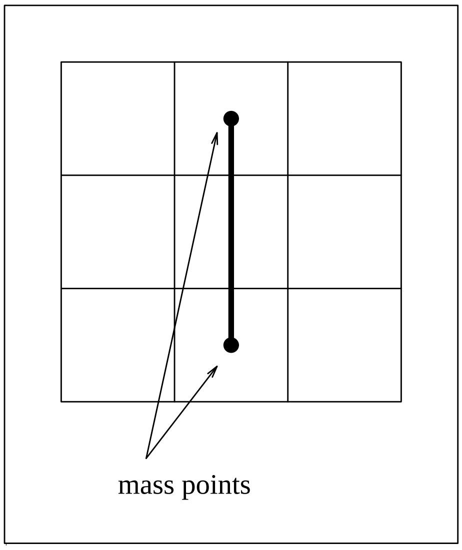

Figure A.1: The rod in the FFD lattice

$$[d_{ij,y}] = C \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{bmatrix}$$

where C is a constant that scales the deformation in accordance with the object definition. For example, in our implementation $C = -100$.

The state vector is

$$\mathbf{q}^T \begin{bmatrix} t_x & t_y & \theta & a \end{bmatrix}$$

Matrix $\mathbf{M}$ is given by (4.10) which for the rod becomes

$$\mathbf{M} = m(\mathbf{J}_1 + \mathbf{J}_2)$$

where $m = 50$Kg for our implementation. The Jacobian is given by the expression $J_{ij} = \partial x_i / \partial q_j$ which evaluates to

$$\mathbf{J}^T = \begin{bmatrix} 1 & 0 & \left[\frac{\partial \mathbf{R}}{\partial \theta} \sum_{i=0}^{3} \sum_{j=0}^{3} (a\mathbf{d}_{ij} + \mathbf{L}_{ij,loc}) B(l_x, i) B(l_y, j)\right]_x \\ 0 & 1 & \left[\frac{\partial \mathbf{R}}{\partial \theta} \sum_{i=0}^{3} \sum_{j=0}^{3} (a\mathbf{d}_{ij} + \mathbf{L}_{ij,loc}) B(l_x, i) B(l_y, j)\right]_y \end{bmatrix} \quad cont.$$

$$\begin{matrix} \left[\mathbf{R} \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{d}_{ij} B(l_x, i) B(l_y, j)\right]_x \\ \left[\mathbf{R} \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{d}_{ij} B(l_x, i) B(l_y, j)\right]_y \end{matrix} \Bigg]$$

$$= \begin{bmatrix} 1 & 0 & \left[\frac{\partial \mathbf{R}}{\partial \theta}(\mathbf{lP})\right]_x & \left[\mathbf{R}\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{d}_{ij}B(l_x,i)B(l_y,j)\right]_x \\ 0 & 1 & \left[\frac{\partial \mathbf{R}}{\partial \theta}(\mathbf{lP})\right]_y & \left[\mathbf{R}\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{d}_{ij}B(l_x,i)B(l_y,j)\right]_y \end{bmatrix}$$

where $\mathbf{lP}$ are the initial local coordinates of an object point.

We can calculate the derivatives of $\mathbf{J}$ required in the calculations of the derivatives of matrix $\mathbf{M}$, using (4.13) as follows:

$$\begin{aligned}
\frac{\partial \mathbf{J}}{\partial t_x} &= 0 \\
\frac{\partial \mathbf{J}}{\partial t_y} &= 0 \\
\frac{\partial \mathbf{J}}{\partial \theta} &= \begin{bmatrix} 0 & 0 & \left[\frac{\partial^2 \mathbf{R}}{\partial \theta^2}\mathbf{lP}\right]_x & \left[\frac{\partial \mathbf{R}}{\partial \theta}\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{d}_{ij}B(l_x,i)B(l_y,j)\right]_x \\ 0 & 0 & \left[\frac{\partial^2 \mathbf{R}}{\partial \theta^2}\mathbf{lP}\right]_x & \left[\frac{\partial \mathbf{R}}{\partial \theta}\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{d}_{ij}B(l_x,i)B(l_y,j)\right]_y \end{bmatrix} \\
\frac{\partial \mathbf{J}}{\partial a} &= \begin{bmatrix} 0 & 0 & \left[\frac{\partial \mathbf{R}}{\partial \theta}\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{d}_{ij}B(l_x,i)B(l_y,j)\right]_x & 0 \\ 0 & 0 & \left[\frac{\partial \mathbf{R}}{\partial \theta}\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{d}_{ij}B(l_x,i)B(l_y,j)\right]_y & 0 \end{bmatrix}
\end{aligned}$$

Appendix B

Cartoon Laws of Physics

The cartoon laws of physics as presented in [23], but without the comments of the original document. The document was sent to IEEE by Deborah Nordstrom after receiving it from Hari Saini (California). Nobody really knows its origin.

1. Any body suspended in space will remain in space until made aware of its situation.
2. Any body in motion will tend to stay in motion until solid matter intervenes suddenly.
3. Any body passing through solid matter will leave a perforation conforming to its perimeter.
4. The time required for an object to fall 20 stories is greater than or equal to the time it takes for whoever knocked it off the ledge to spiral down 20 flights to attempt to capture it unbroken.
5. All principles of gravity are negated by fear.
6. As speed increases objects can be in several places at once.
7. Certain bodies can pass through solid walls painted to resemble tunnel entrances; others cannot.
8. Any violent rearrangement of feline matter is impermanent.
9. A cat will assume the shape of its container.
10. Everything falls faster than an anvil.

11. For every vengeance there is an equal and opposite revengeance.

Appendix C

Interpreter Input

Typical interpreter input:

```
load_def def_pot # load deformation file
trans 500 400 0 # translation
set spring F
set debug F
set gravity T
assign 1 400000 250000
assign 2 20000000 12500000
assign 3 1200000 7500000
set codamA 250000
set spdam 0
set grconst 4500
set frconst 50
set frsin 0.1
set grdam 550
set time 0
set end 25
set dt 0.1
set_vec q 0 0 0 0 0 0
set_vec dq 0 0 0 0 0 0
simul
```

Bibliography

[1] Arunabha Bagchi. *Optimal Control of Stochastic Systems.* Prentice Hall, 1993.

[2] David Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics*, 24(4):19–28, August 1990.

[3] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics*, 26(2):303–308, July 1992.

[4] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An Introduction to Splines for Use in Computer Graphics & Geometric Modelling.* Morgan Kaufmann Inc., 1987.

[5] R. Barzel and A. Barr. Modeling with dynamic constraints. In *Tutorial 17 Notes.* SIGGRAPH, 1987.

[6] Arthur Earl Bryson and Yu-Chi. *Applied Optimal Control; Optimization Estimation and Control.* Waltham, Mass. Blaisdell Pub. Co., 1969.

[7] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered construction of deformable animated characters. *Computer Graphics*, 23(3):243–252, July 1989.

[8] Michael F. Cohen. Interactive spacetime control for animation. *Computer Graphics*, 26(2):293–302, July 1992.

[9] Sabine Coquillart. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. *Computer Graphics*, 24(4):187–193, August 1990.

[10] Sabine Coquillart and Pierre Jancène. Animated free-form deformation: An interactive animation technique. *Computer Graphics*, 25(4):23–26, July 1991.

[11] Yuval Davidor. *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization*, volume 1 of *World Scientific series in Robotics and Automated Systems*. Singapore; Teaneck, NJ : World Scientific, 1991.

[12] Lawrence Davis. *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann Publishers, 1987.

[13] Richard C. Dorf. *Modern Control Systems*. Addison-Wesley,Reading, 1989.

[14] Eugene Fiume and Marc Ouellette. On distributed, probabilistic algorithms for computer graphics. *Graphics Interface*, pages 211– 218, 1989.

[15] James Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics Priciples and Practice*, chapter 21. Addison-Wesley Publishing Company Inc., second edition, 1990.

[16] Marie-Paule Gascuel. An implicit formulation for precise contact modeling. In *Computer Graphics Proceedings*, Annual Conference Series, pages 313–320, 1993.

[17] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., 1989.

[18] H. Goldstein. *Classical Mechanics*. Addison-Wesley, 2nd edition, 1980.

[19] Radek Grzeszczuk. Automated learning of muscle-based locomotion through control abstraction. Master's thesis, University of Toronto, 1994.

[20] William W. Hager. *Applied Numerical Linear Algebra*. Prentice Hall, Englewood Cliffs, New Jersey 07632, 1988.

[21] J. K. Hodgins and M. H. Raibert. Biped gymnastics. *International Journal of Robotics Research*, 9(2):115–132, April 1990.

[22] J. H. Holland. Adaptation in natural and artificial systems. Ann Arbor,MI: University of Michigan Press, 1975.

[23] Cartoon laws of physics. Article in After 5 At The Institute, October 1994.

[24] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation part i. Distributed as course material.

[25] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[26] J. Lasseter. Principles of traditional animation applied to 3-d computer animation. *Proceedings of SIGGRAPH '87*, 21(4):35–44, 1987.

[27] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[28] Jerry B. Marion and Stephen T. Thornton. *Classical Dynamics of Particles and Systems*. Harcourt Brace Jovanovich, Publishers, third edition, 1988.

[29] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. *Proceedings of SIGGRAPH '90*, 22(2):29–38, August 1990.

[30] G. S. P. Miller. The motion dynamics of snakes and worms. *Proceedings of SIGGRAPH '88*, 22(4):169–178, August 1988.

[31] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, August 1988.

[32] J. Thomas Ngo and Joe Marks. Spacetime constraints revisited. *Computer Graphics Proceedings, Annual Conference Series*, pages 343–350, August 1993.

[33] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, N.J., 1982.

[34] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, July 1989.

[35] John C. Platt and Alan H. Barr. Constraint methods for flexible models. *Proceedings of Siggraph*, 22(4):279–288, August 1988.

[36] William H. Press, William T. Vetterling, Saul A. Teukolsky, and Brian P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[37] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *Proceedings of SIGGRAPH '91*, 25(4):349–358, July 1991.

[38] J. D. Schaffer. *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann Publishers Inc., June 1989.

[39] Thomas W. Sederberg and Scott R. Parry. Free-form deformations of solid geometric models. *Proceedings of Siggraph*, 20(4):151–160, August 1986.

[40] Karls Sims. Artificial evolution for computer graphics. *Computer Graphics*, 25(4):319–328, July 1991.

[41] John M. Snyder, Adam R. Woodbury, Kurt Fleischer, Bena Currin, and Alan Barr. Interval methods for multi-point collisions between time-dependent curved surfaces. In *Computer Graphics Proceedings*, Annual Conference Series, pages 321–334, 1993.

[42] A. J. Stewart and J. F. Cremer. Beyond keyframing: An algorithmic approach to animation. *Proceedings of Graphics Interface '92*, pages 273–281, 1992.

[43] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2):185–194, July 1992.

[44] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *Visual Computer*, 4:306–331, 1988.

[45] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[46] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, July 1987.

[47] Demetri Terzopoulos, John Platt, and Kurt Fleischer. Heating and melting deformable models. In *Graphics Interface '89*, pages 219–226, June 1989.

[48] Demetri Terzopoulos and Hong Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, April 1994.

[49] Demetri Terzopoulos, Xiaoyuan Tu, and Radek Grzeszczuk. Artificial fishes with autonomous locomotion, perception, behavior and learning in a simulated physical world. In *ARTIFICIAL LIFE IV Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, Complex Adaptive Systems series, pages 17–27. The MIT Press, 1994.

[50] David Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface '91*, pages 255–262, 1991.

[51] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception and behavior. *Computer Graphics Proceedings, Annual Conference Series*, pages 43–50, July 1994.

[52] M. van de Panne and E. Fiume. Sensor-actuator networks. *Computer Graphics Proceedings, Annual Conference Series*, pages 335–342, August 1993.

[53] M. van de Panne, E. Fiume, and Z. Vranesic. Physically based modelling and control of turning. *CVGIP:Graphical Models and Image Processing*, 55(6):507–521, November 1993.

[54] Michiel van de Panne. Parameterized gait synthesis for virtual windup toys. In review, 1994.

[55] Michiel van de Panne, Ryan Kim, and Eugene Fiume. Synthesizing parameterized motions. *Fifth Eurographics Workshop on Animation and Simulation, at Oslo*, September 1994.

[56] Michiel van de Panne, Ryan Kim, and Eugene Fiume. Virtual wind-up toys for animation. *Graphics Interface*, pages 228–215, 1994.

[57] A. Witkin, K. Fleischer, and A. Barr. Energy constraints on parameterized models. *Computer Graphics*, 21(4):225–232, July 1987.

[58] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. *Computer Graphics*, 24(2):11–22, March 1990.

[59] Andrew Witkin and Michael Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.

[60] Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. *Computer Graphics*, 24(4):243–252, August 1990.

[61] N. M. J. Woodhouse. *Introduction to Analytical Dynamics*. Oxford Science Publications, 1987.

[62] D. Zeltzer. Motor control techniques for figure animation. *IEEE Computer Graphics and Applications*, pages 53–59, November 1992.