An Automatic Collision Response Algorithm

by

Sonja Struben

B.Sc., University of Calgary, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

we accept this thesis as conforming
to the required standard

The University of British Columbia

August 1998

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

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date Aug. 31, 1998

DE-6 (2/88)

Abstract

Many animations depict two or more objects interacting and potentially colliding. Collision response is a complex process if the objects are intended to respond like soft bodies and to exhibit the properties of real objects. Physically-based models calculate contact forces to incorporate into the calculation of velocities and positions of the control mesh. Some physically-based models, for example those that model cloth, strive for visually realistic results. Until recently the magnitude of the calculations required for physically-based modeling have precluded real-time interaction. A complaint with physically-based models is the correlation between the parameters, such as forces and torques, and the resulting 'look' of the response are sometimes difficult for the user to understand.

The work presented in this thesis does not strive for the simulation of real object properties. Instead it tries to remove the interpenetration between two objects while providing a set of controls for the animator to adjust the 'look' of the collision response. A set of data points within the interpentration region of the two colliding objects is determined by the algorithm and each object interpolates those data points to remove the interpenetration. The position of the data points is a function of the relative rigidity of the two objects. Locality or globality of the response is achieved by allowing the user to specify the amount of response absorbed by different levels of a hierarchical B-spline modeling primitive. Combinations of deformational, translational and rotational collision response mechanisms give more

options for the look of the response. Empirical results suggest the algorithm's computation time is small enough to allow for a fast preview of the animation, even for moderately complex geometry.

Contents

Abstract ii

Contents iv

Acknowledgements vi

1 Introduction 1

1.1 Motivation 1

1.2 Problem Statement 2

1.3 Thesis Overview 2

2 Previous Work 3

2.1 Physically-Based Modeling 3

2.2 Free Form Deformations 6

2.3 Displacement Schemes 8

2.4 Collision Response and Animators 10

2.5 ACRA's Goals 10

3 Parametric Curve and Surface Fundamentals 12

3.1 Parametric Curve Overview 13

3.2 B-Spline Curves 14

3.3 B-Spline Surfaces 18

3.4 Hierarchical B-Spline Surfaces 20

4 Collision Response Algorithm **25**

4.1 Multi-Resolution Interpolation 25

4.1.1 Curve Interpolation 26

4.1.2 Surface Interpolation 28

4.1.3 Hierarchical Surface Interpolation 30

4.2 Using Multi-Resolution Interpolation for Collision Response 33

4.2.1 Rigidity 33

4.2.2 Deformation, Translation and Rotation 34

4.2.3 Deformational Response 37

4.3 Translational and Rotational Response 38

4.3.1 Translational Response 39

4.3.2 Rotational Response 39

4.3.3 Restriction 40

4.3.4 Prolongation 41

4.4 Summary 42

5 Results and Future Work **43**

5.1 Implementation and Empirical Results 43

5.2 Goals Revisited 44

5.3 Effectiveness of the Algorithm 45

5.4 Future Work 56

6 Summary **58**

Bibliography **60**

Glossary **64**

Index **66**

Acknowledgements

First and foremost I'd like to acknowledge my supervisor David Forsey. He was there from the start with suggestions and in turn willing to listen to my half-thought out ideas. More than once he was there to help me get over my self-doubt or to let me explore a topic on my own. Jason Harrison deserves a hearty thanks for his thorough review of my thesis and all the helpful articles he sent my way. I'm also indebted to my second reader, Alain Fournier, for both his helpful thesis suggestions and for guiding me through the topic of physically-based modeling. I had the pleasure of interviewing some animators at Totally Hip Software and their suggestions both confirmed some of my own opinions and helped clarify what is really important for them. Specifically I'd like to thank Greg McConnell, Trevor Bentley, Ann-May Rhodes, Anthony Law and especially Brian Leeners for organizing the interview.

Finally my love and thesis dedication go to my husband David Marwood. I would never have had the courage to complete this work without his support. Thank you David.

SONJA STRUBEN

The University of British Columbia

August 1998

Chapter 1

Introduction

1.1 Motivation

The main goal of computer animation is to achieve a desired effect such as laughter, enthralment or sorrow on the viewer. An animator approaches the task of creating the animation with a set of ideas in his or her mind or on paper. The ease or realizing those ideas is heavily influenced by the capabilities of the available tools.

Physically-based modeling systems generate responses to collisions by calculating the contact forces during a collision and then incorporating these forces into the position and velocity calculations of each object's control mesh. Although some animations of physically-based models are very visually appealing, the complexities of creating such a model and manipulating the model's parameters to achieve a specific look are sometimes daunting. Most animators still rely on manually adjusting the parameters that directly control an object's shape to resolve the interpenetration of two objects. While the control over the resulting shape is excellent, the effort and time required to resolve the interpenetration over many frames is considerable. Resolving collisions in this manner is a tedious task and the motivation for this work is the desire to relieve this tedium through the development of a better tool for basic collision response.

1.2 Problem Statement

By basic collision response we mean the removal of the interpenetration between the objects, or at least enough of the interpenetration so that only a small amount of post-processing 'clean-up' removes the remainder. How real objects change shape during a collision is based on many factors including rigidity and structure. The difficulty of providing a collision response algorithm that simulates real objects is apparent when one considers the vast variety of deformations exhibited by real objects.

This work describes a tool that provides a variety of controls for the animator to adjust the look of a basic collision response. One of its goals is to generate the responses fast enough so that the animator can quickly preview the animation. It is believed that a quick preview of the collision response will give the animator a sense of motion flow and allow for iterative adjustment of the collision response parameters.

1.3 Thesis Overview

The next chapter presents a description of previous work in this field, an animator's perspective on the collision response problem and a summary of the goals of the collision response algorithm. The fundamentals of parametric curves, surfaces and hierarchical B-spline surfaces in Chapter 3 provide a common basis for the collision response algorithm described in Chapter 4. Much like the animator's process, the collision response algorithm moves surface control vertices to resolve the interpenetration of two objects. Chapter 4 describes how the control vertices will be moved, the controls over the 'look' of the response. Some results, both empirical and visual are presented in Chapter 5 and an overall summary is given in Chapter 6.

Chapter 2

Previous Work

Animating complex deformable objects is tedious work for the animator because the traditional method of choice is to manipulate surface shape parameters at each frame. A large variety of schemes have been proposed to aid in the animation of deformable objects. Some schemes try to simulate specific objects such as cloth or simulate material properties such as plasticity or wrinkling. The driving force behind the research in this area today is for algorithms that are fast, flexible, easy to control and/or produce visually realistic results.

This chapter will describe three different areas of research that try to tackle the problem of animating deformable objects. A section summarizing an interview with a group of animators and their specifications for a deformation algorithm follows. Finishing the chapter is a description of the automatic collision response algorithm's (ACRA) goals.

2.1 Physically-Based Modeling

By *physically-based modeling* we mean any system that uses some laws of physics to iteratively calculate the positions, velocities, forces or other properties of the objects. Using a physically-based model to simulate the motion or shape of objects

has many advantages and is the foundation for much of the work done in robotics and simulation. Two of the main benefits of animations using physically-based models is the visual realism and automatic object motion. The specification of key-framed motion is not needed, provided the parameters of the system are set up correctly. There is a double-edge to physically-based models in the sense that exaggerated or unrealistic effects are difficult to create and the affect of changing system parameters can be difficult to predict.

Much work has been invested for developing schemes to animate rigid bodies [3] [4] [27]. The animation of non-rigid bodies opened a new realm of animation problems, specifically the deformation of a non-rigid body such as cloth [37] [28] [21] [30] [36] [5] during and after a collision. Most cloth models approximate cloth with a deformable surface composed of a uniform grid of point masses. A typical point mass is connected to its neighbors by springs. External forces act on the point masses to deform the surface while the internal elastic forces from the springs maintain cohesion. The change in velocity and position of a point mass due to internal and external forces is calculated using numerical methods to solve the differential equation specified by Newton's law. A numerical method calculates the change in position or velocity during a discrete time step. The position or velocity is incremented by the calculated changed and used as the initial value for the next time step. The size of the time step and the numerical method chosen impacts the behavior of the dynamics[1] system since a large time step could cause the numerical solution to be inaccurate or unstable whereas a small time step increases the number of calculations. Baraff and Witkin [5] have shown that the use of implicit methods and adaptive time steps can decrease the system's calculation time.

Several physically-based models have been introduced specifically for the purposes of simulating properties or visual effects such as elasticity, inelasticity, garment wrinkle formation [22], garment crumpling[33] and dressing a virtual human

[1] *Dynamics* refers to the application of Newton's laws to a system to calculate positions and velocities.

[12] [39]. Terzopoulos and Fleischer [32] simulate three different inelastic behaviors, viscoelasticity, plasticity and fracture. An object that exhibits *elastic* properties will return fully to its original shape when all external forces are removed. If an object returns to its original shape slowly or only partially then the object is said to be *inelastic*. Inelastic deformations may depend on the history of the applied forces. A material whose behavior includes the characteristics of a viscous fluid together with elasticity exhibits *viscoelastic* properties (e.g. Silly Putty flows under sustained force but also bounces elastically when subjected to bursts of force). The amount that a material sustains permanent deformations is a rating of its *plasticity*. *Fracture* occurs when a material deforms beyond a certain limit. Cracks develop according to internal force or deformation distributions and their propagation is affected by local variations in material properties. Terzopoulos and Fleischer's physically-based model is similar to a cloth model in that the object is a grid (possibly 3D) of points connected by combinations of *units*. The three units described are an elastic unit that acts like a spring, a viscous unit whose rate of deformation is proportional to the force, and a plastic unit that does not respond to a force until a certain threshold is reached. For example, the simulation of viscoelasticity is achieved by modeling the connection in the grid by a combination of an elastic unit and a viscous unit, since this combination simulates internal forces that depend on the deformation magnitude and rate.

Several researchers have developed models with the specific purpose of generating a visual effect. Kunii and Gotoda [22] presented a cloth model that generates wrinkles as it deforms. Their cloth model is a mass/spring mesh including diagonal springs between point masses. They claim that certain characteristic points on the cloth will describe the shape of the wrinkles while moving and manipulating these points animates the wrinkles. By perturbing the position of a point mass the energy of the system will change and any perturbation that lowers the overall energy is retained. Perturbing multiple point masses will form wrinkles in the cloth as the

system reaches a lower energy state.

Wu et. al [38] investigated a variant of the wrinkle formation problem for the purposes of generating facial wrinkles. Their physically-based skin model has a muscle layer, a fat tissue layer and a skin mesh. The skin mesh is connected to the muscle layer through springs which represent connective fat tissue. Revolution surfaces, such as cylinders or ellipsoids, represent muscles. Movement of the muscles causes the skin to deform which in turn causes wrinkles to form. As a person gets older the skin's elasticity decreases while the plasticity increases. Their model also takes into account a plastic-visco-elastic process that changes the skin's rest position to simulates wrinkling as a person ages.

Proponents of physically-based modeling claim that models based on physical laws can generate motion and deformation from a set of initial conditions and a set of applied forces over time. While physically-based modeling systems have generated some exceptionally realistic animations, animators have indicated that specifying the forces, constraints and parameters for a *particular* motion is often nonintuitive.

2.2 Free Form Deformations

Free form deformations were first introduced by Sederberg and Parry [31] as a method for deforming a complex model within a parallelepiped region of space. In short, any object within the Bézier solid that defines the parallelepiped can calculate new positions for its vertices such that the object deforms as the parallelepiped deforms. The deformation process (adapted from [34]) is:

- Impose a local coordinate system within the parallelepiped. Any point $\mathbf{P}$ within the parallelepiped can be specified by,

$$\mathbf{P} = \mathbf{O} + u\vec{u} + v\vec{v} + w\vec{w} \tag{2.1}$$

where **O** is the origin. The (u,v,w) coordinates, given in Equation 2.2, are the parametric coordinates of **P** in the parallelepiped.

$$
\begin{aligned}
u &= \frac{(\vec{v} \times \vec{w}) \cdot (\mathbf{P} - \mathbf{O})}{(\vec{v} \times \vec{w}) \cdot \vec{u}} \\
v &= \frac{(\vec{u} \times \vec{w}) \cdot (\mathbf{P} - \mathbf{O})}{(\vec{u} \times \vec{w}) \cdot \vec{v}} \\
w &= \frac{(\vec{u} \times \vec{v}) \cdot (\mathbf{P} - \mathbf{O})}{(\vec{u} \times \vec{v}) \cdot \vec{w}}
\end{aligned} \tag{2.2}
$$

- Define a three dimensional lattice of control points within the parallelepiped.

$$\mathbf{F_{i,j,k}} : 0 \leq i \leq l,\ 0 \leq j \leq m,\ 0 \leq k \leq n \tag{2.3}$$

where

$$\mathbf{F_{i,j,k}} = \mathbf{O} + \frac{i}{l}\vec{u} + \frac{j}{m}\vec{v} + \frac{k}{n}\vec{w}. \tag{2.4}$$

- Deform the lattice of control points to new points $\mathbf{F'_{i,j,k}}$.

- Calculate the deformed points $\mathbf{P'}$ from the deformed lattice and the trivariate Bézier function using the same parametric coordinates from Equation 2.2.

$$\mathbf{P'} = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} \mathbf{F'_{i,j,k}} B_i^l(u) B_j^m(v) B_k^n(w) \tag{2.5}$$

A character animation system developed by Chadwick et. al [13] uses free form deformations to animate secondary features like muscle bulging. Their character model has four layers, a motion specification layer, a skeleton, a muscle layer and a skin and clothing layer. The animator controls the motion of the character by kinematically specifying joint angles and constraints on the skeleton layer. Interaction between the layers is specified by parameters and constraints set by the animator. Because each layer uses the specified constraints to drive the motion of the next layer the computer can relieve the animator of the burden of managing the interaction between the layers. Additional work required to create the character model and specify the constraints between the layers is recovered by easier

specification of the animation. A muscle primitive is represented as a pair of adjoining free form deformations. The skin layer, which is the actual geometric layer that is rendered, is contained within the muscle layer and it deforms as the muscle undergoes a free form deformation. Squash and stretch behavior in the form of bulging and bending muscles is automatically calculated from the kinematic state of the articulated skeleton. Soft body structures such as flesh or fatty tissue are simulated through free form deformations controlled by dynamics. The lattice of the free form deformation surrounding the soft body structure is mapped to point masses. A dynamic simulation is applied to the point masses and their resultant motion is mapped back to the free form deformation to drive the muscle deformations. Unfortunately the real-time interactivity of the free form deformations and usability of the system are not commented on so the applicability of the algorithms for general object deformation is in unknown.

The work by Zheng et. al [40] provides some tools for easily modifying the shape of free form curves. User defined sculpting tools are pushed into the free form curve to alter its shape. For finer control the region of the curve that is deformed can be limited to a user defined span. Sculpting tools to deform a curve is an intuitive and appealing interface but would not be applicable for ACRA since an extension of the method to surfaces is not presented and the burden of adjusting surface shape at each animation frame remains.

Free form deformations provide a way to calculate the deformation of an object inside a deformed parallelepiped with the deformation of the parallelepiped depending either on a dynamics system or animator manipulation.

2.3 Displacement Schemes

Displacement schemes are a new approach to modeling deformations without the use of a physical model or dynamics. They strive for visually convincing deformations, not the simulation of physically realistic deformation. Several ideas from the three

projects presented in this section are incorporated into ACRA.

Gascuel and Desbrun [10] [11] generate object deformation through an implicit surface layer that coats an internal model that is either a mass/spring network, an articulated structure composed of one or more rigid objects, or a particle system. Their scheme has three main benefits: the implicit surface provides efficient collision detection, an exact contact between colliding objects is found and the object volume is preserved. During a collision a negative field term, which models compression, is added to the portion of the surface within the interpenetration region. A propagation region is defined around the interpenetration region that gains a positive field term to simulate the transverse propagation of the deformations (i.e. bulging). The key difference between this technique and the ones mentioned in the previous sections is that the deformation itself is not generated through dynamics but the resultant deformation can be used to calculate a response force. One drawback of their method is the inherently rounded or blobby look of implicit surfaces.

Palazzi [29] adapts the idea of displacement constraints from Gascuel [18] to animate deformable objects. Deformable objects are modeled with rigid line segments connected at their end points to form a grid. The system streamlines the dynamic computations by treating each rigid line segment independently without imposing cohesion constraints. External forces acting on each rigid line segment are found and used to calculate the new positions and orientations of the line segments. A second step re-links the line segments by enforcing the constraints acting on the grid. Palazzi introduced a multilevel approach to distribute the external forces across the object. His deformable object is a hierarchy of grids where different levels of the hierarchy are assigned portions of the external force to generate a spectrum of local to global behavior.

Harrison [20] used a piecewise linear multi-resolution surface to model deformable objects. His deformations are kinematically driven in the sense that displacements, calculated from the interpenetration of two objects, move specific sur-

face points towards specific goal points at a certain rate. Different amounts of the displacement are absorbed by different levels in the hierarchy to generate a spectrum of local to global deformations. His method is fast and easy to control but lacks the smoothness of a spline surface.

2.4 Collision Response and Animators

Four animators [1] were interviewed to survey their opinions about animating deformable objects and the features they would find useful. Unanimously the most prevalent concern was maintaining control over the animation. Any system that imposed movements or deformations on the objects without a mechanism for overriding the movements was considered unusable. When specifically questioned about physically-based models and dynamics systems they indicated such features would be useful provided the parameters were easy to specify and the results from the dynamics system could be overridden if desired. Furthermore, the animators already have a plan for how the objects will be moved and react before they begin the animation process so a system that calculates motion is neither needed nor desired. Beneficial features are a fast preview of the response, controls that provide a spectrum of deformations and a database of default configurations that could be adjusted for customized behavior.

2.5 ACRA's Goals

Traditionally, animators manipulate the control vertices of the deforming objects at each frame or employ morphing techniques. These techniques provide a large amount of control over the progression of the deformation but require the animator to generate all motions of the animation by hand.

The automatic collision response algorithm (ACRA) is a prototype that is designed to try and remove the interpenetration between two objects as opposed

to simulating real object responses. It falls in with the other displacement schemes of Section 2.3 to provide a middle ground between physically-based systems and control point manipulation done by hand. The following goals were chosen to make ACRA a fast and flexible tool that an animator could use for basic collision response which can later be enhanced by hand if desired.

- **Unintrusive.** The key-framed motion or position of objects should not be altered by the algorithm. If the animator wants an object to change position or velocity after the collision then it is the animator's responsibility to key frame such motion. Consequently, any non-rigid object can deform indefinitely since the system does not generate fractures. If two non-rigid objects interact, the one with the highest rigidity will deform less. This provides the animator with a tool capable of generating unrealistic deformations.

- **Fast previews.** The algorithm should try to minimize the computation time so that the animator only waits a small amount of time ($\approx$ 1 second/frame) to preview the animation. ACRA uses two strategies to minimize its computation time. First, the algorithm does not need to maintain the state of the system so it is only executed at the time of the collision and possibly afterwards if the algorithm is extended to return the object back to tits rest shape. Second, ACRA narrows its collision response to the portion of the object that is interpenetrating with the other object.

- **Broad spectrum of responses.** A multi-resolution, smooth surface is a good choice for the modeling primitive used by ACRA. The multi-resolution aspect of the surface can be exploited to generate a spectrum of local to global responses.

- **Animator Override.** If the broad spectrum of responses still does not produce the desired look then the surface shape should be adjustable by the animator.

Chapter 3

Parametric Curve and Surface Fundamentals

Several excellent textbooks have been written on the subject of parametric curves and surfaces [7, 8, 14, 15] thus this chapter will only overview the fundamentals of the topic and establish a common notation. First, properties of general cubic parametric curves are described and some common types of curves are mentioned. Since the modeling primitive used by ACRA is based on B-splines the properties of B-spline curves and the extension to surfaces are covered. To lead into the chapter on ACRA, the last section describes hierarchical splines and previews their utility for collision response.

Although the algorithm in Chapter 4 describes how a collision response can be achieved if the modeling primitive is a hierarchical B-spline surface, the underlying ideas could be applied to other primitives such as Bézier surfaces or piecewise linear surfaces. In fact, much of this work is an exploration of the specific benefits of using hierarchical B-spline surfaces for collision response since Harrison [20] has already shown how to model deformable objects with piecewise linear multi-resolution surfaces.

3.1 Parametric Curve Overview

The most basic of modeling primitives are points and lines. A *piecewise linear* approximation is a series of line segments that approximates the shape of other primitives such as curves. A curve can also be approximated by a *piecewise polynomial* representation where each segment $\mathbf{Q(t)}$ is given by three functions $x(t)$, $y(t)$ and $z(t)$ and the parameter t varies over the interval $[0, 1]$. This is known as a *parametric representation* because any point on the curve is represented by a single parameter. Cubic polynomials are often used for the three functions of $\mathbf{Q(t)}$ because they are the lowest order polynomial that can interpolate two endpoints and specify tangents at each endpoint. Higher order polynomials require more conditions to control both the shape and the additional inflection points that can cause undesirable artifacts or "wiggles". ACRA uses a modeling primitive based on cubic splines so the remainder of the chapter will be restricted to splines of degree 3.

A cubic curve segment $\mathbf{Q(t)} = [x(t)\ y(t)\ z(t)]$ is defined by the following cubic polynomials,

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x, \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y, \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z, \quad 0 \le t \le 1. \end{aligned} \tag{3.1}$$

A succinct matrix form for the curve segment is,

$$\mathbf{Q(t)} = [x(t)\ y(t)\ z(t)] = T \cdot C \tag{3.2}$$

where

$$\begin{aligned} T &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \\ C &= \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \end{aligned} \tag{3.3}$$

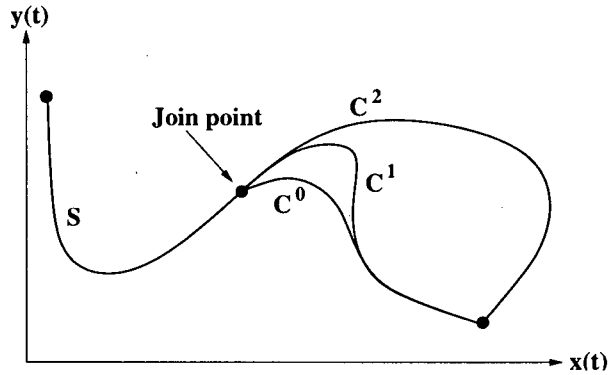


Figure 3.1: A two-dimensional curve with two segments. The difference in segment shape is shown when the join point has C^0, C^1 and C^2 continuity. [Source: [15], p. 481]

The parametric tangent vector of the curve is the derivative of $\mathbf{Q(t)}$ with respect to the parameter t. If the tangent vectors at the join point of two adjacent curve segments are equal in both direction and magnitude then the curve is C^1 continuous. In general, if each derivative $d^n/dt^n[\mathbf{Q(t)}]$ through to the nth derivative is equal in magnitude and direction at the join point then the curve is C^n continuous. Figure 3.1 shows two curve segments where the join point exhibits C^0, C^1 or C^2 continuity.

There are several methods for specifying the four constraints on curve shape: *Hermite* curves define two endpoints and two endpoint tangent vectors, *Bézier* curves define two endpoints and two additional points to control the endpoint tangent vectors and *B-splines* define four control points which the curve does not necessarily pass through.

3.2 B-Spline Curves

Maintaining C^0, C^1 and C^2 continuity is easier with a B-spline than Hermite and Bézier curves since a B-spline segment shares control points with adjacent segments. A B-spline curve with $m+1$ control vertices, $\mathbf{V_0}, \mathbf{V_1}, \ldots, \mathbf{V_m}$, $m \geq 3$, has $m-2$

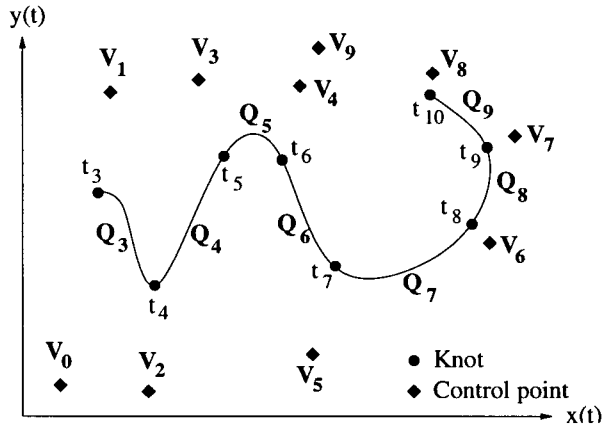


Figure 3.2: A B-spline curve with 7 segments. [Source: [15], p. 492]

segments, $\mathbf{Q_3}, \mathbf{Q_4}, \ldots, \mathbf{Q_m}$. The parameter range for a segment $\mathbf{Q_i}$ is $t_i \leq t < t_{i+1}$, for $3 \leq i \leq m$. A *knot* is a join point between $\mathbf{Q_{i-1}}$ and $\mathbf{Q_i}$ and has a parameter value, or *knot value*, of t_i. The endpoints of the curve segment have knot values of t_3 and t_{m+1} and are also called knots. A diagram of a two-dimensional B-spline curve illustrating the knots and control points is shown in Figure 3.2.

The coefficient matrix of equation 3.3 is usually rewritten as $C = M \cdot G$, where M is a 4×4 *basis matrix* and G is a 4×1 *geometry vector*. A segment is a weighted sum of the elements in the geometry vector where the weights are cubic polynomials of t. These cubic polynomials, known as *blending functions* or *B-spline basis functions*, have a matrix form $B = T \cdot M$. It is important to recognize that the elements of the geometry vector are the four variables that control the curve's shape. During an affine transformation only the geometry vector needs to be transformed since the curve is generated from the vector. Equation 3.4 shows the B-spline geometry vector $G_{B_{S_i}}$ for a segment $\mathbf{Q_i}$ with four control vertices defining

the segment.

$$G_{B_{S_i}} = \begin{bmatrix} \mathbf{V_i} \\ \mathbf{V_{i+1}} \\ \mathbf{V_{i+2}} \\ \mathbf{V_{i+3}} \end{bmatrix}, \; 0 \leq i \leq m-3. \tag{3.4}$$

With the B-spline basis matrix M_{B_S} given by Equation 3.5 a curve segment $\mathbf{Q_i(t)}$ is calculated in Equation 3.6.

$$M_{B_S} = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}. \tag{3.5}$$

$$\begin{aligned} \mathbf{Q_i(t)} &= T \cdot M_{B_S} \cdot G_{B_{S_i}}, \\ &= \frac{(1-t)^3}{6}\mathbf{V_i} + \frac{3t^3 - 6t^2 + 4}{6}\mathbf{V_{i+1}} + \frac{-3t^3 + 3t^2 + 3t + 1}{6}\mathbf{V_{i+2}} \\ &\quad + \frac{t^3}{6}\mathbf{V_{i+3}}, \;\; 0 \leq t < 1 \end{aligned} \tag{3.6}$$

The B-spline blending functions B_{B_S} are easily picked out from Equation 3.6.

$$\begin{aligned} B_{B_S} &= [B_0 \; B_1 \; B_2 \; B_3] \\ &= \left[\frac{(1-t)^3}{6} \;\; \frac{3t^3 - 6t^2 + 4}{6} \;\; \frac{-3t^3 + 3t^2 + 3t + 1}{6} \;\; \frac{t^3}{6}\right], \; 0 \leq t < 1 \end{aligned} \tag{3.7}$$

As shown in Figure 3.3, the B-spline blending functions exhibit the important property of being everywhere nonnegative and summing to 1. These conditions indicate each segment is contained within the convex hull of its four control points. Detecting collisions between two curves is simplified because non-colliding curves are quickly eliminated by testing the intersection of their convex hulls.

Subdivision is a key concept when dealing with hierarchical surfaces so the topic will be introduced here. Simply put, a segment is subdivided into two segments

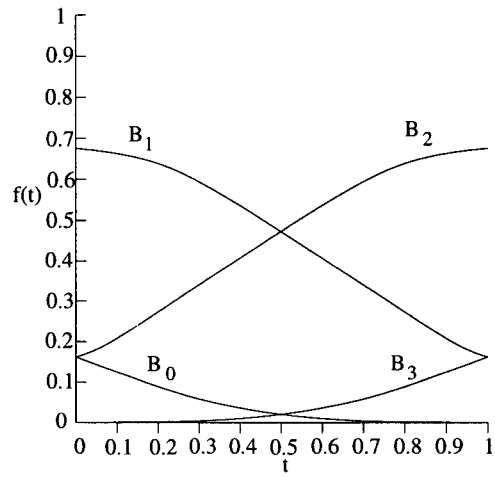


Figure 3.3: The B-spline blending functions. Notice only B_3 is zero at $t = 0$ and B_0 is zero at $t = 1$. This indicates the segment before $t = 0$ will be influenced by $\mathbf{V_i}, \mathbf{V_{i+1}}, \mathbf{V_{i+2}}$ but not $\mathbf{V_{i+3}}$. Similarly the segment after $t = 1$ will be influenced by $\mathbf{V_{i+1}}, \mathbf{V_{i+2}}, \mathbf{V_{i+3}}$ but not $\mathbf{V_i}$.

for the purposes of adding control vertices and gaining a finer control over the curve's shape. For example, a Bézier segment can be subdivided into two segments, both of which are coincident on the original segment and share only one end point. As shown in Figure 3.4, the original segment has control vertices $\mathbf{V_i}$, $0 \leq i \leq 3$ and the two new segments have control vertices $\mathbf{L_i}$ and $\mathbf{R_i}$ respectively. A total of five new control vertices (since $\mathbf{V_0} = \mathbf{L_0}$, $\mathbf{V_3} = \mathbf{R_3}$ and the endpoints $\mathbf{L_3}$ and $\mathbf{R_0}$ are shared) replace the two old vertices $\mathbf{V_1}$ and $\mathbf{V_2}$. For B-splines a similar scheme is derived in [15] where the four control points of the original segment are replaced by five new control points. Unfortunately the segments adjacent to the one being subdivided are still defined by some of the original control points so changing the position of any of the five new control points or the four old control points will cause a crack in the spline, as shown in Figure 3.5. The Oslo algorithm [7] adds knots to the knot sequence and finds a new set of control vertices that represents the same curve. This takes us out of the realm of uniform B-splines since the difference between knot values is not necessarily constant. Section 3.4 will show

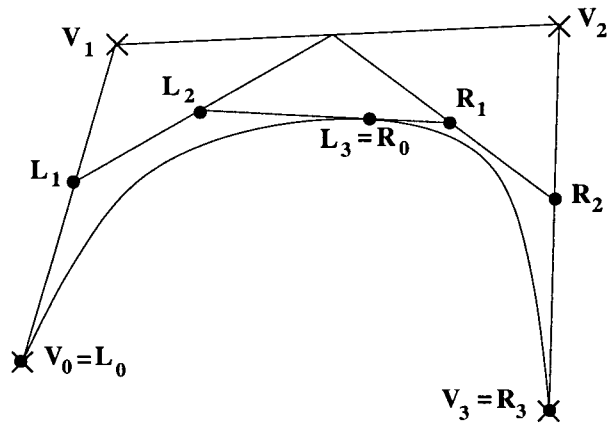


Figure 3.4: A single Bézier curve segment is subdivided into two segments. [Source [15], p.508]

that hierarchical splines retain a uniform knot difference by creating a new level of the hierarchy when they are subdivided.

3.3 B-Spline Surfaces

The representation of a parametric surface is similar to a curve except the elements of the geometry vector are not constants but instead are themselves parametric cubic curves. Two parameters, u and v, represent a point on the surface. If one of the parameters is fixed then varying the other over the interval $[0, 1]$ maps out a parametric cubic curve (Figure 3.6). In matrix form a parametric bicubic surface is represented as,

$$\mathbf{Q}(\mathbf{u}, \mathbf{v}) = U \cdot M \cdot \begin{bmatrix} \mathbf{G_1}(\mathbf{v}) \\ \mathbf{G_2}(\mathbf{v}) \\ \mathbf{G_3}(\mathbf{v}) \\ \mathbf{G_4}(\mathbf{v}) \end{bmatrix} \tag{3.8}$$

where $\mathbf{G_i}(\mathbf{v})$ is a parametric cubic curve given by

$$\begin{aligned} \mathbf{G_i}(\mathbf{v}) &= V \cdot M \cdot [\mathbf{V_{i1}}\ \mathbf{V_{i2}}\ \mathbf{V_{i3}}\ \mathbf{V_{i4}}]^T \\ &= [\mathbf{V_{i1}}\ \mathbf{V_{i2}}\ \mathbf{V_{i3}}\ \mathbf{V_{i4}}] \cdot M^T \cdot V^T. \end{aligned} \tag{3.9}$$

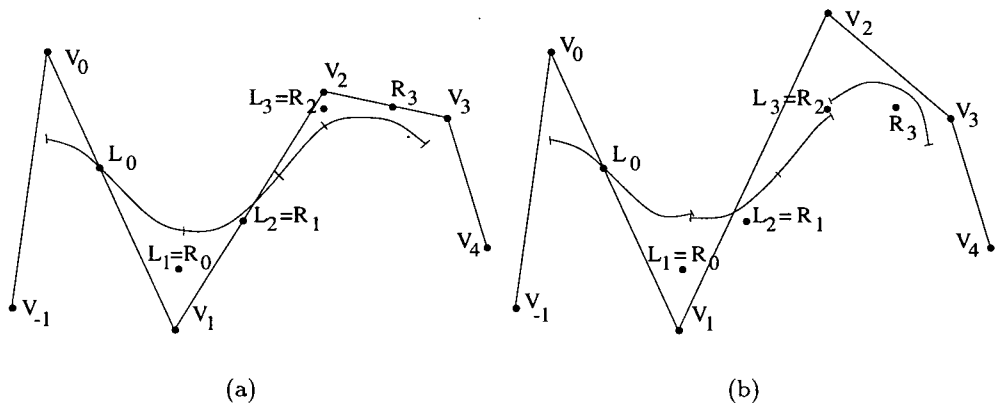


Figure 3.5: (a) Three B-spline curve segments with the middle segment subdivided into two segments. The middle segment is defined by five new control vertices ($L_0, L_1 = R_0, L_2 = R_1, L_3 = R_2, R_3$). (b) Control vertex V_2 from the original curve is moved and cracks form.

A variable $\mathbf{V_{i,j}}$, $0 \leq i, j \leq 3$, is a control vertex and V denotes the 1×4 vector $[v^3\ v^2\ v\ 1]$. The surface representation of Equation 3.8 is rewritten as,

$$\mathbf{Q(u,v)} = U \cdot M \cdot G \cdot M^T \cdot V^T, \quad 0 \leq u, v \leq 1 \tag{3.10}$$

$$\mathbf{Q(u,v)} = U \cdot M \cdot \begin{bmatrix} \mathbf{V_{11}} & \mathbf{V_{12}} & \mathbf{V_{13}} & \mathbf{V_{14}} \\ \mathbf{V_{21}} & \mathbf{V_{22}} & \mathbf{V_{23}} & \mathbf{V_{24}} \\ \mathbf{V_{31}} & \mathbf{V_{32}} & \mathbf{V_{33}} & \mathbf{V_{34}} \\ \mathbf{V_{41}} & \mathbf{V_{42}} & \mathbf{V_{43}} & \mathbf{V_{44}} \end{bmatrix} \cdot M^T \cdot V^T \tag{3.11}$$

For a B-spline patch the surface equation is given in Equation 3.12 where the 4×4 geometry matrix G_{B_S} is the 16 control vertices of the patch.

$$\begin{aligned} \mathbf{Q(u,v)} &= U \cdot M_{B_S} \cdot G_{B_S} \cdot M_{B_S}^T \cdot V^T, \quad 0 \leq u, v \leq 1 \qquad (3.12) \\ &= B_{B_S}(u) \cdot G_{B_S} \cdot B_{B_S}(v)^T \\ &= \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{V_{i,j}} B_i(u) B_j(v) \end{aligned}$$

To maintain C^0 continuity at an edge between two patches the edge curve of each patch must be identical, which means the control vertices must be identical.

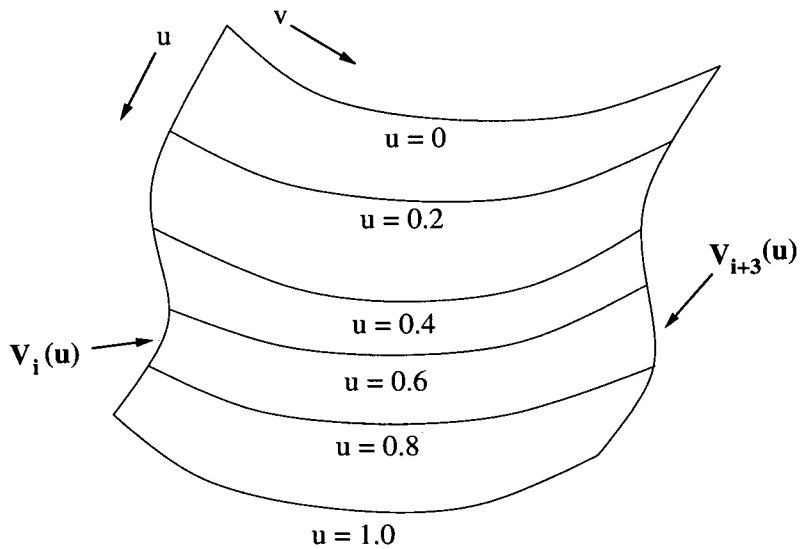


Figure 3.6: A bicubic parametric surface patch showing some parametric curves with constant u.

Just as with curves, C^1 continuity across two patches requires C^0 continuity and the edge tangents have equal direction and magnitude. Higher orders of continuity are found by enforcing the equality of higher order partial differential equations with respect to the non-constant parameter. A B-spline surface automatically maintains C^0, C^1 and C^2 continuity across patches because adjacent patches share control points and basis functions.

One final important note before turning attention to hierarchical spline surfaces involves the B-spline blending functions. In the same manner that the B-spline blending function indicates a control vertex of a cubic B-spline curve influences four segments, a control vertex of a bicubic B-spline surface influences 16 patches.

3.4 Hierarchical B-Spline Surfaces

The Oslo algorithm [7] mentioned in Section 3.2 describes a method for refining a bicubic B-spline surface by inserting additional knots into the knot sequence and replacing the original control vertices by a new set of vertices. When applied to a

bicubic B-spline the entire row and column that the patch is a member of has its control vertices replaced. Both the editing complexity and the memory requirements increase with so many additional, potentially unneeded, control vertices. Forsey [17] [16] introduces hierarchical B-spline surfaces that have the advantageous property of *local refinement* by adding only those control vertices needed for local editing of the patch. As the name suggests, hierarchical B-splines are a hierarchy of B-spline surfaces where a finer level k in the hierarchy contains the refined control vertices of a coarser level $k-1$. If a level k of the surface $\mathbf{Q}^{[\mathbf{k}]}(\mathbf{u},\mathbf{v})$ with an $a \times b$ array of control vertices $\mathbf{V}^{[\mathbf{k}]}_{\mathbf{i,j}}$ and basis functions $B^{[k]}_i(u)$ and $B^{[k]}_j(v)$ is given by,

$$\mathbf{Q}^{[\mathbf{k}]}(\mathbf{u},\mathbf{v}) = \sum_{i=0}^{a}\sum_{j=0}^{b} \mathbf{V}^{[\mathbf{k}]}_{\mathbf{i,j}} B^{[k]}_i(u) B^{[k]}_j(v) \tag{3.13}$$

then the surface at level $k+1$ is written as,

$$\mathbf{Q}^{[\mathbf{k+1}]}(\mathbf{u},\mathbf{v}) = \sum_{i=0}^{c}\sum_{j=0}^{d} \mathbf{V}^{[\mathbf{k+1}]}_{\mathbf{i,j}} B^{[k+1]}_i(u) B^{[k+1]}_j(v). \tag{3.14}$$

The basis functions on level $k+1$ are the refined basis functions from level k with some additional knots (δ_u and δ_v) in the u and v direction.

$$\begin{aligned} B^{[k+1]}_i(u) &= \sum_{r=0}^{a+\delta_u} \alpha_i(r) B^{[k]}_r(u) \\ B^{[k+1]}_j(v) &= \sum_{s=0}^{b+\delta_v} \alpha_j(s) B^{[k]}_s(v) \end{aligned} \tag{3.15}$$

The α coefficients give a relationship between the control vertices on level k and level $k+1$.

$$\mathbf{V}^{[\mathbf{k+1}]}_{\mathbf{r,s}} = \sum_{i=0}^{c}\sum_{j=0}^{d} \alpha_i(r)\alpha_j(s) \mathbf{V}^{[\mathbf{k}]}_{\mathbf{i,j}} \tag{3.16}$$

A set of four B-spline patches and the local refinement process is shown in Figure 3.7. Hollow circles denote original control vertices and filled circles show new control vertices on the finer level.

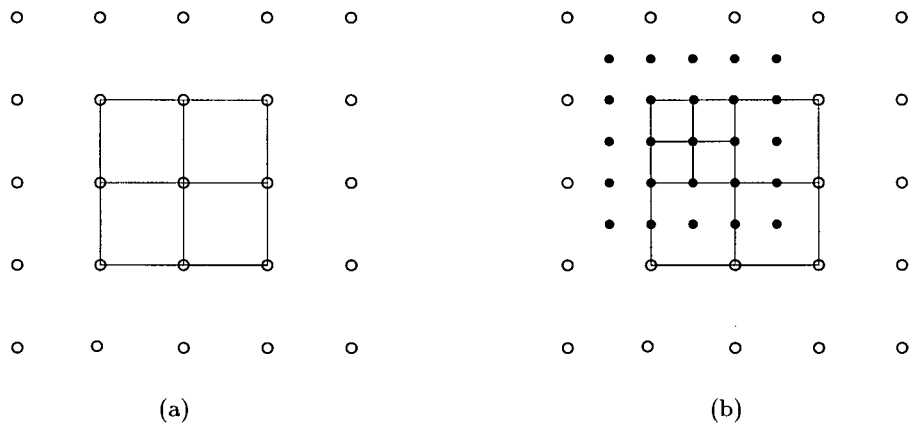


Figure 3.7: (a) Four B-spline patches and their control vertices. (b) Local refinement of the top left patch. The four corner vertices of the original patch still exist in the data structure but have been removed from the diagram for clarity.

To maintain C^2 continuity only those control vertices whose basis functions are zero at the boundary to the coarser level are allowed to move. For example, in Figure 3.8 the refinement of several patches and the additional control vertices at each step that are free to move are shown. In Chapter 4 an algorithm is described to interpolate a patch through a point in space. This algorithm calculates the displacement of each patch control vertex such that the patch interpolates the point. To execute the interpolation algorithm ACRA must first ensure all the control vertices of the patch are moveable which may mean patches neighboring the interpolating patch need refinement.

The vector indicating the position of a level $k+1$ control vertex before it has been displaced from its position after the refinement is called the *derived* vector. When a level $k+1$ control vertex is displaced, an *offset* vector, relative to the local frame of reference on level k, is stored. The vector addition of the derived vector and the offset vector[1] gives the position of the control vertex as shown in Figure 3.9. This technique is more flexible than *displacement mapping* since displacement mapping

[1]The offset vector must be transformed into the derived vector's space before the vector addition is performed.

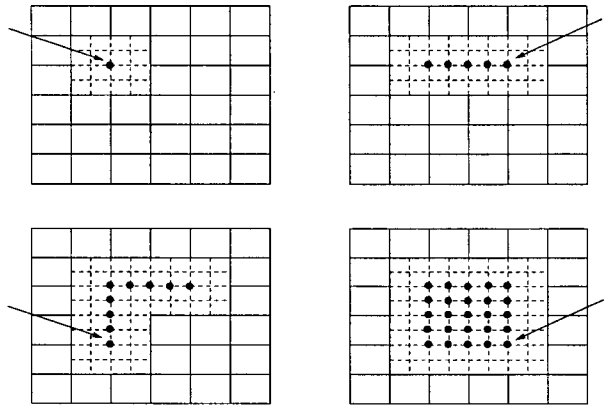
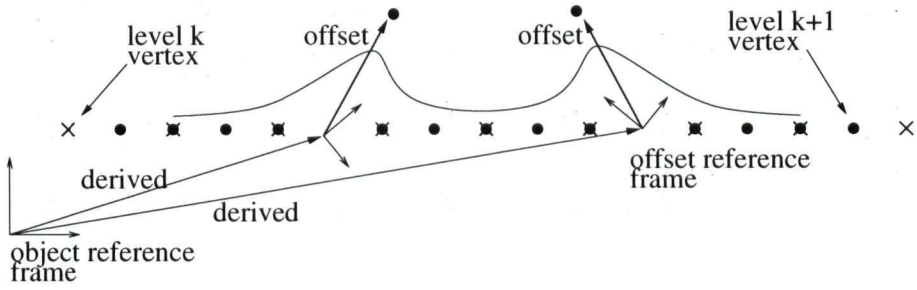


Figure 3.8: Local refinement of adjacent patches. The common corner control vertex of the four adjacent patches that are being refined is shown with the arrow. The black dots indicate which control vertices on the finer level are free to move without breaking C^2 continuity at the boundary to the coarser level. [Source [16], p.67]

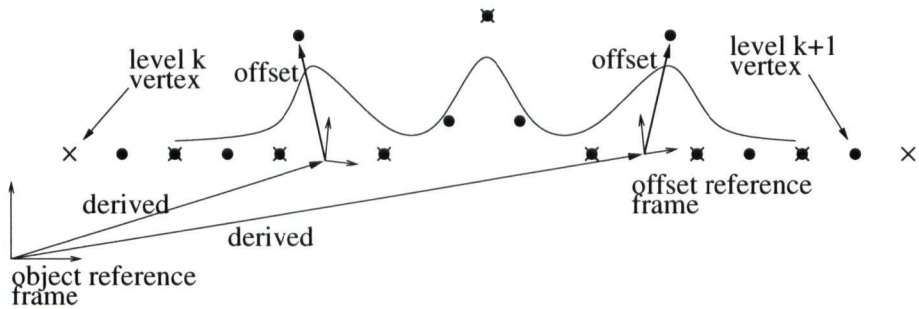
only displaces a surface point along the surface normal as defined by a pattern or map.

An important effect of storing a derived and an offset vector is the automatic finer level displacements when a coarser level control vertex moves (Figure 3.10). Typically control vertices at finer levels have a more localized control over the surface shape whereas coarser level vertices have influence over a larger portion of the surface[2]. The next chapter describes how ACRA makes use of this behavior to provide a spectrum of local to global collision responses.

[2]It is possible to model an object where a finer level control vertex can influence a larger portion of the object than a coarser level control vertex. In this case the animator must take care when specifying the amount of collision response each level of the hierarchy is responsible for.

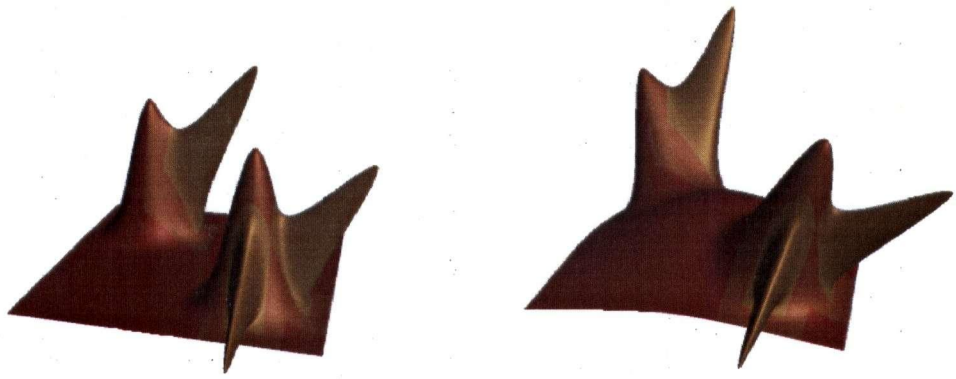


(a)



(b)

Figure 3.9: (a) A 2D representation of a derived and offset vector for a pair of vertices on level $k+1$. (b) Moving a vertex on level k causes the shape on level $k+1$ to follow along.



(a) (b)

Figure 3.10: A 3D example of the displacement of a coarser level vertex.

Chapter 4

Collision Response Algorithm

This chapter details the collision response algorithm and the mechanisms for controlling the response. The evolution of an interpolation algorithm, from a basic curve interpolation to a hierarchical surface interpolation, is described. Hierarchical surface interpolation is used to change an object's shape in response to a collision. A description of the controls for specifying the locality of the collision response and two additional collision response mechanisms, namely a translational and a rotational response finish the chapter.

4.1 Multi-Resolution Interpolation

The interpolation method proposed by Bartels and Beatty [6] serves as a basis for the extension of the method to hierarchical surfaces. Interpolation of multiple data points by a multilevel B-spline is presented in [23]. Independently, Archer [2] used a multipoint surface interpolation algorithm for the purposes of craniofacial reconstruction. This chapter will show surface interpolation is also applicable for object deformation.

4.1.1 Curve Interpolation

Bartels and Beatty's algorithm [6] solves the problem of interpolating any point on a parametric curve through an arbitrary data point. Control vertices of the curve segment containing the point are displaced such that the curve changes shape and interpolates the data point. A benefit of the technique is the ability to control the locality of the curve deformation simply by controlling the number of control vertices that are displaced.

As seen in Chapter 3, the equation for any point on a cubic B-spline segment is given by

$$\mathbf{Q}(\mathbf{u}) = B_0(u)\mathbf{V_0} + B_1(u)\mathbf{V_1} + B_2(u)\mathbf{V_2} + B_3(u)\mathbf{V_3} \tag{4.1}$$

where u, the parametric value for the curve, is defined on the interval $[0, 1]$. The points $\mathbf{V_0}$, $\mathbf{V_1}$, $\mathbf{V_2}$, and $\mathbf{V_3}$ are the control vertices for the segment and the functions $B_0(u)$, $B_1(u)$, $B_2(u)$, $B_3(u)$ are the B-spline basis functions. If the point $\mathbf{Q}(\mathbf{u})$ is to interpolate a new point $\mathbf{Q}'(\mathbf{u})$ while maintaining a constant parametric value then the equation for $\mathbf{Q}'(\mathbf{u})$ is written as

$$\mathbf{Q}'(\mathbf{u}) = B_0(u)\mathbf{V'_0} + B_1(u)\mathbf{V'_1} + B_2(u)\mathbf{V'_2} + B_3(u)\mathbf{V'_3} \tag{4.2}$$

The displacement, $\Delta\mathbf{Q}(\mathbf{u})$, between $\mathbf{Q}(\mathbf{u})$ and $\mathbf{Q}'(\mathbf{u})$ is found by subtracting Equation 4.1 from Equation 4.2.

$$\Delta\mathbf{Q}(\mathbf{u}) = B_0(u)\Delta\mathbf{V_0} + B_1(u)\Delta\mathbf{V_1} + B_2(u)\Delta\mathbf{V_2} + B_3(u)\Delta\mathbf{V_3} \tag{4.3}$$

Each of the $\Delta\mathbf{V_i}$'s represents the displacement of the control vertex necessary for $\mathbf{Q}(\mathbf{u})$ to interpolate $\mathbf{Q}'(\mathbf{u})$. The displacement $\Delta\mathbf{Q}(\mathbf{u})$ can be thought of as a vector describing the direction and magnitude of the interpolation. Setting each of the $\Delta\mathbf{V_i}$'s to $\Delta\mathbf{Q}(\mathbf{u})$ would satisfy Equation 4.3 because the B-spline basis functions sum to one. The curve segment would be translated and the shape would remain

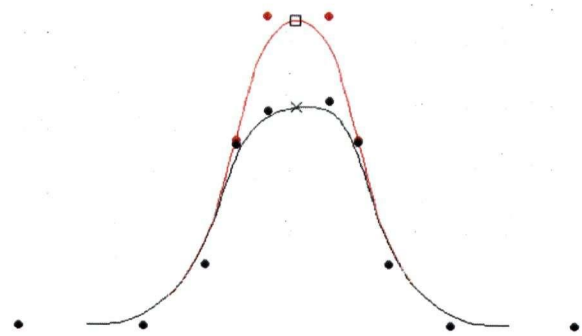


Figure 4.1: Interpolation of a single data point. The cross is the data point that $\mathbf{Q}(\mathbf{u})$ (the box) should interpolate. Notice only the four control vertices defining $\mathbf{Q}(\mathbf{u})$'s segment are displaced from the original position shown in red.

constant. To affect a minimum distance shape change[1] Bartels and Beatty [6] propose that the displacement be a weighted displacement of $\Delta\mathbf{Q}$[2]. The weights are chosen such that Equation 4.3 is satisfied.

$$\Delta\mathbf{V_i}(\mathbf{u}) = \Delta\mathbf{Q}(\mathbf{u}) \cdot w_i(u) = \Delta\mathbf{Q}(\mathbf{u})\frac{B_i(u)}{B_0^2(u) + B_1^2(u) + B_2^2(u) + B_3^2(u)} \tag{4.4}$$

Equation 4.4 suggests the amount each control vertex is displaced is proportional to its proximity to the moving point. Figure 4.1 shows the displacement of each of the control vertices as the spline interpolates a data point.

Modifying the interpolation algorithm to allow the interpolation of multiple data points allows for a greater range of resultant shapes. If different segments of the spline are involved in the interpolation then more of the spline will be deformed. To interpolate the $k = 0, \ldots, n$ data points, $p = 0, \ldots, m$ control vertices need to be moved. The displacement of each data point is written as

$$\Delta\mathbf{Q}(\mathbf{u_k}) = \Delta\mathbf{V_0}B_0(u_k) + \Delta\mathbf{V_1}B_1(u_k) + \Delta\mathbf{V_2}B_2(u_k) + \Delta\mathbf{V_3}B_3(u_k). \tag{4.5}$$

A control vertex, $\mathbf{V_p}$, will have multiple non-zero displacements if it influences segments from two of more data points. Equation 4.4 is modified to calculate

[1]The minimum distance shape change is the change in curve shape such that the curve point $\mathbf{Q}(\mathbf{u})$ interpolates $\mathbf{Q}'(\mathbf{u})$ but the control vertices undergo a minimal displacement.

[2]For a theoretical justification of the weights in Equation 4.4 the interested reader is referred to [6].

the displacement of each control vertex for each data point.

$$\begin{aligned}\Delta\mathbf{V_p}(\mathbf{u_k}) &= \Delta\mathbf{Q}(\mathbf{u_k}) \cdot w_i(u_k) \\ &= \Delta\mathbf{Q}(\mathbf{u_k}) \frac{B_i(u_k)}{B_0^2(u_k) + B_1^2(u_k) + B_2^2(u_k) + B_3^2(u_k)} \end{aligned} \tag{4.6}$$

The index i, $0 \leq i \leq 3$, is the index of $\mathbf{V_p}$ in the segment containing $\mathbf{Q}(\mathbf{u_k})$. The overall displacement for $\mathbf{V_p}$ is a weighted sum of all the $\Delta\mathbf{V_p}(\mathbf{u_k})$.

$$\Delta\mathbf{V_p} = \sum_{k=0}^{n} W_p(u_k)\Delta\mathbf{V_p}(\mathbf{u_k}) \tag{4.7}$$

Using a weighted displacement suggests the curve will not interpolate the multiple data points perfectly. The actual displacement of each curve point is

$$\Delta\widetilde{\mathbf{Q}}(\mathbf{u_k}) = \Delta\mathbf{V_0}B_0(u_k) + \Delta\mathbf{V_1}B_1(u_k) + \Delta\mathbf{V_2}B_2(u_k) + \Delta\mathbf{V_3}B_3(u_k) \tag{4.8}$$

The weights $W_p(u_k)$ should be chosen such that the difference between the desired displacement $\Delta\mathbf{Q}(\mathbf{u_k})$ and the actual displacement $\Delta\widetilde{\mathbf{Q}}(\mathbf{u_k})$ is a minimum. The method of least squares [25] is used to find appropriate weights.

$$\begin{aligned} s &= \sum_{k=0}^{n} (\Delta\mathbf{Q}(\mathbf{u_k}) - \Delta\widetilde{\mathbf{Q}}(\mathbf{u_k}))^2 \\ \frac{\partial s}{\partial W_p(u_k)} &= 0 \\ W_p(u_k) &= \frac{B_i^2(u_k)}{\sum_{k=0}^{n} B_i^2(u_k)} \end{aligned} \tag{4.9}$$

To reduce the errors between the desired and the actual displacements the curve interpolation can be iterated. Figure 4.2 shows a curve with 8 control vertices interpolating 4 data points. In Figure 4.2(b) the curve has been re-interpolated 5 times to achieve an excellent fit of the data points.

4.1.2 Surface Interpolation

Thus far the presented method has applied to B-spline curves. Objects in animation are usually represented as surfaces so the method must be adapted for a surface interpolating a set of data points.

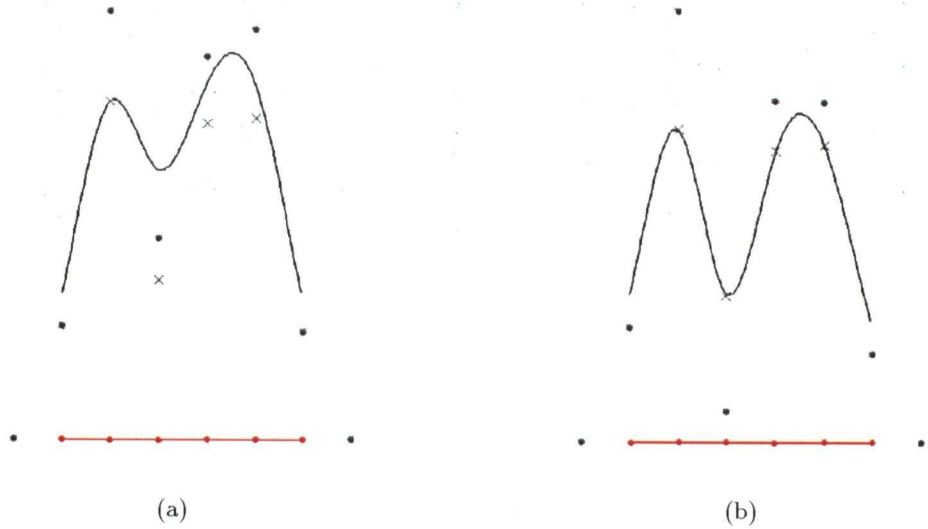


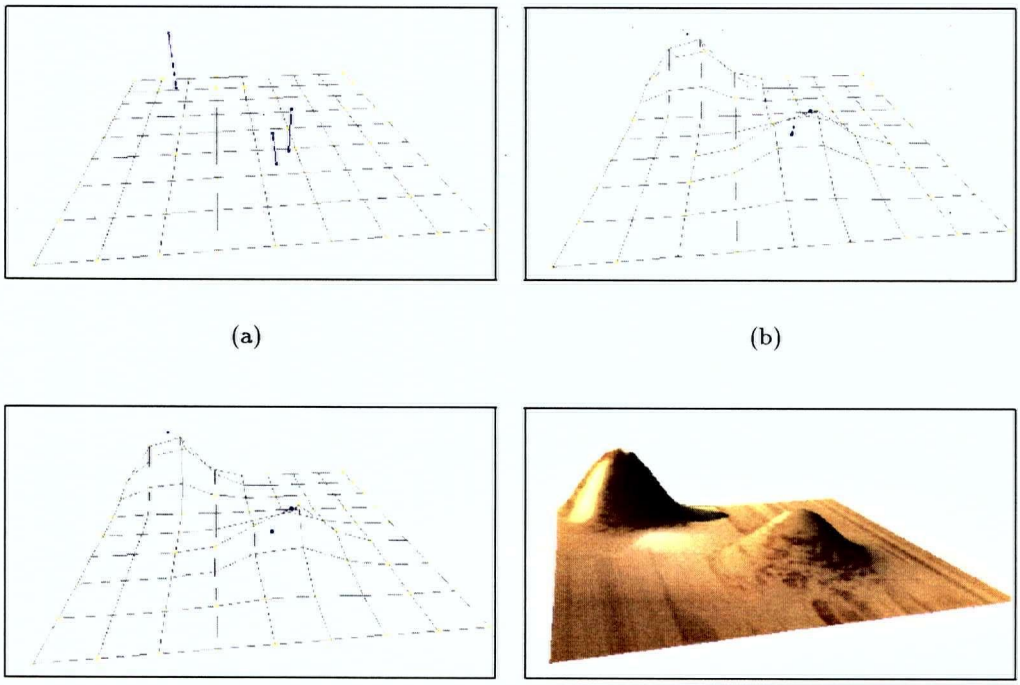
Figure 4.2: Interpolation of multiple data points. The original curve is shown in red, the data points are crosses and the control vertices are dots. (a) After one interpolation. (b) After five interpolations.

A set of control vertices $\mathbf{V_p}$ where $p = 0, \cdots, m$ will be displaced for the surface to interpolate the $k = 0, \cdots, n$ data points. The displacement of a control vertex due to data point k is given by

$$\begin{aligned} \Delta \mathbf{V_p}(\mathbf{u_k}, \mathbf{v_k}) &= \Delta \mathbf{Q}(\mathbf{u_k}, \mathbf{v_k}) \cdot w_{i,j}(u_k, v_k) \\ &= \Delta \mathbf{Q}(\mathbf{u_k}, \mathbf{v_k}) \frac{B_i(u_k) B_j(v_k)}{\sum_{l=0}^{3} \sum_{m=0}^{3} (B_l(u_k) B_m(v_k))^2}. \end{aligned} \tag{4.10}$$

The overall displacement of each control vertex is a weighted sum of all the individual displacements due to the k data points. As in section 4.1.1 the weights are found with the method of least squares.

$$\begin{aligned} \Delta \mathbf{V_p} &= \sum_{k=0}^{n} W_p(u_k, v_k) \Delta \mathbf{V_p}(\mathbf{u_k}, \mathbf{v_k}) \\ W_p(u_k, v_k) &= \frac{(B_i(u_k) B_j(v_k))^2}{\sum_{k=0}^{n} (B_i(u_k) B_j(v_k))^2} \end{aligned} \tag{4.11}$$



(a) (b)

(c) (d)

Figure 4.3: Interpolation of multiple data points. (a) Initial configuration. (b) After one iteration. (c) After 5 iterations. (d) Shaded surface.

Again, due to the weighted displacement of the control vertices the interpolation of a surface will not be perfect and the method can be iterated. Figure 4.3 shows a surface with 11×11 control vertices and 3 data points. Notice in Figures 4.3(b) and (c) how the patch with one surface point converges to the data point with fewer iterations than the patch with multiple surface points.

4.1.3 Hierarchical Surface Interpolation

As described in Chapter 3, a hierarchical surface has a hierarchy of levels where each level is a subdivision of the previous level. The idea that a broader spectrum of surface shapes is available if the interpolation algorithm is applied to various levels in the hierarchy is adopted from Palazzi [29] and Harrison [20]. If a coarse

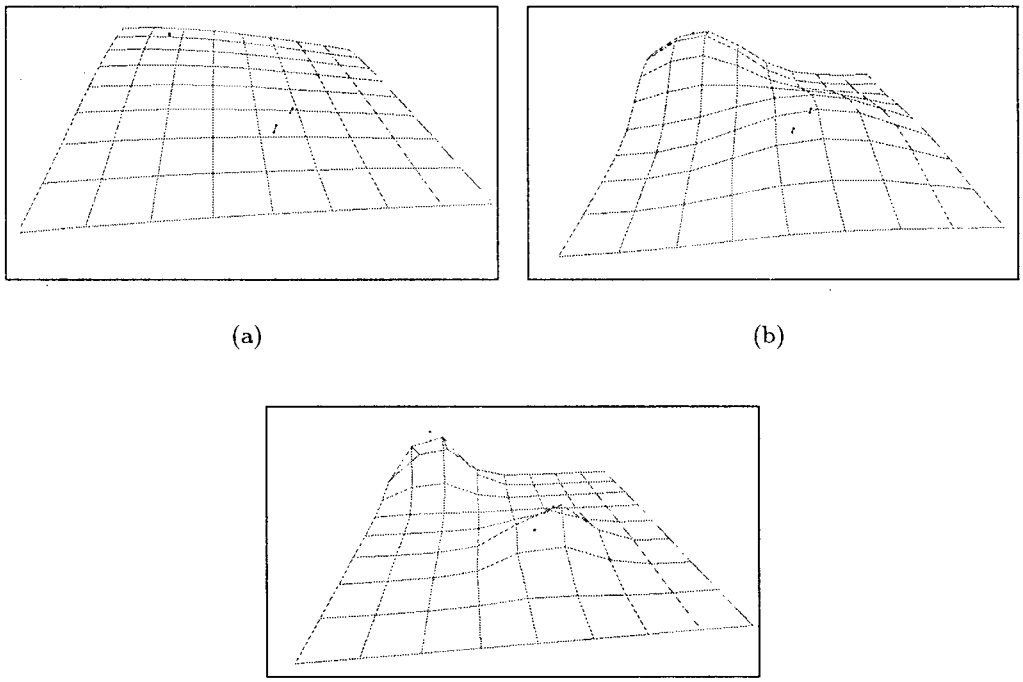


Figure 4.4: Hierarchical interpolation of multiple data points. (a) Coarsest level absorbs all the interpolation (5 iterations). (b) In between level absorbs all the interpolation (5 iterations). (c) Finest level absorbs all the interpolation (5 iterations).

level of the hierarchy interpolates the data points then a larger region of the surface will change shape. Conversely, a localized shaped change occurs if the finest level interpolates the data points. This is the core mechanism used by the collision response algorithm to provide control over the locality of the object deformations. Figure 4.4 shows the same surface as Figure 4.3 but the interpolation is done by the coarsest level (Figure 4.4(a)), an in between level (Figure 4.4(b)), and the finest level (Figure 4.4(c)).

Blending the interpolation from various levels is known as *multi-resolution* interpolation. If some of the interpolation is achieved or 'absorbed' by a coarse

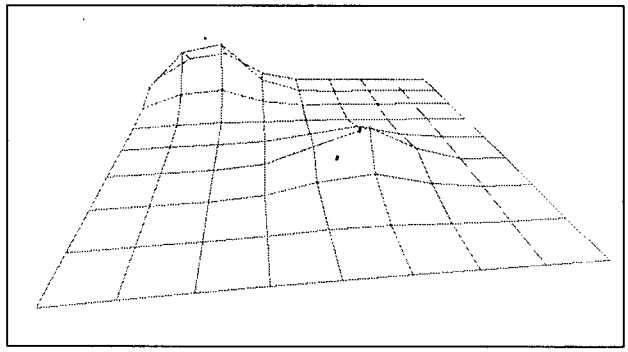


Figure 4.5: Multi-resolution interpolation of multiple data points. The coarsest level absorbs 30% of the interpolation and the finest level absorbs 70%.

level and another portion absorbed by a fine level then the overall shape would be gradual. Each level, l, where $l = 0$ is the coarsest and $l = m$ is the finest, can absorb a percentage of the interpolation $a^{[l]}$. The displacement of a data point k on l will be

$$\Delta\mathbf{Q}^{[l]}(\mathbf{u_k}, \mathbf{v_k}) = a^{[l]} \cdot \Delta\mathbf{Q}(\mathbf{u_k}, \mathbf{v_k}) \tag{4.12}$$

where

$$\sum_{l=0}^{m} a^{[l]} = 1, \quad a^{[l]} \geq 0 \tag{4.13}$$

Before any interpolation is performed $\Delta\mathbf{Q}(\mathbf{u_k}, \mathbf{v_k})$ is calculated as the displacement between the kth data point and the surface point $\mathbf{Q}(\mathbf{u_k}, \mathbf{v_k})$. Each level, starting with $l = 0$, first determines the points $\mathbf{Q}^{[l]}(\mathbf{u_k}, \mathbf{v_k})$ and then interpolates the level through the data points,

$$\mathbf{Q}'^{[l]}(\mathbf{u_k}, \mathbf{v_k}) = \mathbf{Q}^{[l]}(\mathbf{u_k}, \mathbf{v_k}) + a^{[l]} \cdot \Delta\mathbf{Q}(\mathbf{u_k}, \mathbf{v_k}) \tag{4.14}$$

Figure 4.5 shows the surface from Figure 4.3 with 30% of the interpolation absorbed by level 0 and 70% absorbed by level m.

Hierarchical B-spline surfaces allow for local refinement, thus a surface point $\mathbf{Q}(\mathbf{u_k}, \mathbf{v_k})$ may be in a region where the subdivision level is less than the finest

level that absorbs some of the interpolation. Thus, interpolating a data point on level q, where q is not the finest level m, may not remove all the interpenetration if $\sum_{l=q+1}^{m} a^{[l]} > 0$. ACRA incorporates two solutions for handling such data points. The first alternative refines the patch containing the data point until that patch is at the finest level that absorbs any of the interpolation. A second alternative retains the refinement level of the surface and obeys the absorption amounts by only interpolating up to the level the surface point is on. In the first case the surface is refined and the response will be more local while the second case will still maintain a global response but some of the interpenetration may remain.

4.2 Using Multi-Resolution Interpolation for Collision Response

During a collision an interpenetration will occur between the two objects if no response is applied. Any non-rigid object involved in the collision will need to deform to resolve the interpenetration. The collision response algorithm finds appropriate surface points and data points for an object such that the object will be deformed and no longer interpenetrate the other object after the data points are interpolated. A method for finding appropriate surface points and data points is described in this section.

4.2.1 Rigidity

ACRA is only concerned with the response of an object after a collision is detected, another module needs to be used to detect the collisions. The collision response algorithm requires two pieces of information; the point on each surface where the two objects initially made contact and the boundary of the interpenetration region on each surface. If we were to rewind the collision to the instant when surface S_1 and surface S_2 touched, $\mathbf{P_1}$ and $\mathbf{P_2}$ are the contact points on each surface respectively.

On the line between $\mathbf{P_1}$ and $\mathbf{P_2}$ the point $\mathbf{P'}$ is defined as a data point that each surface will interpolate. The position of $\mathbf{P'}$ is determined by the relative rigidity of the two objects. As predicted by Equation 4.15, if S_1 is more rigid than S_2 then $\mathbf{P'}$ will be closer to $\mathbf{P_1}$ than $\mathbf{P_2}$.

$$\begin{aligned} \mathbf{P'} &= R\mathbf{P_1} + (1-R)\mathbf{P_2} \\ R &= \frac{R_1}{R_1 + R_2} \end{aligned} \tag{4.15}$$

The rigidity, R_1 and R_2, of S_1 and S_2 respectively is in the range $[1, 100]$.

Interpolating $\mathbf{P_1}$ through $\mathbf{P'}$, for example, will only move the patch containing $\mathbf{P_1}$, on a given level. The absorptions $a^{[l]}$ may be set such that some other parts of S_1 will still remain within the interpenetration region after the interpolation (see Figure 4.6). The solution is to move all of the patches that are completely or partially contained in the interpenetration region. To do so requires a surface point and a corresponding data point for each patch. Following the example shown in Figure 4.7, $k = 0, \ldots, n$ additional surface points are found at the parametric middle of the patch or the parametric middle of the portion of the patch inside the interpenetration region. By extending a vector from each $\mathbf{A_{1,k}}$ in the direction of $\overrightarrow{\mathbf{P_1P_2}}$ a corresponding surface point is found on S_2. The data points, $\mathbf{A'_{1,k}}$, shown in Figure 4.7(c) are found with Equation 4.16. Figure 4.7(d) and (e) shows a similar process for surface 2.

$$\begin{aligned} \mathbf{A'_{1,k}} &= R\mathbf{A_{1,k}} + (1-R)\mathbf{B_{1,k}} \\ R &= \frac{R_1}{R_1 + R_2} \end{aligned} \tag{4.16}$$

4.2.2 Deformation, Translation and Rotation

ACRA provides three types of collision response, deformational as described in section 4.1, translational and rotational as described in section 4.3. The user is able to

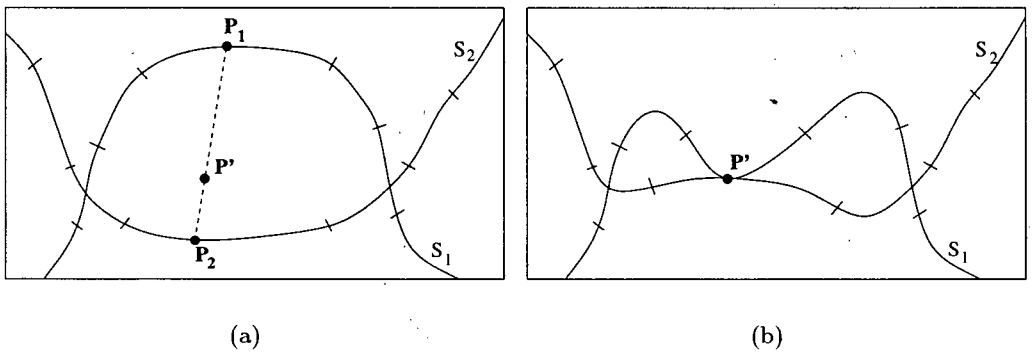


Figure 4.6: Movement of a single patch may not eliminate all the interpenetration. A simplified 2D version of the surfaces is shown to clarify the error.

control both the amount of collision response that is absorbed on each level $a^{[l]}$ and the combination of a deformational $d^{[l]}$, translational $t^{[l]}$ and rotational $r^{[l]}$ response on each level.

$$d^{[l]} + t^{[l]} + r^{[l]} = 1 \tag{4.17}$$

where

$$\begin{aligned} d^{[l]} &\geq 0 \\ t^{[l]} &\geq 0 \\ r^{[l]} &\geq 0 \end{aligned}$$

The user controls the amount of collision response absorbed on each level with percentage sliders as shown in Figure 4.8. The sum of all the active sliders is 100%. Slider controls for deformational, translational and rotational percentages per level are also available.

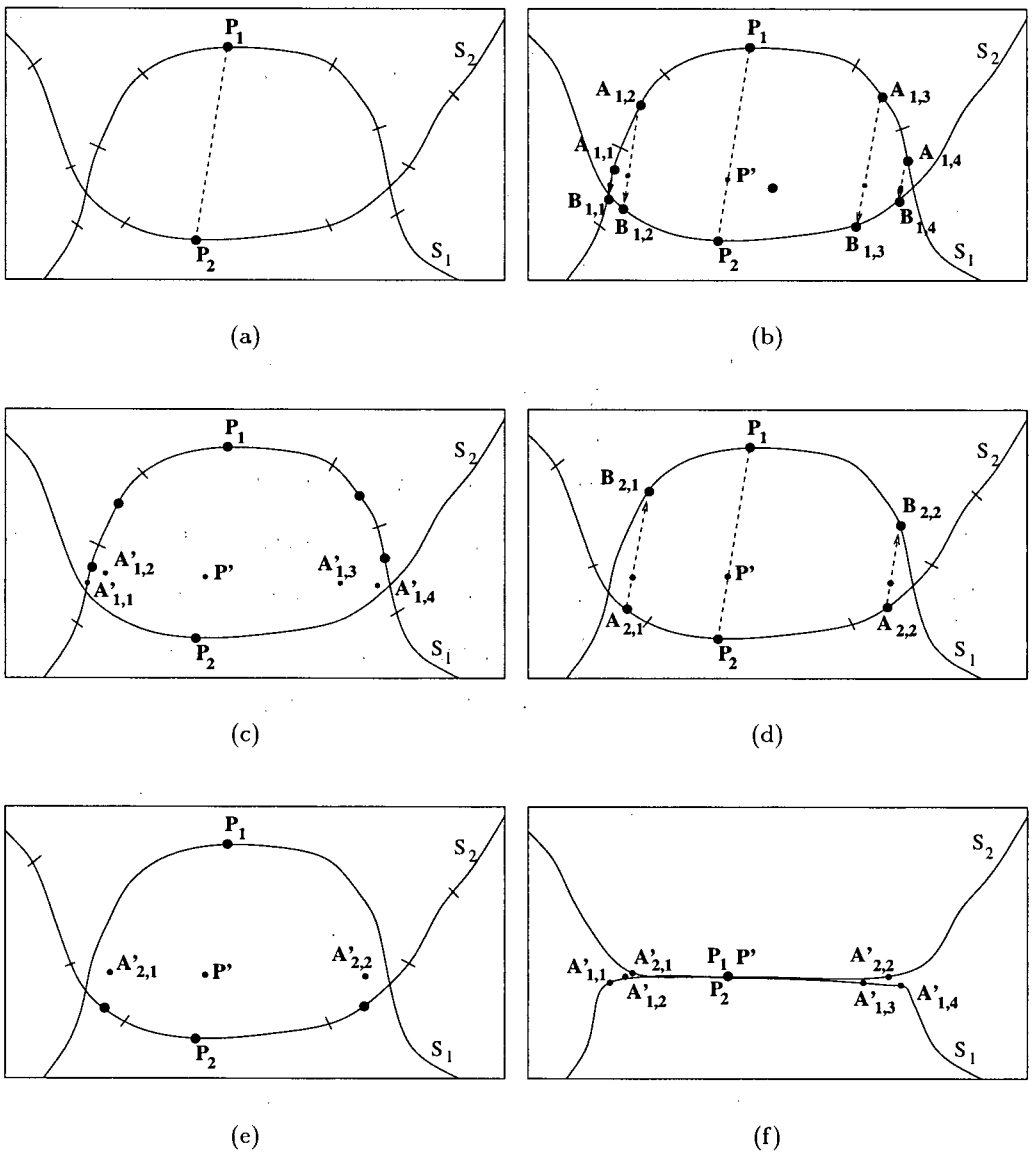


Figure 4.7: Steps for finding additional data points for all patches or portions of patches within the interpenetration region. (a) The ticks on the surfaces indicate patch boundaries. (b) A vector is extended from each additional surface point to the other surface. (c) Data points to be interpolated by surface 1 are indicated by $\mathbf{A}'_{1,j}$. (d) and (e) show the same process for surface 2. (f) After the interpolation.

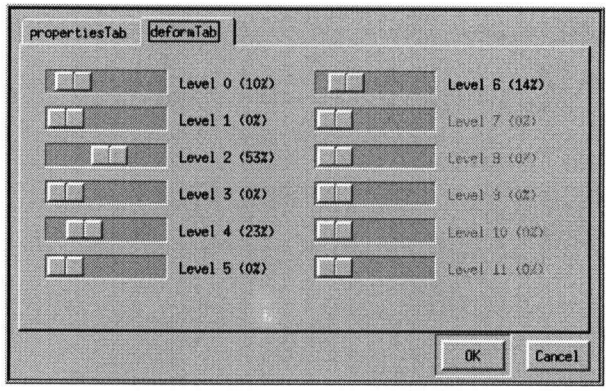


Figure 4.8: Controls for setting the absorption per level. Greyed out sliders indicate the surface has not been refined to that level.

4.2.3 Deformational Response

The points $\mathbf{P}_{\mathbf{i}}^{[l]}$ and $\mathbf{A}_{\mathbf{i,j}}^{[l]}$ on level l are the points that have the same (u, v) coordinates as $\mathbf{P_i}$ and $\mathbf{A_{i,j}}$. The displacement that the entire level should absorb is

$$\Delta\mathbf{P}_{\mathbf{i}}^{[l]} = a^{[l]} \cdot (\mathbf{P}' - \mathbf{P_i}) \tag{4.18}$$

$$\Delta\mathbf{A}_{\mathbf{i,j}}^{[l]} = a^{[l]} \cdot (\mathbf{A}'_{\mathbf{i,j}} - \mathbf{A_{i,j}}) \tag{4.19}$$

The portion of the displacement that the deformational response is responsible for is

$$\begin{aligned} \Delta\mathbf{P}_{\mathbf{i}_{\text{deform}}}^{[l]} &= d^{[l]} \cdot \Delta\mathbf{P}_{\mathbf{i}}^{[l]} \\ \Delta\mathbf{A}_{\mathbf{i,j}_{\text{deform}}}^{[l]} &= d^{[l]} \cdot \Delta\mathbf{A}_{\mathbf{i,j}}^{[l]}. \end{aligned}$$

The data points that the interpolation algorithm will interpolate are

$$\begin{aligned} \mathbf{P}'^{[l]}_{\text{deform}} &= \mathbf{P}_{\mathbf{i}}^{[l]} + \Delta\mathbf{P}_{\mathbf{i}_{\text{deform}}}^{[l]} \\ \mathbf{A}'^{[l]}_{\mathbf{i,j}_{\text{deform}}} &= \mathbf{A}_{\mathbf{i,j}}^{[l]} + \Delta\mathbf{A}_{\mathbf{i,j}_{\text{deform}}}^{[l]}. \end{aligned}$$

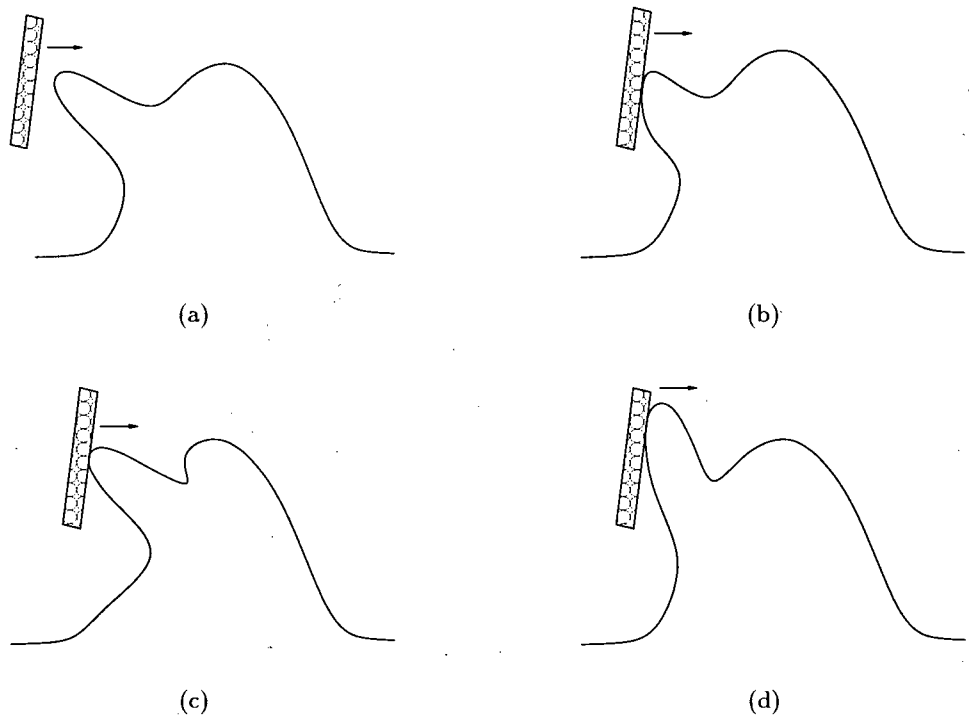


Figure 4.9: Collision response when a surface is specified to (b) deform, (c) translate or (d) rotate.

4.3 Translational and Rotational Response

Two additional methods for collision response are provided to model objects that do not undergo pure deformational collision response. The interpolation algorithm generates a *deformational response* in the sense that it tends to give a dented appearance to the object. Some objects may prefer to retain their shape and respond to the collision by translating to another position while others may translate on a finer level to retain surface details but dent inwards on a global scale. The option to allow a retention of details on a level during collision response is called *translational response*. Similarly, *rotational response* simulates an object being pushed or rolled during a collision. Figure 4.9 shows how an object may respond when the main response mechanism is a deformation, a translation and a rotation on a fine level.

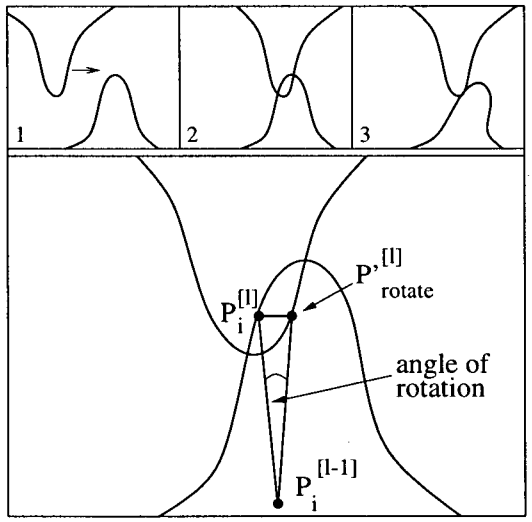


Figure 4.10: The three smaller figures show the system before the collision, during the collision and the resulting rotational response. The bottom object undergoes a rotational response where the axis of rotation is into the page and the angle of rotation is taken from the triangle $\mathbf{P}_{\mathbf{i}}^{[\mathbf{l}-\mathbf{1}]}, \mathbf{P}^{[\mathbf{l}]}, \mathbf{P}'^{[\mathbf{l}]}_{\mathbf{rotate}}$.

4.3.1 Translational Response

A translational response on level l simply translates all the control vertices of the patches containing $\mathbf{P}_{\mathbf{i}}^{[\mathbf{l}]}$ and $\mathbf{A}_{\mathbf{i,j}}^{[\mathbf{l}]}$ by $\Delta\mathbf{P}^{[\mathbf{l}]}_{\mathbf{i}_{\mathbf{translate}}}$ and $\Delta\mathbf{A}^{[\mathbf{l}]}_{\mathbf{i,j}_{\mathbf{translate}}}$ in the direction $\overrightarrow{\mathbf{P}'^{[\mathbf{l}]}\mathbf{P}_{\mathbf{i}}^{[\mathbf{l}]}}$ and $\overrightarrow{\mathbf{A}'^{[\mathbf{l}]}_{\mathbf{i,j}}\mathbf{A}^{[\mathbf{l}]}_{\mathbf{i,j}}}$. The equations for the displacements are,

$$\begin{aligned}
\Delta\mathbf{P}^{[\mathbf{l}]}_{\mathbf{i}_{\mathbf{translate}}} &= t^{[l]} \cdot \Delta\mathbf{P}_{\mathbf{i}}^{[\mathbf{l}]} \\
\Delta\mathbf{A}^{[\mathbf{l}]}_{\mathbf{i,j}_{\mathbf{translate}}} &= t^{[l]} \cdot \Delta\mathbf{A}^{[\mathbf{l}]}_{\mathbf{i,j}}.
\end{aligned}$$

4.3.2 Rotational Response

As shown in Figure 4.9(d) a rotational response moves the features on a level by what appears as a rotation. The initial contact point $\mathbf{P}_{\mathbf{i}}^{[\mathbf{l}]}$ is used to calculate the

interpolation point $\mathbf{P}'^{[l]}_{\text{rotate}}$ in the same manner as the deformational response case.

$$\begin{aligned}\Delta\mathbf{P}^{[l]}_{\mathbf{i}_{\text{rotate}}} &= r^{[l]} \cdot \Delta\mathbf{P}^{[l]}_{\mathbf{i}} \\ \Delta\mathbf{A}^{[l]}_{\mathbf{i,j}_{\text{rotate}}} &= r^{[l]} \cdot \Delta\mathbf{A}^{[l]}_{\mathbf{i,j}}\end{aligned}$$

$$\begin{aligned}\mathbf{P}'^{[l]}_{\text{rotate}} &= \mathbf{P}^{[l]}_{\mathbf{i}} + \Delta\mathbf{P}^{[l]}_{\mathbf{i}_{\text{rotate}}} \\ \mathbf{A}'^{[l]}_{\mathbf{i,j}_{\text{rotate}}} &= \mathbf{A}^{[l]}_{\mathbf{i,j}} + \Delta\mathbf{A}^{[l]}_{\mathbf{i,j}_{\text{rotate}}}.\end{aligned}$$

A plane is defined by the three points $\mathbf{P}^{[l-1]}_{\mathbf{i}}, \mathbf{P}^{[l]}_{\mathbf{i}}, \mathbf{P}'^{[l]}_{\text{rotate}}$ (see Figure 4.10). The normal of that plane will be the axis of rotation and the angle of rotation is the angle between $\overrightarrow{\mathbf{P}^{[l-1]}_{\mathbf{i}}\mathbf{P}^{[l]}_{\mathbf{i}}}$ and $\overrightarrow{\mathbf{P}^{[l-1]}_{\mathbf{i}}\mathbf{P}'^{[l]}_{\text{rotate}}}$. A quaternion, composed of the axis of rotation and the angle, is applied to the offset of each control vertex influencing the patch containing $\mathbf{P}^{[l]}_{\mathbf{i}}$. An identical process is used to determine the angle of rotation and axis of rotation for each $\mathbf{A}^{[l]}_{\mathbf{i,j}}, \mathbf{A}'^{[l]}_{\mathbf{i,j}_{\text{rotate}}}$ pair. If a control vertex is influenced by more than one quaternion then a spherical linear interpolation of the quaternions is performed.

4.3.3 Restriction

The numerical multi-grid method [9] [26] originally inspired the idea for using multiple levels in a surface to simulate local or global behavior. For interpolating a surface through a set of data points a coarse interpolation using the coarser levels of the hierarchy will give the system a better starting point for a finer level interpolation. Coordinating behavior within the hierarchical surface requires a restriction and prolongation operator. A *restriction* operator passes information from a finer level to a coarser level. After a collision is detected the initial contact point, the additional points and the data points are determined for the finest level. The displacements, $\Delta\mathbf{P}_{\mathbf{i}}$ and $\Delta\mathbf{A}_{\mathbf{i,j}}$, are calculated for object i. The restriction operator passes the displacements and the (u, v) coordinates of the initial point and the additional points to each coarser level.

In a future version of ACRA the restriction operator could propagate a negative displacement outside of the interpenetration region to the coarser levels. A negative displacement could simulate volume conservation through a bulging effect.

4.3.4 Prolongation

Starting at the coarsest level the algorithm performs the following steps where the prolongation operator is the last step to pass information from a coarser level to a finer level.

- Calculate the initial contact point and additional points on this level, l, from the (u, v) coordinates provided by the restriction operator.
- Using Equation 4.18 and Equation 4.19, find the amount of displacement the initial point and additional points on this level will absorb.
- Carry out the deformational response.
- Recalculate the initial contact point and the additional points on this level. This step is necessary because the previous step has altered the shape of the surface.
- Carry out the translational response.
- Recalculate the initial contact point and the additional points on this level.
- Carry out the rotational response.
- Prolongate the new surface configuration to level $l + 1$.

In future versions of ACRA, the prolongation operator would also be responsible for ensuring no additional collisions occur due to a negative displacement restricted to level l. Surface fairing [24] on a global or local scale could occur if the prolongation operator applied a fairness function.

Each of the three methods for collision response may introduce further collisions to the objects. An iterative loop of collision detection followed by collision response may be necessary to reduce the interpenetration to the point where post-processing 'clean-up' removes the remainder.

4.4 Summary

The interpolation algorithm described in this chapter describes a method for altering the shape of a surface. Locality or globality of the collision response is controlled by altering the amount of interpolation each level absorbs. Three types of collision response are available and the user can choose a combination of the types on each level. Iterative calculations at each frame or time step are not needed because control vertex displacements are only calculated when a collision is detected. Some results and the boundaries of the problem set where ACRA is applicable is the topic of the next chapter.

Chapter 5

Results and Future Work

This chapter will address the issue of ACRA's effectiveness in terms of real-time execution speed, range of responses and the scope of models ACRA is suitable for. First some empirical results from an animation of a large model colliding with a small model are presented. Frames from various animations with changes in relative rigidity, locality of collision response and different types of response are presented to show the effect of the response controls. ACRA's goals from Section 2.5 are briefly revisited and some some ideas for avenues of future work and extensions to this foundational algorithm are discussed.

5.1 Implementation and Empirical Results

To collect data about ACRA's execution speed and to compare execution speeds for models of varying size, an animation of a large model colliding with a smaller model was generated. The animation has a total of 30 frames (one of which is shown in Figure 5.1), and 4 frames required a collision response. Although this thesis only deals with the issue of collision response, collision detection is an important predecessor to the collision response process. A library called RAPID [19] [35] is used for collision detection but the output from RAPID and the inputs required for

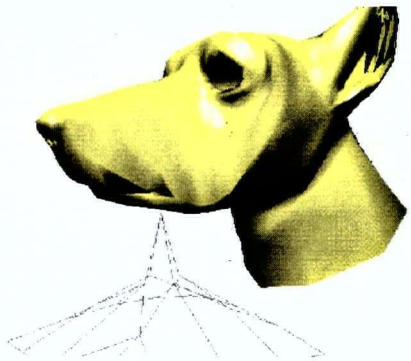


Figure 5.1: Animation frame. The large model of a dog head has 7 hierarchy levels and a total of 2085 control vertices while the small model of a spike has 3 levels and 90 control vertices.

ACRA were not entirely compatible. In fact a rather significant amount of code is needed to convert the outputs from RAPID to the inputs needed for ACRA. To give some idea of the relative importance of these three components, execution speeds were gathered for each component and tabulated in Figure 5.2.

RAPID, like many collision detection schemes, has two phases: an initial setup and the actual collision detection. It stores the geometry of the model in its local space and the translation and rotation in global space, thus if the object undergoes a translation or rotation only those matrices need updating. If local geometry changes, RAPID needs to execute its initial setup phase again. Each frame in the animation requires a query to RAPID's collision detection phase but only those frames that have a collision require the conversion of the collision detection outputs to ACRA's inputs, a collision response and a new initial setup for RAPID.

5.2 Goals Revisited

- **Unintrusive.** Key-framed motion of the object is obeyed by ACRA since the algorithm only changes the shape of an object, not the object's position.
- **Fast previews.** The performance results of Figure 5.2 indicate a collision

	Number of frames requiring process	Wall-clock time (ms)
RAPID (initial setup)	5	Large model - 3919 Small model - 12
RAPID (collision detection)	30	16
Conversion to collision response inputs	4	358
Collision response	4	Large model - 590 Small model - 14

Figure 5.2: Performance results for collision detection and response from a 30 frame animation rendered on a Pentium 150MHz with 64MB RAM. The numbers given are averages over the frames.

response requires less than a second of real time even for the large model, suggesting ACRA can provide fast preview for moderately complex geometry.

- **Broad spectrum of responses.** Using a multi-resolution surface such as a hierarchical B-spline surface allows a spectrum of local to global responses by adjusting the absorption amount on various levels of the hierarchy. Adjusting the three different response types on each level of the hierarchy gives the user the capability to generate denting, translating and rotating effects.

- **Animator Override.** Since the algorithm does not need to track the system's state, the animator is free to manipulate the surface by hand. If, in the future, the algorithm is extended to relax the surface back to a rest shape then care needs to be taken to specify how ACRA will behave when both the animator and ACRA want to adjust the position of the same control vertex.

5.3 Effectiveness of the Algorithm

Several frames from four animations of two key-framed objects colliding are shown in Figures 5.3-5.6. In the first animation the bottom object is much more rigid than

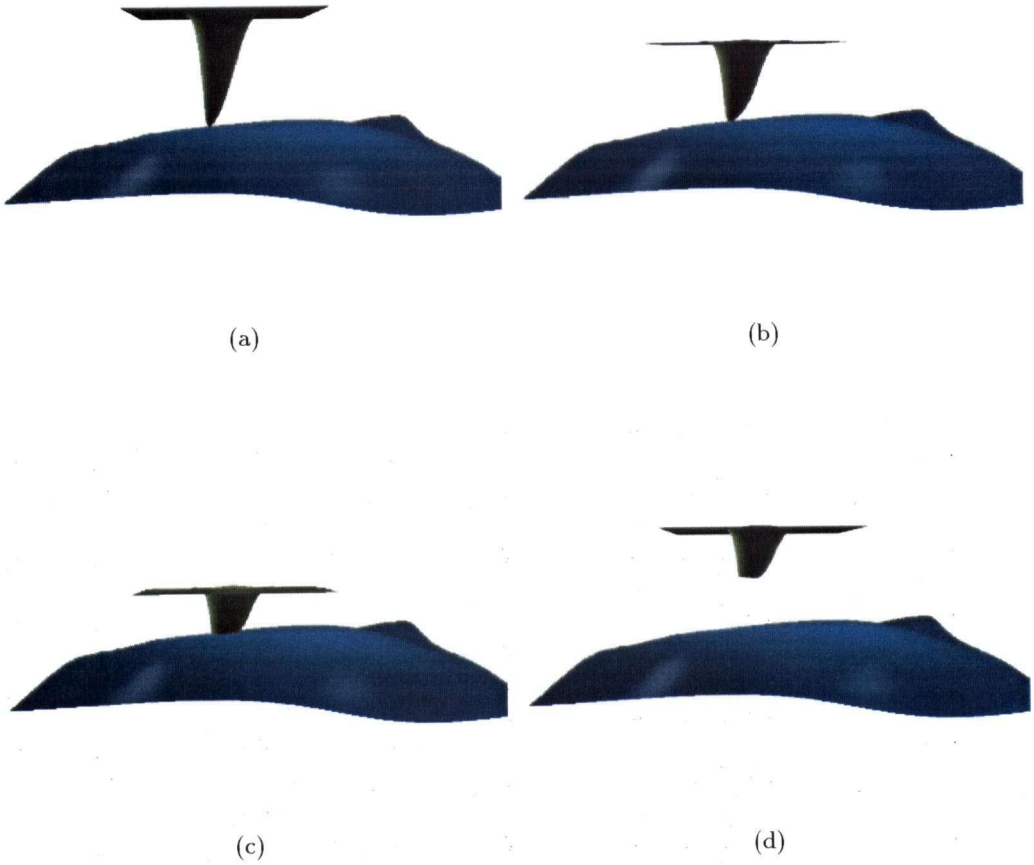


Figure 5.3: Frames 10, 15, 20 and 25 from an animation where the bottom object is highly rigid and the top object undergoes a deformational collision response at the finest two levels of its hierarchy. Both objects have 5 levels in their hierarchy.

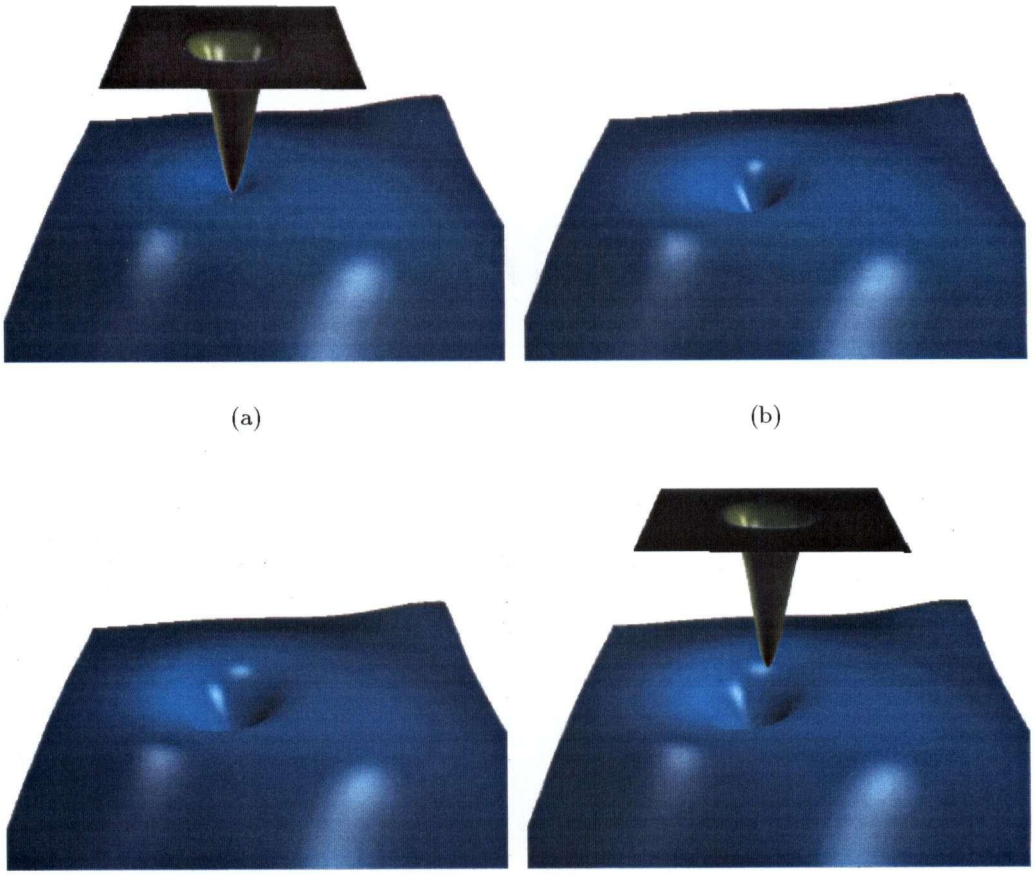


Figure 5.4: Similar to Figure 5.3 except the top object is more rigid than the bottom object. For clarity, the top object wasn't rendered in b and c.

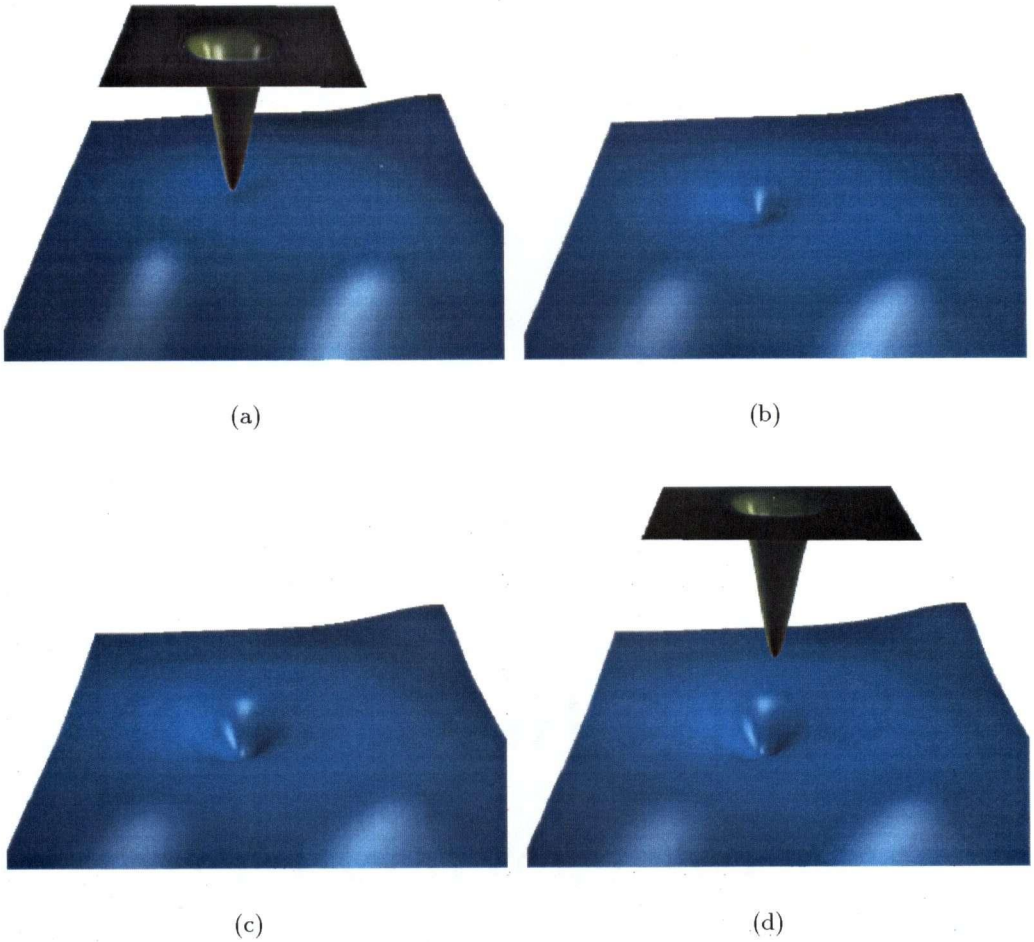


Figure 5.5: Similar to Figure 5.4 except the bottom object distributes the deformation absorption equally levels 1-3. For clarity, the top object wasn't rendered in b and c.

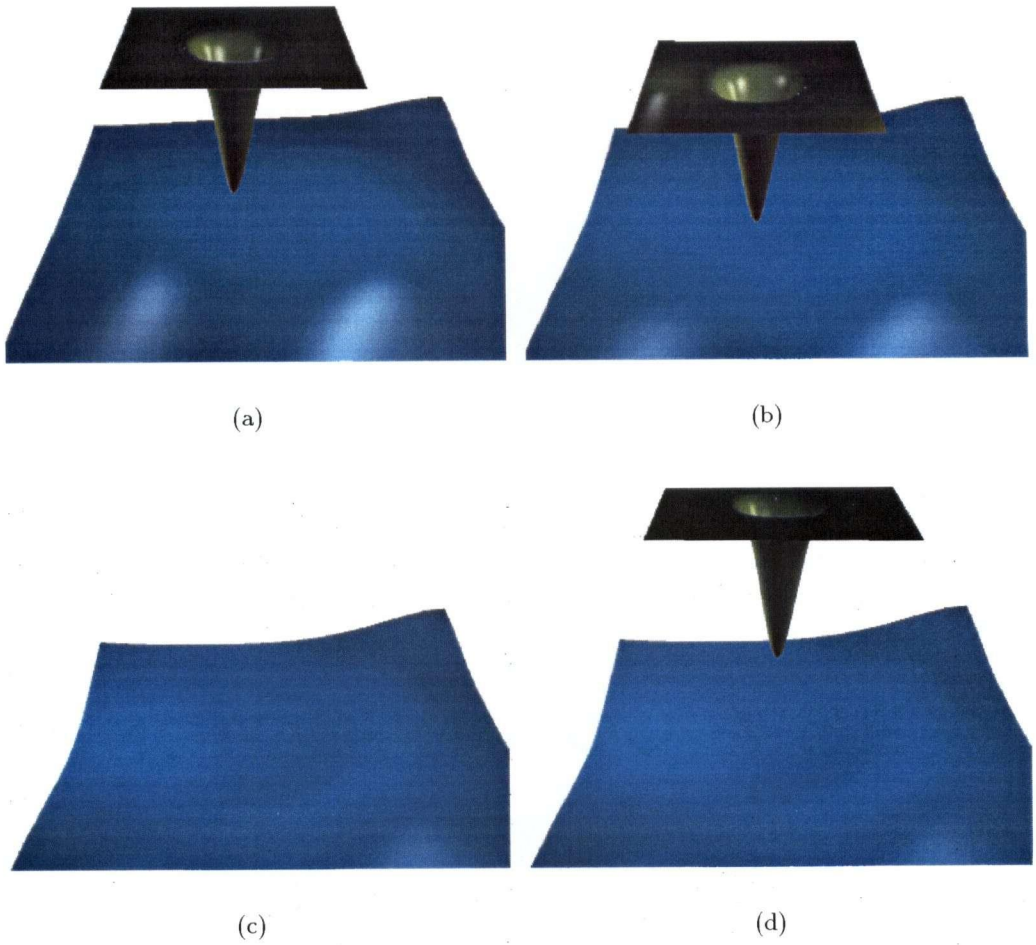


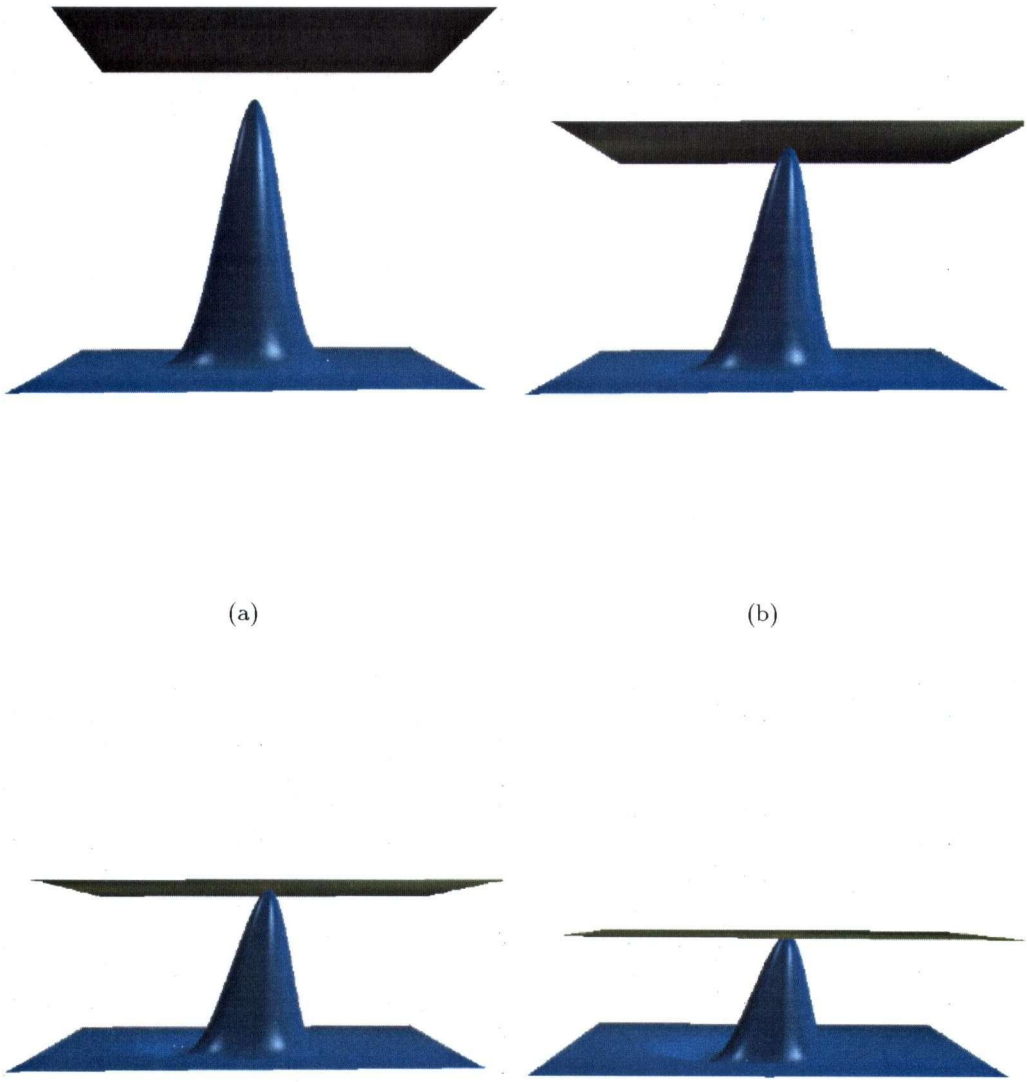
Figure 5.6: Similar to Figure 5.4 levels 1 and 2 of the bottom object absorb all the deformation.

the top object, the collision response is purely deformational and the finer levels were set to absorb the deformation. The top object was made much more rigid than the bottom object in the second animation (Figure 5.4). Figures 5.5 and 5.6 show how adjusting the absorption of the deformation to different levels of the hierarchy give various responses. Levels 1-3 of the bottom object are set to absorb the same amount of deformation in Figure 5.5 whereas levels 1 and 2 of the hierarchy absorb the deformation in Figure 5.6.

The impact of changing the response type from a deformational to translational is shown in Figures 5.7- 5.9. In both cases all of the response is absorbed by the finest level of the hierarchy and the plane pushing down on the spike is totally rigid. To show more clearly how the bottom object responds the top object is not rendered in the last three frames but the side view gives information about the relative positions of the two objects.

A rotational response is shown in the bottom object of Figure 5.10 where the bottom object absorbs all of the response on level 2 (since that is the level where the control vertices were offset to create the bump). Notice how the bump eventually rotates far enough to allow the top surface to slide past. To show a combination of a rotational and deformational response the bottom object of Figure 5.11 was set to absorb half of the response on level 1, as a deformational response, and the other half of the response on level 2, as a rotational response.

ACRA is not suitable for every animation that involves collisions since there will be some complex deformations that are difficult or even impossible to generate within the boundaries of ACRA's controls. For example, objects that are prone to self-intersection will not exhibit appropriate collision response. Some techniques for rendering an animation preview use only a sample of the complete set of frames. ACRA does not behave well in this scenario since the algorithm only uses the current state of the system. Thus if there are large changes in state, which is likely when there is a sampling of frames, then ACRA will generate an equally large response.



(a) (b)

(c) (d)

Figure 5.7: Deformational response side view.

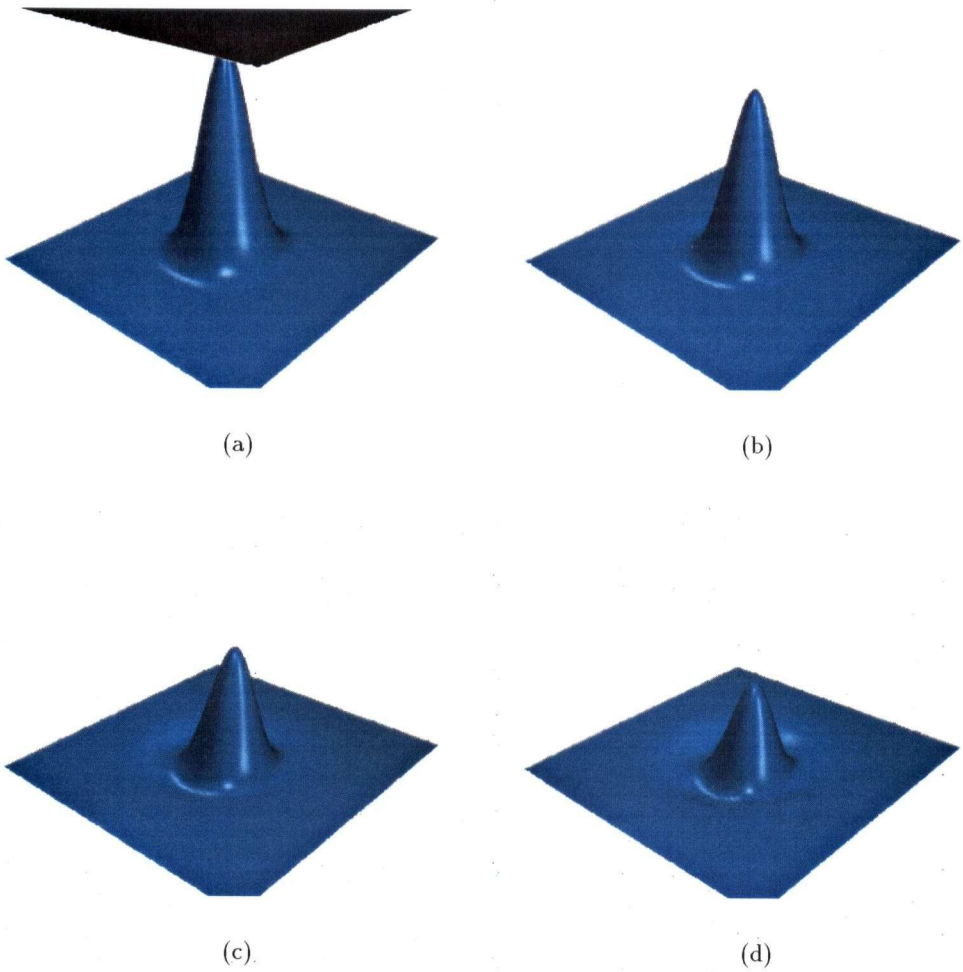


Figure 5.8: Deformational response.

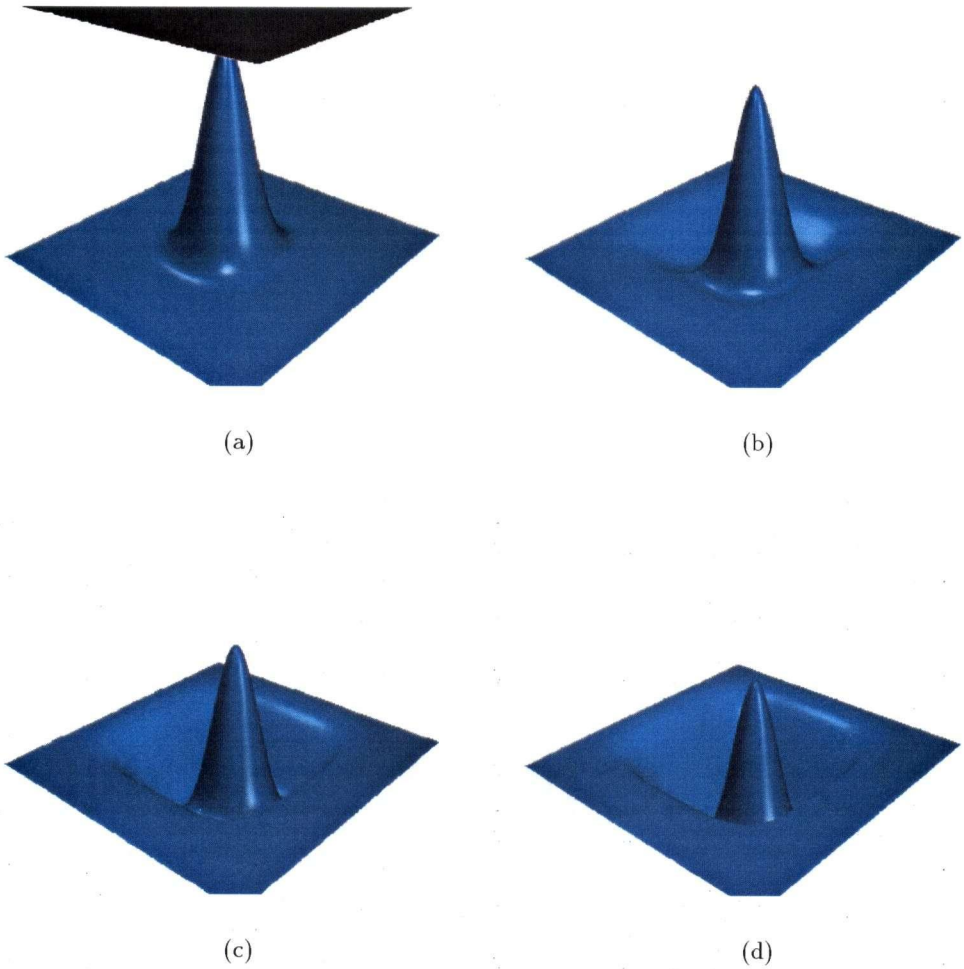
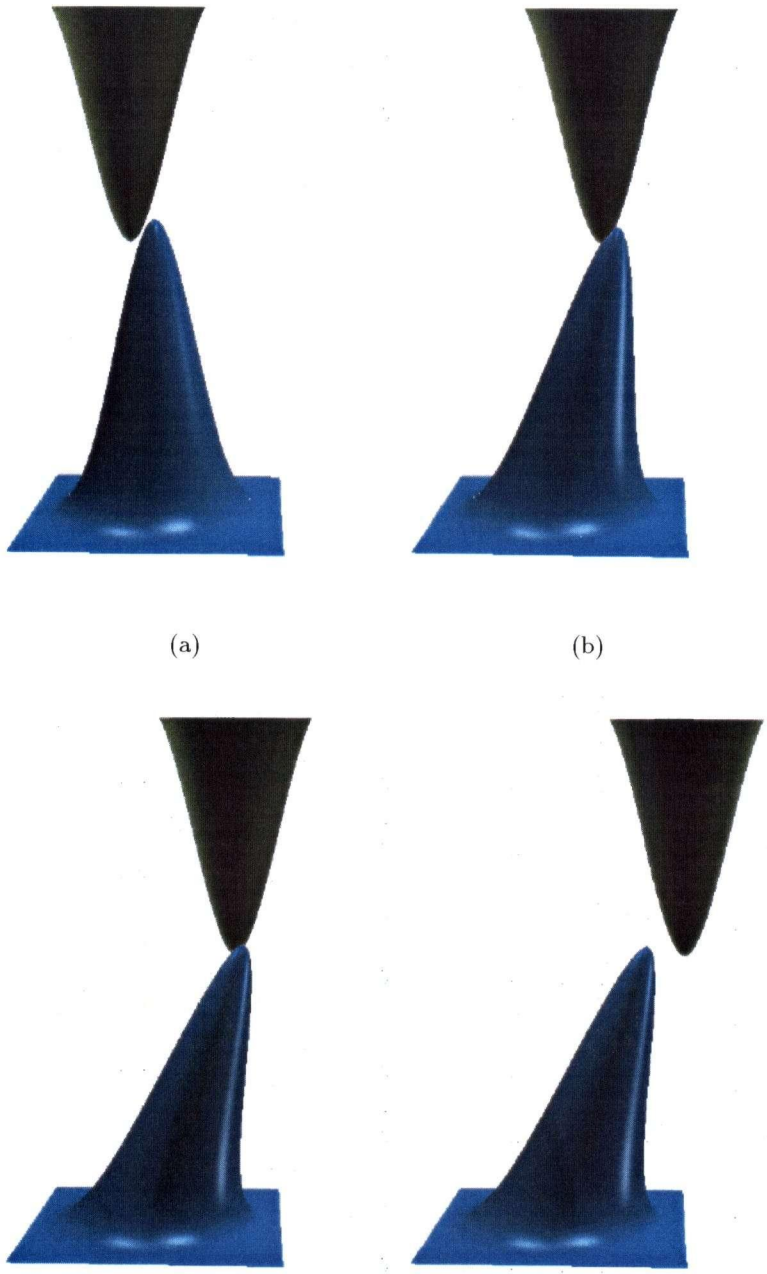


Figure 5.9: Translational response.



(a) (b)

(c) (d)

Figure 5.10: Rotational response.

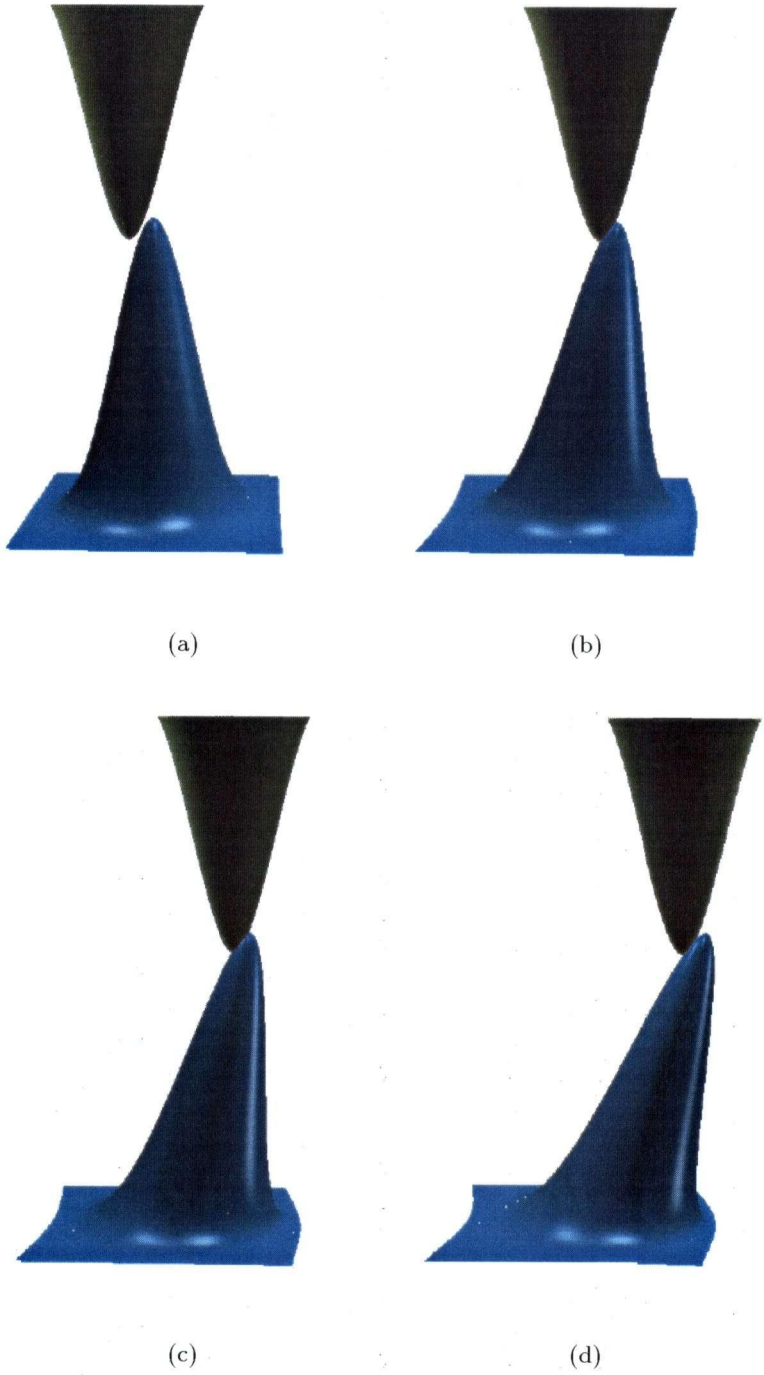


Figure 5.11: Rotational and deformational response.

Other preview techniques use a coarse representation of the object but this may not be a desirable strategy for previewing collision responses since it is difficult to collide objects whose geometry is ill defined. The user should also be aware that ACRA does not guarantee all the interpenetration between two objects will be resolved since no proof has been given that iterating the interpolation algorithm will cause the interpolation error to converge to zero. ACRA was designed, and has proven suitable for quickly removing interpenetrations between objects and providing controls to specify a variety of responses.

5.4 Future Work

One step to convincing animators and members of the graphics community of ACRA's utility is to devise a more robust mechanism for removing the interpenetration between two objects. A contact surface in the interpenetration region that both objects are constrained to would guarantee a resolution of the interpenetration. Gascuel and Desbrun [10] have found contact surfaces when the modeling primitive is implicit surfaces but some investigation needs to be done to determine the applicability of their method to other surface types.

Real objects continue to deform after the collision has finished by relaxing back to a rest shape. Using the methodology outlined by Harrison [20], ACRA could be extended to allow the animator to set the rest shape for an object. Any deformation of the object due to a collision requires a displacement of the control vertices. This displacement could be stored as an offset and, after the collision, reversed to bring the object back to its rest shape. If multiple collisions occur then the offsets for each collision could be stored in a LIFO stack and processing each element of the stack will return the object's rest shape.

Some objects, like plastercine, retain all the deformation applied to them while other objects, like an inflated balloon, will return to the shape they had before the collision. *Plasticity* is a rating of the amount of deformation an object

permanently retains. The animator could set the plasticity for an object via a percentage slider to indicate that ACRA should only restore a fraction of the stored offset. When the object attempts to return to its rest shape by reversing the stored offset it will only return part way. If a control vertex is offset by $\Delta\mathbf{V}$ and the plasticity for object i is p_i on the interval $[0, 1]$ then the stored offset for that control vertex is $(1 - p_i)\Delta\mathbf{V}$.

A real object with high viscosity will take longer to return to its rest shape than an object with low viscosity. Furthermore, an object may return to its rest shape quickly in a global sense but more slowly in a local sense. To simulate a multi-resolution relaxation rate a set of animation curves can be used where each curve describes the relaxation rate of a level.

All objects, provided they don't loose or gain any mass, conserve volume during a deformation and some objects, such as cloth and paper, conserve area (within an epsilon). Providing an option to preserve volume or area during the collision response would be beneficial, especially if the (u, v) parameterization could be adjusted to deform texture maps naturally. Currently, efficient techniques for area and volume conservation are not available so the running time of the algorithm would suffer.

Chapter 6

Summary

Simulating the collision response of real objects is a complex process and research spanning several decades has been devoted to finding algorithms to aid in the animation of collision response for both rigid and non-rigid bodies. Some research work has been targeted to specific collision response effects, such as wrinkling, while others attempt to simulate material properties, such as plasticity or inelasticity. Yet, animators still rely on the manual manipulation of data points to model soft body deformations suggesting more work needs to be done to create usable tools.

ACRA provides a tool for generating collision response that is not focused on simulating real object responses but instead attempts to remove the interpenetration between two objects. It strives to be as unintrusive as possible to the animator's workflow by both providing the animator with a variety of controls to alter the 'look' of the collision response and not altering the key-framed motion of the system. The hierarchical B-spline modeling primitive provides a local or global collision response by allowing the animator to specify the amount of response each level of the hierarchy absorbs. Three response types, deformational, translational and rotational, give the animator a greater range of responses.

ACRA can be expanded to allow for other useful effects like volume conservation, plasticity controls and the relaxation of the surface to a rest shape. In all,

ACRA is a working prototype for a first step towards a tool for generating quick, basic collision responses.

Bibliography

[1] Private communication with four animators, Greg McConnell, Trevor Bentley, Ann-May Rhodes and Anthony Law, from Totally Hip Software.

[2] Katrina M. Archer. *Craniofacial Reconstruction Using Hierarchical B-Spline Interpolation.* University of British Columbia, 1997. M.Sc. thesis. Available at http://www.cs.ubc.ca/nest/imager/th/archer.masc.1997.html.

[3] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *SIGGRAPH '89*, pages 223–232, 1989.

[4] David Baraff. Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation. In *SIGGRAPH '90*, pages 19–28, 1990.

[5] David Baraff and Andrew Witkin. Large Steps in Cloth Simulation. In *SIGGRAPH '98*, pages 43–54, 1998.

[6] Richard H. Bartels and John C. Beatty. A Technique for the Direct Manipulation of Spline Curves. In *Graphics Interface*, pages 33–39, 1989.

[7] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[8] Robert C. Beach. *An Introduction to the Curves and Surfaces of Computer-Aided Design.* Van Nostrand Reinhold, New York, NY, 1991.

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

[10] Marie-Paule Cani-Gascuel and Mathieu Desbrun. Animation of Deformable Models Using Implicit Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, 1997.

[11] Marie-Paule Cani-Gascuel and Mathieu Desbrun. Active Implicit Surface for Animation. In *Graphics Interface*, pages 143–150, 1998.

[12] Michael Carignan, Ying Yang, Nadia Magnenat Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. In *SIGGRAPH '92*, pages 99–104, 1992.

[13] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered Construction for Deformable Animated Characters. In *SIGGRAPH '89*, pages 243–252, 1989.

[14] Gerald E. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, Inc., San Diego, 1988.

[15] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics Principles and Practice.* Addison Wesley Publishing Company, Inc., Reading, Massachusetts, 1996.

[16] David R. Forsey. *Motion Control and Surface Modeling of Articulated Figures in Computer Animation.* University of Waterloo, 1990. Ph.D thesis.

[17] David R. Forsey and Richard H. Bartels. Hierarchical B-Spline Refinement. In *SIGGRAPH '88*, pages 205–212, 1988.

[18] Jean-Dominique Gascuel and Marie-Paule Gascuel. Displacement Constraints: A New Method for Interactive Dynamic Animation of Articulated Bodies. In *Third Eurographics Workshop on Animation and Simulation*, 1992.

[19] S. Gottschalk, M.C. Lin, and D. Manocha. OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. In *SIGGRAPH '96*, pages 171–180, 1996.

[20] Jason Harrison. *A Kinematic Model for Collision Response.* University of British Columbia, 1994. M.Sc. thesis. Available at http://www.cs.ubc.ca/nest/imager/th/harrison.msc.1994.html.

[21] P. Howlett and W.T. Hewitt. Mass-Spring Simulation using Adaptive Non-Active Points. In *Eurographics*, 1998.

[22] Tosiyasu L. Kunii and Hironobu Gotoda. Modeling and animation of garment wrinkle formation processes. In *Computer Animation*, pages 131–147, 1990.

[23] Seungyong Lee, George Wolberg, and Sung Yong Shin. Scattered Data Interpolation with Multilevel B-Splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, 1997.

[24] N. J. Lott and D. I. Pullin. Method for fairing B-spline surfaces. *Computer-Aided Design*, 20(10):597–604, 1988.

[25] Jerrold E. Marsden and Anthony J. Tromba. *Vector Calculus.* W. H. Freeman and Company, New York, 1988.

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

[27] Matthew Moore and Jane Wilhelms. Collision Detection and Response for Computer Animation. In *SIGGRAPH '88*, pages 289–298, 1988.

[28] Hing N. Ng and Richard L. Grimsdale. Computer Graphics Techniques for Modeling Cloth. *IEEE Computer Graphics and Applications*, 16(5):28–41, 1996.

[29] Larry Palazzi. *Deformable Models Using Displacement Constraints.* University of British Columbia, 1993. M.Sc. thesis. Available at http://www.cs.ubc.ca/nest/imager/th/palazzi.msc.1993.html.

[30] Xavier Provot. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In *Graphics Interface*, pages 147–154, 1995.

[31] Thomas W. Sederberg and Scott R. Parry. Free-Form Deformations of Solid Geometric Models. In *SIGGRAPH '86*, pages 151–160, 1986.

[32] Demetri Terzopoulos and Kurt Fleischer. Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture. In *SIGGRAPH '88*, pages 269–278, 1988.

[33] Nikitas Tsopelas. Animating the crumpling behaviour of garments. In *Eurographics Workshop on Animation and Simulation '91*, pages 11–23, 1991.

[34] University of California Davis, Computer Science Department. On-line computer graphics notes. http://graphics.cs.ucdavis.edu/GraphicsNotes/GraphicsNotes.html current as of April 28, 1998.

[35] University of North Carolina, Department of Computer Science. RAPID – Robust and Accurate Polygon Interference Detection. http://www.cs.unc.edu/~geom/OBB/OBBT.html current as of July 2, 1998.

[36] Pascal Vollino, Martin Courchesne, and Nadia Magnenat Thalmann. Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects. In *SIGGRAPH '95*, pages 137–144, 1995.

[37] Jerry Weil. The Synthesis of Cloth Objects. In *SIGGRAPH '86*, pages 49–54, 1986.

[38] Yin Wu, Nadia Magnenat Thalmann, and Daniel Thalmann. A Dynamic Wrinkle Model in Facial Animation and Skin Ageing. *Journal of Visualization and Computer Animation*, 6(4):195–206, 1995.

[39] Ying Yang and Nadia Magnenat Thalmann. An Improved Algorithm for Collision Detection in Cloth Animation. In *First Pacific Conference on Computer Graphics and Applications '93*, 1993.

[40] J. M. Zheng, K. W. Chan, I. Gibson, and Q. Y. Xiong. A New Approach for Direct Manipulation of Free-Form Curve. In *Eurographics*, 1998.

Glossary

Basic collision response: The removal of the interpenetration between two or more objects, or at least enough of the interpenetration so that only a small amount of post-processing 'clean-up' removes the remainder.

Blending functions (also basis functions): The weights applied to the elements of the geometry vector when calculating the parametric curve.

Collision response: Any reaction, whether as a rigid body or a non-rigid body, to a collision such that the interpenetration due to the collision is removed.

Derived vector: Used by a hierarchical B-spline surface to specify the position of a control vertex, before it has been displaced, in the object's local reference frame.

Elastic: A property of an object. An object that returns fully to its original shape when all external forces are removed is said to be elastic.

Fracture: The cracking or tearing of a material when it deforms beyond a certain limit. Cracks develop according to internal force or deformation distributions and their propagation is affected by local variations in material properties.

Free form deformations: A method for deforming a model within a parallelepiped region of space. An object within the Bézier solid that defines the parallelepiped can calculate new positions for its vertices such that the object deforms as the parallelepiped deforms.

Geometry vector: A column vector of geometric constraints. The geometric constraints are the conditions, such as endpoints or tangent vectors, that define the parametric curve.

Inelastic: A property of an object. Inelastic objects return to their original shape slowly or only partially when all external forces are removed.

Knot: A join point between two adjacent B-spline segments that has a knot value associated with it.

Local refinement: A technique used by hierarchical B-spline surfaces of adding only those control vertices needed for editing purposes while retaining the unedited portion of the surface in its original definition.

Minimum distance shape change: The minimum distance shape change of a curve is the change in curve shape such that the curve point $\mathbf{Q}(\mathbf{u})$ interpolates a specified point in space but the control vertices of the curve segment containing $\mathbf{Q}(\mathbf{u})$ undergo a minimal displacement.

Offset vector: The displacement of a control vertex from its position indicated by the derived vector. The offset vector is specified in the local frame of reference at the position indicated by the derived vector.

Physically-based modeling: Any system that uses some laws of physics to iteratively calculate the positions, velocities, forces or other properties of an object.

Piecewise linear: Refers to a series of line segments that approximates the shape of other primitives such as curves or surfaces.

Piecewise polynomial: Refers to the approximation of a curve or surface by a series of polynomial segments. Cubic polynomials are used most often because they are the lowest order polynomial that can interpolate two endpoints and specify tangents at each endpoint.

Plasticity: A rating of the amount of deformation an object permanently retains.

Prolongation operator: An operator that passes information from a coarser level to a finer level of a multi-resolution surface.

Restriction operator: An operator that filters information from a finer level to a coarser level of a multi-resolution surface.

Viscoelastic: A property of an object. A viscoelastic object's behavior includes the characteristics of a viscous fluid together with elasticity.

Viscosity: The property of a fluid that resists the forces tending to cause flow.

Index

additional points, 34
animation preview, 2, 50

B-spline
    basis matrix, 15
    blending functions, 14–16, 20, 26
    geometry vector, 14–16, 18
    knot, 15, 20
    knot value, 15
B-spline basis functions, *see* B-spline blending functions

clean-up, *see* post-processing
cloth, 4
collision detection, *see* RAPID
conservation
    area, 57
    volume, 57
contact points, 33
continuity, 14, 19, 22
control mesh, 1, 4
curve
    B-spline, 14–18, 26
    Bézier, 14, 17
    Hermite, 14
    parametric, 13–14

dynamics, 4

elasticity, 4–6

fracture, 4–5, 11
free form deformation, 6–8

grid, *see* control mesh

H-spline
    derived vector, 22–23
    offset vector, 22–23, 56

inelasticity, 4–5

local refinement, 21, 22, 32

multi-resolution interpolation, 31

numerical methods, 4

physically-based modeling, 1, 3–6
    springs, 4, 5
    using FFD, 8
plasticity, 4–6, 56
point masses, *see* control mesh
post-processing, 2, 42
preview, *see* animation preview
properties
    elasticity, 4–6
    fracture, 4–5, 11
    inelasticity, 4–5
    plasticity, 4–6, 56
    viscoelasticity, 4–5

RAPID, 43–44
relaxation rate, 57
response
    deformational, 34–37, 50
    rotational, 34–35, 39–40, 50
    translational, 34–35, 38–39, 50

rest shape, 56
rigidity, 2, 11, 33–34

spring mesh, *see* control mesh
subdivision
    B-spline, 16–18, 20
    Bézier, 16–18
    H-spline, 32
surface
    B-spline, 18–20
    Bézier, 12
    implicit, 9, 56
    piecewise linear, 9, 12

time step, 4

viscoelasticity, 4–5

weights
    curve interpolation, 27–28
    surface interpolation, 29–30
wrinkle
    cloth, 5–6
    facial, 6