Animated Surface Pasting

by

Clara Tsang

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1998

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Surface pasting is a composition method in which a feature surface is attached to a base surface to provide details on the underlying surface. In computer animation, a lot of time is spent modeling and surface pasting is a modeling method that can quickly and easily model faces and other objects with small details. Thus, surface pasting may be a useful animation technique because it reduces the modeling time. However, it is unclear whether pasted surfaces will have problems such as distortion in animation. In this thesis, surface pasting was combined into the animation process to see how pasted surfaces behave in animation. In general, while surface pasting behaved as desired in animation, there are some situations where it behaves poorly. Most of these bad situations can be avoided or corrected by the animator.

Acknowledgements

I would like to thank my supervisor, Stephen Mann, for his invaluable guidance, suggestions, support and patience throughout my research, revisions of the drafts of this thesis and delicious ice-cream over the years. I would also like to thank my co-supervisor, Richard Bartels, who suggested the research idea and made suggestions on the development of my research, and I would like to thank my readers, Chrysanne DiMarco and Rick Kazman, for their time, feedback and suggestions. I would also like to thank Mark Riddell for drawing the animation background, helping me to figure out the animation sequence and making the video titles, and Patrick Gilhuly for recording the video.

Financial support for the research presented in this thesis was provided by ITRC/CITO.

Finally, I dedicate this work to my family and Danny for their unconditional love and support.

Trademarks

SGI and Open Inventor are registered trademarks of Silicon Graphics, Inc.

The Splines Library is under the copyright of the Computer Graphics Laboratory at the University of Waterloo.

All other products mentioned in this thesis are trademarks of their respective companies. Our use of general descriptive names, trade names, and trademarks, even if not precisely identified, is not an indication that such names may be used freely in all circumstances.

Contents

1 Introduction **1**

1.1 Overview of Chapters 4

2 Background **5**

2.1 B-Spline Curves 5

2.1.1 Knot Multiplicity And Continuity 6

2.2 Tensor Product Surfaces 7

2.3 Surface Pasting 9

2.3.1 The Diffuse Coordinate System 9

2.3.2 Greville Displacement B-Splines 10

2.3.3 Pasting Greville Displacement B-Splines 12

2.3.4 Continuity of Pasted Surfaces 14

2.4 Motivation of Surface Pasting 14

2.5 Related Work 16

2.6 Keyframe Animation 17

3 Animated Surface Pasting **21**

3.1 Motivation And Goals 21

3.2 Puppet Model 22

3.2.1 Skinning 23

3.2.2 Pasted Details 33

3.3 Keyframe System 37

3.3.1 Catmull-Rom Splines 37

3.3.2 Special Cases 38

3.4 User Interface Details 42

3.4.1 Keyframe System User Interface 42

3.4.2 Parameter User Interface 44

4 Results **47**

4.1 Description of Animation 47

4.2 How Pasted Surfaces Behave 49

5 Summary and Conclusions **57**

5.1 Animated Surface Pasting 58

5.2 Conclusions 59

5.3 Future Work 60

Bibliography **61**

List of Tables

3.1 Degree of Freedom on Puppet's Joints 22

3.2 Mouse Functions 22

3.3 A List of Offset Values 28

List of Figures

2.1 Two Curves Join With C^1 Continuity 7

2.2 Two Surface Patches Join With C^1 Continuity 8

2.3 Greville Displacement B-Spline 11

2.4 Surface Pasting Procedure 13

2.5 Knot Insertion Procedure 15

3.1 Puppet Model 24

3.2 Skeleton Version of Puppet Model With Key Points 26

3.3 Puppets With Different Offset Values 27

3.4 Smooth Shoulder Problem 29

3.5 Puppet Model With Highlighted Shoulders 30

3.6 Shoulder Connector To Arm 31

3.7 First Surface Patches of Shoulder 32

3.8 Unpasted Nose And Ear Model 33

3.9 Eye Model 33

3.10 Five Lip Models 34

3.11 Finger Models 36

3.12 Catmull-Rom Spline 40

3.13 Special Case 5 Solution 40

3.14 Special Case 5 Counter-Example 41

3.15 User Interface of Keyframe System 43

3.16 Parameter User Interface Dialog Box 44

4.1 The Scream Frame 48

4.2 Different Poses of Head and Facial Features 52

4.3 Modeling Problem 53

4.4 Bigger Pupil of Eye Problem 54

4.5 Fingers Distortion Problem 55

4.6 New Grasping Motion 56

Chapter 1

Introduction

Computer animation is now used widely in many different areas, such as movies, advertisements and education. There are three steps for making an animation. In traditional animation, the first step is to create a storyboard and design the characters of the animation. The second step is for a skilled animator to draw the *keyframes* and the last step of producing an animation is for less-skilled artists to draw the remaining animation frames. In computer animation, the first step is the same as traditional animation, but the second and the last steps are different. The second step of computer animation is to model the characters through computers and to define all the details of animation, such as the animation procedure and the animation sequences. Finally, the last step is to render the animation frames by computer. Even though faster than traditional animation, the computer animation process is still time intensive, requiring a lot of human time for modeling and setting up the animation.

For modeling task, the animator cares about how he/she can build a model

of what he/she wants easily and in an efficient way. For example, *Alias Maya*[1] provides a full complement of NURBS surface and polygon modeling tools. An animator can define his/her own curves, surfaces or polygons, or use the NURBS primitives, such as spheres, cubes, cylinders, cones and plane, to model an object. After building the shape of the object, the animator can make the object more realistic by adding textures, colors and backgrounds. The animator can combine several objects together by adding joints to build a skeleton.

Side Effects Houdini[2] has similar functionalities to *Maya*, but *Houdini* is built on a procedural animation paradigm. The user builds networks of operators, each of which controls one task. The user can insert operators into the network, remove them from the network or change them without endangering the previous work.

Most of the animation tasks are related to the motion control techniques. The most common motion control techniques for animating articulated structures are keyframe animation and kinematics. *Alias Maya* provides two types of kinematics: forward kinematics and inverse kinematics, so the user can choose the way that he/she wants to pose the skeleton. *Houdini* has an additional kinematics technique, called follow-curve kinematics on multi-bone chains. Free form deformation is one common motion control technique for animating soft objects. *Maya* has different types of deformers: sculpt deformer, lattice deformer, wire deformer, cluster deformer and blend shape deformer. *Houdini* has three surface deformations: lattice and magnet deformation and multi-object interpolation. Both *Maya* and *Houdini* have similar functionalities for springs dynamics. The motion control techniques for procedural animation include particle set animation and behavioral procedural animation. In both *Maya* and *Houdini*, the user can create collisions between

[1] Maya is a trademark of Alias|Wavefront, a division of Silicon Graphics Limited

[2] Houdini is a trademark of Side Effects Software Inc.

rigid objects and particles system features, such as fire, smoke, rain, fireworks or explosions effects.

Most of the time for making an animation is spent on modeling. With standard modeling techniques, the process of modeling the required detail involves trial and error manipulation of many control points, polygons, etc. Unfortunately, this detailed work can rarely be reused; for example, to create two faces, two models are usually constructed from scratch completely independently even though many of the details may be the same. If the modeling process can be shortened, the life of the animator will be easier.

Surface pasting is a new method for modeling. With this technique, the animator can model, for example, a puppet with different appearances by selecting different features from a database of features and pasting them on the base model without having to re-model all the details of each feature. Side Effects Software Inc. has recently incorporated a surface pasting technique into *Houdini*, but the method is new enough that *Houdini* users have had little time to use and test the technique. Previous work has shown surface pasting to be a useful modeling technique. However, it is unknown how well surface pasting features will behave in an animation system. In this thesis, I tested how pasted surfaces behave in animation. I tested this by implementing a keyframe animation system, which used surface pasting for modeling a puppet. I used my system to generate an animated short, which allowed me to observe how well surface pasting works in animation.

From the animation short, I found that pasted surfaces behaved well under deformation of a base surface. However, they did not behave correctly as they moved on a base surface themselves.

1.1 Overview of Chapters

In Chapter 2, I present an overview of B-spline curves and tensor product surfaces and illustrates how the surface pasting method works. Then, I explain why we use the surface pasting technique and list the related work. Finally, I review the idea of keyframe animation.

In Chapter 3, I motivate my research and describe its design. Finally, I give a detailed description of the testing application.

The description of the animation and the results of the tests are described and analyzed in Chapter 4.

Chapter 5 summarizes the results of my research and what I found during my research and lastly, discusses the possibility of future research.

Chapter 2

Background

This chapter introduces the background material for surface pasting and animation. In particular, I discuss the concept of a tensor product surface, describes the surface pasting procedure [Bar94, BBF95], shows some related surface pasting work [Bar94, Cha96] and presents the basic idea of keyframe animation.

2.1 B-Spline Curves

A B-spline curve is a piecewise polynomial defined by control points P_i and a set of basis functions $B_i^m(u)$:

$$C(u) = \sum_{i=0} P_i \, B_i^m(u) \, .$$

Each basis function B_i^m is defined by a knot vector, which is a sequence of nondecreasing knot values

$$u_0 \leq u_1 \leq u_2 \leq \cdots \leq u_M \, .$$

The univariate B-spline basis functions of order m or degree $m-1$ are defined by the following recursive formulas:

$$B_i^1(u) = \begin{cases} 1 & \text{if } u_i \le u \le u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_i^m(u) = \frac{u - u_i}{u_{i+m-1} - u_i} B_i^{m-1}(u) + \frac{u_{i+m} - u}{u_{i+m} - u_{i+1}} B_{i+1}^{m-1}(u)\,,$$

where $B_i^m(u)$ is non-zero over m subintervals of the range of u. There is a relationship between the number of knot values no_kv and the number of control points no_cp

$$no\_kv = no\_cp + order - 1 = no\_cp + degree\,.$$

Only cubic curves are considered in this thesis. Hence, the order of the basis function is four (m = 4) and the degree is three.

2.1.1 Knot Multiplicity And Continuity

As mentioned above, a knot vector is a sequence of nondecreasing knot values. *Knot multiplicity* is the number of times a knot value occurs in a knot vector. If the multiplicity equals the degree of the curve, the knot vector has full multiplicity. If a knot vector has multiplicity k at knot value u_i, then a degree n B-spline curve is with C^{n-k} continuous at u_i.

The de Boor algorithm is one method to interpolate the control points and evaluate the curve. The point on the curve at value t can be evaluated by doing repeated knot insertion until knot t has full multiplicity. Thus, the curve will pass through a control point if the knot vector has full multiplicity at the corresponding knot value. Knot vectors are usually set with full multiplicities at both ends, so the curve will start and end with the first and last control points respectively.

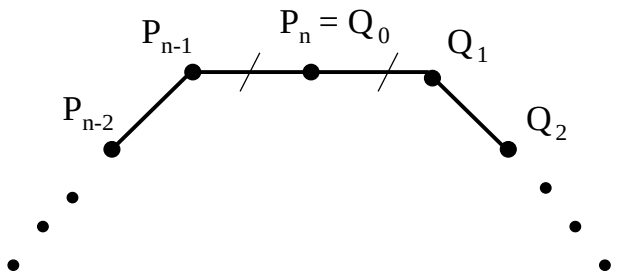


Figure 2.1: Two Curves Join With C^1 Continuity

Two B-spline curves can be joined together with different continuities. C^0 continuity is trivial: for two curves with full multiplicity of their end knots to meet C^0, the last control point of the first curve must equal the first control point of the second curve. For C^1 continuity, the two curves have to meet with C^0 continuity and then the vector between the last two control points on the first curve and the vector between the first two control points on the second curve have to be the same (see Figure 2.1). Geometrically, length is not required for a smoothly connecting curve. Two curves still meet smoothly if these two vectors are parallel but of different length.

2.2 Tensor Product Surfaces

A tensor product surface is determined by a rectangular mesh of three-dimensional control points and a patch basis function, which is the product of curve basis functions $B(u)$ and $B(v)$, over a two-dimensional domain D. The two curve basis functions $B(u)$ and $B(v)$ are determined by two sets of knot vectors $u_0, \cdots, u_M$ and $v_0, \cdots, v_N$.

The indexing scheme of control points $P_{i,j}$ is used to reflect the geometric arrangement of the control points. Since the control points $P_{i,j}$ are arranged in a

rectangular topology, the tensor product B-spline surface $S(u, v)$ can be expressed by a double summation:

$$S(u,v) = \sum_{i=0} \sum_{j=0} P_{i,j}\, B_i^m(u) B_j^n(v)\,,$$

where m and n are the orders of the first and second curve basis functions respectively. The curve basis functions $B_i^m(u)$ and $B_j^n(v)$ are defined as in Section 2.1.

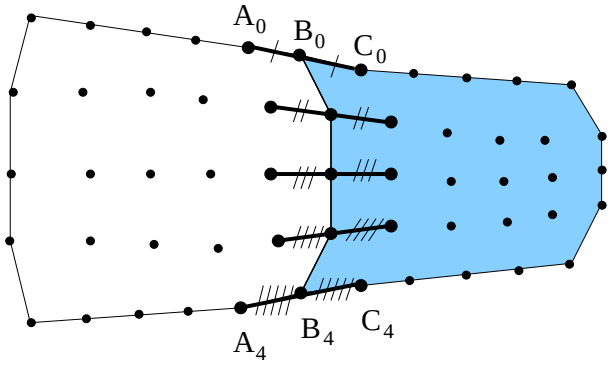


Figure 2.2: Two Surface Patches Join With C^1 Continuity

We use the idea of curve continuity for tensor product surface continuity. If we want two tensor product surface patches to meet with some level of continuity along a boundary, we just need to construct corresponding rows (or columns) of control points, which when treated as curves, meet with that continuity. For example, in Figure 2.2, to join two tensor product surface patches with C^1 continuity, the boundary curve of one patch has to meet with the boundary curve of the other patch and for each row of control points, each boundary control point B_i must be at the midpoint of the segment spanned by the two control points (A_i and C_i) on either side. Again, a relaxed form of continuity allows each B_i to lie anywhere along the corresponding $\widehat{A_iC_i}$ segment, as long as each B_i breaks the corresponding $\widehat{A_iC_i}$ segment into the same ratios.

2.3 Surface Pasting

Surface pasting is a process in which a feature surface is attached on a base surface as a detail. The base surface can be a single surface or a composite surface. After pasting, the original underlying surface is not changed and the feature surface retains the character of its unpasted shape and reflects the topography of the underlying surface as well. Details are added to surfaces by applying displacement maps as described below.

The surface pasting method, which will be presented in this section, was developed by Barghiel [Bar94, BBF95] from a technique proposed by Forsey and Bartels [BF91]. This section first presents the Diffuse Coordinate System and Greville Displacement B-Splines and outlines the procedures of surface pasting.

2.3.1 The Diffuse Coordinate System

In a Diffuse Coordinate System, there is a diffuse coordinate space (*DCS*). Under *DCS*, each control vertex has its own associated local coordinate frame $\mathcal{F}_{i,j} \in DCS$. Each control vertex can be written as an origin $O_{i,j}$ plus an offset vector $\boldsymbol{d}_{i,j}$ in the diffuse coordinate space:

$$\begin{aligned} S(u,v) &= \sum_{i=0}\sum_{j=0} P_{i,j}\, B_i^m(u) B_j^n(v) \\ &= \sum_{i=0}\sum_{j=0} (O_{i,j} + \boldsymbol{d}_{i,j}) B_i^m(u) B_j^n(v) \end{aligned}$$

The quality of the pasting operation and the behavior of pasted surfaces depend on the choice of origin $O_{i,j}$ and displacement vector $\boldsymbol{d}_{i,j}$. Barghiel [Bar94] used the Greville Displacement to define the origin and the displacement vector.

2.3.2 Greville Displacement B-Splines

In surface pasting, control points are represented as offsets from local coordinate frames. This section describes the details of those offsets. Each B-spline surface has its corresponding two-dimensional domain, which is determined by the knot values in each parametric direction [BBB87].

For each control vertex $P_{i,j}$ of a B-spline surface, there is an associated domain point, called a *Greville point*, where the surface point at this domain point is maximally influenced by $P_{i,j}$ [Far94]. The Greville point $\gamma_{i,j}$ associated with control vertex $P_{i,j}$ is defined as a pair of *Greville abscissae*, one for each parametric direction:

$$\gamma_{i,j} = (\gamma_i, \gamma_j)\,,$$

where the Greville abscissae γ_i and γ_j are the average of d (surface degree) knots in parametric directions i and j, respectively.

The two-dimensional Greville point can be embedded into a three-dimensional space by adding an additional coordinate (e.g., 0) to each point. Then the three-dimensional Greville point is

$$\Gamma_{i,j} = (\gamma_{i,j}, 0) = (\gamma_i, \gamma_j, 0)$$

The three-dimensional Greville point $\Gamma_{i,j}$ is chosen to be the origin $O_{i,j}$ of the local frame. Then the displacement vector $\boldsymbol{d}_{i,j}$ will be a vector from the origin $\Gamma_{i,j}$ to the control vector $P_{i,j}$:

$$\boldsymbol{d}_{i,j} = P_{i,j} - \Gamma_{i,j}\,,$$

which is called the *Greville displacement*. Thus, the diffuse representation of a B-spline point is:

$$S(u,v) = \sum_{i=0} \sum_{j=0} (\Gamma_{i,j} + \boldsymbol{d}_{i,j}) B_i^m(u) B_j^n(v)$$

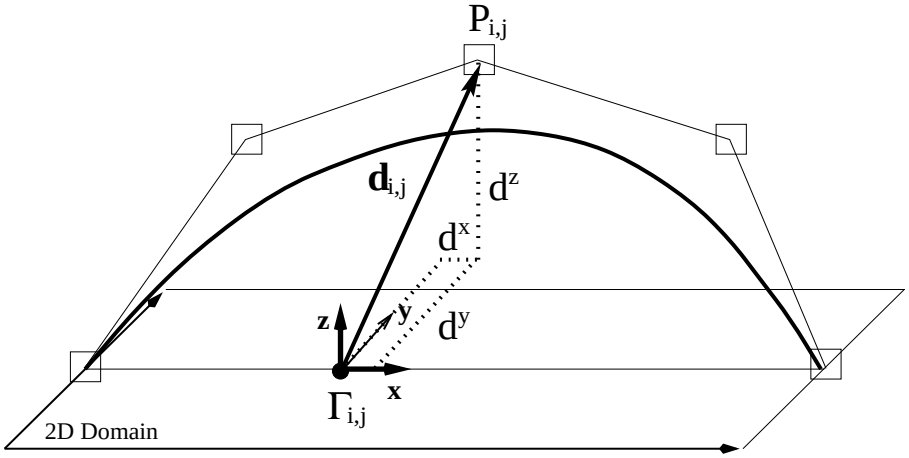


Figure 2.3: Greville Displacement B-Spline

as illustrated in Figure 2.3.

The basis ($\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$) for the local frame is constructed with $\boldsymbol{x}$ and $\boldsymbol{y}$ as the two parametric directions of the embedded domain and $\boldsymbol{z} = (0, 0, 1)$. The Greville displacement $\boldsymbol{d}_{i,j}$ can be expressed as a linear combination of the coordinate vectors of the local frame:

$$\boldsymbol{d}_{i,j} = d^x_{i,j}\, \boldsymbol{x} + d^y_{i,j}\, \boldsymbol{y} + d^z_{i,j}\, \boldsymbol{z}$$

where $d^x_{i,j}$, $d^y_{i,j}$ and $d^z_{i,j}$ are the x, y and z components of the Greville displacement vector, respectively.

Thus, a B-spline point can be rewritten as a collection of local displacements from the Greville abscissae and represented vectorially as a *Greville displacement B-spline* by removing the origins $\Gamma_{i,j}$:

$$S(u, v) = \sum_{i=0} \sum_{j=0} \boldsymbol{d}_{i,j}\, B^m_i(u) B^n_j(v)\,.$$

This is the representation of the control point being used for surface pasting.

2.3.3 Pasting Greville Displacement B-Splines

Figure 2.4 shows the process of surface pasting. There are at least two surfaces involved in the pasting procedure: a *base surface* S_B on which to paste, defined over the base domain, and a *feature surface* S_F that pastes onto S_B, defined over the feature domain. Surface pasting is an operation that manipulates the location of the feature's control vertices, provided the displacement vectors remain unchanged.

The process of pasting a feature surface onto a base surface starts with mapping the feature domain into the base domain by a transformation T. This transformation T is used to determine the location and the size of the feature surface relative to the base surface.

After mapping the feature domain into the base domain, the origin $\Gamma_{i,j}$ of the feature local frame becomes $T(\gamma_i, \gamma_j) = (\gamma_i', \gamma_j')$. The base surface point $S_B(\gamma_i', \gamma_j')$, which is the image of the mapped $\Gamma_{i,j}$, is used as a new origin for positioning the feature displacement vector $\boldsymbol{d}_{i,j}$ during pasting. The basis $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ maps to $(\boldsymbol{x'}, \boldsymbol{y'}, \boldsymbol{z'})$ where $\boldsymbol{x'}$ and $\boldsymbol{y'}$ match the embedded feature domain's parametric directions, and $\boldsymbol{z'}$ remains perpendicular to the embedded feature domain.

Before the feature displacement vector $\boldsymbol{d}_{i,j}$ is applied to the mapped origin $S_B(\gamma_i', \gamma_j')$, a local base coordinate frame $\boldsymbol{\mathcal{F}}_B$ at $S_B(\gamma_i', \gamma_j')$, where $\boldsymbol{\mathcal{F}}_B = \{S_B(\gamma_i', \gamma_j'), \boldsymbol{v}(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k})\}$ has to be constructed. To reflect the curvature of the base surface at the mapped origin $S_B(\gamma_i', \gamma_j')$, the components $\boldsymbol{i}$ and $\boldsymbol{j}$ of the vector $\boldsymbol{v}$ are calculated as the partial derivatives of the base surface S_B at $S_B(\gamma_i', \gamma_j')$ in the $\boldsymbol{x'}$ and $\boldsymbol{y'}$ directions ($\frac{\partial S_B}{\partial x'}$ and $\frac{\partial S_B}{\partial y'}$), respectively, and $\boldsymbol{k}$ is the cross product of $\boldsymbol{i}$ and $\boldsymbol{j}$.

Finally, the pasted control vertex $P_{i,j}'$ results from embedding the displacement vector $\boldsymbol{d}_{i,j}$ of the feature into the local base coordinate frame $\boldsymbol{\mathcal{F}}_B$. The pasted

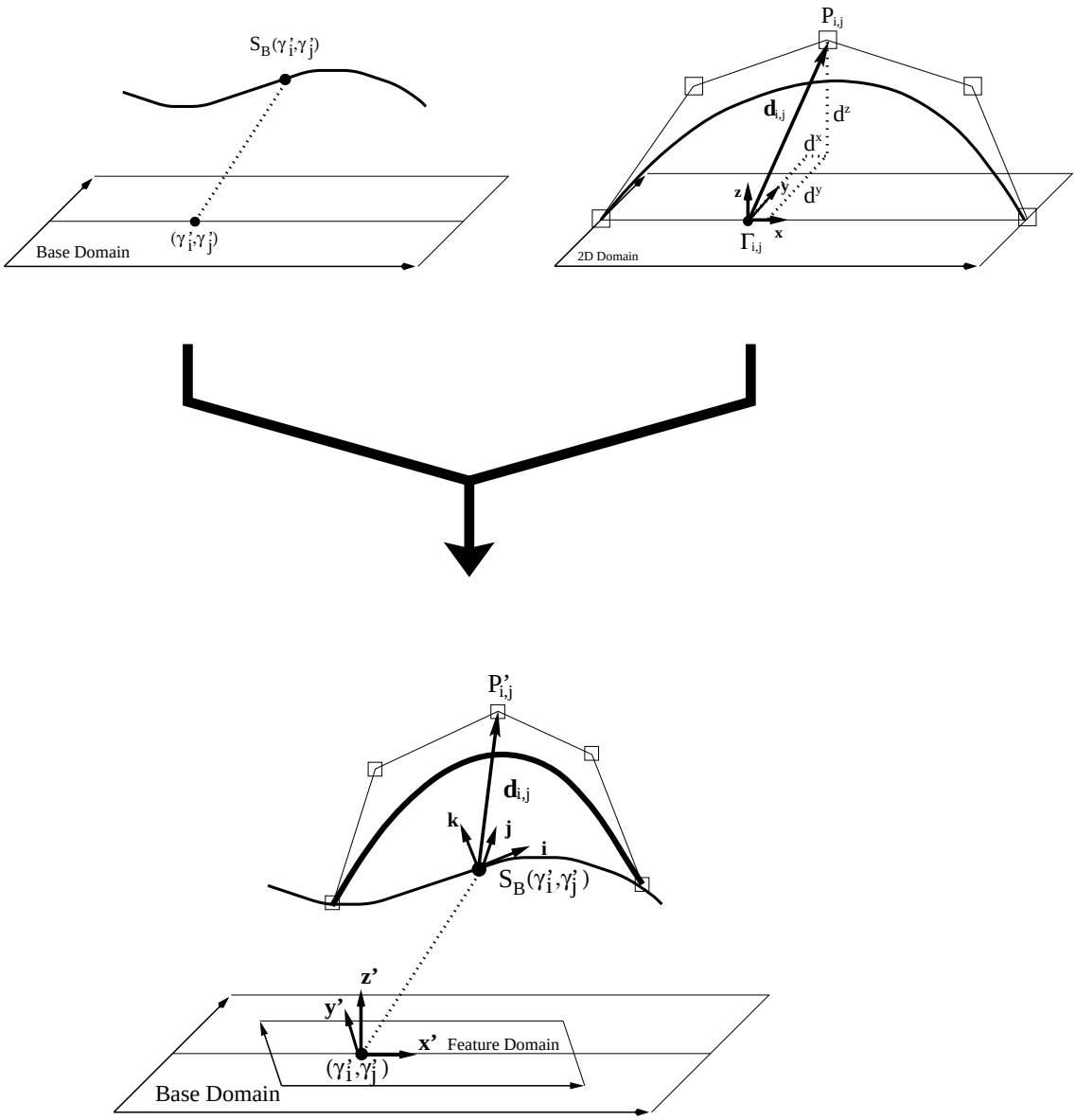


Figure 2.4: Surface Pasting Procedure

control vertex $P'_{i,j}$ is calculated by a point-vector addition performed in the given frame context:

$$\begin{aligned} P'_{i,j} &= S_B(\gamma'_i, \gamma'_j) + \boldsymbol{d}_{i,j} \\ &= S_B(\gamma'_i, \gamma'_j) + d^x_{i,j}\,\boldsymbol{i} + d^y_{i,j}\,\boldsymbol{j} + d^z_{i,j}\,\boldsymbol{k} \end{aligned}$$

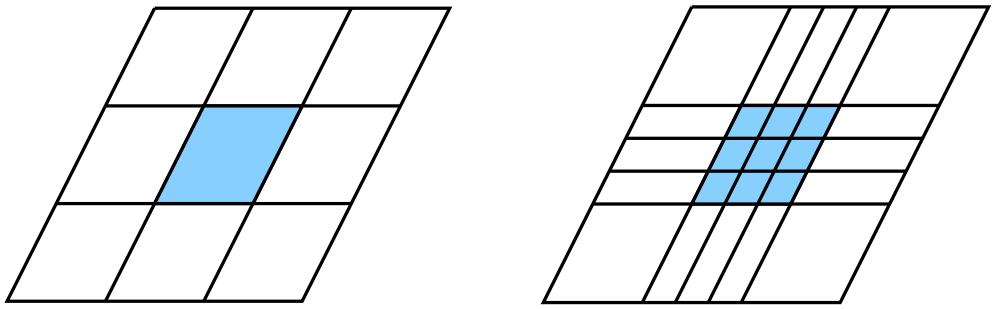
2.3.4 Continuity of Pasted Surfaces

Surface pasting is an approximation technique; the pasted surfaces do not meet a base surface even with C^0 continuity. However, if we set each control point of the boundary at Greville domain points, those unpasted control points will lie on the x-y plane and will paste on the base surface. Hence, the connection between the pasted surface and the base surface will be nearly C^0 provided the base surface's curvature is not too great relative to the spacing of the points $S_B(\gamma'_i, \gamma'_j)$. Moreover, we can get an approximation of the feature boundary to any level of error tolerance by refining the knot vector of the feature surface.

If we additionally place the second layer of control points around the boundary at the Greville domain points, then the feature will meet the base with approximate C^1 continuity because this will put the appropriate control points nearly in the configuration of Figure 2.2.

2.4 Motivation of Surface Pasting

If we want to add details to a B-spline tensor product surface, the number of control vertices must increase. One way to increase the number of control vertices is to refine the surface with knot insertion.



(a) before knot insertion (b) after knot insertion

Figure 2.5: Knot Insertion Procedure

Boehm's algorithm [Boe80] and the Oslo algorithm [BBB87] are popular methods for knot insertion. To split a knot in a patch (shaded area in Figure 2.5(a)), an entire row and/or an entire column across the entire surface has to be refined. Hence, several rows and/or several columns of control vertices are added inside the refined row and column as shown in Figure 2.5(b). Thus, adding details to a particular region/patch will increase the complexity of the surface. Further, the additional control vertices outside the refined patch are unnecessary. This results in a higher cost to evaluate the surface.

Surface pasting is a method of adding details on the surface without increasing the complexity of the base. Under the surface pasting method, the underlying original surface is not changed; we merely add another B-spline tensor product surface as detail on top of the underlying surface. Therefore, there is no increase in the complexity of the surface nor any addition of unnecessary control vertices.

As mentioned before, there is an alternative way to add details: hierarchical B-splines, which were presented by Forsey and Bartels [FB88]. Their method was aimed at minimizing the number of control vertices in the surface by only

inserting control vertices within the editing region by doing knot insertion. The editing/refinement process can be localized by keeping the first two layers of boundary control vertices within the editing region static to maintain C^1 continuity and manipulating only the central control vertices. If we fix the first three layers of boundary control vertices, then we will have C^2 continuity. Each control vertex is represented in terms of an *offset* relative to a hierarchy of local reference frames. In this method, detail surfaces, called *overlays*, are formed on top of a base surface in a hierarchical fashion by repeating this approach on the interior of an overlay, regarding it as the parent surface for the creation of further overlays.

Although hierarchical B-splines can maintain the number of control vertices as low as possible and keep mathematical continuity as high as possible, there are some advantages of surface pasting over hierarchical B-splines. The main advantage of pasted surfaces is arbitrary placement and orientation of features. This allows us to use a variety of useful modeling techniques. For example, we can create a database of features to use for surface pasting. Consider modeling a new car. The designer can choose one car handle model from the database of car handle models and paste it on the car model to see how the new car looks. Such databases are not easy to implement with hierarchical B-splines.

Moreover, surface pasting allows more flexible modeling paradigms, such as rotation and translation of the features. These paradigms are difficult to implement with hierarchical B-splines.

2.5 Related Work

Barghiel [Bar94] and Chan [Cha96] have done previous work on surface pasting. Bartels and Forsey [BF91] extended a displacement method proposed by Forsey and

Bartels [FB88]. Barghiel [Bar94] designed the data structures necessary to support interactive pasting and implemented the first surface-pasting editor. Barghiel implemented a spline surface-pasting editor called *PasteMaker* to demonstrate the concepts of surface pasting interactively, in real-time. In PasteMaker, the user can control the location and the size of the features by manipulating their domains in a Domain Space User Interface, which is two-dimensional.

Chan [Cha96] extended Barghiel's PasteMaker so that the user can manipulate the three-dimensional composite surface directly, instead of having to manipulate the two-dimensional surface domains. Chan implemented a *World Space User Interface* called *PasteInterface* to demonstrate the concepts of three-dimensional manipulated user interface. The main task of PasteInterface is to convert three-dimensional operations to the underlying two-dimensional domain functions used in surface pasting [Bar94]. In PasteInterface, the user can perform all domain operations provided by the domain space user interface, but interact with the three-dimensional model directly.

2.6 Keyframe Animation

Keyframe systems take their name from the traditional production systems such as the one used by Walt Disney [WW92]. In these systems, skilled animators designed the 2-D figure at various positions in a particular sequence, which were so-called *keyframes*. Keyframes mark the changes in the characteristics of the figure. Then, the drawing process was passed on to less skilled animators to produce *inbetween* frames based on the given keyframes. The more keyframes the skilled animator produced, the more control he/she could have on the animation.

In 3-D computer animation, keyframe animation is a scheme of dividing the

work between the animator and the computer system [WB76]. The animator builds keyframes at specified intervals in an animation sequence. Basically, a keyframe is a set of parameters, for example, positions, rotation angles, sizes and so on. The animator defines all the parameters for each keyframe. Then, the computer generates the inbetween frames by interpolating the parameters between each pair of keyframes [BW71]. The quality of the animation depends on the choice of interpolation techniques. There are two major interpolation techniques: *linear interpolation* and *spline interpolation.*

Linear interpolation uses the initial and final position of an object that are specified in each of two keyframes and then each inbetween frame is calculated via a linear equation. This kind of interpolation technique often produces poor results because the motion of an object may be jerky across keyframes.

Using a spline interpolation technique, the motion of an object is defined by an interpolating cubic spline curve. The positions of an object specified inside the keyframes are used to define the cubic spline curve. After the spline curve is defined, the inbetween frames are calculated via the spline equation. This kind of interpolation technique produces better results than linear interpolation techniques. Thus, a spline interpolation technique commonly is used in computer animation.

Skeletal animation is a common technique in keyframe systems. With this technique, a stick-figure skeleton is created and used as a foundation for generating keyframes. The skeleton is made of many pieces that are connected at the joints to become an articulated figure. The animator can pose the articulated figure by rotating the joints. The joint angles are keyframed during the animation process. Since each piece in the articulated figure is rotated around the joint and it is the rotation angles that are interpolated, the pieces will not detach. The advantage of using skeletons for making keyframes is that the time for manipulating a skeleton

is faster than rendering a whole model because the skeleton is less complex. To build an animated model, the skeleton can be *skinned* by putting a tensor product surfaces on top of it after the keyframes are produced. Hence, the process for making keyframes is shortened. An additional modeling advantage is that the relationship between keyframes can be retained, because the next keyframe can be derived from the skeleton of the previous keyframe.

Alias Maya[1] and *Side Effects Houdini*[2] are two examples of keyframe animation systems.

[1]Maya is a trademark of Alias|Wavefront, a division of Silicon Graphics Limited

[2]Houdini is a trademark of Side Effects Software Inc.

Chapter 3

Animated Surface Pasting

3.1 Motivation And Goals

In the animation process, an animator spends a lot of time modeling. As mentioned in Section 2.4, surface pasting is a good method for modeling. Hence, we would like to merge the surface pasting technique into the animation process. However, it is unclear how pasted surfaces will behave at the inbetween frames in a keyframe system. Potentially, the pasted surfaces may distort or separate from the base surface, they may not resize in proportional to the size changes of the base surface, or they may look fine. Therefore, the goal of this thesis is to investigate the behavior of pasted surfaces and features at inbetween frames and decide whether surface pasting is a useful technique in animation.

For this thesis, I modeled a puppet with surface pasting and wrote a keyframe system to animate the puppet. The details of the puppet modeling, the keyframe system and the user interface will be described in the rest of this chapter.

Joint	*Degree of Freedom*	*Directions*
shoulder	2	X, Z
elbow	1	Z
wrist	2	X, Z
head	3	X, Y, Z
neck	2	X, Z

Table 3.1: Degree of Freedom on Puppet's Joints

Rotation Around	*Which Button Is Pressed*	*Which Direction*
X-axis	Left Button	X
Y-axis	Middle Button	Y
Z-axis	Middle Button	X

Table 3.2: Mouse Functions

3.2 Puppet Model

To start the research for this thesis, a skeletal keyframe was needed for implementing an animation. Hence, I needed to build a puppet. Since the purpose of this research is to investigate the behavior of pasted surfaces in animation, some feature surfaces were pasted on the skin of the puppet. I concentrated on the facial features of the puppet. Thus, the puppet does not have legs, but has an elongated body.

The behavior of pasted surfaces can be changed in two ways: changing the shape of a base surface and moving the pasted surface itself. Thus, I organized the testing into two parts. The first part checked the pasted surface when the base surface was deformed. Thus, I implemented a deformable head model. The second part examined the moving feature. I implemented this part by giving the puppet

moving fingers. Additionally, I built some lip models to make the puppet look better.

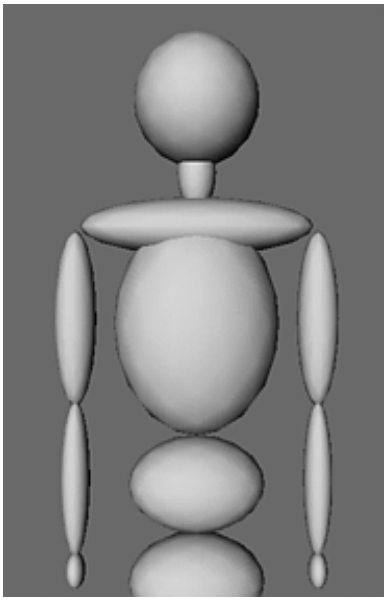
There are two versions of the puppet model, skeleton and skin-only. In the skeleton version, the puppet is made of ellipsoids for the bones as shown in Figure 3.1(a). The keyframe system works with the skeleton version only. Thus, a user can move any part of the puppet in the skeleton version.

The degrees of freedom on each part of the puppet are shown in Table 3.1. The user can pick any part of the puppet to move or rotate by pressing the left or middle button inside that part. The elbow of the puppet is positioned by the mouse position when the left button is pressed. The elbow can only be moved on the hemisphere around the shoulder. The remaining parts of the puppet can be controlled by the mouse movement as shown in Table 3.2.

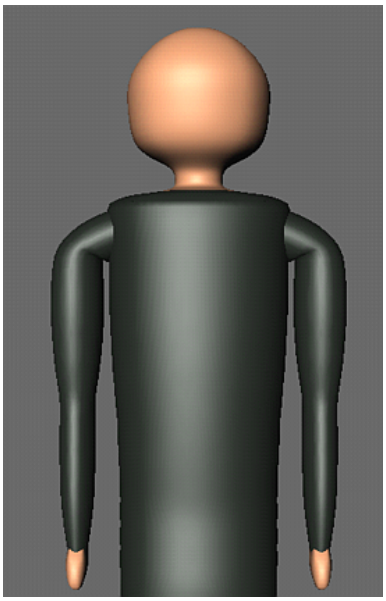
In the skin-only version, the puppet is built as a skin surface for the base surface as shown in Figure 3.1(b) and the features such as the eyes, ears, nose, lips and fingers are modeled and put on top of the skin surface using the surface pasting technique. Figure 3.1(c) shows the skin-only version puppet with the facial features and fingers. The details of skinning and pasted details are discussed in the following subsections.

3.2.1 Skinning

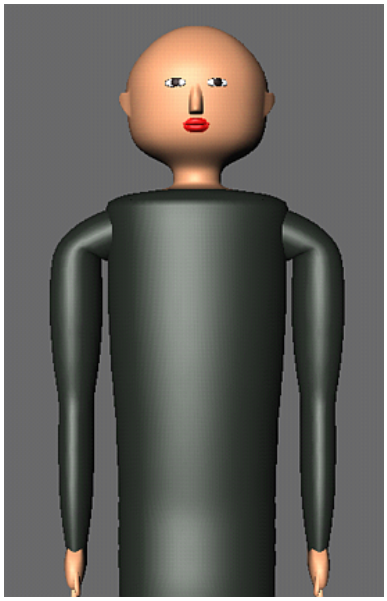
Skinning is a process in which skin surfaces are modeled and associated with a skeleton. With standard methods, the skin surfaces can be modeled out of primitives such as spheres and cylinders, and splines [Mae96]. For example, we can start modeling a character with a sphere. We reshape the sphere to become a torso, resize four unit cylinders to become arms and legs and manipulate the vertices near



(a) Skeleton Version



(b) Base Skin Only



(c) Skin-only Version

Figure 3.1: Puppet Model

the end of each cylinder to join the torso smoothly. Finally, we can add a head, hands and feet to the character by using a Boolean operation.

The second way for modeling skin is to use splines only. For example, we can model a character from two simple spheres that are made of splines. We use one sphere to model the upper body. We rescale the sphere to model the torso, pull some vertices at the poles horizontally away from the sphere to create the arms and reshape the arms to add some details. Then, we can pull the vertices at the top of the torso up to create a neck. After that, we use another sphere to model the lower body. We rescale the sphere along the horizontal axis to make it long, bend the ends of the long sphere down to make a horseshoe shape and then adjust the vertices to show the hips and knees. Finally, we position the legs to intersect the body at the waist and add a head, hands and feet. One advantage of this second method over the first method is that you have fewer pieces to join together with continuity.

For a skeletal keyframe system, the skin needs to be associated with the skeleton. There are two ways this can be done. First, the skeleton can be modeled and then the skin placed on top, or, second, the skin can be modeled and then the skeleton added inside. In both methods, the user associates parts of skin with parts of skeleton. Once this association is made, changes to the pose of the skeleton result in corresponding changes to the skin.

For my work, I skinned the puppet myself because my system is standalone. And I simplified the skinning process by putting many surface patches on the skeleton of the puppet because my research is concentrated on the behavior of pasted surfaces, not on skin modeling. Although the skinning was a side issue in my work, the process is quite involved, and I will give a detailed description of the skinning process to document what I did.

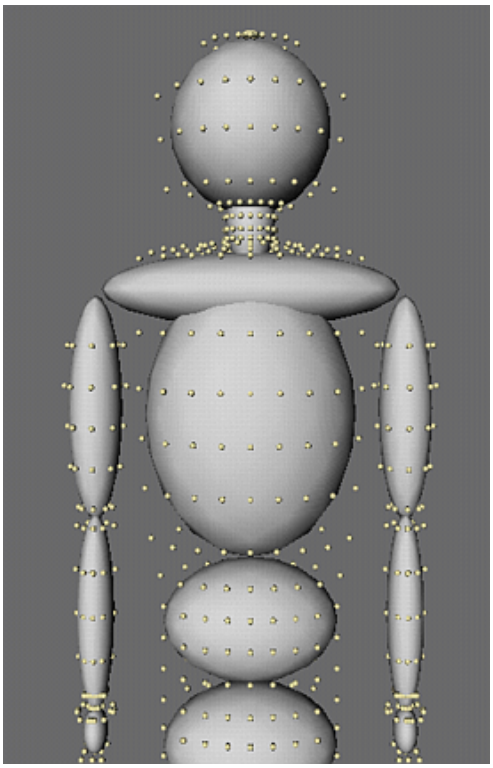
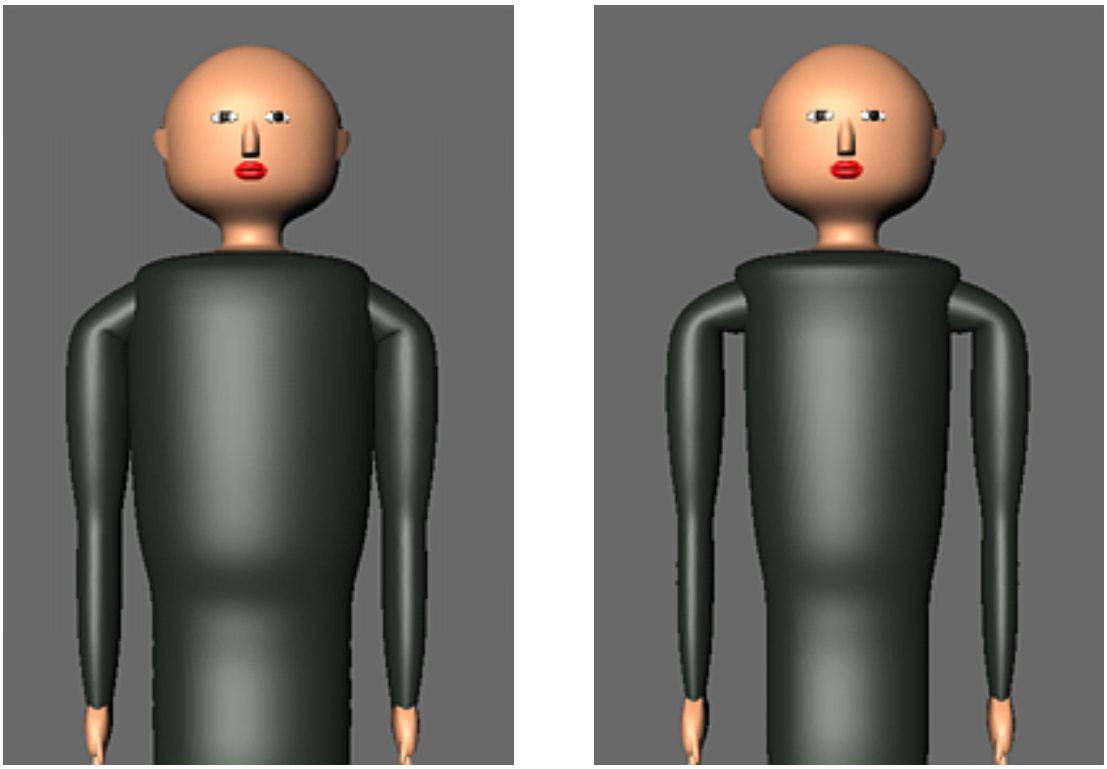


Figure 3.2: Skeleton Version of Puppet Model With Key Points

The skin surface of the puppet is formed by many separated surface patches. Each surface patch is offset from a piece of the skeleton. The control points of a surface patch are found by taking offsets from the key points on the ellipsoid of the skeleton. I manually set the positions of the key points on an ellipsoid by clicking on the ellipsoid and getting the position of that point from OpenInventor. For some body parts such as the body and arms, I offset the key points from the skeleton because I wanted to use one fixed offset value for each body part. Figure 3.2 shows the key points on the skeleton of the puppet. The size of the puppet depends on the values of the offset. If we use big offset values on the body of the puppet, the puppet will have a big tummy (as shown in Figure 3.3(a)). If we use small offset values, the puppet will become a skinny guy (as shown in Figure 3.3(b)). The



(a) Fat Puppet (b) Skinny Puppet

Figure 3.3: Puppets With Different Offset Values

final offset values were selected (as shown in Table 3.3) after experimenting with different offset values.

After the control points of all surface patches are found, the surface patches have to be joined with C^1 continuity. Some body parts are pieces of the same tensor product surface and meet C^2 automatically. I stitched separate tensor product surfaces with either C^0 or C^1 continuity. This stitching occurred between the shoulder and the body, between the shoulder and the arms, and between the forearms and the hands.

Surface	*Body Part*	*Offset Value*
Head	head	15
	neck	10
	part of shoulder	20
Body	body	22
	hip	37
Arm (Left/Right)	upperarm	13
	forearm	13
Hand (Left/Right)	hand	10

Table 3.3: A List of Offset Values

There are three surface patches for the body: one for the torso and two for the hips. Both the torso and hip have joints, so the body can bend and/or turn around. These three surface patches are modeled as one bi-cubic B-spline surface, so they meet with C^2 continuity automatically. For each arm, there are three surface patches: one for an upperarm, one for a forearm and one for a hand. The surface patches of the upperarm and the forearm are modeled as one bi-cubic B-spline surface. For the surface of the hand, we cannot model it as part of the arm surface because the surface of the arm is built with both ends open, but the surface of the hand is closed on one end. I joined the hand into one end of the forearm to meet with C^0 continuity. This joint looks fine around the boundary, so I did not do anything further for connecting the arm and the hand. However, I moved some control points on the hand to make the hand look more realistic.

I modeled the head part that included the head, the neck and the upper part of the shoulder with one bi-cubic B-spline surface. To get different colors for the head and shoulder, this single tensor product surface was rendered as two trimmed

surfaces.

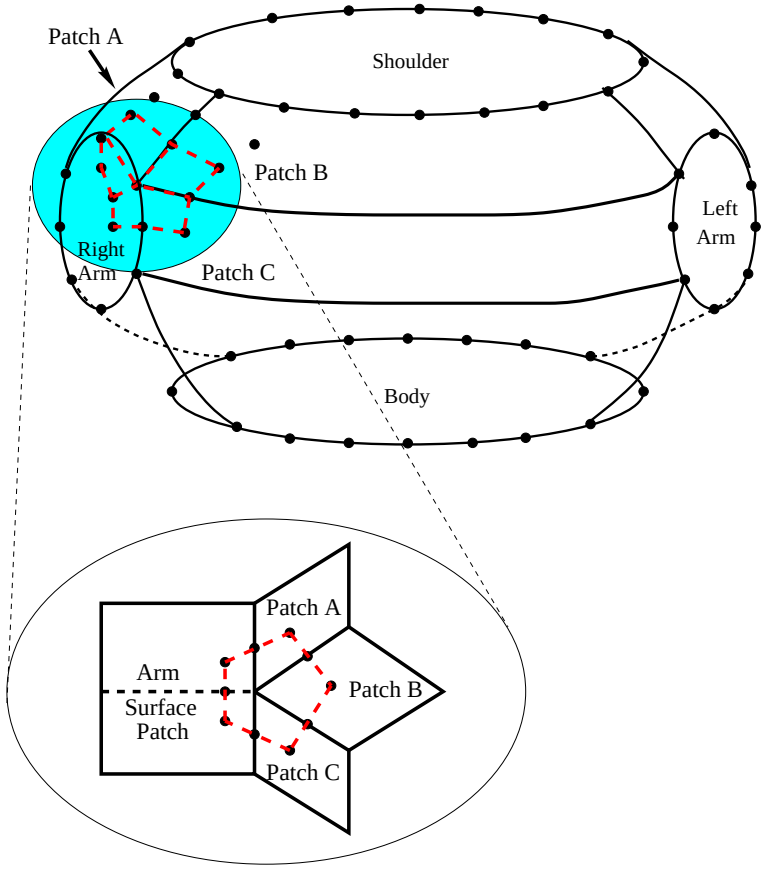


Figure 3.4: Smooth Shoulder Problem

The last and most difficult part was modeling the shoulders. We want to model the shoulder to join the body, neck and the arms smoothly. However, it is hard to model the shoulder to join every part with C^1 continuity. Figure 3.4 shows one possible layout of surface patches for forming the shoulder. Here there are eight tensor product surface patches. From the lower half of Figure 3.4, it is clear that at the shoulder/arm joint five patches meet at one point. To join all five patches with C^1 continuity, each dashed line that joins three control points has to be colinear. It is not easy to find a solution for this requirement. The main concern of my research

is not modeling, so I didn't build the shoulder to connect everything smoothly. In my program, I modeled the shoulders as a single tensor product surface. This shoulder meets the body and the neck with C^1 continuity, but joins connectors to the arms with just C^0 continuity.

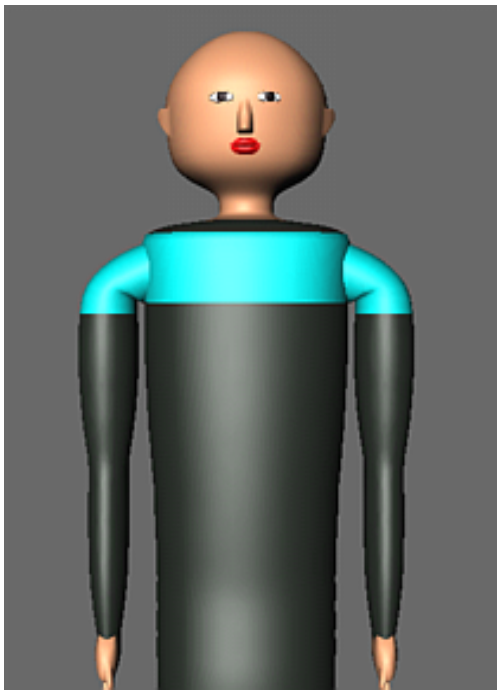


Figure 3.5: Puppet Model With Highlighted Shoulders

There are three separate surface patches for the shoulder (the highlighted surfaces shown in Figure 3.5): one for connecting the head and the body, and the other two for connecting the arms. Figure 3.6 shows the shoulder connector that connects one of the arms to the body. We can build the connectors to meet with the arms with C^1 continuity, but the connectors penetrate the body/shoulder and appear to meet with C^0 continuity. Since the cross section of the arm is a circle, the connector can be modeled with circular ends. Hence, the shape of the connector is like a bent cylinder. The control points of the first row on the connector are

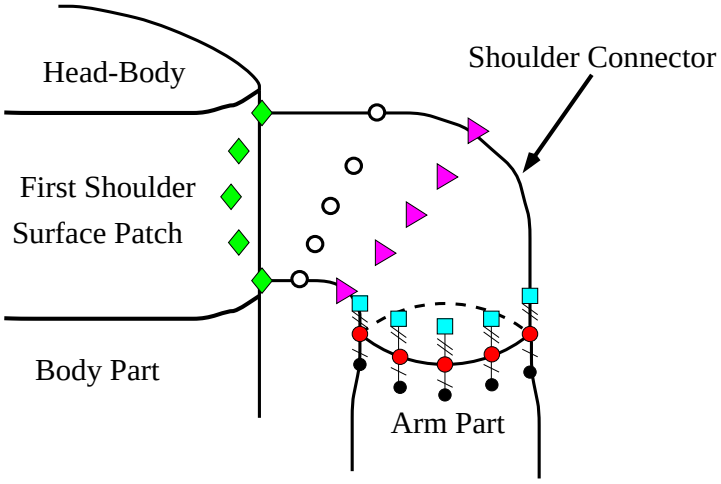


Figure 3.6: Shoulder Connector To Arm

equal to the control points of the first row on the upperarm (the gray dots shown in Figure 3.6). The control points of the last row, which connect to the body, are calculated by rotating and translating the control points of the first row to touch the body in C^0 continuity (the diamonds shown in Figure 3.6). The control points of the third row are found in a similar way, using half the rotation angle and half the translation distance used for the diamonds (the triangles shown in Figure 3.6). To meet with C^1 continuity, the ratio of the length of the last segment on one patch and the length of the first segment on the other patch must be the same along the whole boundary and these two segments must be colinear. Hence we define the control points of the second row based on the first segment of the body surface patch to make the shoulder connector connect to the arm with C^1 continuity (the squares shown in Figure 3.6).

The remaining patches connect the head/shoulder to the body. Figure 3.7 shows the first surface patch of the shoulders. The control points of the surface patch that

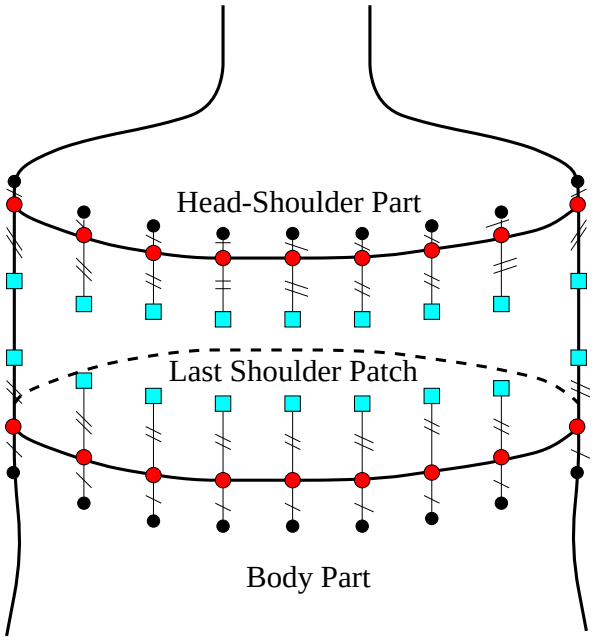
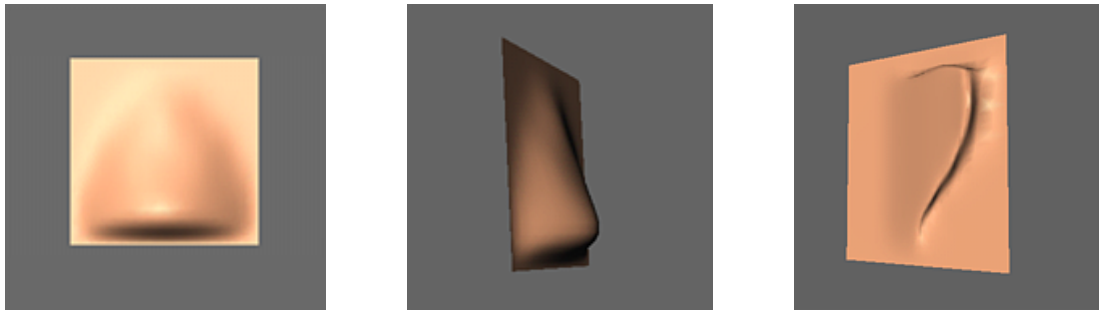


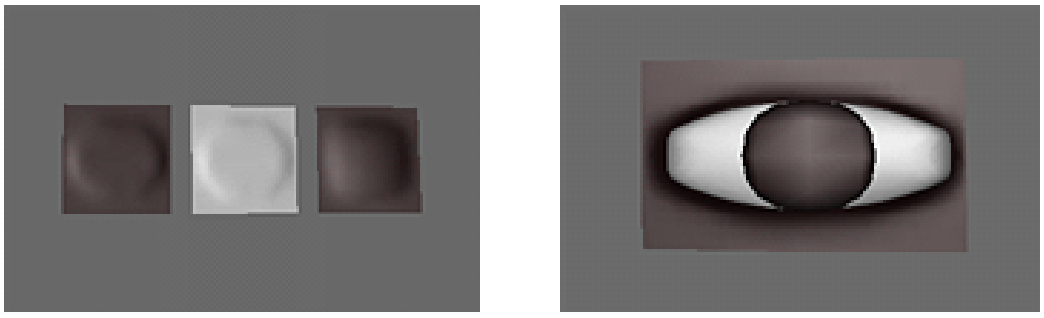
Figure 3.7: First Surface Patches of Shoulder

connects the head and the body are based on the control points of the head surface patch and the body surface patch. First, we want the shoulder to connect the head and the body with C^0 continuity (i.e., they meet in the same places), so the control points of the first row of the shoulder are the same as the control points of the last row of the head and the control points of the last row of the shoulder are the same as the control points of the first row of the body (the gray dots shown in Figure 3.7). To have the head and the body meet with C^1 continuity, we set the control points of the second and third rows on the shoulder based on the last segment of the head surface patch and the first segment of the body surface patch, respectively, as described above (the squares shown in Figure 3.7).



(a) Nose (Front View) (b) Nose (Side View) (c) Ear

Figure 3.8: Unpasted Nose And Ear Model

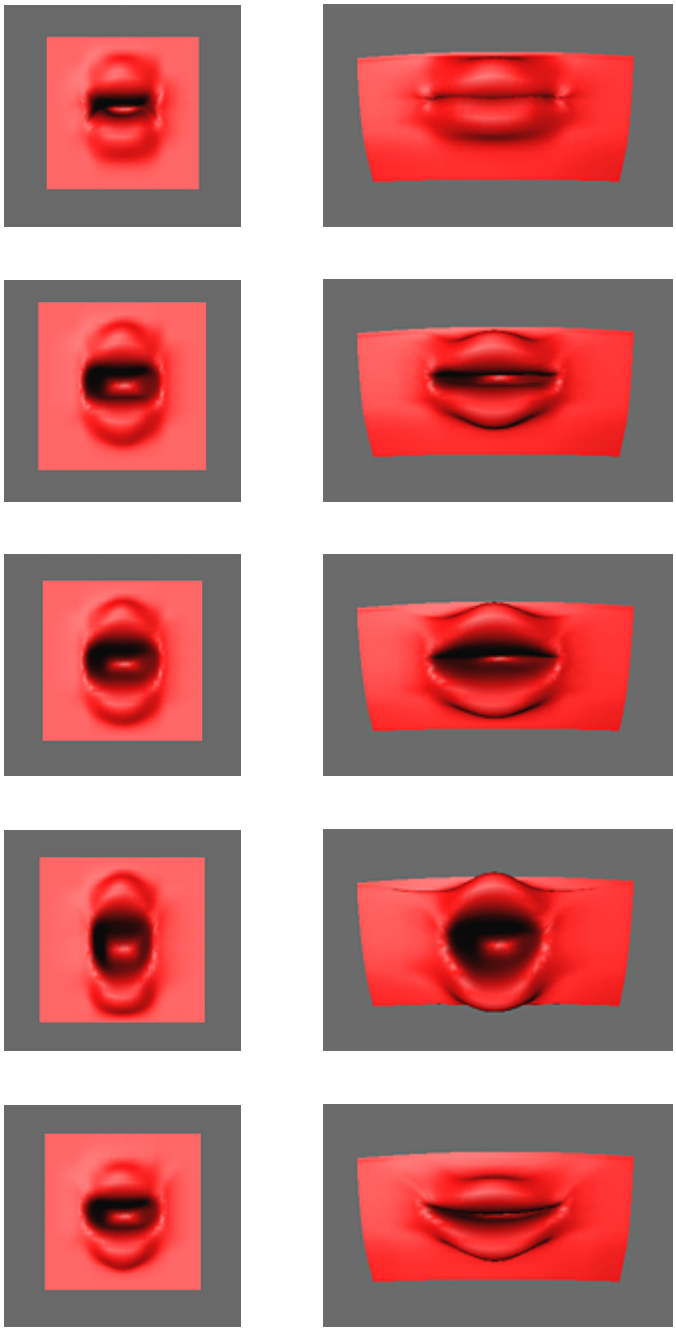


(a) Without Pasting (b) After Pasting

Figure 3.9: Eye Model

3.2.2 Pasted Details

The eyes, ears, nose and lips are pasted onto the skin of the head. Figure 3.8 shows the unpasted ear and nose surfaces. Each eye is formed of three surfaces using hierarchical pasting. The first layer is a black surface, which creates a black ellipse around the eye. The second layer is the white surface, which models the white of the eye. The third layer is another black surface, which models the pupil. Figure 3.9(a) shows the three layers of the eye without pasting and Figure 3.9(b)



(a) Without Pasting (b) After Pasting

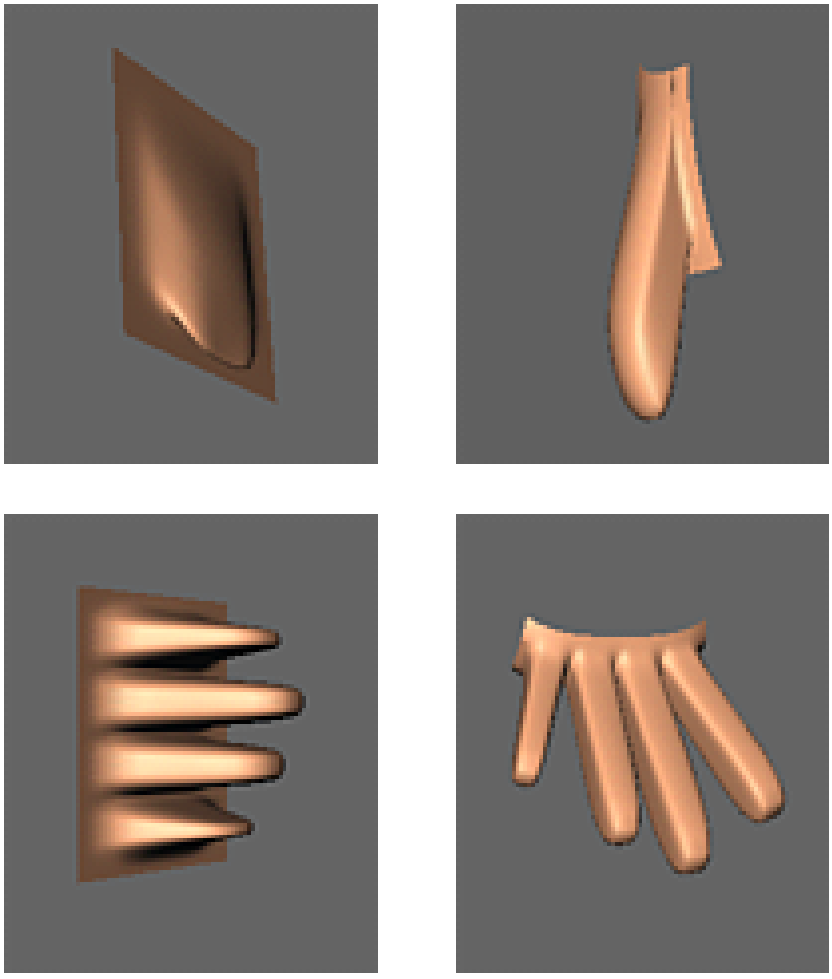
Figure 3.10: Five Lip Models

shows the result of the eye after pasting. The lips are formed by one surface. There are five different lip models, which are used to animate the lips for talking. In Figure 3.10, the left column shows the five lip models without pasting and the right column shows the lips after pasting. To show the inner part of the mouth, part of the head surface (under the lips) has been trimmed. The trimming control points on the head surface are defined based on the model of the lips.

On each hand, the fingers are modeled as two separate surfaces, one for the thumb and the other for the rest of the fingers. In Figure 3.11, the left column shows the thumb and the rest of the fingers without pasting and the right column shows the fingers after pasting. Note that the pasted thumb is much taller than the unpasted thumb because the control points were scaled down in x and y but not z.

The fingers can perform a grasping motion by moving the domain of the finger surface (see Section 3.4.2 for details). Note that this is a poor way to animate such a motion. Normally, you would model the fingers with a skeleton and keyframe their joint angles. I modeled this motion by sliding the domain only because I wanted to test moving features in an animation system.

All the features have been pushed below the base surface a little bit to hide the margin (normally flat in shape) of the pasted surfaces. The pasted features are hardcoded inside the keyframe system application. The user can only manipulate the features through the user interface. For example, the user can move the fingers through the sliding bar of Finger Domain and change the lips model by clicking the corresponding lip model button in the *Parameter User Interface* Dialog Box (see Section 3.4.2). If the user wants to do something else, such as adding or modifying features, he/she has to modify the code directly. I did not add this additional functionality because my goal was to test pasted surfaces and not to make a full-feature animation modeller.



(a) Without Pasting (b) After Pasting

Figure 3.11: Finger Models

3.3 Keyframe System

In a keyframe system, the user enters keyframes and then the computer computes inbetween frames. My keyframe user interface is discussed in the next section. Here, I will discuss my system for computing inbetween frames.

I used Catmull-Rom Splines to compute inbetween frames. Initially, I calculated each inbetween frame using four keyframes: the previous keyframe, the current keyframe and the next two keyframes. However, there were several cases for which Catmull-Rom Splines gave poor results as discussed below.

3.3.1 Catmull-Rom Splines

Given n points $P_0, P_1, \cdots, P_{n-1}$, we want to find a piecewise cubic C^1 curve that interpolates all the data points. Further, we would like a method where changes to one data point only affect the curve locally. I chose Catmull-Rom Splines for my interpolation technique because they have those properties. For Catmull-Rom Splines, the curve on each segment $P_i\widehat{P_{i+1}}$ is determined by four points P_{i-1}, P_i, P_{i+1} and P_{i+2}. The tangents to the curve at P_i and P_{i+1} are $P_{i+1} - P_{i-1}$ and $P_{i+2} - P_i$ respectively.

If we use the Hermite basis functions, P_i, P_{i+1}, $P_{i+1} - P_{i-1}$ and $P_{i+2} - P_i$ are the coefficients. However, I chose to use the monomial basis because it is less expensive to evaluate. (Using monomial basis, there is one fewer multiplication per inbetween frame than using Hermit basis.) In the monomial basis, we can define a cubic curve segment as

$$P_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \,.$$

We want $P_i()$ to pass through P_i and $P_i + 1$, with the respective slopes P_i' and P_{i+1}'.

For Catmull-Rom Splines, we can find the values of the constants a_i, b_i, c_i, d_i by the following system of linear equations:

$$\begin{aligned} P_i(0) &= d_i & &= P_i \\ P_i(1) &= a_i + b_i + c_i + d_i & &= P_{i+1} \\ P_i'(0) &= c_i & &= P_{i+1} - P_{i-1} \\ P_i'(1) &= 3a_i + 2b_i + c_i & &= P_{i+2} - P_i \end{aligned}$$

3.3.2 Special Cases

After the interpolation method was tested with different types of keyframes, I found some cases in which the original Catmull-Rom Splines method would give poor animation results. The rest of this section will show the special cases I found during testing and the solutions I used for each case.

Case 1: Two Keyframes Only

In this case, there are only two keyframes. Although most animations will have more than two keyframes, I handled this case for completeness. I calculated the inbetween frames by linearly interpolating between the two keyframes.

Case 2: The First Two Keyframes P_0P_1 or The Last Two Keyframes $P_{last-1}P_{last}$

Since segment P_0P_1 is defined by the first two keyframes, there is no P_{-1}. Hence, we cannot use the equation for $P_i'(0) = P_{i+1} - P_{i-1}$. There are two different methods for solving this problem.

Method 1	*Method* 2
$P_i'(0) = \vec{0}$	$P_i'(0) = P_{i+1} - P_i$

Similarly, for the last two keyframe, there are two different solutions for $P_i'(1)$:

Method 1	*Method* 2
$P_i'(1) = \vec{0}$	$P_i'(1) = P_{i+1} - P_i$

In my system, the user can choose either method by selecting the method number from the *Interpolation Method* menu.

Case 3: Same Current Keyframes $P_i = P_{i+1}$

When the current two keyframes are the same, we want the inbetween frames to be the same. Thus, the tangent of the curve at P_i and P_{i+1} must be both zero. Hence, the equations for both $P_i'(0)$ and $P_i'(1)$ have to be changed to $P_i'(0) = \vec{0}$ and $P_i'(1) = \vec{0}$.

Case 4: Same Previous Keyframes $P_{i-1} = P_i$ or Same Next Keyframes $P_{i+1} = P_{i+2}$

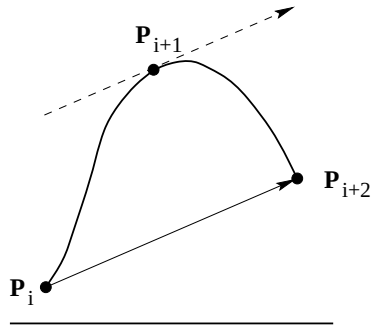
Because of Case 3, we need to set $P_i'(0) = \vec{0}$ to make the spline curve C^1 continuous. Similarly, for $P_i'(1) = \vec{0}$.

Case 5: Next Keyframe P_{i+1} or Current Keyframe P_i is Maximum/Minimum

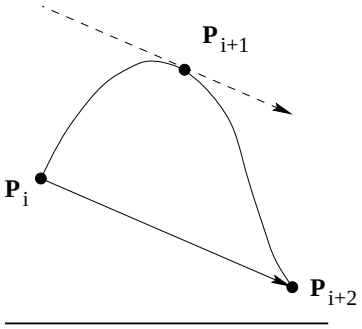
To make things easier for the animator, when the animator sets a local maximum/minimum in the keyframes, we should make that keyframe be a local maximum/minimum in our frame sequence. Unfortunately, Catmull-Rom Splines do not have this property, as shown in Figure 3.12.

Hence, I modified the Catmull-Rom technique when there is a local maximum/minimum by setting the tangent of the curve at P_{i+1} (or, P_i) to zero, so no inbetween frame will be above the maximum point or below the minimum point as shown in Figure 3.13. I.e., $P_i'(1) = \vec{0}$ (or, $P_i'(0) = \vec{(0)}$).

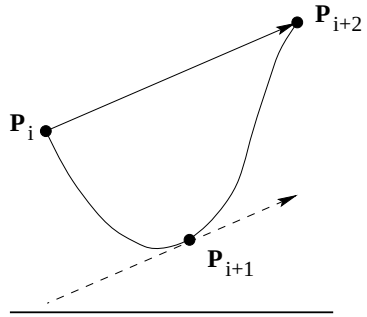
There are still some data where the curve will exceed a maximum/minimum



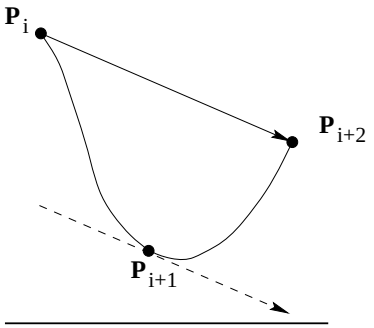
Case 1a: Maximum Point



Case 1b: Maximum Point

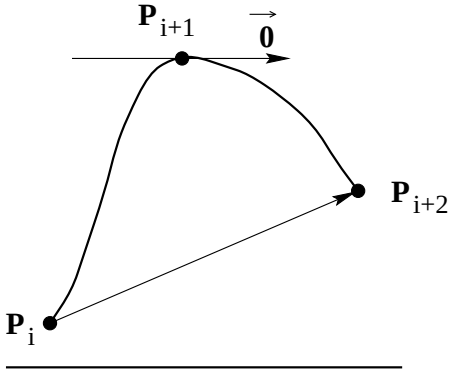


Case 2a: Minimum Point

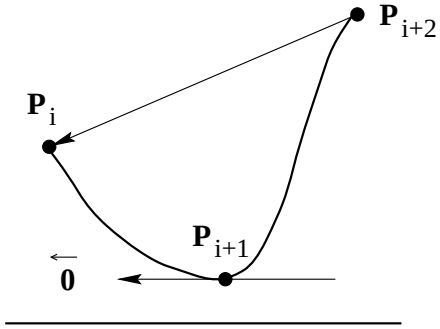


Case 2b: Minimum Point

Figure 3.12: Catmull-Rom Spline



Case 1: Maximum Point



Case 2: Minimum Point

Figure 3.13: Special Case 5 Solution

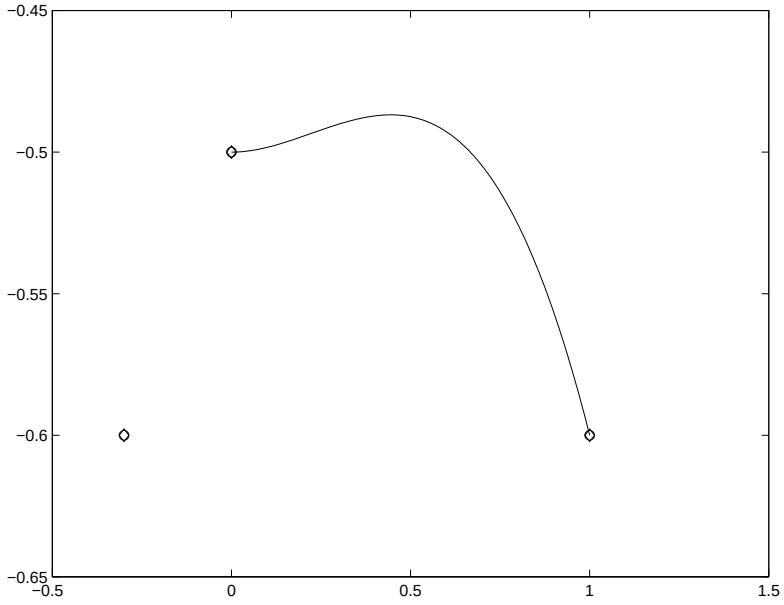


Figure 3.14: Special Case 5 Counter-Example

keyframe value. For example, when the absolute value of the derivative at P_{i+2} (or P_{i+1}) is very big. In this case, the inbetween frames will still be above the maximum point or below the minimum point even though we set $P_i'(1) = \vec{0}$ (or, $P_i'(0) = \vec{(}0)$). Figure 3.14 shows an example of this problem. In this figure, the data points are illustrated by the circles (the fourth point is below the bottom of the figure at $(1.3, -1.08)$). $P_i'(0) = 0$ and $P_i'(1) = -0.5$. The Catmull-Rom Spline curve is $f(x) = -0.3x^3 + 0.2x^2 - 0.5$. Clearly, some inbetween points near the beginning of the curve are greater than the local maximum point (starting point).

There are two ways to fix this problem. First, the animator could add more keyframes to prevent the absolute value of the derivative from being very large. Hence, this problem will disappear. Second, we could modify the Catmull-Rom Spline method to solve this problem by adjusting the derivative value if it is very big. The main concern of my research is to test how pasted surfaces behave and

not to build a perfect animation system. Therefore, I used the first method in my program. However, in a real system, the interpolation method would have to be modified to fix this problem.

3.4 User Interface Details

The user interface of my system is built on top of OpenInventor. OpenInventor provides an *examiner viewer* that allows the user to view an object from different positions by moving the camera [Wer94]. The user can rotate the whole object with a track ball or translate it in three dimensions. On top of this, I added an interface for my keyframe system. The details are discussed below.

3.4.1 Keyframe System User Interface

The main task in the animation process is to define the keyframes. Figure 3.15 shows the user interface of my keyframe system. Initially, the user is provided with the first keyframe in which the puppet is in the default position and the default size. The user can define new keyframes by inserting them anywhere from one frame before the first keyframe to one frame after the last existing keyframe, copying them from the current selected keyframe to the next of the current keyframe, saving the current selected keyframe with the new information, or deleting the current selected keyframe. The current selected keyframe number is indicated for the user's convenience.

After defining each keyframe, the user needs to specify a frame number for the keyframe. On the right side of the main window, there is a list of frames for the user to define the relationship between the frames and the keyframes. The user can

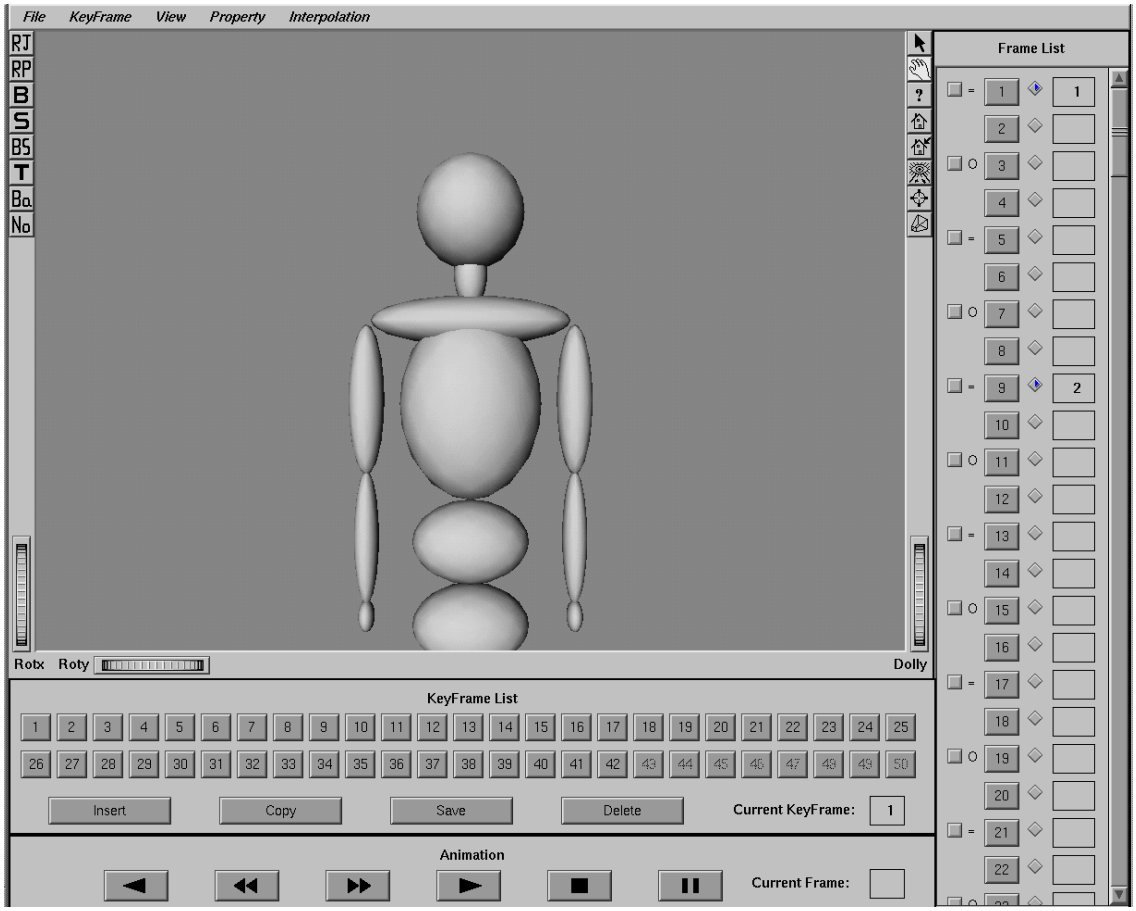


Figure 3.15: User Interface of Keyframe System

click on the radio buttons on the right side of the frame number buttons to select that frame to associate with one keyframe. The keyframe number associated with a frame is assigned in increasing order automatically. Moreover, the user can toggle the talking motion with the existing keyframe animation by selecting the square toggle buttons on the left side of the frame number buttons.

After the user defines the whole keyframe animation, the user may want to see the whole animation or a particular inbetween frame. The user can view the animation with the same functions as a normal VCR (play, rewind, fast forward,

step forward, step backward, stop and pause). And then, the user can view one particular inbetween frame by pressing the corresponding frame number button. When the animation is playing, the current frame is shown beside the VCR function buttons, so the user knows which frame is showing.

3.4.2 Parameter User Interface

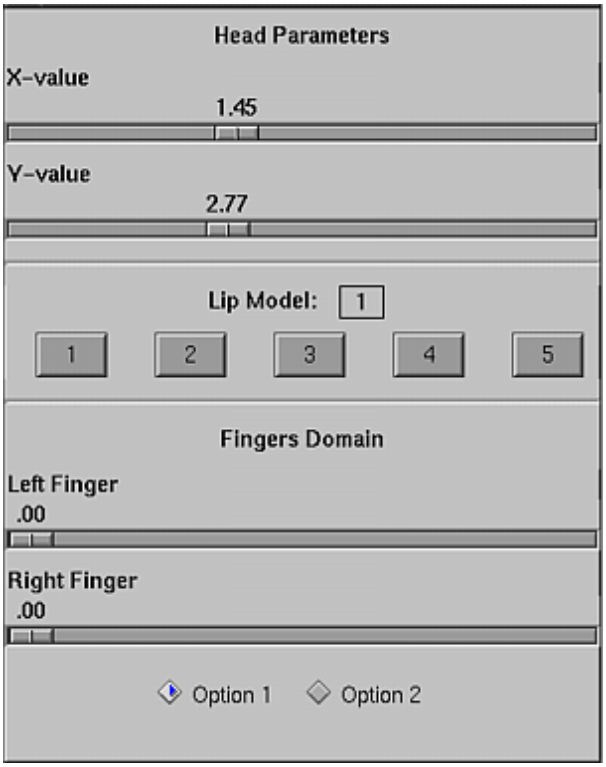


Figure 3.16: Parameter User Interface Dialog Box

To test how pasted surfaces behave at inbetween frames, the head of the puppet can be deformed in various ways. The user can change the size of the head in the *Parameter User Interface* dialog box as shown in Figure 3.16. In the Parameter User Interface dialog box, the first option specifies the head parameters that are

used to control the width and the height of the head. The user can change the width and the height of the head by moving the sliding bar sideways.

The second option of the Parameter User Interface is the lip model. The user can pick a particular lip model by pressing the appropriate number button. The last two options in the dialog box are related to the pasted fingers. The user can specify where the fingers are pasted on the hand by moving the sliding bar and define how the fingers are sliding on the hand to perform a grasping motion by pressing one of the radio buttons.

Chapter 4

Results

4.1 Description of Animation

The idea of this animation is based on The Scream by Edvard Munch. The animation starts with a man standing on a bridge. The man seems to be looking for something. Suddenly, there is a very bright flashing light. The man gets scared and does not know what to do. He just holds his head with his hands. At this time, some strange things happen. His head gets deformed because he squeezes his head too hard and his head does not return to its normal shape after he removes his hands. He attempts to shake his head to make it normal. However, it does not work. So, he pulls both ears to stretch his head outside. He releases his hands after he cannot stretch his head anymore. The deformation is gone, but the head is shortened because the man stretched his head too much. Then, he holds his chin with one hand and pulls the top of his head with another hand to elongate the head. Afterwards, the man's head goes back to normal shape and size. The man is glad that no more exotic things happened – everything becomes normal.

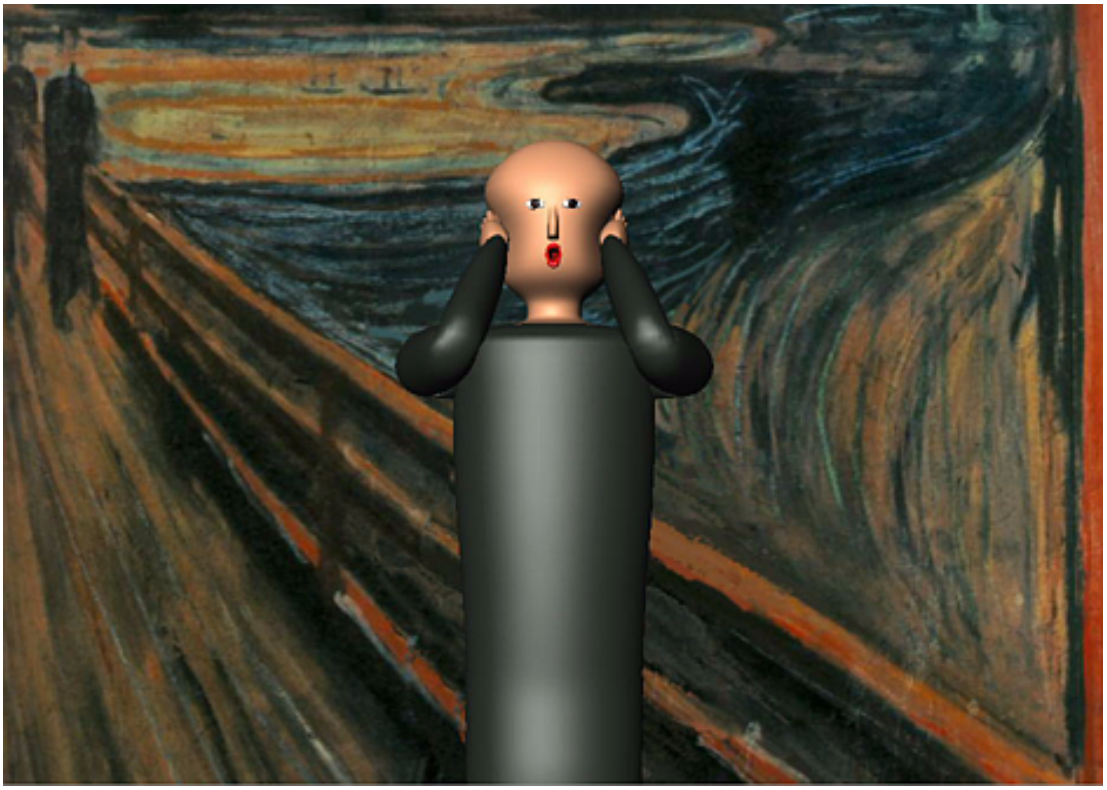


Figure 4.1: The Scream Frame

For this animation, I built forty-two keyframes and generated one hundred and ninety-one inbetween frames. The total run-time of the animation is approximately thirteen seconds. The background of the animation was painted by Mark Riddell based on The Scream. Figure 4.1 shows the scream frame of this animation. This animation is available on the Web at http://www.cgl.uwaterloo.ca/Projects/Splines/pasting/animate98/index.html and on CD-Rom. In the following sections, the discussion is based on inbetween frames extracted from the animation.

4.2 How Pasted Surfaces Behave

The main task of this research was to test how pasted surfaces behave in animation. Thus, we have to look at the facial features, such as eyes, nose, ears, lips, and fingers to see how they behave when the head is deformed and when the fingers move on the hands.

For the most part, the facial features behave properly when the head is deforming. Figure 4.2(a) shows the facial features in the original size and the head without any deformation. When the head is squished, the eyes, ears, nose and lips are all narrowed down correctly as shown in Figure 4.2(b). Next, when the head is stretched sideways, the facial features are elongated in width and are shortened in height appropriately, as shown in Figure 4.2(c). Finally, the heights of the facial features and the head are lengthened at the same time, as in Figure 4.2(d).

Although the facial features behave reasonably on head deformation, there are some details that the animator has to take special care with when creating an animation. Figures 4.3(a) and 4.3(b) show the puppet with the normal head and wide-opened mouth. However, there is a problem with the lip after the head of the puppet has been stretched sideways. Figures 4.3(c) and 4.3(d) show that part of the mouth now sticks into the nose. The reason is that the height of the head has been reduced. Every pasted surface has been compressed along the y-axis. Thus, the nose and the mouth get closer and closer and until they stick together. The animator has to watch for such problems when producing the animation and possibly modify inbetween frames to avoid penetration of surfaces.

Normally, the pupil of the eye should be inside the white of the eye as shown in Figures 4.4(a) and 4.4(c). However, in Figures 4.4(b) and 4.4(d), the pupil of the eye is bigger than the white of the eye. The reason is that the shape of the three

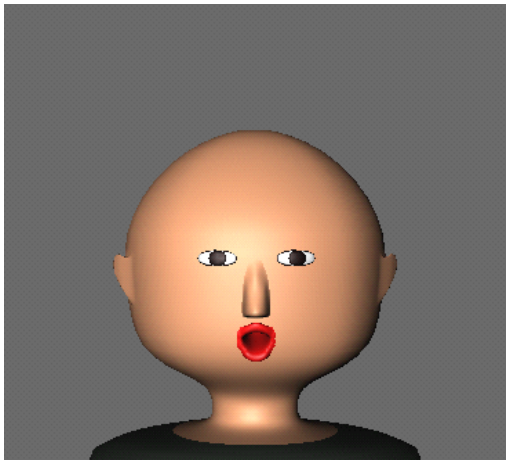
eye surfaces are bump-like and so the displacement vectors of the second boundary layers are pointing outside. When the head is stretched sideways, the height of the eye surface is shortened. Then, the displacement vectors point outside more. Since the three eye surfaces are of similar shape and pasted hierarchically, the effect of the displacement vector pointing outside cumulative. Thus, the last pasted surface, the pupil, lies outside the white of the eye. Figure 4.4(e) illustrates this problem. On the top figure, it shows the white of the eye and the displacement vectors of the pupil on the unstretched head. It is clear that the pupil is placed within the white of the eye. However, the bottom figure shows that the pupil now extends outside the white of the stretched eye.

There is also a distortion of the fingers in a grasping motion. The grasping motion is implemented by sliding the fingers on the hand. Unfortunately, the fingers enlarge when the fingers are slid to one side of the hand as shown in Figure 4.5(b) (Figure 4.5(a) shows the initial pose of the hand and the fingers).

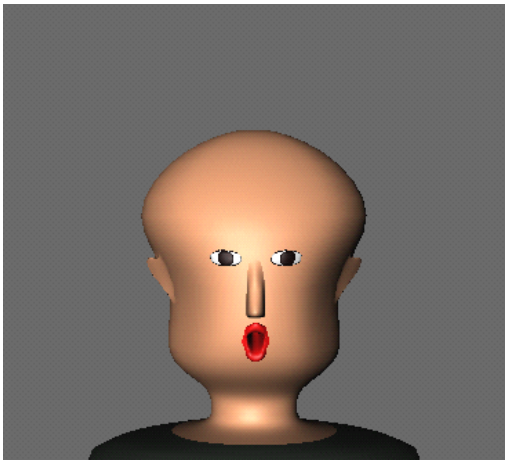
The fingers get fatter because of the shape of the base surface – the hand. The width of the feature domain relative to the base domain was maintained in this sequence. However, the hand has a non-uniform parameterization (i.e., equally spaced points in the domain mapped to non-equally spaced points in the range) and a fixed-sized finger domain as we slide the fingers over the hand. Thus, the fingers will span different areas on the hand. As a result, the boundaries of the fingers move further apart and the fingers thicken.

The reason why the fingers become larger due to the lengthening of the width of the base surface covered by the feature surface. I corrected this problem by shortening the width of the fingers domain. After experimenting, I found a constant factor that could be combined with the sliding offset to form a compression factor. Then, the compression factor is used to rescale the fingers domain when the fingers

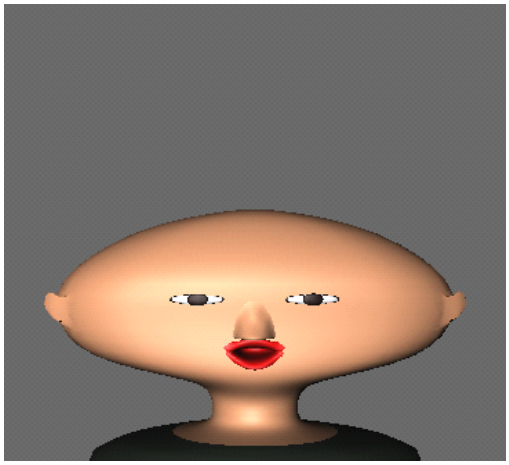
are sliding sideways. Now, the fingers domain is shortened as well as the finger surfaces. This method worked in my testing (see Figure 4.6). However, in a real system, the software may need to do more calculations to solve this problem and maintain the actual width of the fingers no matter where the fingers are on the hand.



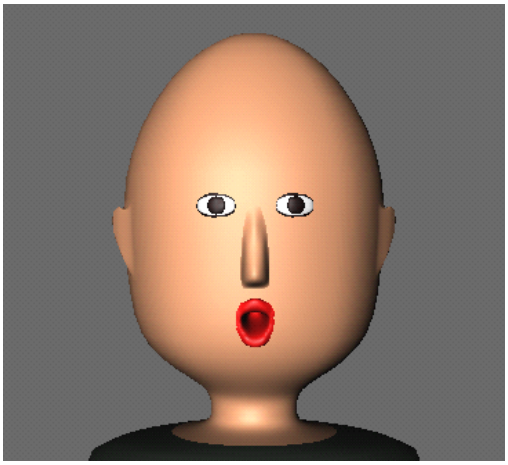
(a) Initial Pose



(b) Squishing Pose

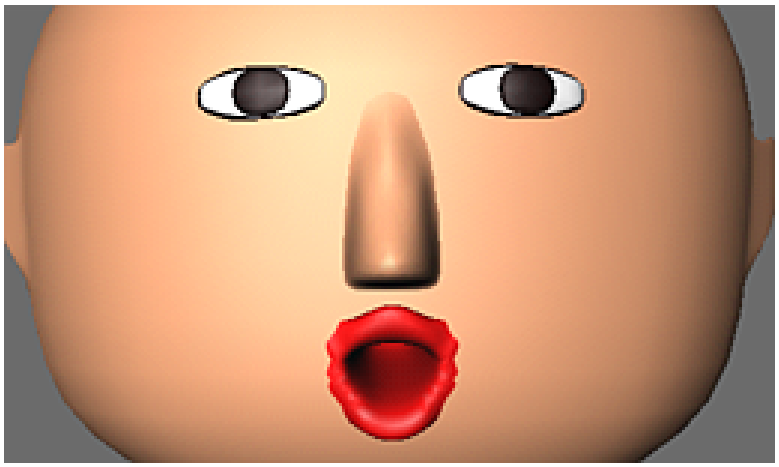


(c) Stretching Aside Pose



(d) Stretching Up Pose

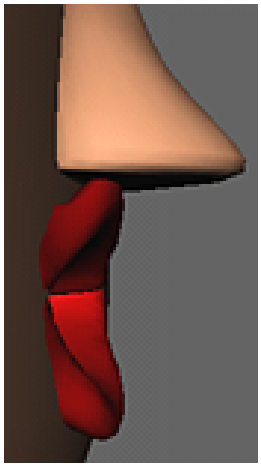
Figure 4.2: Different Poses of Head and Facial Features



(a) Normal Head Front View

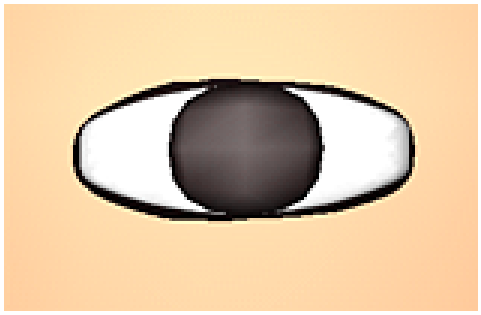
(b) Normal Head Side View

(c) Stretching Head Front View

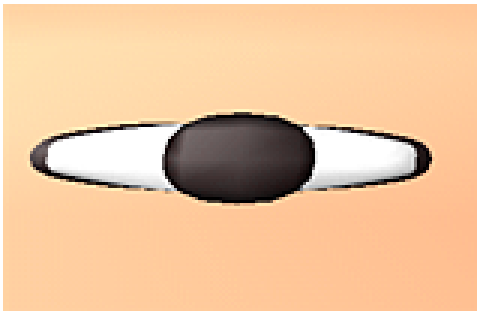


(d) Stretching Head Side View

Figure 4.3: Modeling Problem

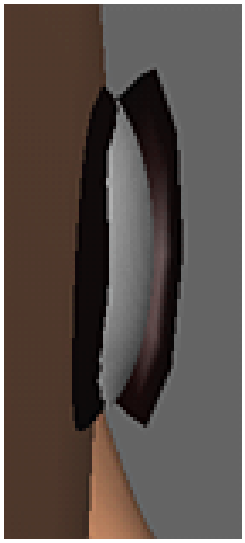


(a) Normal Head Front View

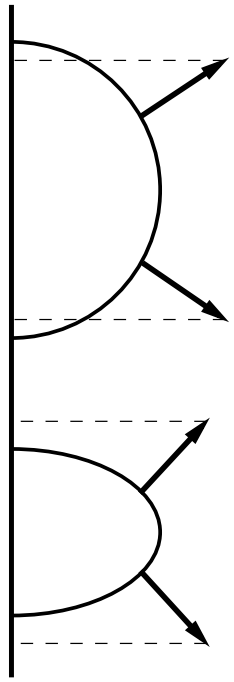


(b) Stretching Head Front View

(c) Normal Head Side View

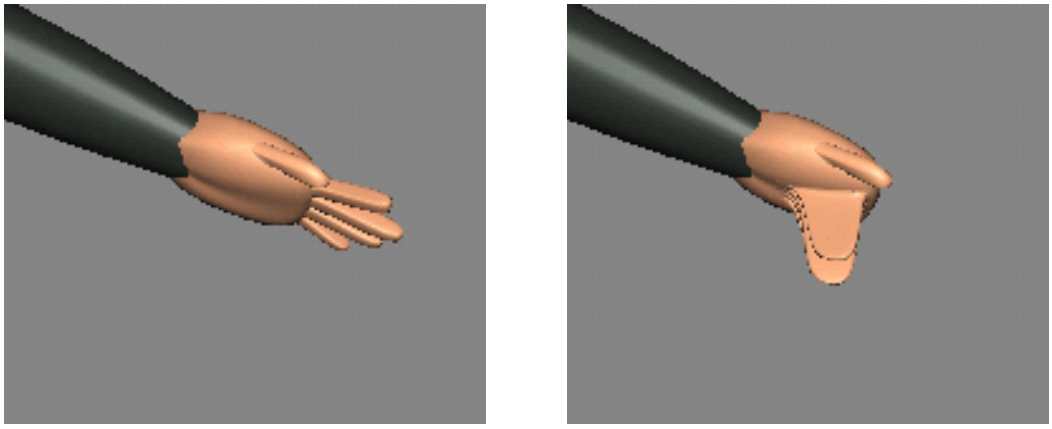


(d) Stretching Head Side View



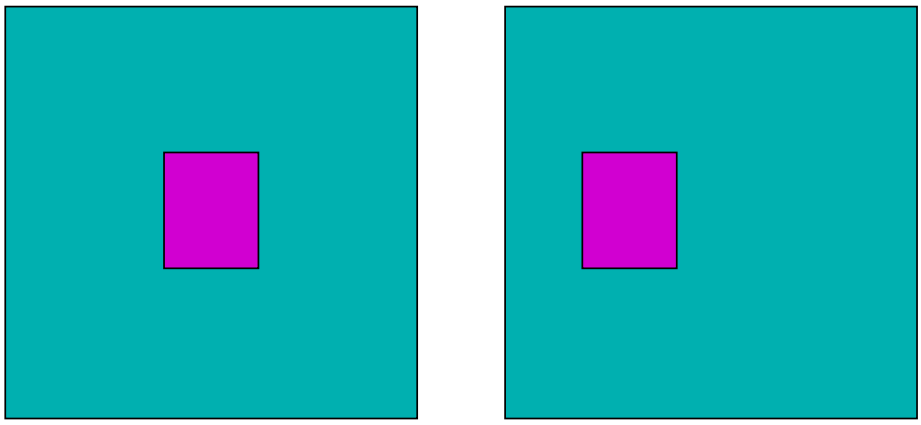
(e) Problem Illustration

Figure 4.4: Bigger Pupil of Eye Problem



(a) Initial Pose

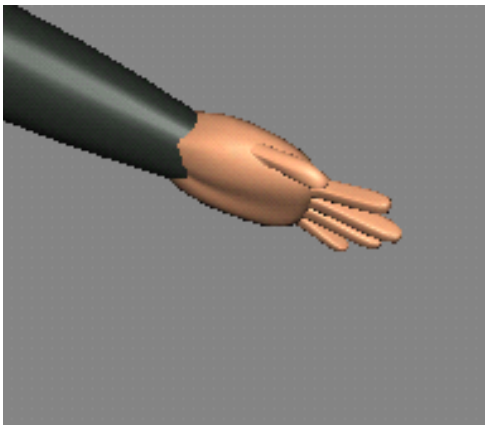
(b) Grasping Pose



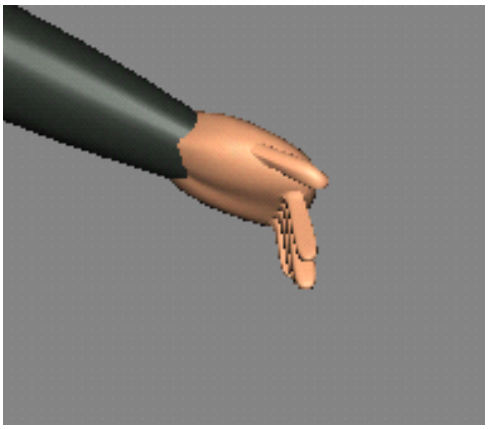
(c) Initial Pose Domain

(d) Grasping Pose Domain

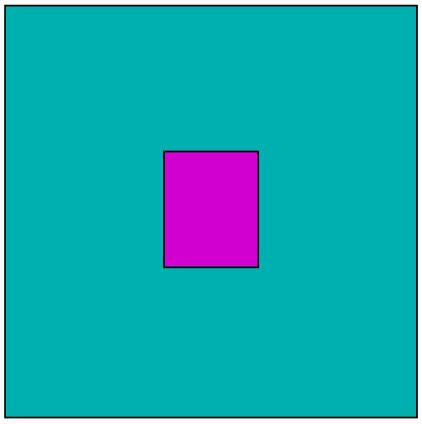
Figure 4.5: Fingers Distortion Problem



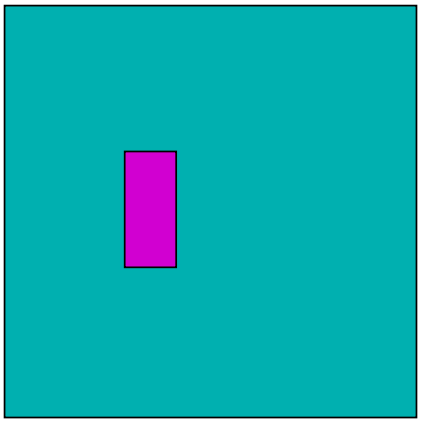
(a) Initial Pose



(b) Grasping Pose



(c) Initial Pose Domain



(d) Grasping Pose Domain

Figure 4.6: New Grasping Motion

Chapter 5

Summary and Conclusions

Surface pasting is a composition method that adds the detailed features on top of a base surface. Surface pasting is a good technique for modeling. However, it had never been used in animation. Therefore, I built a puppet for testing the behavior of pasted surfaces. The skeleton version of the puppet was used to build a set of keyframes. I then added a skin to the puppet and generated the inbetween frames. To test the behavior of pasted surfaces, the skin of the puppet was built as a base surface and the facial features such as eyes, ears, nose, lip and fingers were pasted on the base surface. The pasted surfaces were tested in two ways: deforming the base surface and moving the pasted surfaces.

While most of the implementation was straightforward, there were a few problems. The biggest problem I encountered was in constructing the skin of the puppet. I had hoped to make a C^1 skin surface, which seemed reasonable to expect since the C^1 conditions between two patches are quite simple. However, to connect the arm to the body created a point on the surface where five patches met. At such a point, the simple C^1 conditions no longer apply and something more complex must

be used. Since skinning was not the focus of my research, I decided not to address this problem and used a C^0 join instead.

The second problem I encountered was with Catmull-Rom Splines. I used Catmull-Rom Splines to generate the inbetween frames because they would give smoother motion and as a result require fewer keyframes. However, in a key framing system, there are additional requirements of which I was unaware until I began testing my system. In particular, if the animator puts a local maximum/minimum in a keyframe, then the inbetweener should preserve this maximum/minimum. Unfortunately, Catmull-Rom Splines do not preserve such maxima/minima and so I had to modify the Catmull-Rom method to meet this requirement.

5.1 Animated Surface Pasting

In my animation, I found that, in general, the facial pasted features behave as we expected. That means pasted surfaces behave properly under deformation of a base surface. As the base stretches and deforms, the features stretch and deform in a similar fashion without separating from the base. However, there were some problems with the features. First, at a few spots in my animation, some of the features that did not touch on the undeformed head would interpenetrate in the deformed head. This is a problem with many animation techniques and it highlights the importance of allowing the animator access to the inbetween frames to modify them as needed.

Second, although simple features behaved well under distortion, the one hierarchical feature (the eye) did not behave as well. As the eye narrowed, the pupil bulged out beyond the white of the eye. This bulging resulted from a change in the directions of the normals to the sides of the eye that caused the feature to distort

in an unwanted manner. This suggests that care should be taken when designing features, especially hierarchical features, to avoid pasting over regions of high curvature. Alternatively, we could use a different pasting process for the interior control points to minimize distortion.

The second test I conducted for animated features was to slide the fingers over the hand to make a grasping motion. Initially, I performed this motion by sliding the feature (finger) domain within the base (hand) domain, where I kept the relative sizes of the two domains constant. With this motion, the fingers became greatly distorted, increasing in width by more than a factor of two. An analysis of the base surface (the hand) revealed that it had a non-uniform parameterization.

To fix this problem with the fingers, I rescaled the size of the finger domain within the hand domain and was able to keep the fingers at a nearly constant thickness. One more general approach would be to allow the user to keyframe the domain width. However, it might also be possible to have the system automatically adjust the relative domain sizes in an attempt to keep the feature width constant. Note that a similar problem occurs when modeling with pasted surfaces. However, in a modeler, the user can easily adjust the features manually. In a keyframe system, it is more important to try to maintain constant (or keyframed) size at inbetween frames. As a side note, these size problems can be alleviated if the base surface has a reasonable parameterization and thus bases should be designed that way when possible.

5.2 Conclusions

Pasted Surfaces animated well on a deformable base surface. There are some bad cases over regions with high curvature, but generally we would expect pasted sur-

faces to behave well as they did in my sample animation. Even so, the animator will need to edit inbetween frames to fix interpenetration problems.

Sliding features over a base surface does not work as well, mainly due to non-uniform parameterization of the base surface. It is not clear how useful this is as an animation technique, but if it is needed, then it would need more research to make it work well.

It is difficult to compare my work directly to other animated modeling techniques, but I expect it to compare well, as surface pasting should reduce modeling time and leave the animator free to concentrate on an animation motion. However, this can only be determined once surface pasting is fully integrated into an animation package and tested by production animators.

5.3 Future Work

As mentioned above, areas of future research include the automatic resizing of features for inbetween frames and the alternative placement of interior control points to minimize distortion. In addition, the behavior of pasted surfaces should be investigated in other animation techniques, such as physically based animation.

Bibliography

[Bar94] C. Barghiel. Feature oriented composition of b-spline surfaces. Master's thesis, University of Waterloo, Computer Science Department, Waterloo, Ontario, March 1994. (Available as Computer Science Department Technical Report CS-94-13).

[BBB87] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, Inc., 1987.

[BBF95] C. Barghiel, R. Bartels, and D. Forsey. Pasting spline surfaces. In L. Schumaker, M.Daehlen, and T.Lyche, editors, *Mathematical Methods for Curves and Surfaces*, pages 31–40. Vanderbilt University Press, 1995.

[BF91] R. Bartels and D. Forsey. Spline overlay surfaces. Technical Report CS-92-08, University of Waterloo, Computer Science Department, Waterloo, Ontario, 1991.

[Boe80] W. Boehm. Inserting new knots into b-spline curve. *Computer-Aided Design*, 12:199–201, 1980.

[BW71] N. Burtnyk and M. Wein. Computer-generated key-frame animation. *Journal of the Society of Motion Picture and Television Engineers*, 80(3):149–153, March 1971.

[Cha96] L. Chan. World space user interface for surface pasting. Master's thesis, University of Waterloo, Computer Science Department, Waterloo, Ontario, September 1996. (Available as Computer Science Department Technical Report CS-96-32).

[Far94] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, Inc., 3rd edition, 1994.

[FB88] D. Forsey and R. Bartels. Hierarchical b-spline refinement. *Computer Graphics*, 22(4):205–212, August 1988.

[Mae96] George Maestri. *Digital Character Animation.* New Riders Publishing, 1996.

[WB76] M. Wein and N. Burtnyk. Computer animation. In J. Belzer, A.G. Holzman, and A. Kent, editors, *Encyclopedia of Computer Science and Technology*, volume 5, pages 397–436. Marcel Dekker, Inc., 1976.

[Wer94] J. Wernecke. *The Inventor Mentor Programming Object-Oriented 3D Graphics with Open Inventor, Release 2.* Addison-Welsey Publishing Company, 1994.

[WW92] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques Theory and Practice.* Addison-Wesley, 1992.