Computer Aided Ferret Design

by

Selina Siu

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, 2003

©Selina Siu 2003

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Ferrets are amusing, flexible creatures that have been under represented in computer models. Because their bodies can assume almost any curved shape, splines are the natural tool for modelling ferrets. Surface pasting is a hierarchical method of modelling with spline surfaces, where features are added onto a base surface. Existing surface pasting techniques are limited to modelling rectilinear shapes. Using the task of modelling a ferret as a driving force, I propose a method of pasting cylinders in world space; I looked at methods for reducing distortion of pasted features; and I created a method for pasting trimmed features to allow for features that do not have the rectilinear shape of standard pasting. With my methods, modelling ferrets with surface pasting is easier, and the resulting models are closer to a real ferret.

Acknowledgements

I would like to thank my supervisor Stephen Mann for putting up with me and my ferrets for so many years. Besides being a tolerant supervisor, Steve has also been wonderful ice cream maker, a good friend and a caring mother duck. My thesis and my years at CGL would have been lacking without his invaluable suggestions and his sense of humour.

I would like to thank my readers Richard Bartels and Craig Kaplan for taking the time to read my thesis and giving me many helpful comments. Richard has directed me in understanding many obscure details. Because of Craig, my thesis may even be mostly grammatically correct. I would also like to thank Bill Cowan for putting in words the relationship between ferrets and splines, Patrick Gilhuly for fixing one of my bugs every year, Stefanus Du Toit for help with debuggers, and Elodie Fourquet for her 500 words a day reminders. Financial support for this thesis is provided by NSERC, OGS ST, and CITO.

My time at CGL would not be the same (or as long) without all my non-thesis endeavours and all the support I got for it. I am grateful to Steve for his support with all my side projects, especially in my art work. Many thanks to Art Green, who started by being such a fun teacher I couldn't stop taking his courses. I am also indebted to Art for his supervision of my animated painting project, and Bill for his technical help with it and making it my back up thesis project. Steve, Josée Lajoie, Don Mackay, Doug Kirton, Ian Bell provided many suggestions that made a good excuse to take time away from my thesis. I also wouldn't have as much fun without Josée, Pascale Proulx, Art, Bill, Don and Doug to guide me during our teaching endeavours in animation.

I would also like to thank my friends/roommates Ina Sinzig and Thanh Gundy, who went as far as grocery shopping and litterbox cleaning for me. To Yoshie Kotaka, it's good to have a friend I can call in the middle of the night. To Linda Hagan, my calming influence on my stressful days. To my friends and members of the CGL who were there to offer me advice, technical support, or make fun of me, as necessary. And to my family for their continuing support in my misguided life.

Finally, I would like to thank Birch, Dief, Sprite, Storm, Sand, Bear, Dart and Clef for their love and inspiration. Nothing would be worthwhile without my kids.

This work is dedicated to my ferrets.

Trademarks

OpenGL is a registered trademark of SGI.
Houdini is a registered trademark of Side Effects Software.
The Ferret Trading Post is a trademark of Crunchy Concepts Inc.

All other products mentioned in this thesis are trademarks of their respective companies.

Contents

List of Tables

List of Figures

Chapter 1

Ferrets, pasting, and pasting ferrets

Ferrets [29] are the only domesticated member of the mustelid family. They are small, usually with a masked face, and have a bundle of personality. Historically, they have had a close working relationship with humans in warrening and rodent extermination. Currently, they are popular as a charming if rather misunderstood[1] pet.

Ferrets need to turn easily in small confined spaces, so they are long and narrow, with a body that can assume almost any curved shape (Figure 1.1). Therefore, splines are the natural tool for modelling their bodies.

Surface pasting is a spline-based hierarchical modelling technique that is implemented only in Side Effects Houdini [30]. Creating 3D computer models is a time consuming task with a steep learning curve for an artist. Surface pasting is based on the idea that refined features can be added onto a simpler base surface, much as a child would add details such as eyes and ears to an oval head when modelling in clay. Because surface pasting is conceptually similar to modelling in clay, I believe it is easier for an artist to visualize the modelling process and create 3D computer models using surface pasting, compared to methods such as subdivision surfaces [31].

[1]An opinion, formed after taking my ferrets for many walks in the park, and shared by many ferret owners.

Figure 1.1: Ferrets can assume almost any curved shape, as demonstrated here by Birch.

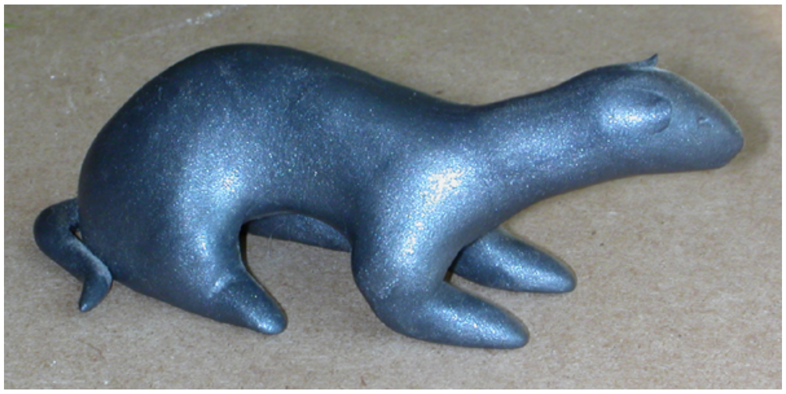


Figure 1.2: A ferret modelled in clay.

1.1 Clay modelling paradigm

If we were to model in clay, we can imagine making a ferret as follows: first we model a tubular body with a tail on one end and a vague shape for the head on the other; then we model the front legs and hind legs as elongated tubes, and stick them onto the body; finally we make ears, eyes and nose and stick them onto the head to get our final ferret. Figure 1.2 shows a ferret modelled in clay. The clay joins have been smoothed out, but evidence that the limbs and ears were added onto a body is evident in the close ups (Figure 1.3).

Modelling in clay is a task most people are familiar with from childhood, but there are no equivalents in software. If we were to follow the clay modelling paradigm in a commercial package such as Houdini, it is easy to model the ferret's

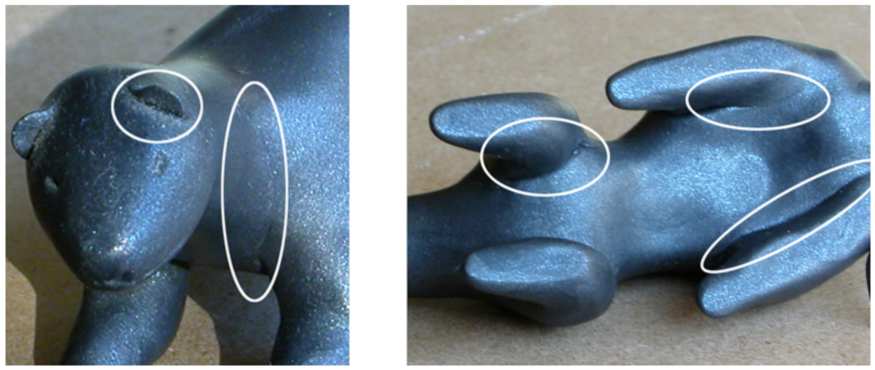


Figure 1.3: Close ups of the ferret display the barely visible evidence that the ears and limbs are stuck on and smoothed out.

long body, the fore legs and hind legs, with cross sections, skinning and capping. Attaching the features is more complex. To attach a leg, we have to create a blend from the leg to the body. First, we define a closed curve on the body. Next, we join the curve on the body to the boundary curve on the leg. We may need to adjust the parameterization and start of the curves to ensure that the blend is not twisted. This will give us a C^0 join; a smooth join requires more work.

1.2 Motivation for improving surface pasting

Surface pasting is closer to the clay modelling paradigm than other hierarchical modelling methods. Suppose we want to create a digital model of a ferret using surface pasting. We would like to follow the steps as for our clay model. First, we can create a tubular body with a generalized cylinder as the base surface on which to stick features. Next, we create feature patches to paste on. With patch pasting [2, 3, 8], the features are limited to rectangular bumps, which are inadequate for modelling arms, legs, eyes or ears. The underlying rectangular structure of the feature is often visible as well. The feature warps to the curvature of the surface it is pasted on, so that the result is not always predictable. Figure 1.4 shows the ferret model I made using patch pasting. Patch pasting is supported in Houdini, but as we can see from the model, the objects we can create with it are somewhat limited.

Based on these observations, I want to improve surface pasting as follows: find a better way to create tubular shapes such as arms or legs; allow arbitrarily shaped bumps as features; and maintain better control over the warping of the features. With the introduction of cylindrical pasting, arms and legs could potentially be

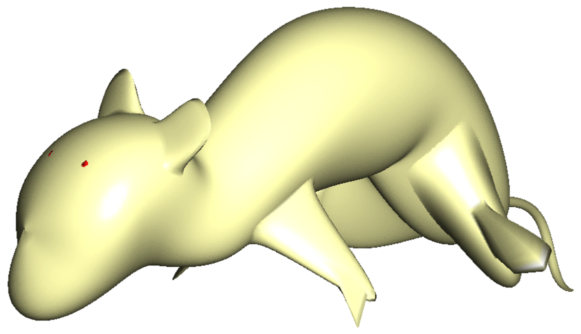


Figure 1.4: A ferret model I built in 1999 using patch pasting.

modelled more easily. However, existing work [24, 25] merely shows that cylinders can be pasted, and the user interface aspects of making and pasting cylinders need further exploration before this technique becomes useful as a modelling tool.

1.3 This thesis

For this thesis, I investigate some of the user interface issues of cylindrical pasting. Then I take some of the ideas I used in cylindrical pasting and apply them to patch pasting to support the creation and pasting of arbitrary shape features. I also touch on a method of improving the join between the feature and its base surface.

In Chapter 2, I give an overview of B-spline curves and surfaces, the different techniques used in surface pasting, and some tools used in building cylindrical objects and user interfaces for surface pasting.

In Chapter 3, I describe the tools I built for creating ferrets. I first describe the improvements I made to cylindrical pasting, and then how I applied these techniques to patch pasting, and finally how I modified patch pasting to allow for arbitrary shaped features. These techniques can be combined to achieve different modelling goals. I also explain a technique I used to improve the boundary between feature and base.

In Chapter 4, I illustrate the techniques I have developed in my program. I also compare my boundary improvement with the standard method of pasting. I give a step by step description of how I build a ferret, and show other models that can be built with improved surface pasting.

In Chapter 5, I discuss the pros and cons of the tools I have developed, and give suggestions for improvements. I end with some thoughts on modelling and how much further I have to go to achieve a truly computer aided ferret design.

Chapter 2

Ferreting out the background

2.1 Little splines

The word ferret comes from the latin word *furonem*, which means thief. Ferrets, *Mustela putorius furo*, are the only domesticated species in the family Mustelidae, which has members such as weasels, otters, polecats, badgers and wolverines [14]. They were domesticated about 2500 years ago, with the first known references in Greek plays by Aristophanes around 400 BC [29]. Ferrets were first used in warrening, rodent control, and later as transporters. They have played an indispensable role in an English sport named ferret-legging [14]. More recently, they act as pampered pets in many households.

A typical ferret lives between 6 to 8 years, although some have lived up to 11 or 12 years. Ferrets are full grown by a year, and become geriatric around 3 to 4. A female ferret weighs between 0.3 to 1.1 kilograms, and is about 16 to 18 inches long, including the tail. A male generally weighs 0.9 to 1.6 kilograms, and is about 18 to 20 inches long. The most common ferret colour is sable, recognized by yellow-buff fur with patches of black or dark brown on the tail and limbs, with light facial fur and a dark mask over dark brown eyes [14]. Albinos are also seen frequently, and mutations have produced a wide variety of colours and patterns.

Ferrets have poor eye sight, can see a limited range of colours, but have a superior sense of smell and sensitive paws. Fueled by their natural curiousity, they can get into no end of trouble. A ferret at play behaves like a toddler on a double expresso [29], dooking and hopping forward and sideways, with an arched back and a bushy tail (Figure 2.1). Ferrets are burrowing critters that engage in digging, stealing, and hoarding. They have developed certain anatomical adaptations because of their burrowing behaviour, such as the placement of the carotid arteries to

Figure 2.1: Diefenbaker with an arched back and a bushy tail.

help maintain cerebral blood flow when they turn in tightly confined spaces. They have a flexible vertebral column that is large in comparison to their size, with 7 massive cervical vertebrae, 15 thoracic vertebrae, 5 to 7 lumbar vertebrae, 3 sacral vertebrae, and 18 caudal vertebrae forming the tail [14]. Ferrets are the physical manifestation of splines, and ideal candidates for modelling and animation.

2.2 From curves to surfaces

Piecewise polynomial curves are used in modelling. Two common representations are *Bézier curves* and *B-spline curves*. In this section, I will define and briefly describe B-spline curves. More details can be found in Farin's book [12] and Mann's course notes [22].

2.2.1 B-spline curves

B-spline curves are a generalization of Bézier curves. The "B" stands for basis and the term B-spline was coined as a short way of saying Shoenberg's spline basis functions. A degree d B-spline curve $B(t)$ is defined by a number of control points C_i and a *knot sequence* t_i from which the B-spline curve is calculated. The control points can also be derived from the polynomial curve's *blossom function*, a symmetric, multi-affine map $f(t_1, t_2, \ldots, t_d)$ such that $f(t, t, \ldots, t) = B(t)$. Each control point C_i is the value $f(t_i, t_{i+1}, \ldots, t_{i+d-1})$. If we have n control points, then

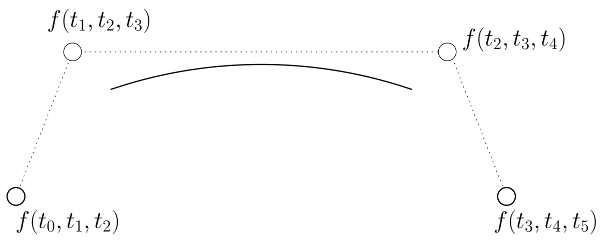


Figure 2.2: A cubic B-spline curve.

we need $n + d - 1$ knots[1], and the B-spline curve will be $n - d$ segments long. The domain of the curve ranges from t_{d-1} to t_{n-1}. The mathematical representation is

$$B(t) = \sum_{i=0}^{n} C_i N_i^d(t),$$

where $N_i^n(t)$ are the B-spline of the curve.

As illustrated in Figure 2.2 a cubic B-spline curve with one segment will have four control points, and a knot vector $\{t_0, t_1, t_2, t_3, t_4, t_5\}$. The domain of the curve is $[t_2, t_3]$.

Segment i of a B-spline curve is defined by control points C_i through to C_{i+d} and knots t_i to t_{i+2d-1}. The valid range of this segment is $[t_{i+d-1}, t_{i+d}]$. In general, each segment shares d control points (and thus $d - 1$ derivatives) with its neighbouring segments. However, if a knot is repeated, then one less control point is shared and there will be a segment of zero length. A B-spline curve of degree d is generally C^{d-1} continuous, but each time a knot is repeated, the continuity drops by one at that knot. If a knot has full multiplicity, that is, it is repeated d times, then the adjacent segments share only one control point (the end point) and the B-spline curve is generally only C^0 continuous at that point.

We can extend a B-spline curve at either end by adding additional control points and knots. To add control to a B-spline curve in the middle, we find additional control points by knot insertion. In Figure 2.3, we insert a knot $t \in (t_2, t_3)$ to get a new knot sequence $\{t_0, t_1, t_2, t, t_3, t_4, t_5\}$. The new control points can be calculated using blossoming. The original control points are replaced by the ones in bold outline.

One method of evaluating a B-spline curve is by using *repeated knot insertion* (Figure 2.4). When a knot t has full multiplicity, the value of the curve at t is

[1]Traditional definitions of B-spline curves require an additional knot at each end of the knot sequence.

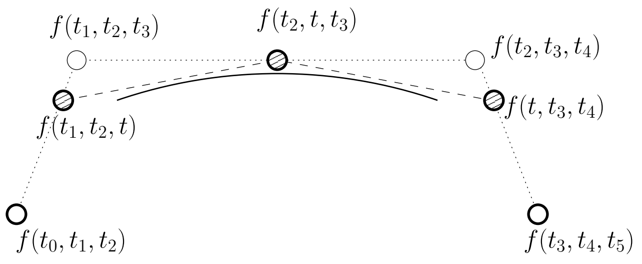


Figure 2.3: Adding a knot to a cubic B-spline curve

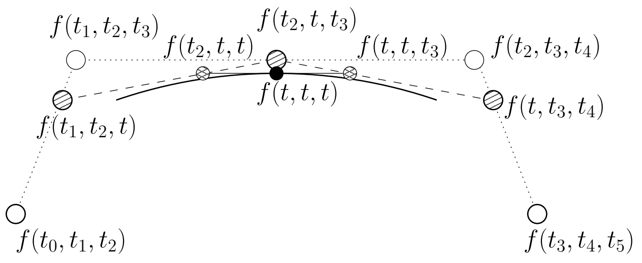


Figure 2.4: Repeated knot insertion. t is inserted into $\{t_0,t_1,t_2,t_3,t_4,t_5\}$ three times.

the blossom value of $f(t,t,..,t)$. This process is known as the *de Boor algorithm.* Figure 2.5 shows the data flow diagram for one segment of the cubic B-spline curve.

The derivative of $B(t)$ or $f(t,t,t)$ can be expressed in terms of its previous level of the data flow diagram:

$$B'(t) = d\frac{f(t_3,t,t)-f(t_2,t,t)}{t_3-t_2}.$$

Subsequent derivatives can be expressed in terms of the previous derivatives, e.g., the derivative of $f(t_3,t,t)$ can be expressed as

$$(d-1)\frac{f(t_3,t_4,t)-f(t_2,t_3,t)}{t_4-t_2},$$

and so the second derivative of $B(t)$ can be expressed as

$$B''(t) = \frac{d(d-1)}{t_3-t_2}\left[\frac{f(t_3,t_4,t)-f(t_2,t_3,t)}{t_4-t_2} - \frac{f(t_2,t_3,t)-f(t_1,t_2,t)}{t_3-t_1}\right].$$

$$\begin{tikzcd}
 & & & f(t,t,t) & & & \\
 & & f(t_2,t,t) \arrow[ur, "\frac{t_3-t}{t_3-t_2}"] & & f(t_3,t,t) \arrow[ul, "\frac{t-t_2}{t_3-t_2}"'] & & \\
 & f(t_1,t_2,t) \arrow[ur, "\frac{t_3-t}{t_3-t_1}"] & & f(t_2,t_3,t) \arrow[ul, "\frac{t-t_1}{t_3-t_1}"'] \arrow[ur, "\frac{t_4-t}{t_4-t_2}"] & & f(t_3,t_4,t) \arrow[ul, "\frac{t-t_2}{t_4-t_2}"'] & \\
f(t_0,t_1,t_2) \arrow[ur, "\frac{t_3-t}{t_3-t_0}"] & & f(t_1,t_2,t_3) \arrow[ul, "\frac{t-t_0}{t_3-t_0}"'] \arrow[ur, "\frac{t_4-t}{t_4-t_1}"] & & f(t_2,t_3,t_4) \arrow[ul, "\frac{t-t_1}{t_4-t_1}"'] \arrow[ur, "\frac{t_5-t}{t_5-t_2}"] & & f(t_3,t_4,t_5) \arrow[ul, "\frac{t-t_2}{t_5-t_2}"']
\end{tikzcd}$$

Figure 2.5: Data flow diagram for the de Boor algorithm

2.2.2 Closed B-spline curves

Sometimes we need to model with a closed curve. To construct a closed curve, we can simply make the first and last control points of the curve be the same and raise the end knots to full multiplicity so the curve touches. However, this curve is only C^0 continuous at its ends.

In a B-spline curve, two segments meet with C^{d-1} continuity when they share d control points. So for the first and last segment to meet with C^{d-1} continuity, the last d control points should be the same as the first d control points, and the knot spacing should also be the same. For example, Figure 2.6 shows a cubic B-spline curve with knot sequence $t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9$. The first three control points are replicated so that the curve meets with C^2 continuity at its start and end. The knot spacing for the last three control points must be the same as the initial three, so our constraints on t_6, t_7, t_8, and t_9 are

$$\begin{aligned}
t_6 &= t_5 + t_1 - t_0 \\
t_7 &= t_6 + t_2 - t_1 \\
t_8 &= t_7 + t_3 - t_2 \\
t_9 &= t_9 + t_4 - t_3.
\end{aligned}$$

Now suppose we want to do knot insertion in this cubic B-spline curve. The valid range for this B-spline curve is $[t_2, t_7]$. If we insert $t \in (t_4, t_5)$, then we can

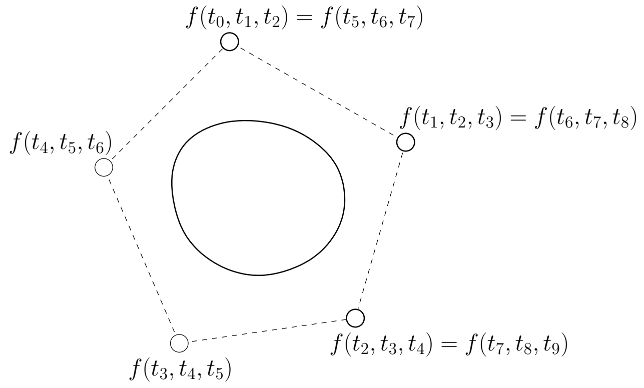

$f(t_0,t_1,t_2) = f(t_5,t_6,t_7)$
$f(t_1,t_2,t_3) = f(t_6,t_7,t_8)$
$f(t_4,t_5,t_6)$
$f(t_2,t_3,t_4) = f(t_7,t_8,t_9)$
$f(t_3,t_4,t_5)$

Figure 2.6: A closed cubic B-spline curve. The first three control points are replicated and the spacing between t_0, t_1, t_2, t_3, t_4 and t_5, t_6, t_7, t_8, t_9 is the same.

proceed as with a regular B-spline curve. However, suppose we insert a knot in the first two segments of the curve, say $a \in (t_2, t_3)$ (Figure 2.7). Our first five knots, which specify the first three control points, are now t_0, t_1, t_2, a, t_3. Our last three control points must be adjusted to reflect the new control points and our last five knots must be adjusted so that the spacing matches the first five knots:

$$\begin{aligned} b &= t_6 + a - t_2 \\ t'_8 &= b + t_3 - a \\ t'_9 &= t'_8 + t_4 - t_3. \end{aligned}$$

Our new knot sequence is $t_0, t_1, t_2, a, t_3, t_4, t_5, t_6, t_7, b, t'_8, t'_9$. If we insert a knot in the last two segments of the curve, we would have to adjust the first five knots so the knot spacing matches the new knots at the end.

We can also think of closed B-spline curves as the result of curves defined over a periodic parameter, such that $t + T = t$ for all t and some fixed T. We can think of matching the knot spacing and control points at the beginning and end as cutting and unrolling the the periodicity to impose the periodic form onto a linear domain. When we knot insert, we have to do additional work to keep the beginning and end coordinated.

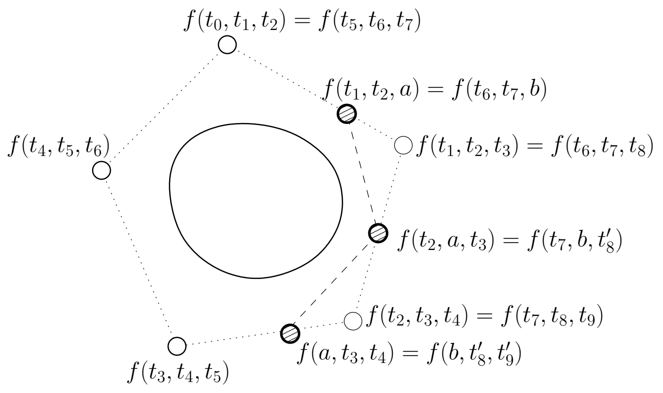

$f(t_0,t_1,t_2) = f(t_5,t_6,t_7)$
$f(t_1,t_2,a) = f(t_6,t_7,b)$
$f(t_1,t_2,t_3) = f(t_6,t_7,t_8)$
$f(t_4,t_5,t_6)$
$f(t_2,a,t_3) = f(t_7,b,t_8')$
$f(t_2,t_3,t_4) = f(t_7,t_8,t_9)$
$f(a,t_3,t_4) = f(b,t_8',t_9')$
$f(t_3,t_4,t_5)$

Figure 2.7: Inserting a knot in a closed cubic B-spline curve. The last few knots must be adjusted to match the spacing for the new initial knots. The control points in bold are the new control points specifying the curve.

2.2.3 Greville points

Greville points are the averages of knots. Each control point has a corresponding Greville point. For a degree d B-spline curve, each Greville point is defined as

$$g_i = \frac{1}{d}(t_i + t_{i+1} + \ldots + t_{i+d-1}).$$

(Note that $f(t_i, t_{i+1}, \ldots, t_{i+d-1})$ is the blossom value of the corresponding control point.) The point on the B-spline curve evaluated at g_i is the point that is maximally influenced by control point C_i. For an open curve, if the Greville point does not lie within the valid range, it is clamped to the valid range. For a closed curve, if the valid range is $[t_a, t_b]$, then a Greville point t outside of the valid range is wrapped around:

$$t' = \begin{cases} t + t_b - t_a & \text{if } t < t_a \\ t - t_b + t_a & \text{if } t > t_b. \end{cases}$$

In this thesis, any open curves I use will have full end knot multiplicity, so the Greville points are never clamped. I will also refer to the 2-D or 3-D points on a curve or surface evaluated at Greville points as *curve* or *surface Greville points*, to differentiate from the domain Greville points.

A curve Greville point G_i corresponding to a control point C_i is the result of an

$$\begin{tikzcd}
 & & & 1 \arrow[dl, "\frac{t_3-g}{t_3-t_2}"'] \arrow[dr, "\frac{g-t_2}{t_3-t_2}"] & & & \\
 & & N_0^1(g) \arrow[dl, "\frac{t_3-g}{t_3-t_1}"'] \arrow[dr, "\frac{g-t_1}{t_3-t_1}"] & & N_1^1(g) \arrow[dl, "\frac{t_4-g}{t_4-t_2}"'] \arrow[dr, "\frac{t-g_2}{t_4-t_2}"] & & \\
 & N_0^2(g) \arrow[dl, "\frac{t_3-g}{t_3-t_0}"'] \arrow[dr, "\frac{g-t_0}{t_3-t_0}"] & & N_1^2(g) \arrow[dl, "\frac{t_4-g}{t_4-t_1}"'] \arrow[dr, "\frac{g-t_1}{t_4-t_1}"] & & N_2^2(g) \arrow[dl, "\frac{t_5-g}{t_5-t_2}"'] \arrow[dr, "\frac{g-t_2}{t_5-t_2}"] & \\
N_0^3(g) & & N_1^3(g) & & N_2^3(g) & & N_3^3(g)
\end{tikzcd}$$

Figure 2.8: Running the de Boor algorithm backwards for a cubic B-spline curve. We feed 1 into the root and evaluate at a Greville point g to obtain the values of the B-splines at g.

evaluation at $B(g_i)$, which can be written as

$$B(g_i) = \sum_{i=0}^{n} C_i N_i^d(g_i).$$

The B-splines N_i^d evaluated at each g_i remains the same as long as the knot sequence is unchanged. We can calculate the values of the B-splines at each Greville point by running the de Boor algorithm backwards (Figure 2.8). Then for any open curve, we can calculate and store the B-splines evaluated at its Greville points. If the control points are changed while the the knot sequence remains the same, the new curve Greville points can be calculated as

$$G = NC,$$

where G is the vector of curve Greville points, C is the vector of control points and N is an $n \times n$ matrix where each row i consists of the B-splines for the first segment j to which C_i belongs, evaluated at g_i. The entries in row i start at j, and the remaining entries are 0. For example, for a cubic B-spline curve two segments long, we have

$$\begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} = \begin{bmatrix} N_0^3(g_0) & N_1^3(g_0) & N_2^3(g_0) & N_3^3(g_0) & 0 \\ N_0^3(g_1) & N_1^3(g_1) & N_2^3(g_1) & N_3^3(g_1) & 0 \\ N_0^3(g_2) & N_1^3(g_2) & N_2^3(g_2) & N_3^3(g_2) & 0 \\ N_0^3(g_3) & N_1^3(g_3) & N_2^3(g_3) & N_3^3(g_3) & 0 \\ 0 & N_1^3(g_4) & N_2^3(g_4) & N_3^3(g_4) & N_4^3(g_4) \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix},$$

where N_k^d indicates the kth B-spline used to weight the kth control point.

If we are working with a closed curve, then the last d Greville points will be repeated and need not be calculated. We can also use the first d control points in our calculations since the control points are repeated. The entries for the matrix N that we create will wrap around and N can be generated with the following algorithm:

```
for (i=0; i<n-d; i++) {
    set j to segment for g$_i$
    run de Boor backwards for g$_i$ on segment j
    for (k=0; k <= d; k++) {
        set matrix[i][(j+k)%(n-d)] to be N$^d_{j+k}$(g$_i$)
    }
}
```

The cubic closed curve shown in Figure 2.6 has eight control points and five segments. We only have to calculate the first five Greville points, so our equation would look something like this:

$$\begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix} = \begin{bmatrix} N_5^3(g_0) & N_6^3(g_0) & N_7^3(g_0) & 0 & N_4^3(g_0) \\ N_0^3(g_1) & N_1^3(g_1) & N_2^3(g_1) & N_3^3(g_1) & 0 \\ 0 & N_1^3(g_2) & N_2^3(g_2) & N_3^3(g_2) & N_4^3(g_2) \\ N_5^3(g_3) & 0 & N_2^3(g_3) & {}_0 2_3^3(g_3) & N_4^3(g_3) \\ N_5^3(g_4) & N_6^3(g_4) & 0 & N_3^3(g_4) & N_4^3(g_4) \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}.$$

Because of the matching spacing of the knot sequence for the last few control points, $N_0^3(g) = N_5^3(g)$, $N_1^3(g) = N_6^3(g)$ and so on.

Since each B-spline is non-zero on its Greville point, we know by the Schoenberg-Whitney theorem [11] that N is non-singular, and hence an invertible matrix. We can use N^{-1} to calculate the control points from a given set of Greville points:

$$C = N^{-1}G.$$

2.2.4 Tensor product B-spline surfaces

Tensor product B-spline surfaces are one extension of B-spline curves to surfaces. Mathematically, an $n \times m$ tensor product B-spline patch is given as

$$T(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} C_{i,j} N_i^d(u) N_j^e(v),$$

where u, v is a rectilinear domain, $C_{i,j}$ are the control points, and $N_i^d(u)$, $N_j^e(v)$ are the B-splines. This equation can be rewritten as

$$T(u,v) = \sum_{i=0}^{n} \left\{ \sum_{j=0}^{m} C_{i,j} N_j^e(v) \right\} N_i^d(u).$$

Here, $\sum C_{0,j} N_j^e(v)$ can be treated as a degree e curve with control points

$$C_{0,0}, C_{0,1}, \ldots, C_{0,m-1}.$$

There are n such curves composed to form a surface. We can also rearrange the terms and view it as m degree d curves with n control points each parameterized by u joined together to form a surface. The curves in each direction share the same knot sequence, so one knot sequence for u and one for v are used in evaluation of the curves.

Evaluation and knot insertion for tensor product B-splines surfaces generalize from curves. To insert a knot in u, we knot insert into each of the m B-spline curves. Similarly for v. To evaluate, we can apply the de Boor algorithm to all the curves in one parametric direction, say u, which gives us the control points for a curve in v which we evaluate to get a point on the surface.

Using the above method for evaluation, if we want the partial derivatives for the surface in both directions (for example, to compute the surface normal), we will have to evaluate the surface twice, starting with a different parametric direction each time. Mann and DeRose [23] gave a less expensive method of evaluation.

Using a series of closed B-spline curves in one direction, we can build a cylindrical surface.

In a tensor product B-spline surface, Greville points are now (u,v) domain pairs, and surface Greville points can be calculated. We can still apply the equation $G = NC$, but now G and C are both 2-dimensional arrays of points, and N is a 3-dimensional array of B-spline evaluations. For patches that have full end knot multiplicity, we can apply the curve version of $G = NC$ to just the boundary curves

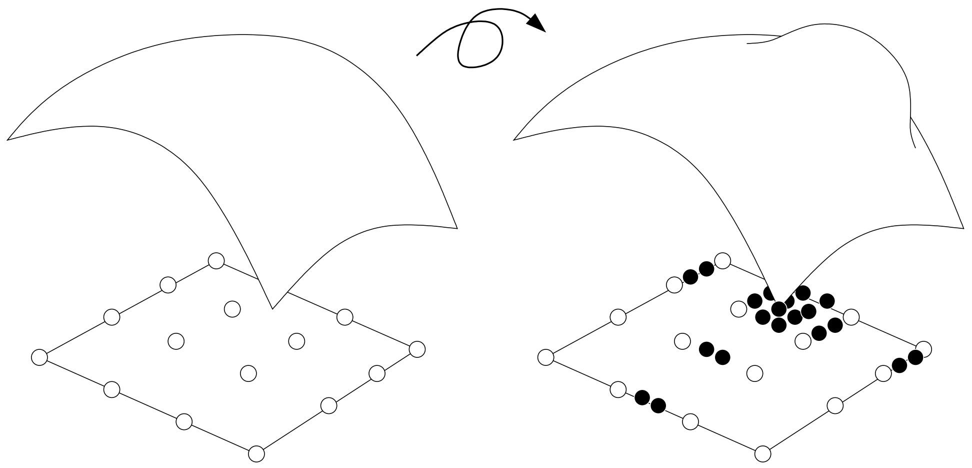


Figure 2.9: Adding details via knot insertion adds complexity to other parts of the surface as well.

to calculate the Greville points.

In this thesis, I will use tensor product B-spline surfaces with full end knot multiplicity in both u and v directions, unless they are cylindrical surfaces. Cylindrical surfaces will have full end knot multiplicity in u and the closed curves will be defined with respect to v.

2.3 Surface pasting

When modelling, we often have mostly smooth surfaces with areas of high detail. One method of adding detail to a surface is by knot insertion (Figure 2.9). However, this method adds complexity to the control mesh even where it is not needed. Methods to overcome this limitation include hierarchical B-splines refinement [13] and surface pasting, a generalization of the hierarchical B-splines refinement method.

Surface pasting is as it sounds, a method to glue surfaces on top of one another. Detailed complex surfaces are pasted on other surfaces, thus preserving the simplicity of base surfaces. The next few sections describe standard surface pasting [2, 3], projective surface pasting [8, 30], and cylindrical surface pasting [24, 25].

2.3.1 Standard surface pasting

Standard surface pasting, developed by Barghiel, Bartels and Forsey [5, 3, 2], consist of a tensor product B-spline surface, called a *feature*, and another tensor product

B-spline surface or a composite surface from pasting, known as the *base* surface.

The feature surface is represented as a *Greville displacement surface* using a *diffuse coordinate system* [5]. In a diffuse coordinate system, each control point has an associated local coordinate frame. The control point $C_{i,j}$ is expressed as an origin $O_{i,j}$ and a displacement vector $d_{i,j}$ in its local coordinate space. The equation for a tensor product B-spline surface becomes

$$\begin{aligned} T(u,v) &= \sum_{i=0}^{n}\sum_{j=0}^{m} C_{i,j}N_i^d(u)N_j^e(v) \\ &= \sum_{i=0}^{n}\sum_{j=0}^{m} (O_{i,j}+d_{i,j})N_i^d(u)N_j^e(v). \end{aligned}$$

In a Greville displacement surface, the domain is embedded in the range by mapping u, v to x, y respectively. The origin of the frame of each control point $C_{i,j}$ is chosen as $C_{i,j}$'s corresponding 3D Greville point $(g_{ui}, g_{vj}, 0)$.

Given a base surface $B(u,v)$ and a feature surface $F(u,v)$, pasting is done in the following way (see Figure 2.10 for an illustration):

1. In a domain window, the user can translate, rotate or scale the feature domain. We initiate the pasting process when the feature domain lies entirely inside the base domain via a transformation T. Each Greville point $g_{i,j}$ in the feature becomes $g'_{i,j} = Tg_{i,j}$.

2. For each transformed Greville point $g'_{i,j}$, we calculate the corresponding point $B(g'_{ui}, g'_{vj})$ on the base surface. This point is taken to be the new origin of the local coordinate frame $F = \{B(g'_{ui}, g'_{vj}), \vec{\imath}, \vec{\jmath}, \vec{k}\}$ of the control point $C_{i,j}$. The basis vectors $(\vec{\imath}, \vec{\jmath}, \vec{k})$ for $C_{i,j}$ is constructed from the partial derivatives of $B(g'_{ui}, g'_{vi})$ as

$$\begin{aligned} \vec{\imath} &= \frac{\partial}{\partial x}B(g'_{ui}, g'_{vj}) \\ \vec{\jmath} &= \frac{\partial}{\partial y}B(g'_{ui}, g'_{vj}) \\ \vec{k} &= i \times j. \end{aligned}$$

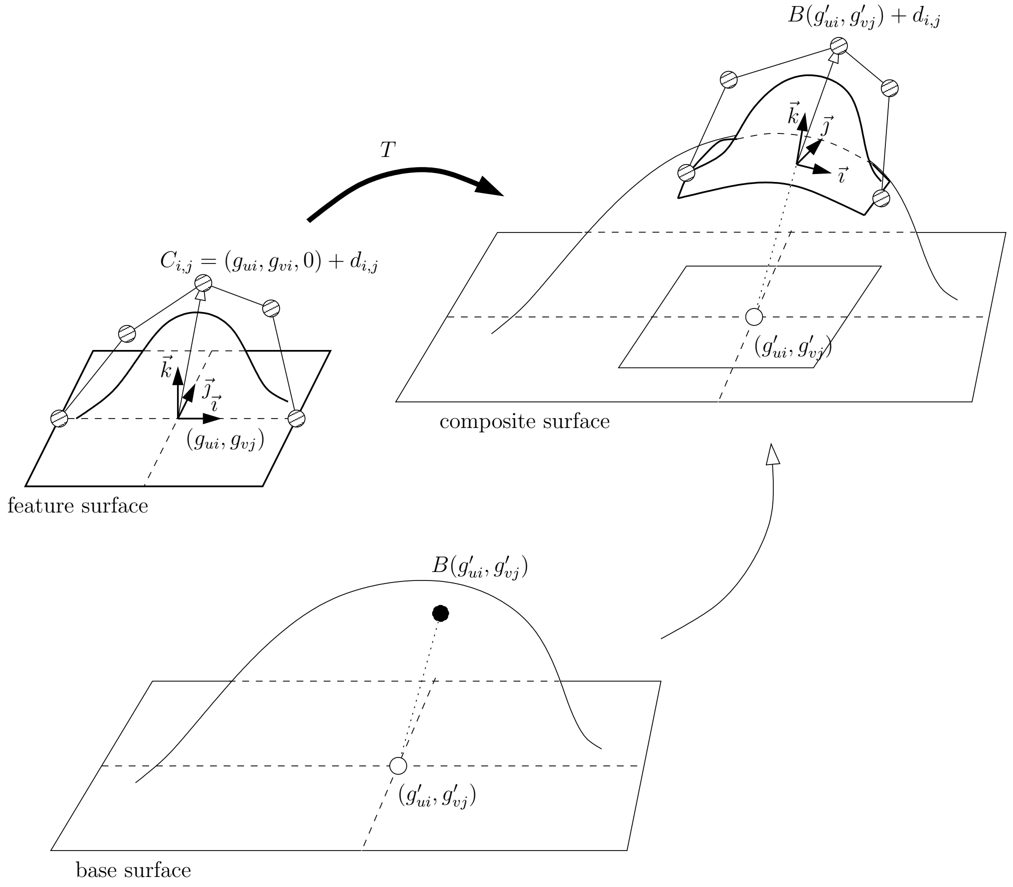


Figure 2.10: Pasting a feature onto a base surface via standard surface pasting.

3. The control hull for the pasted feature is now represented by

$$\begin{aligned} F'(u,v) &= \sum_{i=0}^{n}\sum_{j=0}^{m} C'_{i,j} N_i^d(u) N_j^e(v) \\ &= \sum_{i=0}^{n}\sum_{j=0}^{m} (B(g'_{ui}, g'_{vi}) + d_{i,j}) N_i^d(u) N_j^e(v). \end{aligned}$$

When pasting a feature onto a base, we want the feature to mold itself to the base. If the feature is flat (all $d_{i,j}$ are the zero vector), then we want it to lie precisely on the base surface. However, surface pasting is only an approximation technique. If we look at a pasted boundary curve, each boundary control point lies on the base surface. If the base surface is not flat, then the boundary curve does not touch its

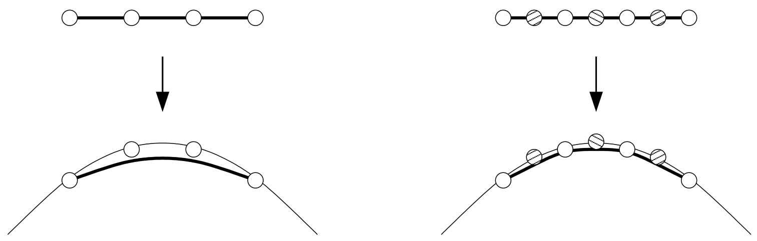


Figure 2.11: Before (left) and after (right) knot insertion. Knot insertion after pasting gives a closer approximation to the base surface.

control point, and does not overlap the base surface exactly. We can improve the boundary approximation to the base surface by knot insertion (Figure 2.11). The more knots we insert, the closer the boundary curve is to the base.

Hence when the boundary control points have zero displacement, we have approximate C^0 continuity. Approximate C^d continuity is achieved by setting the outer most d layers of control point to have zero displacement.

2.3.2 Projective surface pasting

Standard surface pasting is usually implemented by manipulating (scaling, translating, and rotating) the feature domain inside the base domain, which is an unintuitive user interface. A more direct user interface is provided by projective surface pasting [8, 30].

Projective surface pasting (Figure 2.12) differs from standard pasting mainly in the first step. The four corner points of the feature surface are projected on to the base surface. The domain for the feature surface forms a rectangle $D = F_1F_2F_3F_4$. The corresponding domain points for the base are found, and these form a polygon $D' = F'_1F'_2F'_3F'_4$. Since the target polygon is not necessarily a rectangle, we cannot use a linear transformation. Instead, a bi-linear transformation T is used. With a bi-linear transformation, all points inside D can be represented by

$$P(u, v) \;=\; L_1(v) \cdot u + L_2(v) \cdot (1 - u),$$

where

$$\begin{aligned} L_1(v) &= F_1 \cdot v + F_4 \cdot (1 - v) \\ L_2(v) &= F_2 \cdot v + F_3 \cdot (1 - v). \end{aligned}$$

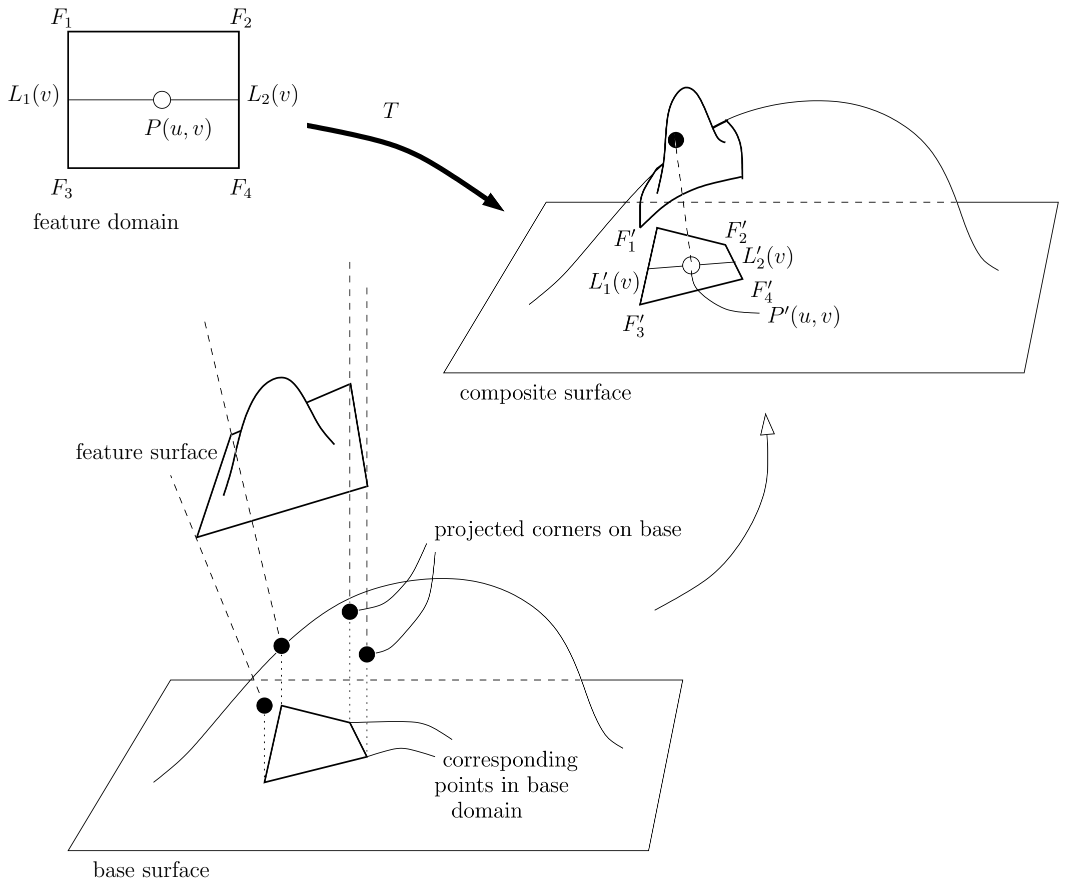


Figure 2.12: Pasting a feature onto a base surface via projective pasting.

Similarly, points in D' are represented by

$$P'(u,v) \;=\; L_1'(v)\cdot u + L_2'(v)\cdot(1-u),$$

where

$$\begin{aligned} L_1'(v) &= F_1'\cdot v + F_4'\cdot(1-v) \\ L_2'(v) &= F_2'\cdot v + F_3'\cdot(1-v). \end{aligned}$$

After transforming the feature domain, we calculate the new control points as before. This method of pasting is more intuitive for the user as they work directly with the 3D model. It also allows skewing of the feature domain.

In the remainder of this thesis, I will refer to this type of pasting as bilinear

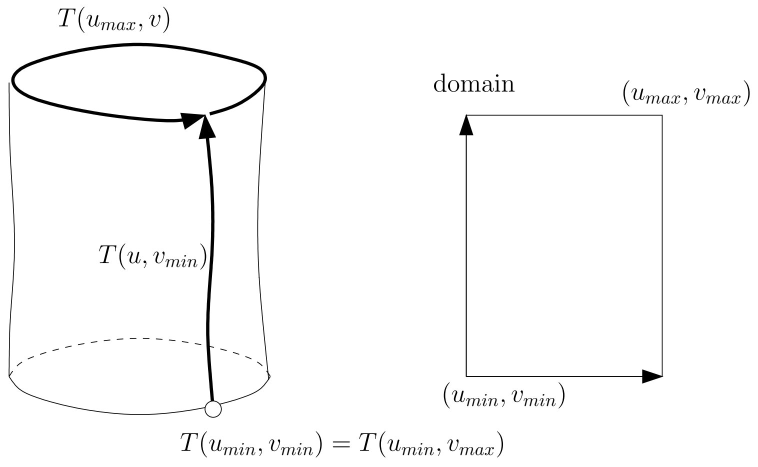


Figure 2.13: Domain mapping for a cylinder.

pasting or projective bilinear pasting, to differentiate from other types of projection based pasting methods I will explore.

2.3.3 Cylindrical pasting

Cylindrical pasting [24, 25] is a method that integrates techniques of parametric trimline-base blending into surface pasting. It takes two base surfaces and creates a smooth transitional cylinder between them. There are two different ways of pasting a cylinder: pasting the cylinder's end onto a surface, or pasting two cylinders end to end. In this thesis, I am only concerned with the former. For details on the latter, refer to Mann and Yeung [24].

A cylindrical tensor product B-spline surface can be built by using a series of B-spline curves closed in one parametric direction. Figure 2.13 shows the domain mapping for a cylinder. The closed B-spline curves are represented in the parametric v direction.

The cylinder is the feature surface. The cylinder used has circular cross sections defined by n unique control points. We paste one of the edges of the feature cylinder onto the base surface via a domain mapping. We denote the control points for this edge L_0. In the base domain, n points are spaced evenly on a circle. For each domain point $d_k = (u_k, v_k)$ in the circle, the control point $L_{0,k}$ is mapped to the base surface $B(u_k, v_k)$ (Figure 2.14). As with standard pasting, the boundary of the pasted feature may not lie directly on the base surface, but it will be an adequate approximation.

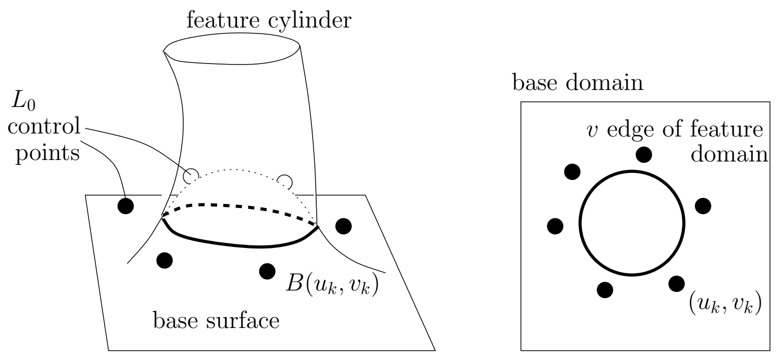


Figure 2.14: Pasting a feature cylinder onto a base surface. The layer of control points forming the edge of the cylinder is mapped onto the base domain.

The above gives an approximate C^0 paste of a cylinder to a base surface. For an approximate C^1 join, we need to map the next layer of control points L_1 as well. Following the rules for standard pasting, we could just map the L_1 layer to a second inner circle in the base domain, and paste the control points directly on the base surface.

There is a second method that gives a better C^1 continuity approximation. A coordinate frame is constructed at each L_0 control point. The frame is constructed such that the basis vectors that lie on the tangent plane to the control point are tangent to the domain circle and perpendicular to the domain circle, as shown in Figure 2.15. Each L_1 control point is then expressed as a displacement relative to its corresponding L_0 frame, and lies on the tangent plane to its corresponding L_0 control point.

2.3.4 Limitations of surface pasting

Surface pasting provides a relatively low cost and low storage method of adding detail to tensor product B-spline surfaces. Modelling is intuitive as features are refined individually and then added to the base surfaces. Base surfaces maintain simplicity, and complexity is created by overlaying features. A world space interface [8] lets users interact directly with their models.

While surface pasting seems ready to revolutionize the world, there remain many problems that need to be solved. Surface pasting is only an approximation technique, and as stated earlier, there are often gaps at the boundaries where feature meets base, although work done by Conrad [9, 10] and Leung [19, 18] improved the boundary approximation and continuity tremendously.

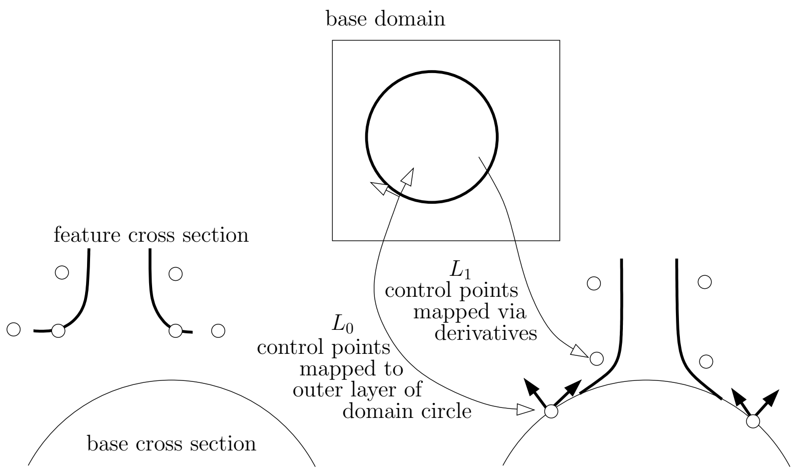


Figure 2.15: Pasting the L_1 layer of control points to achieve an approximate C^1 paste.

The limitation this thesis deals with is one of modelling. Standard pasting is only good for adding details with a rectilinear boundary. With bilinear projective pasting, a quadrilateral domain increases flexibility, but non-convex domains present a problem for bilinear interpolation of the feature domain space. Cylindrical pasting adds to the modelling pool, but is currently limited to an indirect approach to modelling via domain manipulation. And while molding of the feature to the base surface may be desirable sometimes, the feature is often unacceptably distorted (Figure 2.16).

2.4 Reference frames for generalized cylinders

Cylinders are good starting points for modelling elongated shapes such as arms and legs. A cylinder created with arbitrary cross-sections along a 3D curve (Figure 2.17), known as a generalized cylinder, is a common and useful modelling construct. *Reference frames* are positioned along a centroid curve to align each cross-section with its neighbour consistently. The description of generalized cylinders and discussion of reference frames are found in a paper of Bloomenthal [6].

A common type of reference frame is the *Frenet frame*. At any point P on the curve, a Frenet frame consists of a vector T tangent to the curve at P, a principal normal N, and a binormal B. T is simply the normalized first derivative V of the curve. The principal normal is defined be in the direction of the curvature $K = \frac{V \times Q \times V}{|V|^4}$, where Q is the second derivative of the curve. The remaining vectors

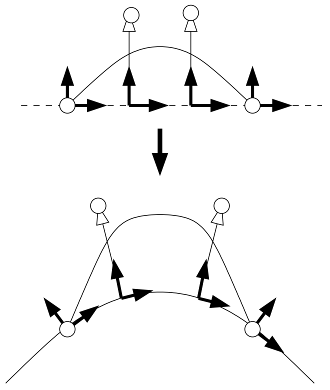


Figure 2.16: Distortion of a feature after pasting.

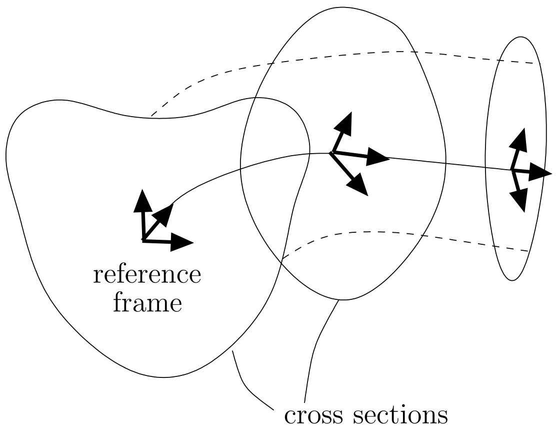


Figure 2.17: Generalized cylinder.

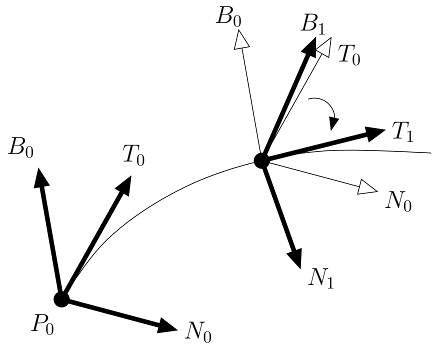


Figure 2.18: Calculating rotational minimizing frames.

are defined to be $N = \frac{K}{|K|}$ and $B = T \times N$.

Frenet frames are convenient because they can be computed independently at arbitrary points on the curve. But they are ill defined wherever the curvature is zero, such as on straight lines or at points of inflection. The normal can also flip after an inflection point, causing a violent twist in the frames along the curve.

Rotational minimizing frames solve some of the problems of Frenet frames. An initial reference frame is defined at the beginning of the curve, and each subsequent frame is calculated using small local rotations with respect to the previous frame (Figure 2.18). Given a frame $F_i = \{P_i, T_i, N_i, B_i\}$ at point P_i, we calculate the normalized tangent T_{i+1} at point P_{i+1}. The new frame F_{i+1} can be calculated by rotating F_i so that T_i aligns with T_{i+1}. N_{i+1} and B_{i+1} are rotated around $T_0 \times T_1$ by the angle $\cos^{-1}\left(\frac{T_0 \cdot T_1}{|T_0||T_1|}\right)$.

Rotational minimizing frames cannot be calculated independently, and the frames generated are dependent on how the initial frame is created. As well, the resulting frames may not be what the user expected, so sometimes a Frenet frame may be preferable.

2.5 Ray intersect spline surface

There are a few instances in which we need to know where a line intersects a spline surface. First, to paste a cylinder on a surface, the user needs to select a paste point on the surface. We can find this point by calculating the direction of the eye to the pixel of the mouse click based on the parameters of the viewing frustum (Figure 2.19), and intersecting this line with the spline surface.

Second, we need to know the intersection of a ray with a spline surface for projective pasting. In bilinear projective pasting, the four corners of the feature

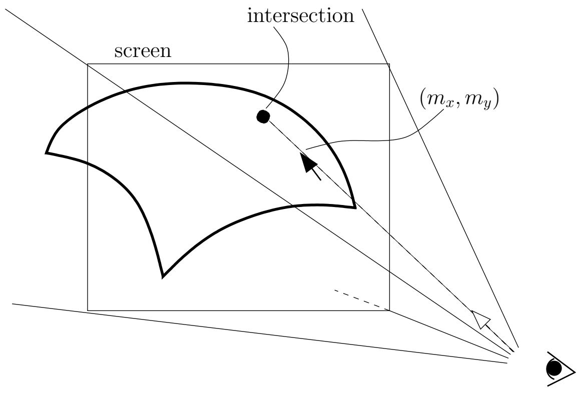


Figure 2.19: Surface point can be calculated by intersecting the spline surface with the direction from the eye point to the mouse

are projected onto the base surface to determine how the the feature domain is transformed in the base domain. For projective cylindrical pasting discussed in the next chapter, I create a gnomon for the selected point with the paste direction corresponding to the surface normal. The tangent of the cylinder spine point to be pasted is matched with this paste vector. I cast lines from the cylinder's current outer control/surface Greville points in the direction of the paste vector. The new control/surface Greville points are determined by where the lines intersect the spline surface.

To compute the intersection of a ray with a spline surface, I looked at two methods, an algorithm given by Kajiya [16] and a method that makes use of hardware via OpenGL.

2.5.1 Kajiya's method

In ray tracing, spline surfaces are subdivided and ray intersections with the subsequent polygons are calculated. This requires high overhead in terms of calculating and storing the mesh. In surface pasting, the surfaces change often, and the number of ray intersect surface calculations are few, so a numerical approach seems more reasonable. The method I chose to investigate was Kajiya's method [16]. It is as follows:

1. Convert the degree $n \times m$ tensor product B-spline surface into degree $n \times m$ Bézier patches via knot insertion.

2. Convert each patch to a monomial representation [12]. If the Bézier control points are given by C, then the monomial coefficients C' are given by M^tCN,

where

$$M_{ij} = (-1)^{j-i} \binom{m}{j} \binom{j}{i}$$

and

$$N_{ij} = (-1)^{j-i} \binom{n}{j} \binom{j}{i},$$

for $i = 1 \ldots n$ and $j = 1 \ldots m$.

3. Represent the ray $p(t) = E + wt$ as the intersection of two planes P_1 and P_2, where the normals of the planes are given by

$$\begin{aligned} N_1 &= \begin{cases} (w_y, -w_x, 0), & \text{if } |w_x| > |w_y| \text{ and } |w_x| > |w_z| \\ (0, w_z, -wy), & \text{otherwise} \end{cases} \\ N_2 &= N_1 \times w. \end{aligned}$$

4. Substitute the monomial surface representation into the equations of the planes to obtain bivariate polynomials

$$a(u,v) = N_1 \cdot (C'(u,v) - E) = 0$$

and

$$b(u,v) = N_2 \cdot (C'(u,v) - E) = 0.$$

After rewriting the equations we have

$$\sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} u^i v^j = 0$$

and

$$\sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij} u^i v^j = 0.$$

Note that the derivation used here differs slightly from Kajiya's paper in that a_i and b_j corresponds to the coefficient of the degree i, j terms, whereas in Kajiya's paper a_i and b_j corresponds to the degree $n-i$, $m-j$ terms.

5. Create the Bezout determinantal form of the resultant with respect to u or v, which ever has the higher degree. To illustrate the method, we will proceed as if v has the higher degree.

The resultant [1] is defined by

$$R(a,b) = a_0^m b_0^n (-1)^{m+n-1} \prod_{i=1}^{n} \prod_{j=1}^{m} (\alpha_i - \beta_j),$$

where a, b are univariate polynomials and α_i and β_j are roots of the polynomials respectively. $R(a,b) = 0$ if and only if a and b share a common root, that is, $\alpha_i = \beta_j$ for some i and j. The resultant can also be expressed as a polynomial in the coefficients of a and b. This allows us to calculate the resultant and use it to determine if the two polynomials share a common root without factoring them. In the case where a and b are bivariate polynomials in u and v, we can treat them as univariate polynomials in u or v with polynomial coefficients in v or u respectively. We can calculate the resultant with respect v so the coefficients will be in u. The value of the resultant is a univariate polynomial in u.

The resultant can be obtained by calculating the determinant of the Sylvester matrix, which is given by

$$\begin{vmatrix}
a_0 & a_1 & \dots & a_n & 0 & \dots & \dots & 0 \\
0 & a_0 & a_1 & \dots & a_n & 0 & \dots & 0 \\
\vdots & & \ddots & & & & & \vdots \\
\vdots & & & \ddots & & & & 0 \\
0 & \dots & \dots & 0 & a_0 & a_1 & \dots & a_n \\
b_0 & b_1 & \dots & b_n & 0 & \dots & \dots & 0 \\
0 & b_0 & b_1 & \dots & b_n & 0 & \dots & 0 \\
\vdots & & \ddots & & & & & \vdots \\
\vdots & & & \ddots & & & & 0 \\
0 & \dots & \dots & 0 & b_0 & b_1 & \dots & b_n
\end{vmatrix}.$$

For two degree n polynomials, the size of the matrix is $2n \times 2n$. A more efficient way to calculate the resultant is to use the Bezout determinantal form which is an $n \times n$ matrix. The entries for this matrix are given by the following code:

```
for (row=0; row<n; row++) {
    for (col=0; col<n; col++) {
        s = col+row+1;
        m[col][row] = 0;
        for (k=0; k <= row; k++) {
            m[col][row] += $a_k b_{s-k} - b_k a_{s-k}$
        }
    }
}
```

The resultant of two degree 4 polynomials would have the following form:

$$\begin{vmatrix} \begin{vmatrix} a_0 & b_0 \\ a_1 & b_1 \end{vmatrix} & \begin{vmatrix} a_0 & b_0 \\ a_2 & b_2 \end{vmatrix} & \begin{vmatrix} a_0 & b_0 \\ a_3 & b_3 \end{vmatrix} & \begin{vmatrix} a_0 & b_0 \\ a_4 & b_4 \end{vmatrix} \\ \begin{vmatrix} a_0 & b_0 \\ a_2 & b_2 \end{vmatrix} & \begin{vmatrix} a_0 & b_0 \\ a_3 & b_3 \end{vmatrix} - \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} & \begin{vmatrix} a_0 & b_0 \\ a_4 & b_4 \end{vmatrix} - \begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix} & \begin{vmatrix} a_1 & b_1 \\ a_4 & b_4 \end{vmatrix} \\ \begin{vmatrix} a_0 & b_0 \\ a_3 & b_3 \end{vmatrix} & \begin{vmatrix} a_0 & b_0 \\ a_4 & b_4 \end{vmatrix} - \begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix} & \begin{vmatrix} a_1 & b_1 \\ a_4 & b_4 \end{vmatrix} - \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix} & \begin{vmatrix} a_2 & b_2 \\ a_4 & b_4 \end{vmatrix} \\ \begin{vmatrix} a_0 & b_0 \\ a_4 & b_4 \end{vmatrix} & \begin{vmatrix} a_1 & b_1 \\ a_4 & b_4 \end{vmatrix} & \begin{vmatrix} a_2 & b_2 \\ a_4 & b_4 \end{vmatrix} & \begin{vmatrix} a_3 & b_3 \\ a_4 & b_4 \end{vmatrix} \end{vmatrix}$$

6. Find the roots of the resultant $r(u)$. The degree of $r(u)$ is at most $2n^2$. I used Laguerre's method [28] to find the roots $u_1, u_2, \ldots, u_q$. $r(v)$ is a polynomial with real coefficients, so the roots are either real or in conjugate pairs of complex numbers.

   There is one special case in calculating the resultant. If one of the polynomials, say $a(v)$, has degree 0 in u, then a, b have a common root only when $a(v)$ has a zero. In this case, we can solve for the roots of $a(v)$, and for each root v_j solve for the roots of $b(u, v_j)$. For this case, we can skip step 7 and proceed directly to the polishing step.

7. Substitute each root u_i to obtain univariate polynomials $a(u_i, v)$ and $b(u_i, v)$. For each pair, calculate the GCD via the Euclidean algorithm to obtain $c(u_i, v)$, which has degree m or less.

8. Find the roots v_j of each $c(u_i, v)$. Polish (u_i, v_j) to get the resulting domain point [15].[2]

   Calculate the surface point $C(u_i, v_j)$ and the distance t from the ray origin E. The difference between the surface point at (u_i, v_j) and the ray point $E + wt$ should be the zero vector:

$$\begin{bmatrix} x(u_i, v_j, t) \\ y(u_i, v_j, t) \\ z(u_i, v_j, t) \end{bmatrix} = C(u_i, v_j) - (E + wt) = \vec{0}.$$

   If (u_o, v_o, t_o) is the current guess, then a better guess is given by

$$\begin{bmatrix} u_{o+1} \\ v_{o+1} \\ t_{o+1} \end{bmatrix} = \begin{bmatrix} u_o \\ v_o \\ t_o \end{bmatrix} - \begin{bmatrix} \frac{\partial x(u_o,v_o,t_o)}{\partial u} & \frac{\partial x(u_o,v_o,t_o)}{\partial v} & \frac{\partial x(u_o,v_o,t_o)}{\partial t} \\ \frac{\partial y(u_o,v_o,t_o)}{\partial u} & \frac{\partial y(u_o,v_o,t_o)}{\partial v} & \frac{\partial y(u_o,v_o,t_o)}{\partial t} \\ \frac{\partial z(u_o,v_o,t_o)}{\partial u} & \frac{\partial z(u_o,v_o,t_o)}{\partial v} & \frac{\partial z(u_o,v_o,t_o)}{\partial t} \end{bmatrix}^{-1} \begin{bmatrix} x(u_o, v_o, t_o) \\ y(u_o, v_o, t_o) \\ z(u_o, v_o, t_o) \end{bmatrix}$$

$$= \begin{bmatrix} u_o \\ v_o \\ t_o \end{bmatrix} - \begin{bmatrix} (\frac{\partial C(u_o,v_o)}{\partial u})_x & (\frac{\partial C(u_o,v_o)}{\partial v})_x & w_x \\ (\frac{\partial C(u_o,v_o)}{\partial u})_y & (\frac{\partial C(u_o,v_o)}{\partial v})_y & w_y \\ (\frac{\partial C(u_o,v_o)}{\partial u})_z & (\frac{\partial C(u_o,v_o)}{\partial v})_z & w_z \end{bmatrix}^{-1} \begin{bmatrix} x(u_o, v_o, t_o) \\ y(u_o, v_o, t_o) \\ z(u_o, v_o, t_o) \end{bmatrix}.$$

9. Keep the root pairs (u_i, v_j) where $0 \leq u_i \leq 1$ and $0 \leq v_j \leq 1$ to obtain the intersection points..

Unfortunately, for bicubic patches, combinatorial constants resulting from the change of basis and expansion exceeds double floating point precision, leading to inaccurate answers. Possibly interval analysis [26] or use of a different basis would solve these problems.

2.5.2 Making use of OpenGL

An alternative method to find intersections between rays and spline surfaces is to use OpenGL's picking and colour buffers to retrieve the domain values of the spline surfaces [20]. I use the following steps to calculate the location under a user's mouse click:

1. The user clicks on a pixel location (m_x, m_y) in the application on the screen.
2. Determine the surface selected via OpenGL's picking.

[2]With many thanks to Richard Bartels for explaining this in detail.

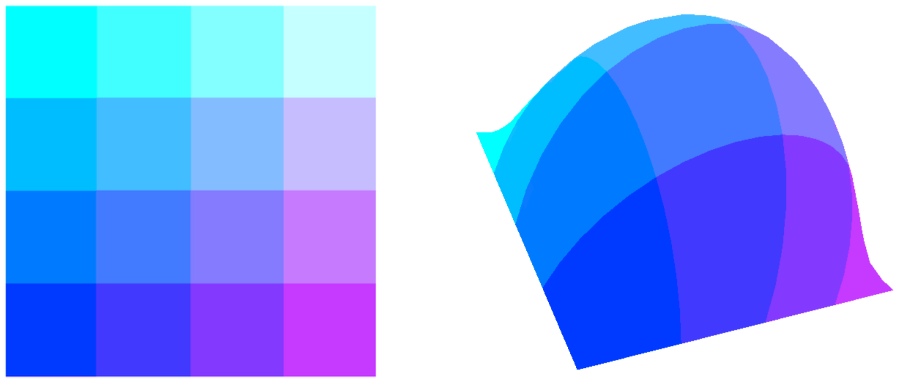


Figure 2.20: A 2-bit domain (left) with its corresponding surface textured in the domain (right). Each pixel sharing the same colour corresponds to one (u, v) location on the surface, inadequate for selection purposes.

3. Render a false coloured version of the surface from the same viewpoint as the user into an OpenGL colour buffer. The r, g, b colours of the surface should be textured such that $r = 0$ corresponds to the minimum u value, and $r = 1$ corresponds to the maximum u value, and $g = 0$ corresponds to the minimum v value, and $g = 1$ corresponds to the maximum v value.

4. Read the r and g values of the colour buffer at (m_x, m_y) and map them to obtain u and v.

This method is limited by the number of bits in the OpenGL's colour buffer, which is machine dependent. I used a machine with a 5-bit colour buffer and one with an 8-bit buffer. With an n-bit buffer, the surface is divided into $2^n \times 2^n$ sections each represented by a unique r and g. The number of sections is independent of the number of surface segments given by the B-spline surface.

With the 8-bit buffer, the surface point calculated usually corresponds to the screen location under the user's mouse click, but with the 5-bit buffer, the section can be large and the surface point can be quite far off. See Figure 2.20 for an example of the problem demonstrated by a 2-bit buffer. This method can be made more accurate by using the (u, v) values as an initial guess in the same root polishing method stated in the previous section. The new surface point calculated corresponds to the screen location.

I can calculate the domain values at which the new control/surface Greville points should be by rendering the surface from each control/surface Greville point in the paste direction, and reading the (u, v) value off the middle of the screen.

A more efficient way would be to render just once and read off all the (u, v) values from the same buffer. We do this the following way:

1. Project the control/surface Greville points on to the plane defined by the paste point and the paste direction.

2. Find the maximum distance between the projected points and the paste point.

3. Render the surface with the eye point at 1 unit behind the paste point in the paste direction, and the up direction from the paste point's gnomon. The near plane should be 1 unit from the eye point. Use an orthographic projection and make sure that the width and height of the viewport are at least twice the maximum distance calculated.

4. Read the r and g values off the colour buffer. The location indices can be calculated using the projected points and the paste points.

The (u, v) pairs for the new control/surface Greville points should also be polished for better accuracy.

Instead of polishing, I can also refine by rendering to hardware multiple times. For each r value read in from an n-bit colour buffer, the corresponding colour range is given by $(\frac{\lfloor r\times(n^2-1)+0.5\rfloor}{n^2}, \frac{\lfloor r\times(n^2-1)+0.5\rfloor+1}{n^2})$. We take the range and convert it to domain values in u. Then we knot insert in the surface to obtain a smaller surface bounded by the u range. We do the same with g to obtain the patch associated with the given (r, g) pair. Now colour this smaller patch in a similar manner, read in the new colour values and convert with respect to the new u, v ranges. This process can be repeated multiple times for additional accuracy.

Chapter 3

Ferret building tools

This chapter discusses in detail the improvements I have made to surface pasting. The first issue I have looked at is creating a more flexible cylindrical pasting paradigm. The ferret's long, sinuous body and tail, as well as its limbs, can be more closely represented by generalized cylinders than by bumps. This cylinder is built around a 3D spine curve. I devised a method to paste the feature cylinder onto a base surface in world space using a single point on the base and the cylinder's spine as guides. The feature cylinder is pasted on to the base by projection in world space rather than domain pasting.

To improve the boundary approximation, I also tested the pasting of surface Greville points. One question with surface pasting was how to place control points so that the feature boundary approximates the base surface better. While control points pasted onto the base lie on the surface, their corresponding surface Greville points may not lie on the base surface, creating gaps. By fixing the locations of surface Greville points such that they lie on the base surface, and then determining the locations of the control points, I can achieve a better C^0 boundary approximation. This improvement can be applied to both cylinders and patches.

To minimize distortion in pasting patches, I took my ideas for cylindrical pasting and applied them to patch pasting. Instead of representing patches as Greville displacement surfaces, I displaced their control points from a single point, and use it as a guide for pasting. The edges of the patch are pasted by projecting all boundary control points onto the base.

My last goal is to allow pasting of arbitrarily shaped patches. Such a patch is created by taking a regular rectangular patch and trimming away the boundary according to a domain curve. It is pasted as if it is an untrimmed patch, and a cylinder is pasted onto the trimmed feature and the base to join the two surfaces smoothly.

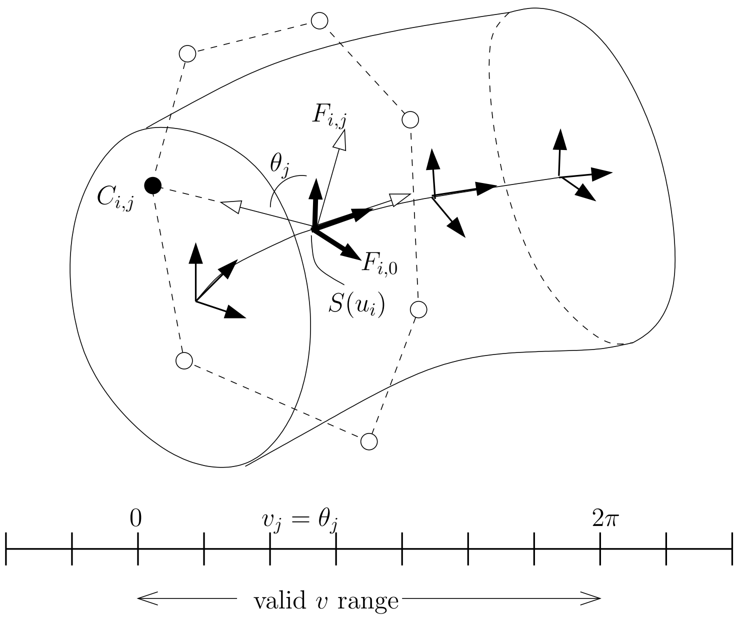


Figure 3.1: Building cylinder $T(u, v)$ from spine $S(u)$. $F_{i,j}$ is calculated by taking the associated frame for spine point $S(u_i)$ rotated by angle θ_j.

I end the chapter by explaining how these different techniques can be combined for hierarchical modelling.

3.1 Projective cylindrical paste

3.1.1 Building a cylinder

The cylinders that I use are tensor product B-spline surfaces wrapped around a spine curve. For a $n \times m$ cylinder that is degree $d \times e$, the spine is a B-spline curve defining the parametric u direction of the surface and shares the degree and knot vector with the surface. Figure 3.1 shows how the cylinder is constructed. The cylinder $T(u, v)$ is built as a displacement surface from the spine $S(u)$, such that

$$\begin{aligned} T(u,v) &= \sum_{i=0}^{n}\sum_{j=0}^{m} C_{i,j} N_i^d(u) N_j^e(v) \\ &= \sum_{i=0}^{n}\sum_{j=0}^{m} (S(u_i) + d_{i,j}) N_i^d(u) N_j^e(v) \end{aligned}$$

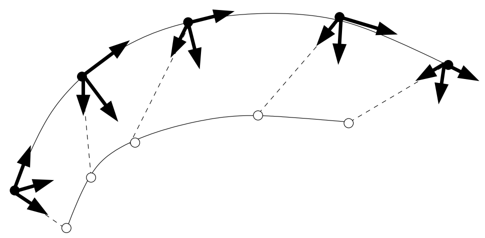


Figure 3.2: The second curve in a double spine determines how the reference frames are oriented.

where u_i is the ith Greville point, $d_{i,j}$ is the displacement for control point $C_{i,j}$, and the local coordinate frame for $C_{i,j}$ is $F_{i,j} = \{S(u_i), \vec{\imath}, \vec{\jmath}, \vec{k}\}$. $F_{i,j}$ is calculated by taking the associated frame for $S(u_i)$ rotated by angle θ_j. The valid range for v is linearly mapped to angles from 0 to 2π, and θ_j is the mapping for v_j, the jth Greville point.

The spine of the cylinders are 3D curves with Frenet frames or rotational minimizing frames as discussed in Section 2.4. Because of the limitations of these two types of construction, I implemented a third option, a double spine (Figure 3.2) where a second user manipulated curve is placed next to first curve. The principal normal for each spine frame is constructed by taking the closest normal that lies on the plane normal to the spine's tangent, and in the direction of the corresponding parametric point on the second curve.

3.1.2 Determining the paste

The user interface for projective pasting is in world space. To paste a surface, the user needs to specify a base surface, the end of the feature cylinder to paste, and the direction to project. My user interface is a multi-step process. See Figure 3.3 for details.

1. The user clicks on a surface B, the base surface for pasting. The location of the surface point where the user clicked, $P = B(u, v)$, is determined using methods discussed in Section 2.5. An initial coordinate frame $F = \{P, \vec{\imath}, \vec{\jmath}, \vec{k}\}$ is built as follows:

$$\begin{aligned}\vec{\imath} &= \frac{\partial B}{\partial u} \\ \vec{\jmath} &= \frac{\partial B}{\partial v} - \left(\frac{\partial B}{\partial v} \cdot \vec{\imath}\right) \times \vec{\imath}\end{aligned}$$

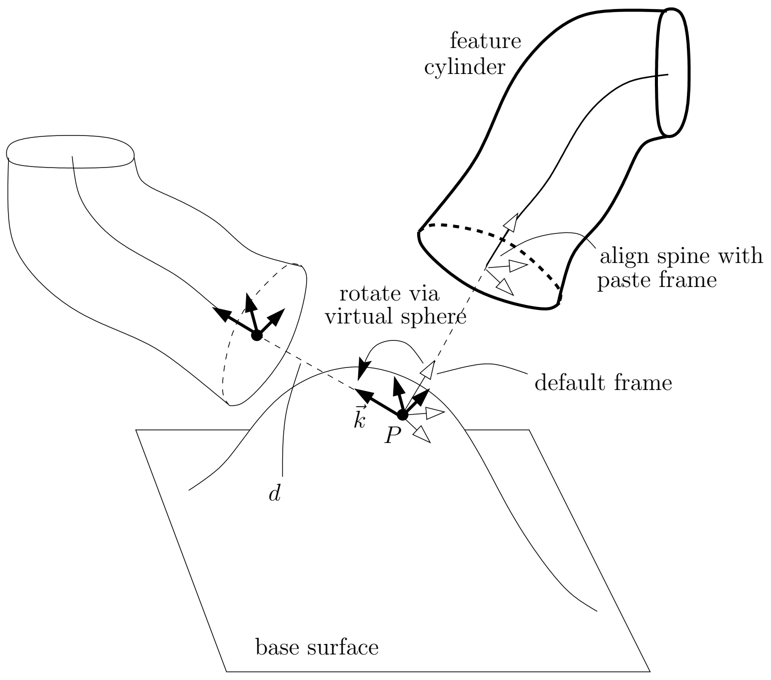


Figure 3.3: Projective cylindrical paste.

$$\vec{k} = \vec{\imath} \times \vec{\jmath}$$

The set of equations used to build the vectors ensures that they form an orthogonal frame, which we normalize to get an orthonormal frame. An orthonormal frame is needed so we can use it to match the coordinate frame, also an orthonormal frame, on a feature cylinder's spine. The default direction for pasting is $\vec{k}$, which is normal to the surface at point P.

2. The user clicks on one end of a cylinder. We first find out the u parameter of the clicked point, and then check if u is within the first quarter of the domain, or the last quarter of the domain. This determines whether we paste the start or end of the cylinder. Let T be the tangent vector at the cylinder's spine end. The cylinder is transformed (rigid body) by aligning the tangent vector T at the cylinder's spine's selected end. The cylinder is transformed by aligning T with the coordinate frame's basis vector $\vec{k}$ on the base surface, and the remaining basis vectors for the spine's end frame are aligned with $\vec{\imath}$ and $\vec{\jmath}$. If we are pasting the end of the cylinder where $u = u_{max}$, the spine's frame will have to be negated before alignment.

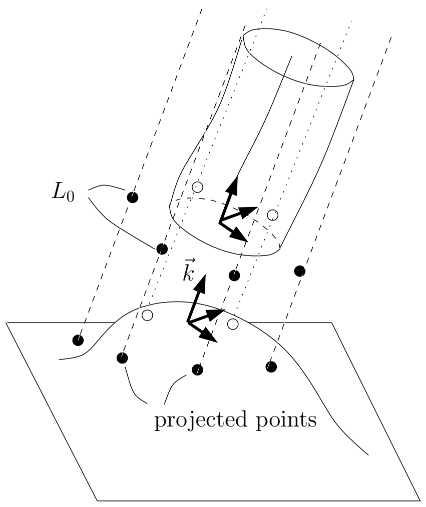


Figure 3.4: Control point paste

3. The initial frame F is built only as a guide. It can be rotated via a trackball interface. The cylinder's selected end is aligned with the paste frame as it is rotated. The user can also specify a distance d at which the cylinder end is displaced from the surface.

4. Finally, the user can choose to paste the cylinder using a control point paste or a surface Greville point paste. The approximate continuity of the paste is specified as well, and there is an option to keep the domain values of the paste even if the surfaces are modified. If conditions are not met, the paste may fail. These details are covered in the next few sections.

3.1.3 C^0 control point pasting

For a control point paste, I project the outermost layer of control points L_0 at the cylinder's selected end onto the base surface in the direction of the paste vector $\vec{k}$ (Figure 3.4). By intersecting the base surface with the projected lines, I can determine the projected points on the base surface. The L_0 layer of control points are modified to be at the positions of their projected counterparts while the remaining control points of the cylinder are not modified. This gives an approximate C^0 paste. The paste fails if any of the projected lines do not hit the base surface.

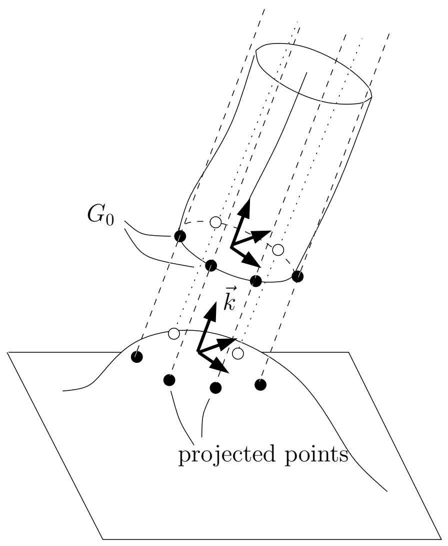


Figure 3.5: Surface Greville point paste

3.1.4 C^0 surface Greville point pasting

In surface pasting the feature is pasted by putting the Greville displacement origins onto the base. If the feature's boundary is flat, the control points are located at the Greville displacement origins, and the feature boundary will approximate the trim curve of the base surface. If the base is curved, then the pasted feature boundary lies below the base trim curve (Figure 2.11, left). Intuitively, it seems that an improvement to the approximation can be made by placing the surface Greville points, which lie on the feature surface boundary, instead of the control points, which may not.

In a surface Greville point paste, I project the outermost layer of surface Greville points G_0 at the cylinder's selected end onto the base surface in the direction of the paste vector $\vec{k}$ (Figure 3.5). I want to position the G_0 points to their projected counterparts on the base surface. To do this, I can use the equation discussed in Section 2.2.3,

$$L_0 = N^{-1}G_0,$$

where L_0 is the layer of boundary control points of the selected cylinder end, and N is the matrix of B-splines that calculates the curve Greville points from a set of control points. After solving for L_0, the outermost layer of control points is moved to L_0. L_0 does not in general lie on the base surface.

This also gives an approximate C^0 paste, where the surface Greville points are

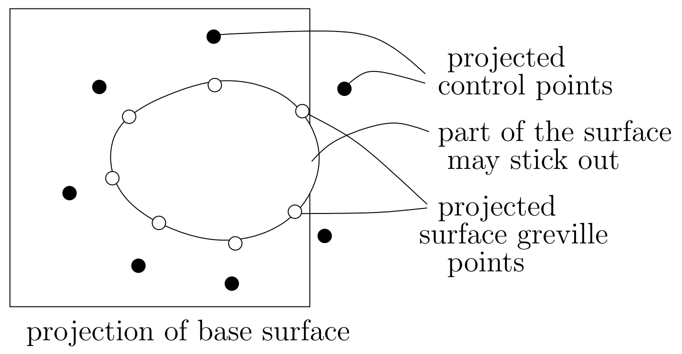


Figure 3.6: Control points may not project onto the base even if surface Greville points do. If this is the case, part of the cylinder's end curve may extend out of the base surface.

guaranteed to touch the base surface. The paste will fail where the projected lines do not hit the base surface, but since the surface Greville points in general lie much closer to each other than control points, the paste is less likely to fail. I will call D the shape formed by the projection of the base surface in the direction $\vec{k}$ in any plane perpendicular to $\vec{k}$. Then, for any convex feature cylinder, if the projected end layer of control points lie in D, so will the end layer of surface Greville points. The reverse is not necessarily true (figure 3.6). On the other hand, if some control points lie outside of D, then part of the paste feature may lie outside of the base surface.

A comparison and analysis between surface Greville point paste and control point paste can be found in Section 4.2.

3.1.5 Paste parameters

If the projected lines do not intersect the front facing side of the base surface at a positive distance, then the paste fails (Figure 3.7). The displacement parameter d translates the cylinder's end from the paste point on the base surface. This parameter is adjusted to ensure the above conditions can be met. These conditions limit the way the pasted surface can distort: if a control point is moved a negative distance in the paste, the surface is more likely to intersect itself; if it pastes on the opposite side of the surface, then the derivative at that point will be pointing away from the surface, causing a warp for an approximate C^1 paste, which is discussed later.

With projective pasting, each control point or surface Greville point has a corresponding domain point on the base surface. The user has the option to keep this

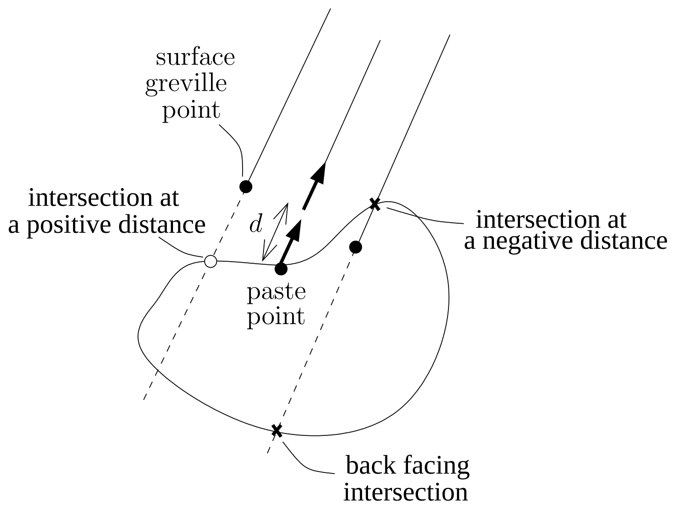

surface greville point
intersection at a positive distance
d
paste point
intersection at a negative distance
back facing intersection

Figure 3.7: The cylinder can only be pasted if all projected points intersect the front face of the surface at a positive distance.

mapping even if the base surface is modified. One reason for keeping the mapping is that the base surface could potentially be animated, and modifications to the base may violate some of the conditions for projective pasting.

As with other kinds of pasting, knot insertion improves the feature approximation to the base surface. When the user chooses the option to keep the current mapping, knot insertion is performed on the feature surface and the mapped domain curve. The newly generated control points are mapped to the base surface at the evaluation of their corresponding domain control point. Keeping the current mapping does not yield the same paste if the paste occurs after knot insertion, but does guarantee that the mapped control points stay in the same location.

3.1.6 Trimming away the base

After the feature cylinder has been pasted, the base surface is no longer a manifold. To get an approximate manifold, we can trim away the base surface inside of the cylindrical paste.

Trim curves in OpenGL must have full end knot multiplicity and start and end on the same control point. They are also specified in the domain space of the surface. For the cylinders pasted via control points, all we have to do is figure out the domain points for each pasted control point, re-use the same sequence of knot values, and then bring the curve to full multiplicity at its end points.

For a surface Greville point paste, we need to do a little more. Since the control points do not necessarily lie on the base surface, there are no domain points in the base domain that evaluate to their location. But the surface Greville points do have corresponding domain points in the base. So we apply matrix N^{-1} to the domain points D_0 for the surface Greville points to get

$$R_0 = N^{-1} D_0.$$

R_0 will give a trim curve where the base surface always touches the feature surface at the surface Greville points.

The base surface for pasting can also be another cylinder. Each cylinder has a seam where the start and end of the closed curve meets. With domain based cylindrical pasting, a closed domain curve is mapped to the pasted end of the feature cylinder. With my method, control points or surface Greville points are mapped to new locations independently and a closed 3D curve is created. Corresponding domain points are found for these points. When the feature cylinder is pasted over the seam of a base cylinder then it is not possible to create a continuous trim curve in the domain. In this case, we can create two separate trim curves that have edges on the seam, but I did not implemented this feature as part of my thesis work.

3.1.7 Capping the cylinder

When we are trimming the base surface for standard surface pasting, we have always trimmed away the inside part of the base surface, onto which the feature is overlaid. For cylinders however, trimming away the outside of the base surface makes sense as well. We can paste a cylinder onto a small patch, trim away the outside of the patch, and cap off the cylinder (Figure 3.8).

A simple algorithm for capping generalized cylinders is to take the centre point of the closed curve at the end, and triangulate. This works well for closed curves with a star kernel [27] but not all concave curves (Figure 3.9). Using pasting as a method of capping, areas outside of curves, convex or non-convex, are always trimmed, so it is a better algorithm for capping generic closed curves.

3.1.8 Approximate C^1 continuity

In existing cylindrical pasting [24, 25], a smooth blend with the base surface is achieved by mapping two circles in the base domain, and then mapping the boundary layer of the pasted cylinder to the outer curve, and the next layer of control points by displacement relative to the coordinate frame at each corresponding

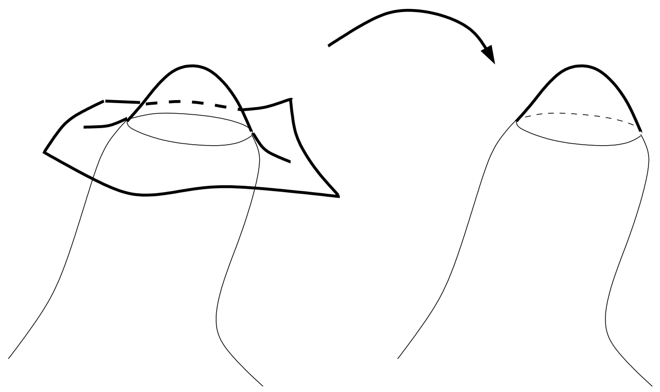


Figure 3.8: The outside of the base is trimmed away when we cap a cylinder.

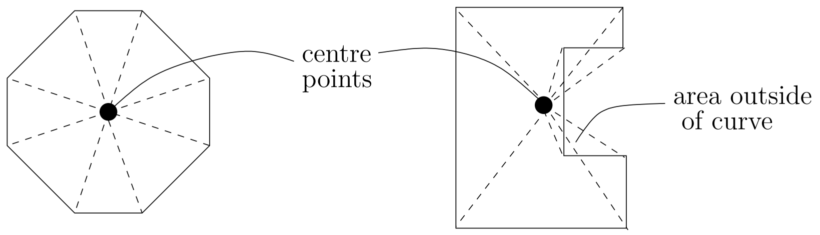


Figure 3.9: A simple algorithm for capping that works well for convex curves (left). For non-convex curves (right), areas outside of the curve may be capped as well.

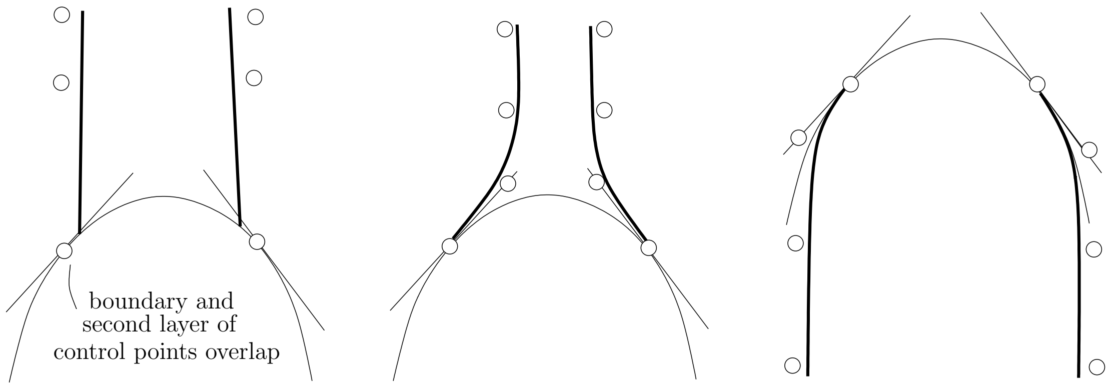


Figure 3.10: Left: when the second layer of control points project onto its boundary layer, we can only achieve C^0 continuity. Center: by making the boundary layer wider than the secondary layer, we can get a smooth approximate C^1 blend. Right: when we make the boundary layer smaller than the secondary layer, we trim the outside of the base to get a smooth cap.

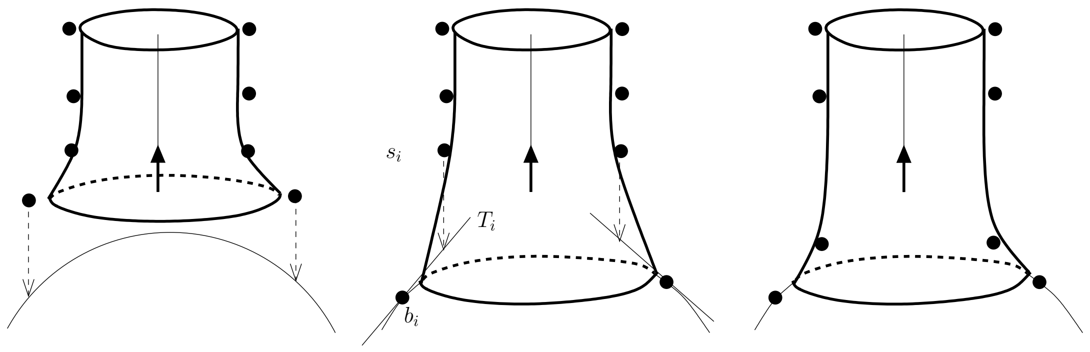


Figure 3.11: On the left, the outer layer of control points is projected onto the base. In the centre, a tangent plane is constructed at each pasted control point on the boundary. The second layer of control points is projected onto corresponding tangent planes to yield an approximate C^1 paste, shown on the right.

pasted outer control point. The displacement is calculated based on the mapping of the control points to the two circles in the domain.

With my method of cylindrical pasting, the cylinder's boundary is mapped to a closed curve, not necessarily convex, in the domain. I could try to map the second layer of control points in a similar manner to existing cylindrical pasting, but it is unclear how to create a smaller inner curve for arbitrary closed curves. For example, if I were to scale the curve smaller around a central point, it is quite possible that the inner curve and outer curve will intersect if the curve is not convex. For example, if the closed curve on the right in Figure 3.9 were to be scaled down around its centre point, the smaller curve would overlap with the original curve.

For this reason, I approximate C^1 continuity as follows. The outer layer of control points is mapped onto the base as with C^0 pastes. For each pasted control point b_i on the boundary, I construct a tangent plane T_i to the base surface at b_i. The corresponding control point s_i on the second layer is placed at the intersection of the line created by the unpasted s_i and the paste direction with the tangent plane T_i. Figure 3.11 illustrates the process.

How smooth the blend appears is determined by the user. An approximate C^1 paste where all the second layer of control points lie in the paste direction from the boundary layer of control points will appear to have a C^0 join, as both layers of control points will be projected and pasted onto the same location (Figure 3.10, left). For a smoother looking join, the boundary layer should be moved further away from the second layer (Figure 3.10, centre). In the case where we are capping a cylinder, we can make the boundary layer smaller than the second layer to yield

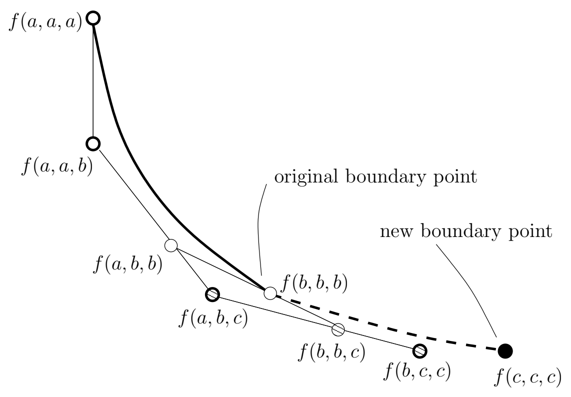


Figure 3.12: Extending a B-spline curve lengthens the curve disproportionately. Here, the original curve has knot sequence a, a, a, b, b, b. The curve is extended by adding control point $f(c, c, c)$, changing the knot sequence to a, a, a, b, c, c, c. Control points $f(a, b, c)$, and $f(b, c, c)$ have predetermined positions that lengthen the original curve. Control point $f(c, c, c)$ can be placed freely, but the extended curve may already be longer than the user desired.

a smooth join to the inside of the trim (Figure 3.10, right).

Extending the cylinder's boundary

As stated above, to create a smooth approximate C^1 join or cap, the user is required to edit the boundary layer of control points manually so that the boundary curve is larger or smaller than the curve defined by the second layer of control points. The distance between the two curves determines the smoothness of the join or cap, with a larger distance giving a smoother appearance. I investigated simplifying this process for the user by automatically extending the boundary curves.

I wanted to generate an additional layer of control points for the user automatically. The question then became how to place this additional layer. If a layer was added and then extended as for trimmed patches (Section 3.3.4), then the boundary segment of the cylindrical feature surface will be changed, something I thought was undesirable.

When I changed the end knot values (necessary to maintain full end knot multiplicity), I can keep the original feature surface unmodified by moving the original boundary layers of control points as well as adding a new layer of control points. The downside of this method is that the new segment created by extension may be longer than the user desired. Figure 3.12 illustrates the problem with respect to extending one control point on the boundary.

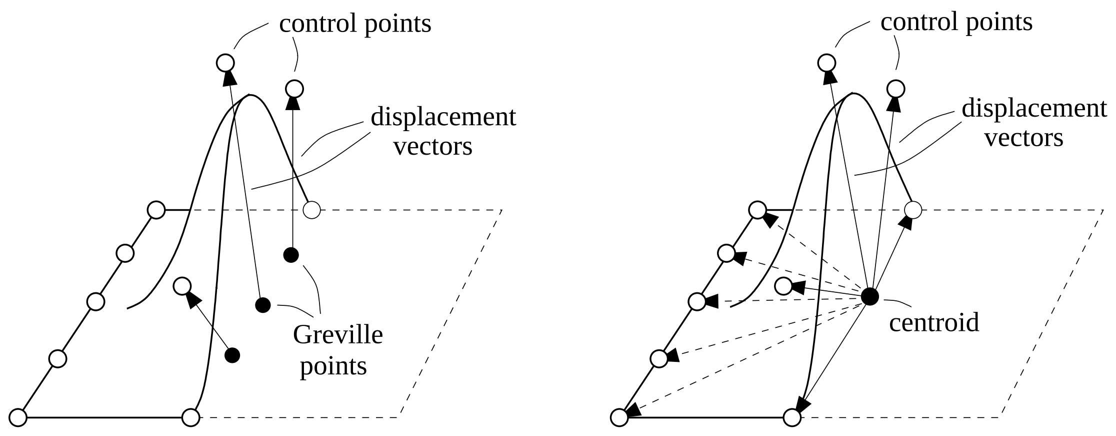


Figure 3.13: Left: a Greville displacement patch. Right: a displacement patch built around a centroid.

I decided not to use the method of extending curves just described because trial curves were lengthened more than I was expecting, often such that the base surface was not large enough for the feature to be pasted on. For now, manual extension of the curves is required.

3.2 Centroid projective patch pasting

3.2.1 Building a patch

I wanted to extend the way cylinders are pasted to pasting patches. In existing surface pasting techniques, patches are represented as Greville displacement tensor product B-spline surfaces. Each control point is calculated from a displacement vector associated with a local diffuse coordinate frame. The feature patch may distort greatly depending on the curvature of the base surface.

In my cylindrical pasting methods, cylinders are built around a spine and pasted in the direction of the selected spine end. I emulate this by building patches around a central point. This centroid is the centre point of the domain embedded in the range. Each control point is calculated from a displacement vector associated with the centroid. Figure 3.13 shows the standard Greville displacement surface and the modified displacement surface.

Leung [19] also uses centroids to displace the interior control points of patches. Our methods differ in how the centroids and their reference frames are constructed: Leung's centroids are computed after boundary control points have been placed,

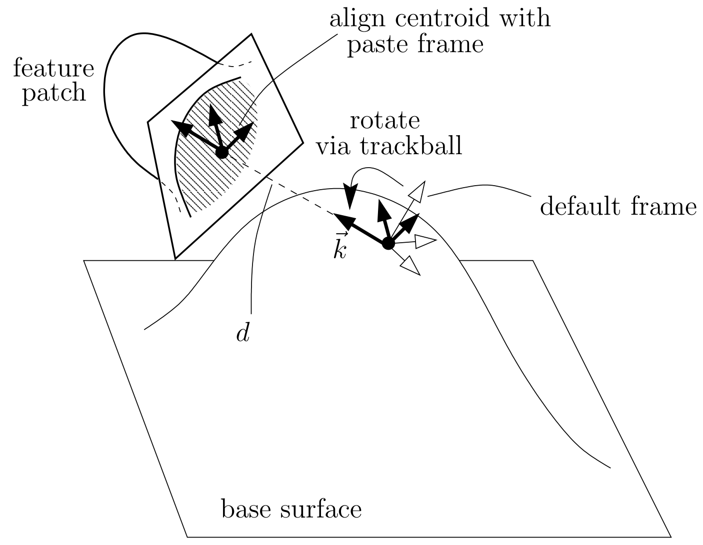


Figure 3.14: User interface for pasting a patch via its centroid

and his reference frames are based on the average planes computed from the pasted boundary control points; the centroids I use determine how patches are pasted, and my reference frames are constructed based on the tangent planes. My methods are described in detail in the next section.

3.2.2 Pasting a patch

Using the centroid, I can paste patches in a similar way to cylinders (Figure 3.14). The user selects a paste point and orients its coordinate frame on the base surface. The centroid is aligned with this coordinate frame. All of the boundary points are projected in the paste direction. The user can select whether to match the boundaries at the control point or the surface Greville points.

The tensor product B-spline surfaces used in this thesis have full end knot multiplicity, so the surface Greville points and the control points are the same at the patch corners (i.e., the corners of the patch always lie on the base surface). With a surface Greville point paste, the boundary curve lies closer to the base surface (see Section 4.2 for error analysis).

With a projective bilinear patch paste (implemented for comparison), the four corners of the patch are projected onto the base surface, and the feature domain is mapped onto a quadrilateral in the base domain (see Section 2.3.2 for details). With the centroid projective paste, the centroid is projected onto the base surface and displaced by a distance d. All the border control points of the feature are projected onto the base surface in the direction of the paste frame and the borders of the feature domain are mapped onto curves in the base domain. As with pasting cylinders, the projected lines must intersect the base surface for the paste to be successful. The interior control points of the feature are not mapped with respect to its domain, but displaced according to the pasted centroid. Figure 3.15 shows the pasting process. Only the boundary of the feature domain is mapped to the base domain; the interior of the feature domain has no corresponding mapping in the base domain.

For a control point boundary match, the boundary control points lie on the base surface, which has corresponding points in the base domains. I used these points as control points for the four 2-dimensional trim curves in the base domain.

For a surface Greville point paste, the surface Greville points have mappings in the base domains, whereas the control points do not. I calculate the control points for each of the four boundary curves by multiplying the inverse of matrix N with the Greville point mappings. This is similar to how I create trim curves for a cylindrical paste. If the knot sequence is not the same, then N_u and N_v will be different matrices.

3.3 Trimmed patch pasting

3.3.1 Creating an irregular patch

With the projective centroid method of pasting patches, I can create patches that have irregular borders for pasting. However, tensor product B-spline patches have a rectilinear structure and creating irregular borders and shapes is unintuitive.[1] Often the shape (displacement of the surface from the plane) of the feature is not rectangular, but the rectilinear structure of the feature as a whole is prominent once the feature has been pasted.

A more intuitive way to create an irregular patch is to trim away the unwanted bits. I can specify a trim curve in the feature domain and use OpenGL to create the irregular patch. However, the border of this patch may not lie on a plane, and it is unclear how to paste the feature using standard pasting or my extension.

3.3.2 Pasting an irregular patch

The general idea behind pasting an irregular patch is to use a cylinder to blend the two surfaces together. First, the user has to specify a trim curve in the feature

[1]Based on personal experience only; others' opinions may differ.

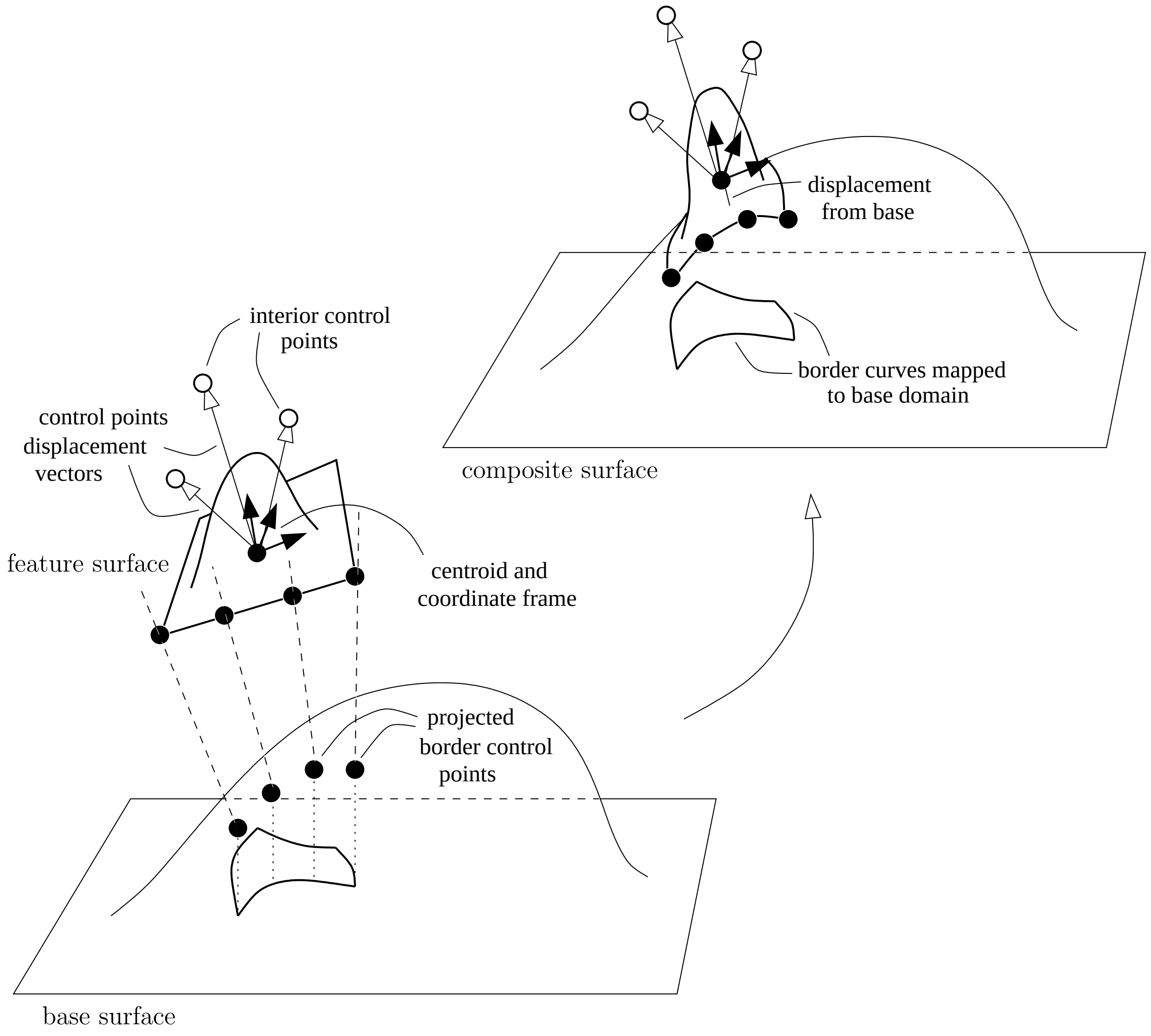


Figure 3.15: The border control points are mapped to curves in the base domain and the interior control points are displaced with respect to the centroid.

domain to create the trimmed patch. The feature is pasted as if I am pasting an untrimmed patch with C^0 continuity. I then trim away the border of the patch and join the trimmed feature boundary to the base surface with a cylindrical paste. Figure 3.16 shows how an irregular patch is pasted.

The cylinder I use for merging the feature with the base has a single segment cubic B-spline curve in the u (spine) direction. Kim and Elber [17] has a similar approach for blending surfaces with specified trim curves that requires more user input. In my implementation, the number of segments in v is calculated based on the size of the feature patch, although there is no real correlation between the two. As with knot insertion, the higher the number of segments, the closer the cylinder's boundary will fit to the trimmed feature. There are a couple of different approaches I can take to achieve the cylindrical blend.

3.3.3 Tangent projective paste

In the first approach, I take evenly spaced samples along the trimmed feature's domain boundary to generate the closed curve of one end of the cylinder. For each sample s, I calculate the vector $\vec{t}$ normal to the boundary curve that also lies on the tangent plane at that point. I then project from s in the direction of $\vec{t}$. If any of the projected lines do not hit the base surface, then a paste is not allowed. If all the lines intersect the base surface, then it is easy to create a cubic blending cylinder with continuity C^0. I take the first ring of control points to be the sample points along the trimmed feature, the last ring of control points to be the intersection points on the base, and the two middle rings to be points between the first and last rings. See Figure 3.17 for an illustration. By construction, the cylinder will join the trimmed feature with approximate C^1 continuity and the base with approximate C^0 continuity. I can also construct a better blending cylinder by taking the above points as surface Greville points, and calculating the control points of the blend cylinder from those using the N^{-1} matrices for the cylinder.

I can get an approximate C^1 paste in a similar manner to projective cylindrical pasting. But first I need to create an outer curve for the blending cylinder so that the second layer of control points can be set to achieve approximate C^1 continuity. This is discussed in the next section.

3.3.4 Extending the boundary for an approximate C^1 paste

For an approximate C^1 continuity paste, I need to project further away from the trimmed boundary. This way I can get a ring of intersection points with the base surface such that the projected feature boundary is inside the ring of points. This

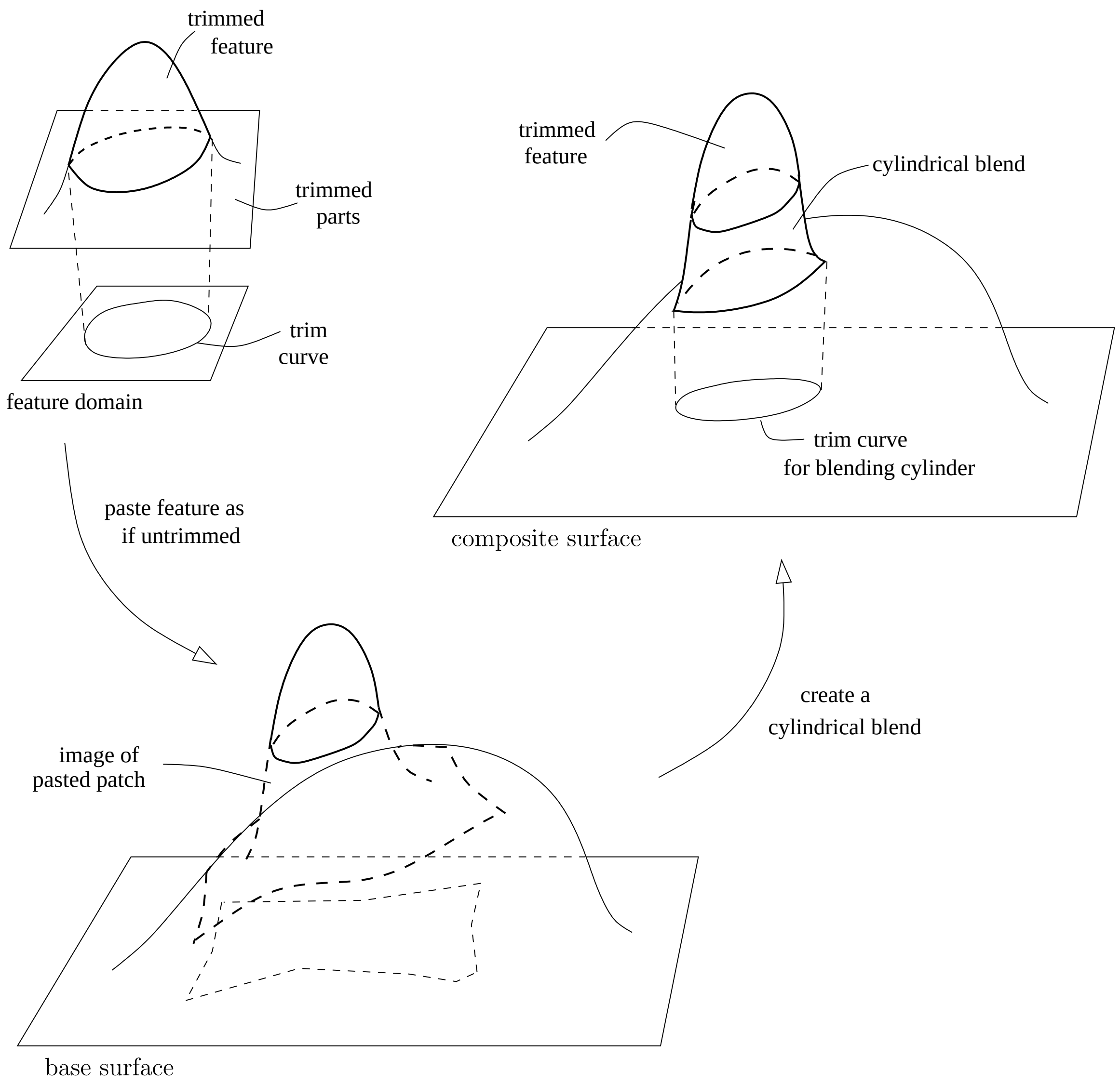


Figure 3.16: Pasting an irregular patch
.

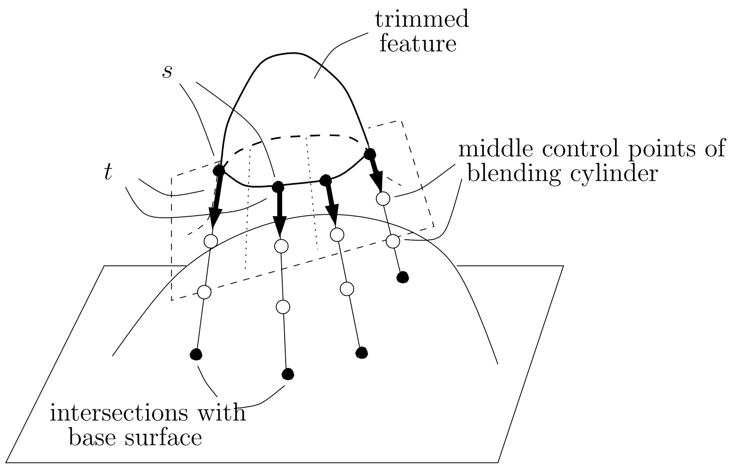

trimmed feature
s
t
middle control points of blending cylinder
intersections with base surface

Figure 3.17: C^0 trimmed tangent projective paste for patches.

lets me manipulate the inner layer of control points to achieve approximate C^1 continuity with the base surface.

There are a few different ways I can project. The user specifies a percentage p, which determines how much further I project from a central point c. One way to project is to extend the projection point from the feature, that is, I take c the centre of the feature domain, s a sample point at the feature trimmed boundary, and project in the direction of $\vec{t}$ (calculated at s) at $s' = (1+p)(s-c)$. However, s' may lie close to the line defined by s and $\vec{t}$, and in this case our ring of points will not be much bigger than the one we get if we do a C^0 paste.

A better way is to project as if we are doing a C^0 paste, and then define s'_b to be $(1+p)(s_b - c_b)$, where s_b is the domain pair for the C^0 intersection at sample s, and c_b is the centre of the feature projected onto the base domain. s' is the corresponding surface point for s'_b, which I use as a point in my ring. The diagram on the left in Figure 3.18 illustrates how each s' is calculated.

Another alternative is to calculate the projection in world space. I take s to be the sample point on the trimmed boundary, c to be the closest point to s that lies on the line defined by the feature centroid and the paste direction. I calculate s_p to be $(1+p)(s-c)$ and project in the direction $\vec{t}$ from s_p to get s', as shown in Figure 3.18 right. I implemented these last two methods for comparison in my thesis.

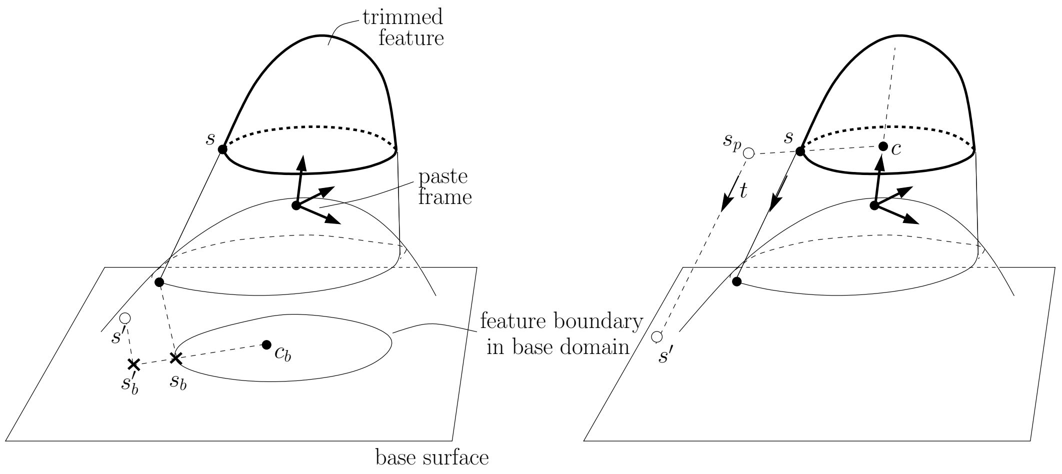


Figure 3.18: Sample points for the approximate C^1 boundary curve can be found by extending points in the domain and calculating the corresponding surface point (left), or by extending points directly in range space (right).

3.3.5 Parallel projective paste

The paste achieved with tangent projection depends on the curvature at the trimmed boundary, which may vary a lot if the control points are moved just a little. The parallel projective method avoids this problem as it does not depend on the curvature. This methods works similarly to tangent projection, with the exception that instead of projecting lines normal to the feature boundary that lie on the tangent plane at each sample, I project in the paste direction. Figure 3.19 illustrates how the points used to construct the blending cylinder are created.

For a C^0 paste, the blending cylinder meets with approximate C^0 continuity at both the feature and the base. For an approximate C^1 paste, the trimmed boundary must be extended to be a bit wider, and this is done as discussed in the previous section. The two middle layers of control points must be set so that the blending cylinder meets both the feature and the base with approximate C^1 continuity.

This is how I determine the location of the middle points. For each sample point s_i on the feature boundary, there is a corresponding point b_i on the base surface found by projecting. I find the midpoint m_i between these two points, and intersect the line defined by m_i and the paste direction with the tangent planes to s_i and b_i. The middle control points of they cylinder are placed at the intersection points to give a smooth blend.

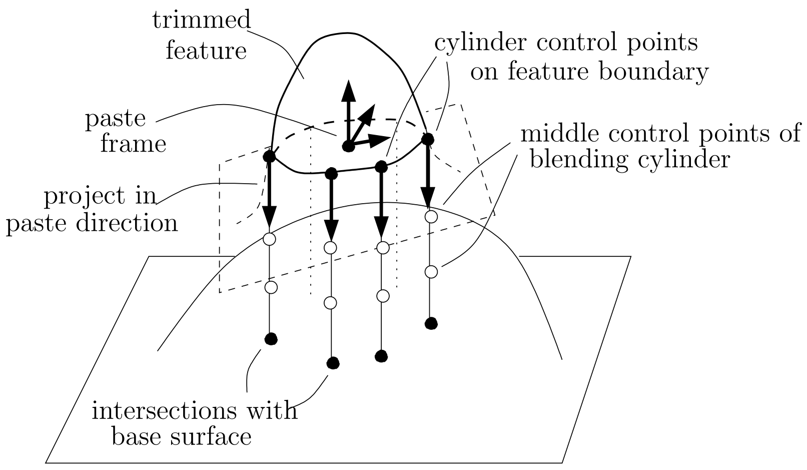

trimmed feature
cylinder control points on feature boundary
paste frame
middle control points of blending cylinder
project in paste direction
intersections with base surface

Figure 3.19: C^0 trimmed parallel projective paste for patches.

3.4 Hierarchical pasting

One of the appeals of surface pasting is that a surface can be augmented repeatedly by pasting on features in a hierarchical manner. With patch pasting, feature domains are transformed onto base domains, so that the respective domains can be embedded in one domain space, and the relation between features and bases forms a directed acyclic graph.

When cylinders are added, one domain space is no longer enough. Cylinders are pasted onto base domains by their ends, which represents one edge in their rectangular domain. Both ends of a cylinder can be pasted as well, so a simple tree or dag structure is inadequate. In my implementation, I use a tree structure to represent the drawing order of the surfaces. At the same time, I store the base and feature relationships. Figure 3.20 shows a set of pasted surfaces, the connection between the domains of the bases and features, as well as the parent/child relationships.

For a discussion on the issues in allowing for more complicated pasting (such as multiple bases and two ends of a cylinder sharing the same base), please see Section 5.5.

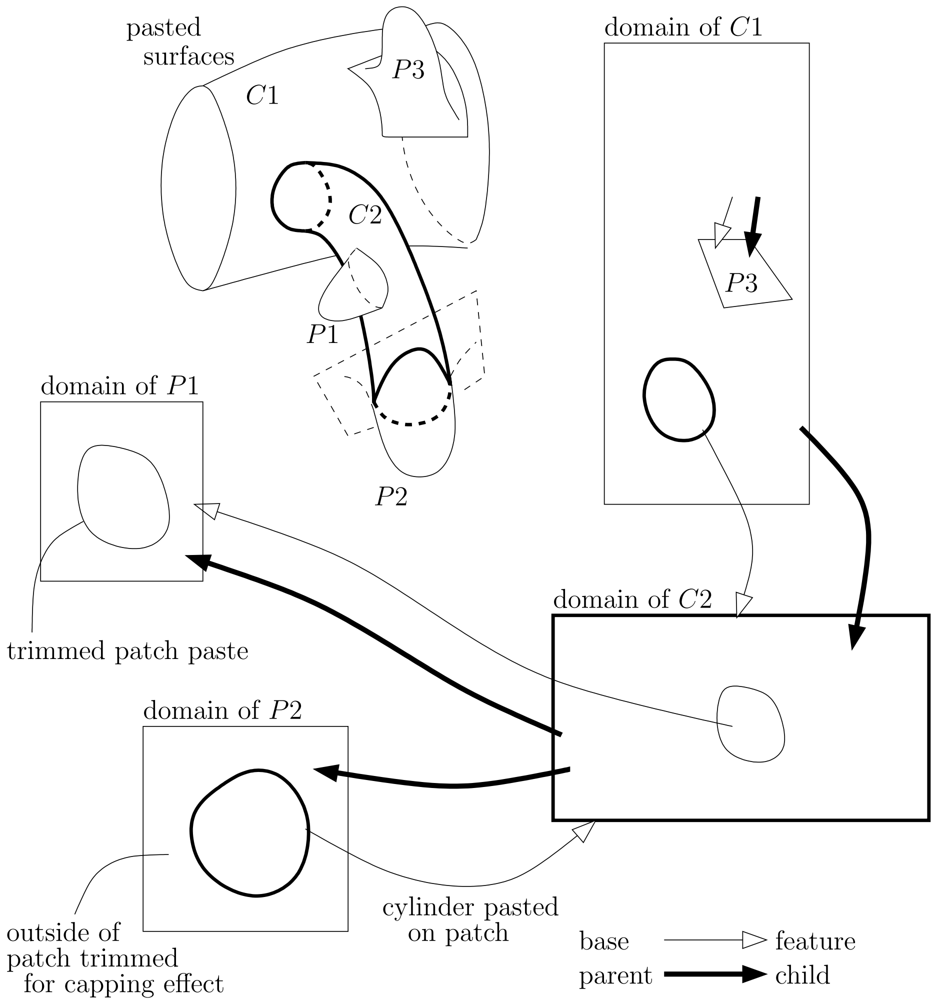


Figure 3.20: An illustration of the various relationships between the feature and base surfaces, compared to the actual child and parent relationship of the paste tree.

Chapter 4

Making a better ferret

In this chapter, I demonstrate the tools I have created. First, I look at the different extensions of pasting. Then I show the improvements in approximation made by using a surface Greville point paste instead of a control point paste. Finally, I illustrate step by step how I build a ferret. With the ferret building tools, I can also create toys for my ferret as well as other models, which I show at the end of this chapter.

4.1 Illustration of pasting techniques

The first extension to surface pasting for better ferret building involves a way to add limbs. Figure 4.1 shows a generalized cylinder before and after it was pasted. On the left, lines parallel to the paste direction in the paste frame project from the surface Greville points. On the right, the resulting approximate C^1 surface Greville paste is shown. The smoothness of the approximate C^1 paste is determined in part by how far apart the user makes the two boundary layers of control points. In the figure, the control points on the outermost layer are not at equal distance from the second layer, so that part of the paste is smoothed out more gradually. Figure 4.2 shows the domain of the base surface with the projected boundary of the pasted feature. With projected cylindrical pasting, the domain curve is not limited to a circle.

In Figure 4.1, the inside of the base surface where the feature is pasted has been trimmed away. If we trim away the outside of the base, we can cap our cylinders. Programs such as Houdini use a simple algorithm for capping surfaces. These simple algorithms are effective for capping cylinders formed by convex curves, but do not work well if we want smooth caps for concave curves. Figure 4.3 shows

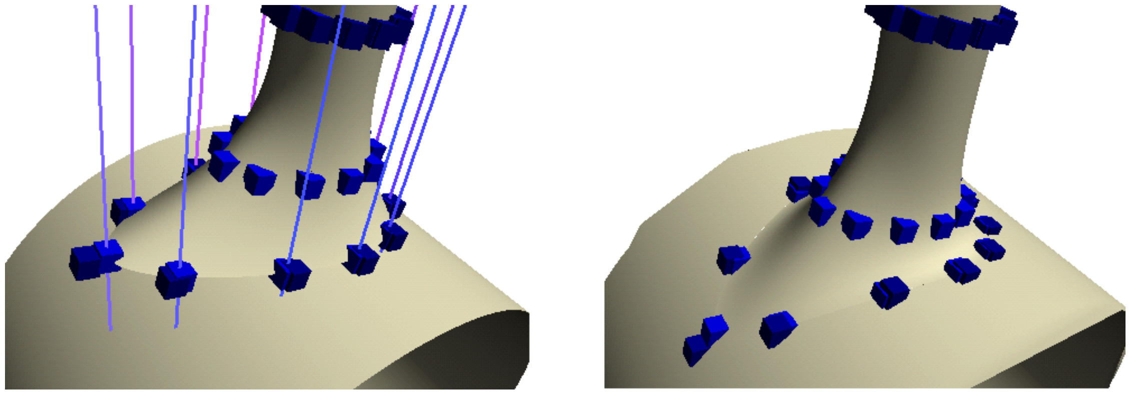


Figure 4.1: Left: the projected cylinder. Right: the cylinder pasted with approximate C^1 continuity

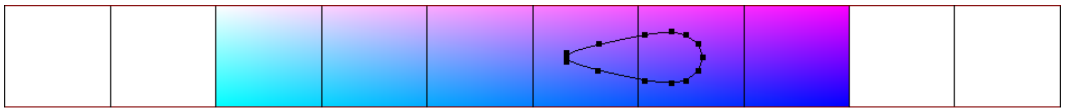


Figure 4.2: Domain of the base surface. The closed curve shows where the boundary of the projected cylinder lies and where the base surface is trimmed.

capping in Houdini. A non-convex, ferret shaped cylinder was created and capped with a faceted flat cap and a smooth tangential cap. The flat cap is represented by one non-convex polygon. The smooth cap is created by taking the centroid of the curve and extending it to the boundary, which causes overlaps when the curve is non-convex. Figure 4.4 shows capping done with my program. The non-convex cylinder on the left is pasted onto a patch, and the outside of the patch is trimmed away. A C^0 paste gives a similar result as a faceted flat cap, and an approximate C^1 paste gives a cap with a smooth join. The trim curve for the cap is shown in Figure 4.5. It is possible to create a cap for a non-convex cylinder in Houdini using other methods. However, more steps are needed than just paste and trim.

As a first step to allow for arbitrary shaped features, I extended patch pasting to a centroid projective paste. This method calculates a centroid for the feature patch and projects the feature boundary control points onto the base surface. Figure 4.6 compares the traditional bilinear projective paste with my method. The feature surface distorts more with a bilinear projective paste, since each interior point has its own local coordinate frame that follows the curvature of the base surface. In a bilinear projective paste, only the corners of the patch are projected, so the domain boundary is always a quadrilateral (Figure 4.7 left). With the centroid

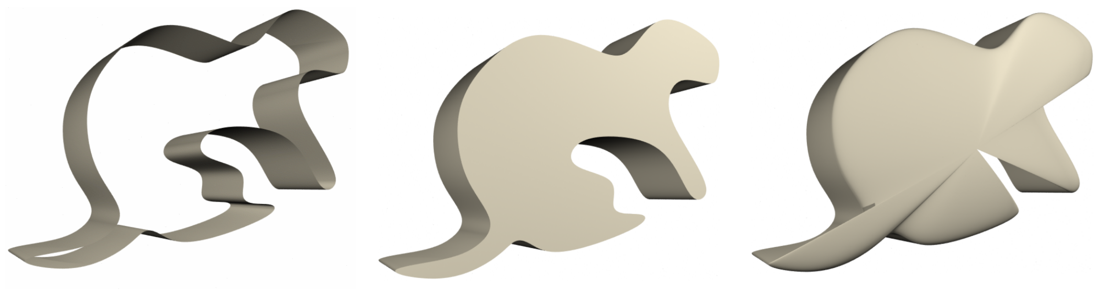


Figure 4.3: Capping a non-convex cylinder in Houdini. From left to right: the cylinder; a polygon cap; a tangential cap.

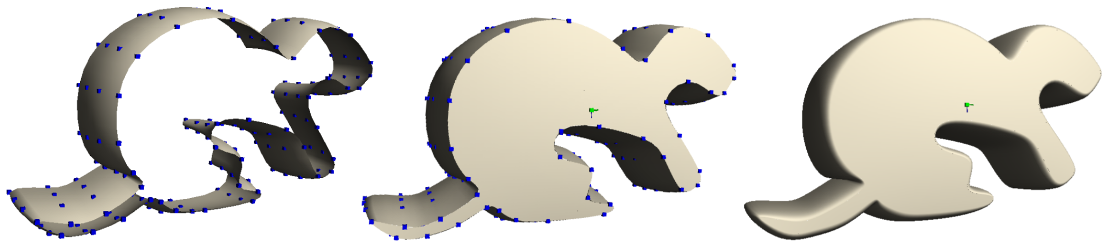


Figure 4.4: Capping a non-convex cylinder by pasting. From left to right: the cylinder; a C^0 cap; an approximate C^1 cap.

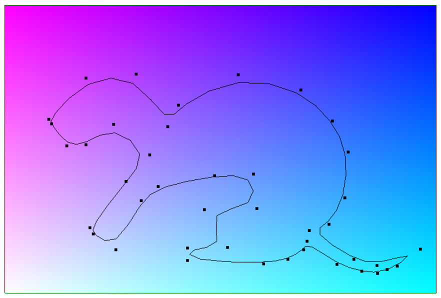


Figure 4.5: The boundary for the non-convex cylinder feature is shown in the base cap.

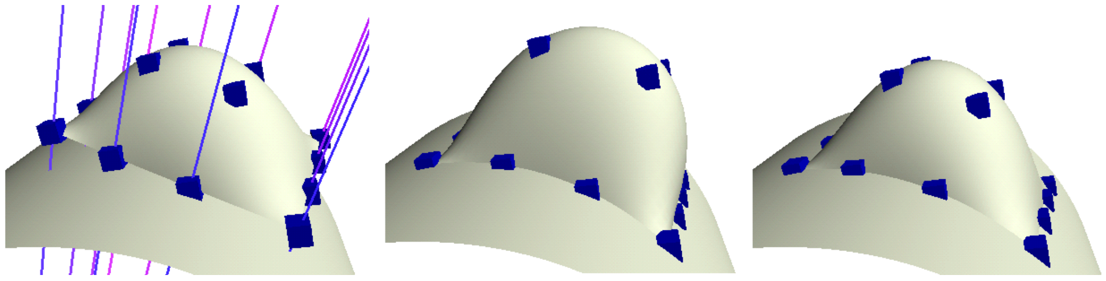


Figure 4.6: From left to right: unpasted surface; C^0 bilinear projective paste; C^0 centroid projective paste.

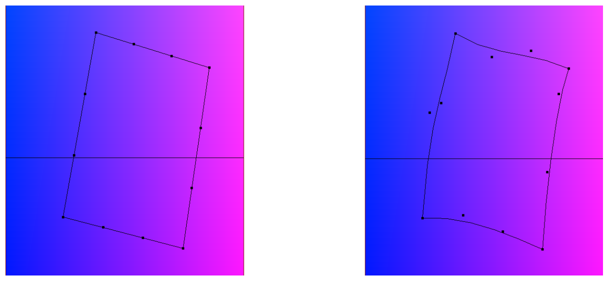


Figure 4.7: Left: the domain boundary of a bilinear projective paste is always a quadrilateral. Right: the domain boundary of a centroid projective paste is formed by four B-splines.

projective paste, the boundary control points can be moved, and each boundary point is projected onto the base. The resulting domain on the right in Figure 4.7 is bounded by four B-splines curves.

Moving boundary control points to create arbitrary shaped features is not an intuitive method. Hence I introduced the pasting of trimmed features. In trimmed features, a closed curve in the feature domain (Figure 4.9, left) defines the trimmed region of the feature patch (Figure 4.8, left). The feature can be pasted via a tangent projective paste (Figure 4.8, centre) or a parallel projective paste (Figure 4.8, right). From the domain curves shown in Figure 4.9 (centre and right), we can see that the curve with a tangent projective paste may resemble a rectilinear shape more closely than a parallel projective paste. This is partly due to the sensitivity of the projection to the curvature at the boundary of the trimmed feature.

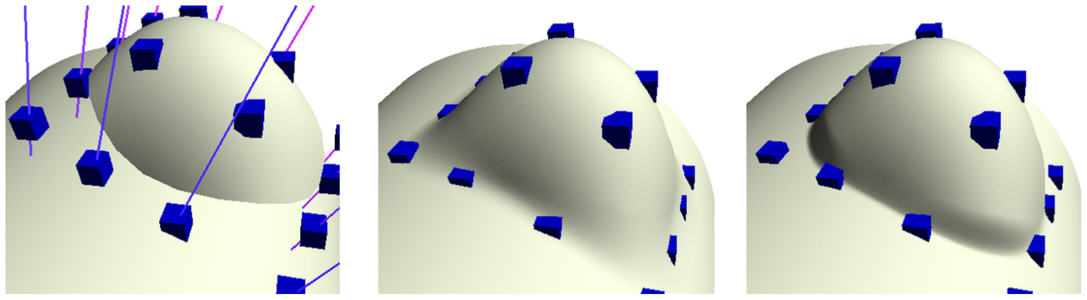


Figure 4.8: From left to right: unpasted trimmed surface; approximate C^1 tangent projective paste; approximate C^1 parallel projective paste.

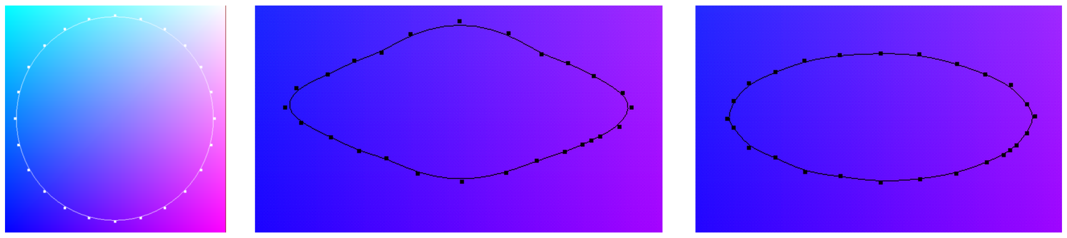


Figure 4.9: Domains for trimmed features. From left to right: unpasted feature is trimmed with circle in the domain; boundary of feature on base domain for a tangent projective paste; boundary of feature on base domain for a parallel projective paste.

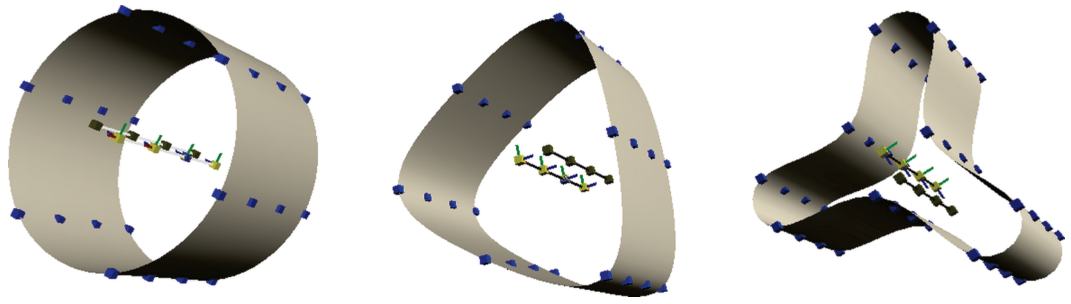


Figure 4.10: Cylinders used for error comparison. From left to right: a standard cylinder; a convex cylinder; a non-convex cylinder

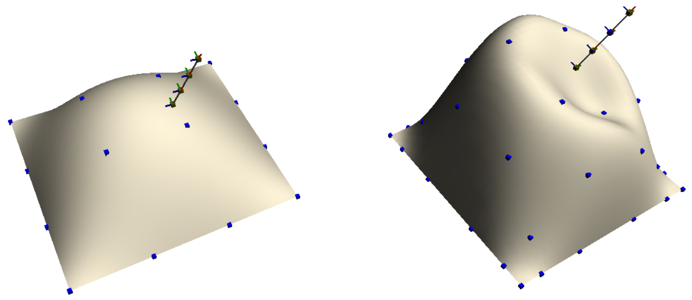


Figure 4.11: The bases used for error calculations are a bump and a bump with an indentation. The spine shows how the test cylinders are pasted.

4.2 Comparison of surface Greville point and control point paste

In this thesis, I implemented a simple improvement to the boundary approximation of pasted surfaces. My basic idea is that by ensuring surface Greville points of the feature lie on the base surface, the feature boundary would be closer than the base. This section describes a few simple tests used to compare the two methods.

The two types of pasting were implemented for both cylinders and patches, but we only need to compare the boundaries for cylinders. I used three test cases for cylinders (Figure 4.10): a standard cylinder; a cylinder with a convex boundary curve; and a cylinder with a non-convex boundary curve. These cylinders are pasted on two different bases (Figure 4.11): a bump and a more complex bump with an indentation. The cylinders are all pasted at the same location and paste direction. Note that when we paste on a flat plane, the error for both a surface Greville point paste and a control point paste is 0. Initially, the cylinders have 6, 6, and 9

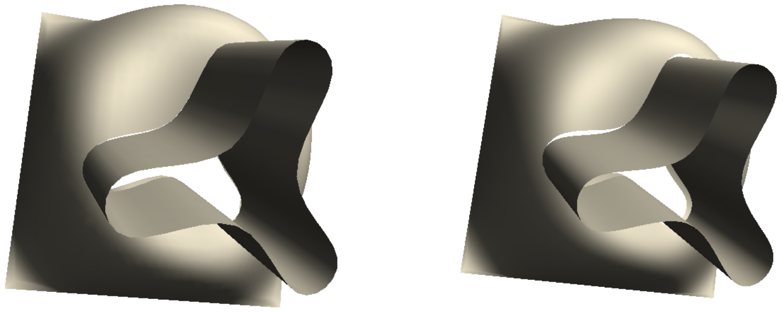


Figure 4.12: A non-convex cylinder pasted on a bump with a surface Greville point paste (left) and a control point paste (right).

control points respectively, not including duplicate control points. At each level of testing, the number of control points is doubled so we can look at improvement via knot insertion as well. I took 960, 960, and 1440 samples along both the feature boundary and the base trimmed boundary to compute the minimum, average and maximum differences between the two curves. The number of samples taken was chosen so that there are at least 20 samples for each unique control point on the feature boundary. Figure 4.12 shows the non-convex cylinder pasted on the more complex bump. Note the larger gaps in the control point paste.

Tables 4.1 to 4.6 show the comparison results. The minimum is not shown as the surface Greville point paste always yields a minimum difference of 0, since by construction the feature's surface Greville points always touch the base surface. Our main interest is in the maximum difference between the two boundary curves, and the ratio of the maximum before knot insertion compared to after knot insertion. From the tables, we can see that surface Greville point paste is always an order of magnitude better than a control point paste, and is comparable to doubling the number of knots. Further, control point paste appears to have order two convergence. However, there is no discernible pattern for surface Greville points.

With a surface Greville point paste, we construct the matrix N and its inverse, and compute the control points from given surface Greville points. For a degree e boundary with m control points, computing N and N^{-1} is an $O(m^3)$ operation, and needs to be calculated once per knot sequence. N is a sparse matrix, and the runtime for calculating its inverse can be improved. Computing the control points is an $O(me)$ algorithm, as each control point is weighed by $e + 1$ surface Greville points with entries from N^{-1}. Surface Greville point paste is a more expensive method of pasting, but it is simple to implement, and keeps the number of control points needed small.

Table 4.1: Cylinder on bump

	Surface Greville paste			Control point paste		
Level	Average	Max	Ratio	Average	Max	Ratio
0	0.010380	0.034020	-	0.143450	0.168916	-
1	0.000829	0.003155	10.782884	0.030356	0.043508	3.882412
2	0.000333	0.001072	2.943097	0.007250	0.010913	3.986805
3	0.000108	0.000407	2.633907	0.001808	0.002704	4.035873

Table 4.2: Cylinder on bump with indentation

	Surface Greville paste			Control point paste		
Level	Average	Max	Ratio	Average	Max	Ratio
0	0.039490	0.134165	-	0.115000	0.215616	-
1	0.005553	0.032865	4.082306	0.028202	0.109243	1.973728
2	0.000176	0.001033	31.815102	0.007297	0.031621	3.454761
3	0.000010	0.000202	5.113861	0.001896	0.009200	3.437065

Table 4.3: Convex cylinder on bump

	Surface Greville paste			Control point paste		
Level	Average	Max	Ratio	Average	Max	Ratio
0	0.013207	0.028753	-	0.170666	0.241144	-
1	0.001092	0.003523	8.161510	0.032933	0.058518	4.120852
2	0.000100	0.000492	7.160569	0.007818	0.015486	3.778768
3	0.000067	0.000567	0.867725	0.001934	0.003780	4.096825

Table 4.4: Convex cylinder on bump with indentation

	Surface Greville paste			Control point paste		
Level	Average	Max	Ratio	Average	Max	Ratio
0	0.044435	0.182811	-	0.395789	0.970510	-
1	0.009100	0.051241	3.567670	0.034223	0.122227	7.940226
2	0.000432	0.002869	17.860230	0.008242	0.038473	3.176955
3	0.000402	0.002787	1.029422	0.002112	0.009769	3.938274

Table 4.5: Concave cylinder on bump

	Surface Greville paste			Control point paste		
Level	Average	Max	Ratio	Average	Max	Ratio
0	0.012482	0.039657	-	0.106587	0.185383	-
1	0.000750	0.003290	12.053799	0.016614	0.032521	5.700409
2	0.000125	0.000635	5.181102	0.003776	0.007609	4.274018
3	0.000034	0.000196	3.239796	0.000928	0.002461	3.091833

Table 4.6: Concave cylinder on bump with indentation

	Surface Greville paste			Control point paste		
Level	Average	Max	Ratio	Average	Max	Ratio
0	0.037005	0.124538	-	0.142876	0.291727	-
1	0.003162	0.013467	9.247642	0.023322	0.064534	4.520516
2	0.000353	0.001956	6.884969	0.005403	0.015519	4.158386
3	0.000100	0.000717	2.728033	0.001354	0.004043	3.838486

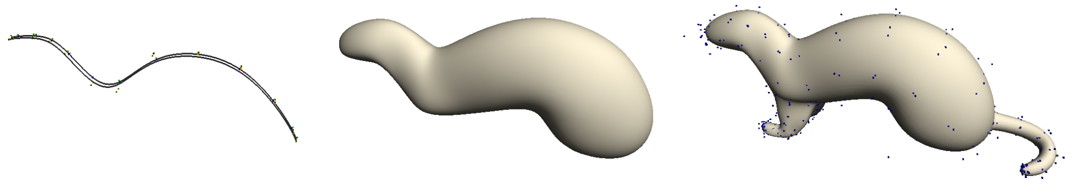


Figure 4.13: Building a ferret

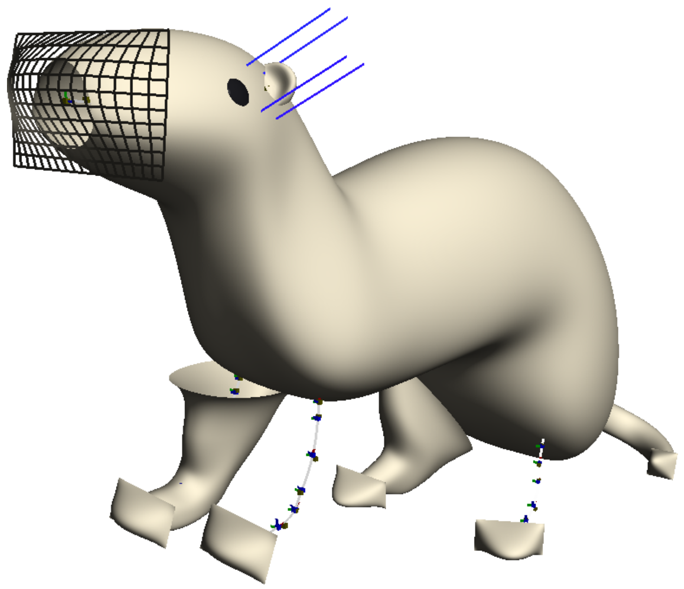


Figure 4.14: The ferret before pasting.

4.3 Building a ferret

The process of building a ferret is illustrated in Figures 4.13 and 4.14. I start by shaping a spline that gives the overall body shape of the ferret (Figure 4.13, left). The cross sections of the generalized cylinder created around this spline are adjusted to fill out the ferret (Figure 4.13, middle). The limbs and tail are generalized cylinders that are pasted onto the body.

Figure 4.14 shows the unpasted features of the ferret body. The legs and tail are generalized cylinders shaped by a central spline. The ends of the cylinders are capped by pasting them on patches and then trimming away the excess. The end of the cylinder forming the head is filled in a similar way. The eyes and ears of the ferret are trimmed patches pasted onto the head. The final ferret is shown in Figures 4.15 to 4.17. A ferret in a standing pose is shown in Figure 4.18. The eyes,

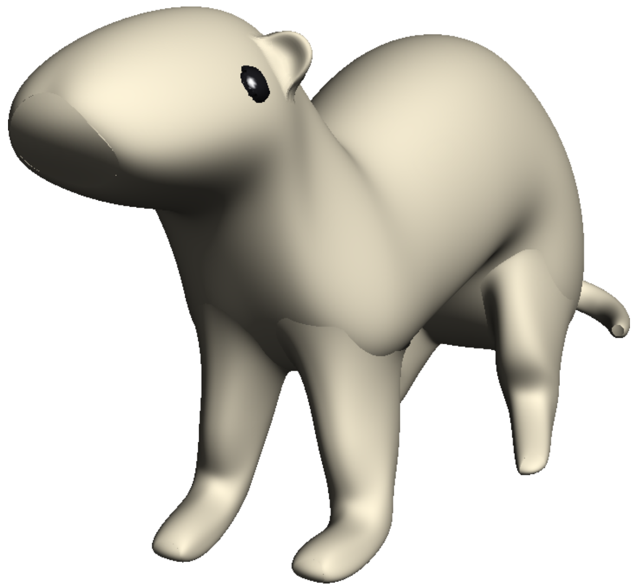


Figure 4.15: The ferret after pasting.

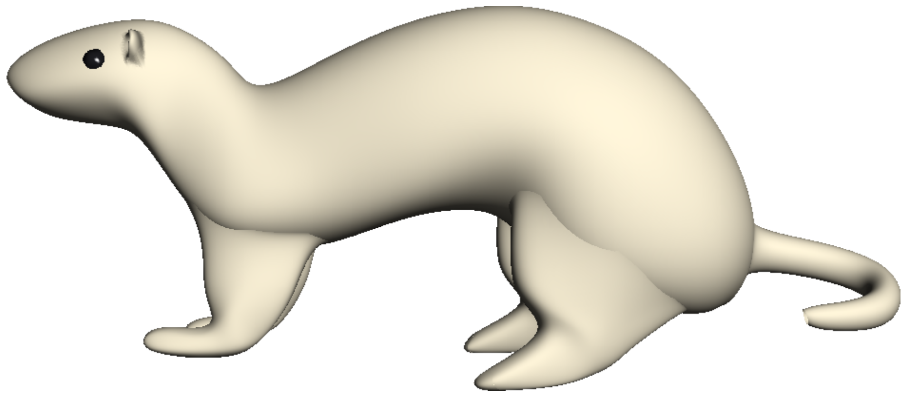


Figure 4.16: The ferret model from the side. Note the creases where the limbs attached to the body. These can be removed by improving the boundary approximation via knot insertion.

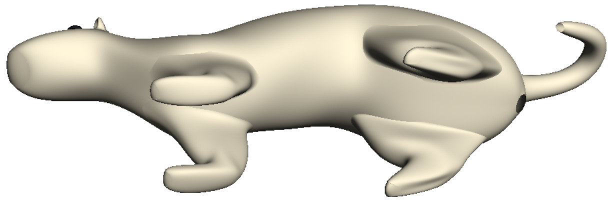


Figure 4.17: View of the ferret model from the bottom.

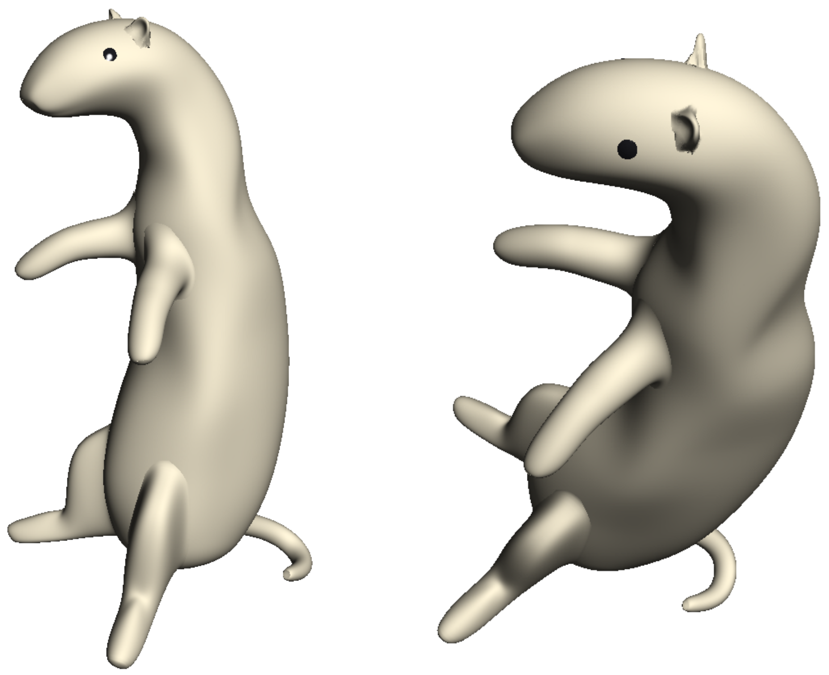


Figure 4.18: A standing ferret.

ears, and tail of the standing ferret are reused features from the previous ferret.

4.4 Other models

In this thesis, I focus on improving the tools for modelling ferrets, but the techniques I have developed can be used to model other less interesting shapes as well. Figure 4.19 shows a bone chew toy for the ferret modelled by capping a cylinder with non-convex ends. Figure 4.20 shows a tree model, built from twisted cylinders pasted together. Figure 4.21 shows a slingshot model. The join at the Y cannot be reduced with knot insertion, because the feature cylinder is so close in radius to the base, and the tangent planes at the side are almost parallel to the paste direction. The fish in Figure 4.22 has a cylinder body and a capped head. The duck in Figure 4.23 is built from several capped cylinders and trimmed patches.

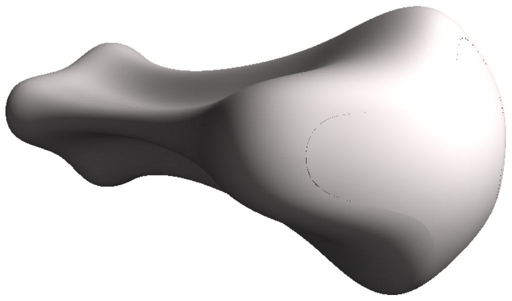


Figure 4.19: A bone chew toy.

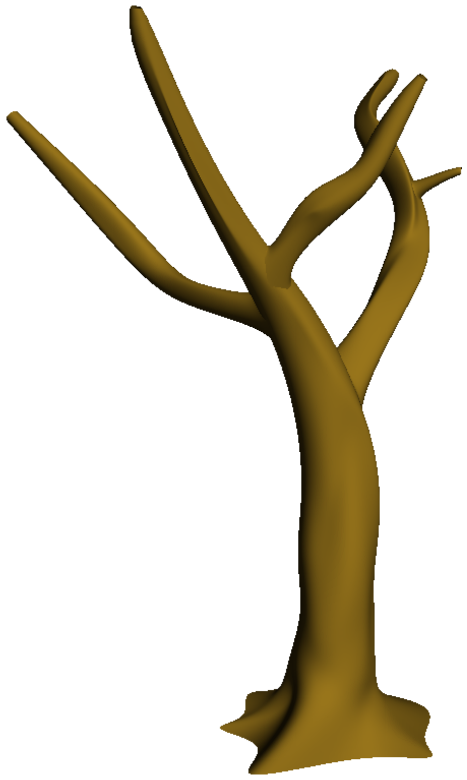


Figure 4.20: A tree.

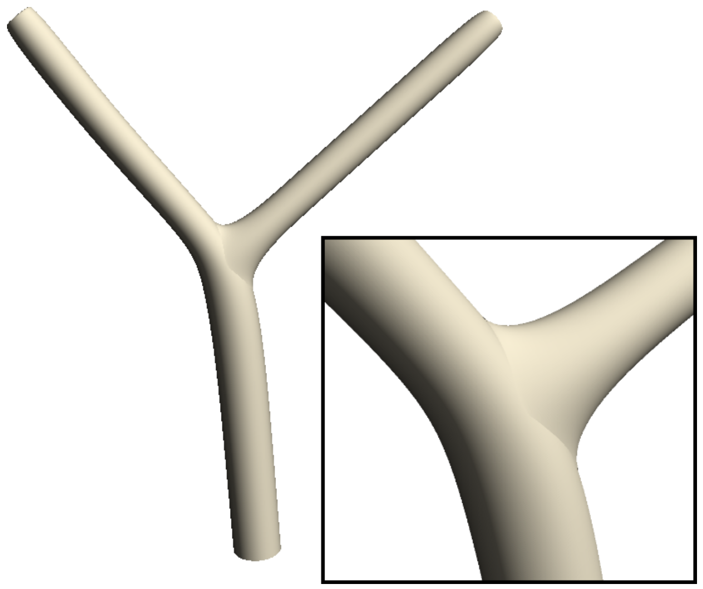


Figure 4.21: A slingshot, requested by Sanjeev Bedi.

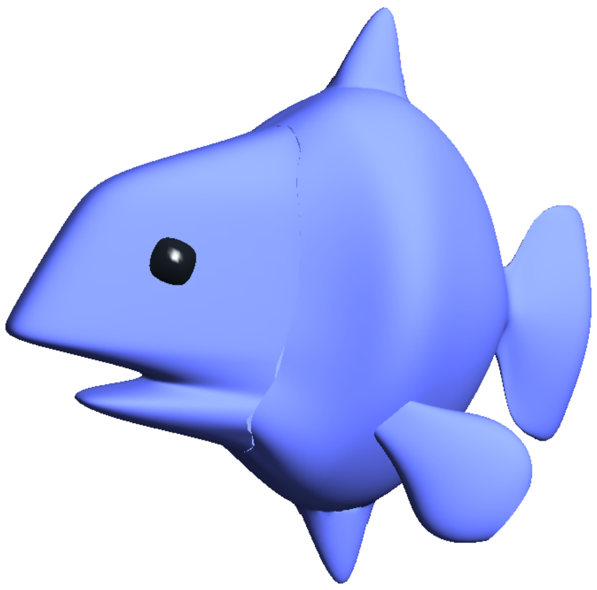


Figure 4.22: A fish.

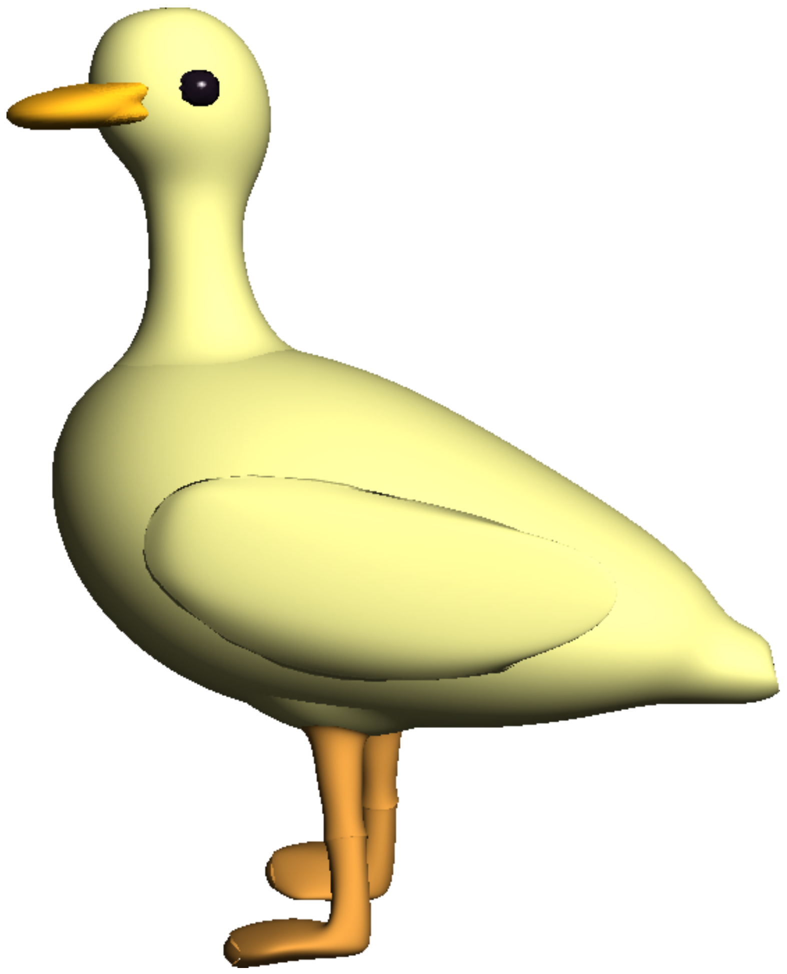


Figure 4.23: A duck, inspired by Steve.

Chapter 5

Ferret on!

The main focus of this thesis was to extend surface pasting to model ferrets more effectively (Figure 5.1). To achieve this goal, I devised a world space user interface for pasting cylinders via a centroid. For better control over warping of features, I extended this method to patches, creating a centroid projective patch paste. I also implemented pasting of arbitrarily shaped features by joining trimmed patches to base surfaces with a cylindrical blend. I made minor improvements to boundary approximations by experimenting with pasting surface Greville points rather than control points.

In this last chapter, I summarize each technique I have devised, discuss their uses and limitations, and present my thoughts on modelling with pasting. As well, I suggest improvements and extensions for future exploration.

5.1 Projective cylindrical pasting

Ferrets have an elongated body, and their arms and legs can be modelled more closely with cylinders than with patches. I used generalized cylinders built around a spline curve to allow easy manipulation of the overall shape of the cylinders. Then I extended domain based cylindrical pasting to world space manipulation to allow more direct pasting of cylindrical features.

Reference frames positioned along a centroid spline curve are used to align cross sections of generalized cylinders. I used three different ways of creating reference frames in this thesis. Frenet frames can be computed independently at arbitrary points, but they may flip after an inflection point, causing violent twists in the cylinder. As well, small changes in the curve may cause the frames to rotate, making it hard to predict how the cross sections will be aligned. With modelling, it

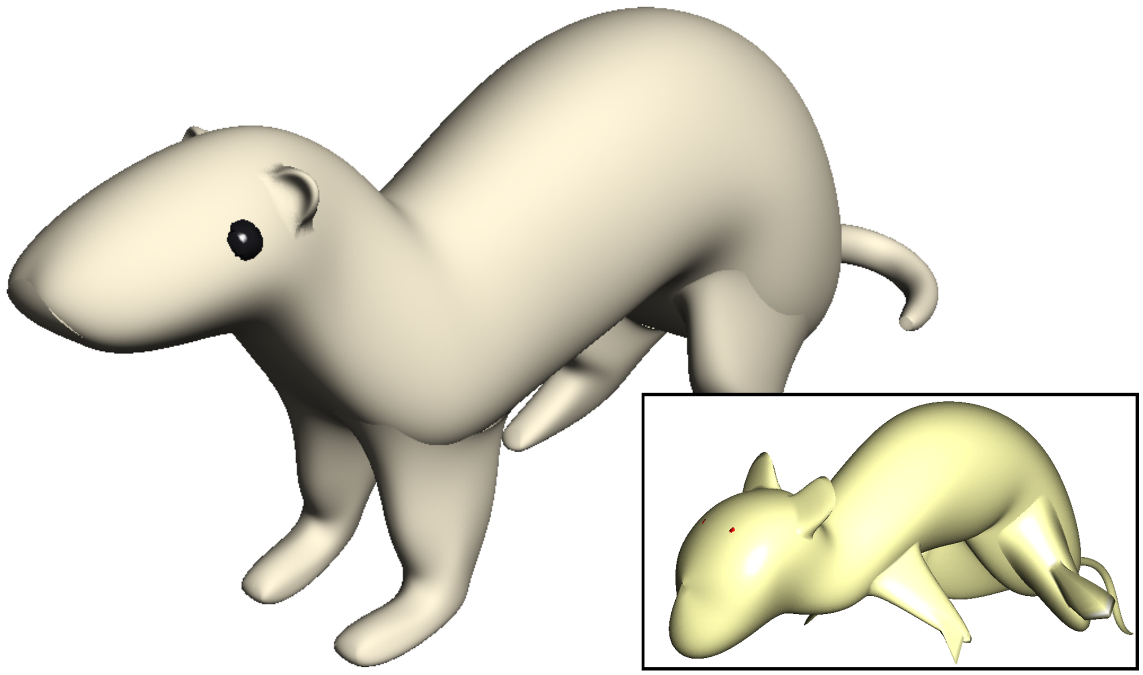


Figure 5.1: Comparison of old and new ferret model.

is important that the orientation of the cross sections do not change drastically with small changes in the centroid curve, so Frenet frames are not feasible. Rotational minimizing frames cannot be computed independently, but eliminate most of the problems with Frenet frames. In particular, small changes to the middle of the curve corresponds to small changes in the reference frames. However, rotational minimizing frames are dependent on an initial frame, which, in my implementation, is a Frenet frame, so small changes to the start of the curve results in drastic changes in the orientation of the subsequent reference frames. To allow for more control of the reference frames, I implemented the double spine, which does not have the above problems. However, editing two curves at the same time is more complicated than editing one, so the interface is not as easy to use as I had hoped. A better scheme for editing the reference frame may be to use rotational minimizing frames and allow the user to orient the initial frame to correct drastic rotations inherent in the creation of Frenet frames.

As with existing patch pasting methods, a world space interface means a projective interface where the feature is projected onto the base and the domain mapping is built from the projection. With generalized cylinders, the centroid curve provides a natural direction for projection. Positioning an object (for projection) in

3D space is a task that requires a lot of manipulation, and I bypass this task by asking users to select a paste point on the base surface from which a paste frame is calculated. The paste frame is easily rotated via a virtual trackball, and matching the feature cylinder to the paste frame allows the user to preview the projection. In the current implementation, the user cannot correct the projection easily if they selected the wrong paste point. Allowing the user to translate the location of the paste point, using techniques described by Chan [8], would make corrections easier. Fixing the paste direction and allowing the user to rotate the paste frame around the paste direction may also help with positioning the paste surfaces.

While a world space projective interface is easier from a user point of view, there are situations where it is not enough. One example is the slingshot model shown in Figure 4.21. The radius of the base cylinder is only slightly larger than that of the projected feature cylinder. Some of the lines projected from the feature boundary intersect at a tangent to the side of the base cylinder, so it is hard to get a smooth approximate C^1 continuity at those locations. In situations such as this, allowing the user to manipulate the trim curves directly may yield a better paste.

Projective pasting also extends the feature edge mapping from circular domain curves to arbitrary shaped closed curves. In particular, I can have cylinders with non-convex cross sections and map them onto non-convex domain curves. Trimming away the outside of the base surface rather than the inside is an effective way to create a smooth cap, and works better than traditional methods for capping non-convex surfaces.

In my implementation, I explored and discarded a method of extending the boundary layer of control points for the user such that the feature surface was extended but the original component was not modified. Instead, I chose to let the user manipulate the boundary manually. I believe that automatic extension of the boundary is more desirable and can be implemented. However, keeping the original feature surface unmodified is unnecessary, since the surface boundary will be changed once the feature is pasted. An implementation where an additional layer of control points is added, similar to that of extending the boundary of trimmed surfaces, and the positions of the original control points are unmodified should be tested. Currently, The user manually makes the boundary curve larger or smaller by scaling it along the curve's normals. Concave curves are less likely to self intersect when scaled in this manner. Extending the boundary of trimmed surfaces along the normal of the boundary curves in this manner should also be investigated.

5.2 Centroid projective patch pasting

Projective patch pasting is undoubtedly a more intuitive interface than domain based patch pasting, for the simple reason that users are manipulating surfaces directly. In current implementations [8, 30], users translate and rotate the feature surfaces in world space to a desired position, which is not an easy task. With pasting cylinders, I implemented an interface where the user has to select and orient a paste point on the base surface. I believe this task is simpler than manipulating a feature surface (the surface may not always rotate around its centre, making it harder to judge how rotation will change its position and orientation) and is equally intuitive. Therefore, I extended this interface to pasting patches, by adding a centroid to feature patches.

I also extended patch pasting in two ways. First, the Greville displacement patches were modified to be centroid displacement patches. When pasting these patches, the centroid, as opposed to the Greville origins, were pasted onto the base. The interior points are displaced with respect to the centroid, so that the feature retains its original shape much better. The disadvantage of this method is that knot insertion will create a larger transition between the boundary points and the interior points, since the number of interior control points changes with each knot inserted but they are always offset from the same location (Figure 5.2). With Greville displacement patches, the displacement offset origins change with knot insertion, so there is a smoother transition.

My second extension was to project all boundary points, rather than just the four corners of the patch. This allows users to create patches with irregular boundaries. I find it is unintuitive to create an irregular shape, such as a star, out of a rectilinear surface. It may be that the editing operations in my program are limited, but the task is not any simpler in a package such as Houdini. My extension was a first step towards pasting of irregular shapes, but it was not a practical method.

5.3 Trimmed patch pasting

A more practical method to create irregular shapes is to use trimmed patch pasting. Commercial packages (such as Houdini) have operations that allow users to extrude rectilinear patches to create irregular bumps easily (an example is extruding fonts). We can paste such patches directly, but the rectilinear boundary is usually visible. By trimming the patch around the feature bump and blending to the base surface with a cylinder, I can get a paste boundary that is much closer to the shape of the bump.

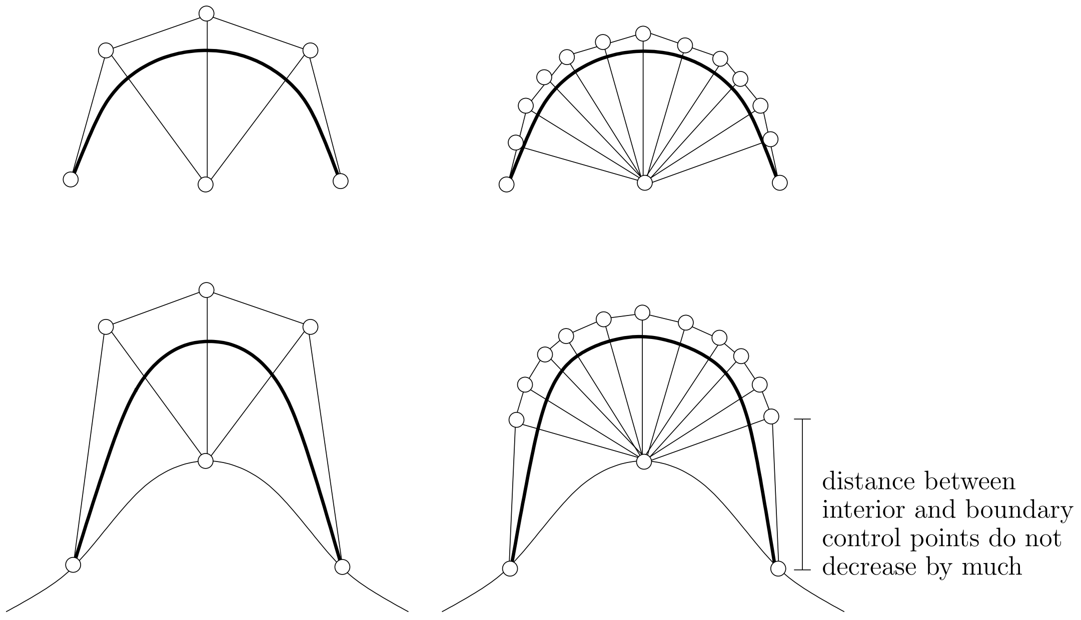


Figure 5.2: The transition between boundary and interior control points does not adjust proportionally to the number of knots inserted.

In my current implementation, there are two different ways to create this blend. With a tangent projective paste, the cylinder is extended along the curvature at the trimmed boundary and blended onto the base surface. The projected boundary may not resemble the trimmed boundary because the projected direction is along the trimmed feature surface, and the feature's curvature may vary along the boundary.

Because of the boundary sensitivity of the tangent projective paste, I implemented a parallel projective paste, where the blend cylinder is extended along the paste direction of the feature. This method gives a paste where the projected boundary is similar to the trimmed boundary. The disadvantage of this method is that a C^0 paste yields two C^0 boundaries at each end of the trimmed cylinder. This is not a concern, since we prefer approximate C^1 continuity for modelling. However, higher levels of continuity should be added for completion. C^2 continuity in a parallel projective paste may give the blend cylinder an overall smoother looking join, rather than just local smoothness at the boundaries. For now, a smoother look can be achieved with a larger paste radius.

There are a number of advantages to trimmed patch pasting. First, we can paste features with irregular boundaries. Second, we do not have to worry about boundary continuity when creating the features. In standard pasting, we need to leave the outer two layers of control points unmodified if we want to achieve approximate C^1 continuity, but with trimmed pasting, continuity is handled by the blend cylinder after trimming. In surface pasting, we generally want to paste features of higher complexity onto a simpler base surface, or knot insert in the feature to improve the paste. This means that features lower in the hierarchy must necessarily have higher knot counts. With trimmed patch pasting, we do not need to worry about the complexity of the feature or base, since the paste can be improved by knot insertion in the blend cylinder only.

One limitation in my implementation is the lack of tools for editing the trim curve. Ideally, the user should be able to manipulate the trim curve directly on the feature surface, and perform knot insertion in the curve as well. Direct manipulation of the trimmed boundary will also make the tangent projective paste more appealing, as users can adjust the boundary to change the projected direction.

5.4 Approximation improvements

Surface pasting is an approximation technique, so reducing the boundary gap between feature and base is useful. I investigated an improvement made by pasting the feature boundary surface Greville points, rather than control points, onto the base surface. Intuitively, the improvement makes sense. The boundary surface Greville points are points on the feature boundary, so by making sure they lie on the base surface the boundary approximation should be better than pasting control points. In practice, this is also the case.

In addition to a better boundary approximation, this method also allows us to paste larger features on the base. Surface Greville points always lie on the surface, whereas the control points may lie quite far from the surface boundary. Therefore, lines projected from the control points will need a larger area in the base to project onto.

However, there are a few concerns with this method. While there is a unique mapping for each set of control points to its corresponding surface Greville points, there is no guarantee on how the control points will change if we change the surface Greville points. Specifically there is no guarantee that the new projected curve will resemble the original curve. This method is also not efficient. For a cylinder with n control points in its cross sections, I have to create an $n \times n$ matrix and calculate its inverse. Calculating the control points after retrieving the positions of the surface Greville point is also an $O(n^2)$ operation. It may be more worthwhile to implement existing approximation techniques [9, 10] into cylindrical pasting.

Because surface pasting is an approximation technique, and rendering spline surfaces is expensive, I feel that improving the boundary approximation serves limited purposes. Current approximation techniques may reduce the gaps between the feature boundary and the base, but do not necessarily remove them. I believe that we should convert pasted surfaces into gapless meshes for rendering, and use surfaces at lower resolution, with fewer knots for interactive pasting.

5.5 Hierarchical pasting

For simplicity's sake, my implementation of the hierarchical pasting structure is a tree, which limits the objects I can model. One particular limitation is that I cannot paste on the blend cylinders of trimmed features.

A more general hierarchical structure should be implemented. With patch pasting, the pasting hierarchy can form a directed acyclic graph, where a feature can be pasted on overlaps in a composite base. With cylinders, a general graph hierarchy should be possible. Both ends of a cylinder can be pasted, and there is no reason why one end of a cylinder cannot be pasted onto itself, forming a loop. This allows modelling of objects with higher topology.

A first step towards a general graph hierarchy that allows self looping would be to investigate data structures for storing graphs efficiently. We also have to consider how to modify cylinders so they can paste onto themselves; a technique presented by Bartels and Beatty [4] may be a good place to start for extending generalized cylinders for pasting loops. There will also be areas on the cylinder where you cannot paste a particular end because it will result in recursive updates. Methods for direct manipulation where a general graph hierarchy is used can be extended based on what is presented by Ma [20, 21].

5.6 Modelling and editing

I implemented limited editing capabilities in my surface paster to allow creation of different features. Translation of selected points is allowed in directions parallel to the view plane. I also allowed translation of control points in the direction of their normals at their surface Greville points so bumps can be created quickly and concave curves can be scaled along their normals for approximate C^1 pasting.

The limited range of editing tools makes me appreciate the complete interface offered by commercial packages such as Houdini, which allows complex manipulation of points. Generalized cylinders make it easier to model and manipulate elongated forms, and it maybe useful to investigate pasting of surfaces with more complex spines similar to skeletal elements used in implicit surfaces [7].

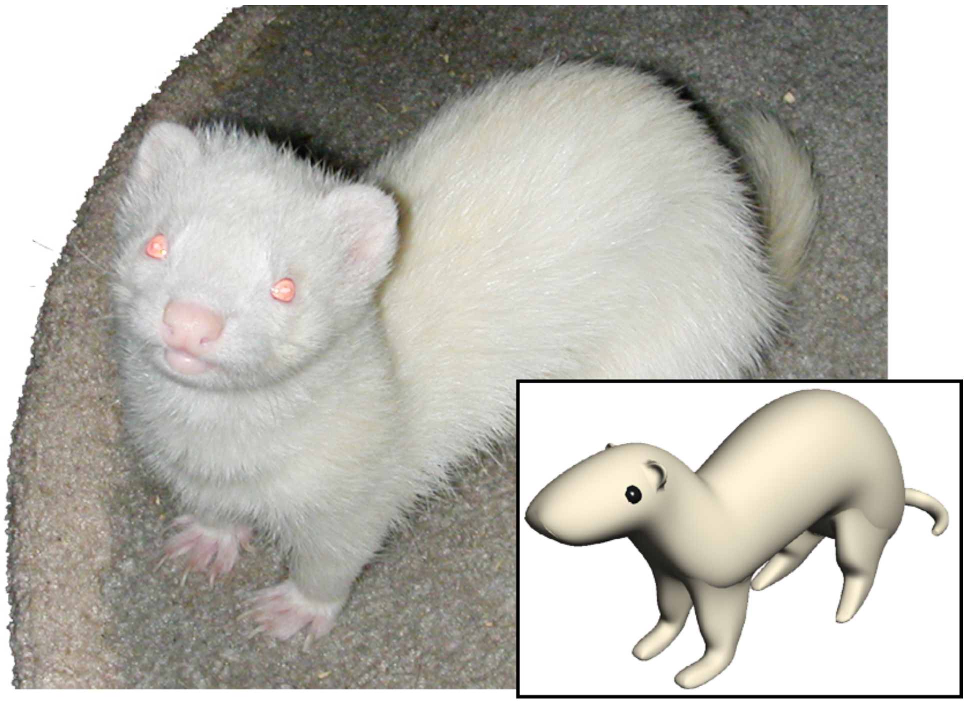


Figure 5.3: The model I created is still far from being similar to an albino ferret like Sprite.

We should also extend pasting to allow features to be pasted on a larger variety of bases. We can approximate any curve on a smooth surface with a B-spline curve, with a better approximation achieved by a larger number of control points. A feature can be pasted on any smooth surface, allowing us to add features to surfaces created by deformation and other methods.

5.7 Ferret on!

This thesis is a first step in creating the digital ferret. I have shown that surface pasting can be an effective method to model the shape of a ferret. We need to investigate the use of textures and fur in surface pasting to properly model sables, cinnamons, silvers and albinos (Figure 5.3). Animating our pasted ferrets to perform the ferret war dance would be the last step needed to bring the digital ferret to life.

Ferrets rule!

Appendix A

Ferret Paster

In this appendix, I describe the basic operations provided by my implementation, *Ferret Paster*. *Ferret Paster* runs on both SGI IRIX and Linux machines.

A.1 Modes and menus

Figure A.1 shows the layout and menu structure of *Ferret Paster*. I assume the use of a three button mouse in my program.

The **File** menu consist of the following commands:

Load Model allows the user to load in an existing file via a dialog. Files used in this program have the extension *.dook*.

Save Model allows the user to save the current model to a file via a dialog.

Save Image saves the contents of the current viewport to a *.png* file via a dialog.

Quit exits the program.

The **View** menu allows the user to manipulate their viewing position.

Reset View Position switches to the default viewing position.

Reset View Orientation switches to the default orientation of the world origin, which looks down the negative z axis.

- X Y plane switches to the default orientation of the world origin to looking down the positive z axis. This and subsequent menu items for the **View** menu are useful for setting the viewport to axis aligned orientations so control points can be manipulated parallel to the axes.

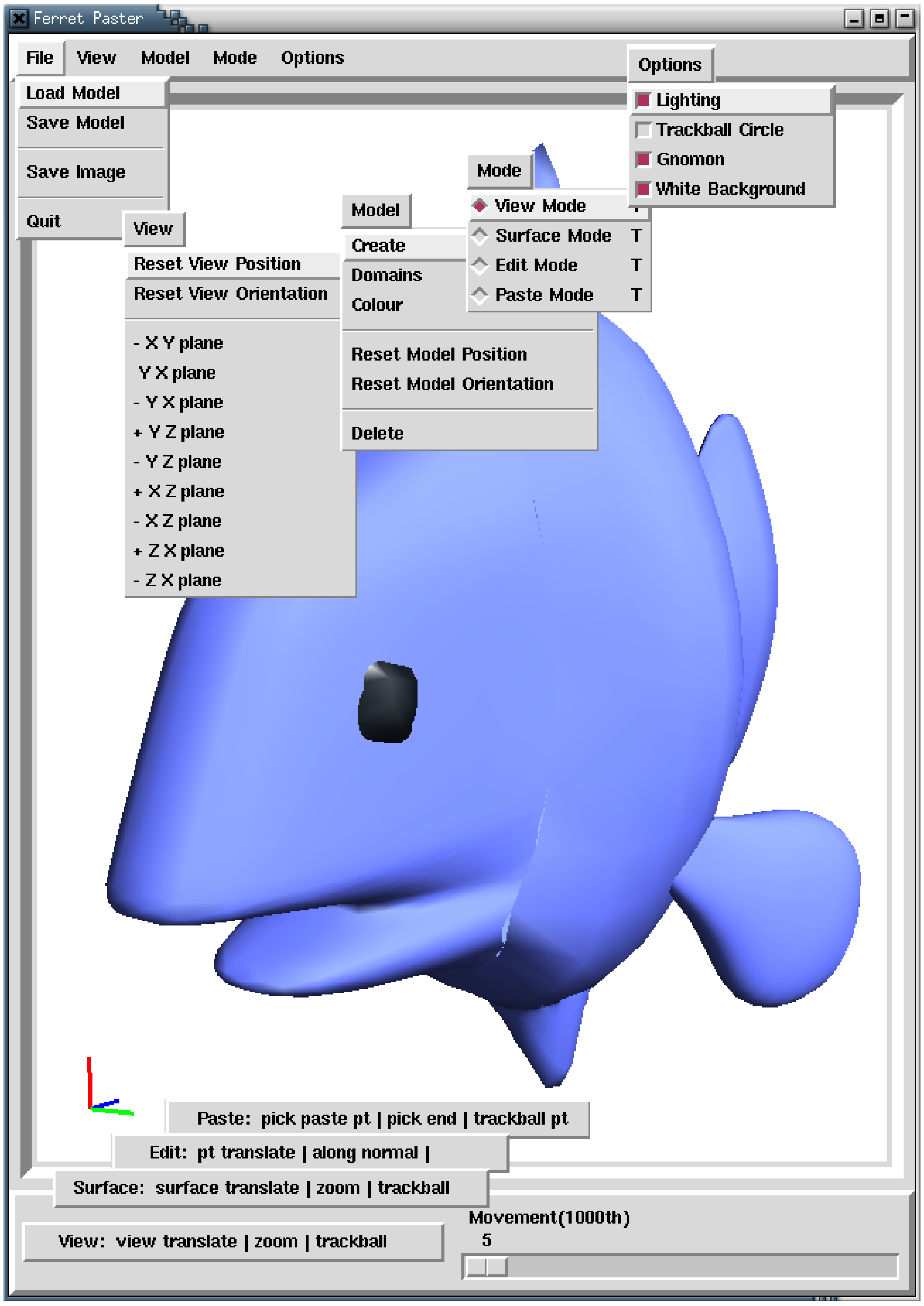


Figure A.1: Menu structure and layout of *Ferret Paster*

The **Model** menu allows the user to create and manipulate models.

Create brings up a dialog for creating patches and cylinders. See Section A.2 for details.

Domains brings up a dialog for domain manipulation of the currently selected surface. See Section A.3 for details.

Colour brings up a dialog for changing the colour of the currently selected surface.

Reset Model Position will move all selected surfaces back to their default positions.

Reset Model Orientation will set all selected surfaces back to their default orientations.

Delete will delete all unpasted selected surfaces that do not have features pasted on them.

There are four different modes for manipulating the different aspects in the program under the **Mode** menu. The user can also switch modes via a button at the bottom left of the screen, which displays the current mode and a short description of the functionality mapped to each mouse button.

View mode allows manipulation of the viewport. Users can translate their eye point in x and y with the left mouse button, and zoom in and out in z with the middle button. The right button allows rotation of the world origin via a trackball.

Surface mode allows manipulation of selected surfaces. Selected surfaces can be translated in x and y with the left mouse button, and moved forward or backward in z with the middle button. Translation is done with respect to viewing coordinates (i.e., y is always up). The right button rotates each surface around its origin via a trackball.

Edit mode allows manipulation of selected control points. The left mouse button will translate each selected point in x and y with respect to viewing coordinates. The middle mouse button will translate each selected control point for a surface in the direction of its current surface normal. This operation is particularly useful for shrinking and expanding cross sections of cylinders. Editing is described in more detail in Section A.2.

Paste mode allows the user to paste surfaces. The left mouse button is used to select a paste point on a surface. The middle button selects a patch or an end of a cylinder to attach to a paste point, and the right button allows the user to change the orientation of the paste point via a trackball. Pasting is described in more detail in Section A.4.

The **Options** menu consists of miscellaneous useful settings.

Lighting turns the lighting on or off. This allows the user to see the undithered surface colouring shown in Figure 2.20 for any selected surface.

Trackball Circle displays a circle in the screen to aid in rotation via the trackball interface.

Gnomon draws a set of gnomon indicating the world origin, and a set of gnomon in the corner of the viewport.

White Background changes the background colour to white.

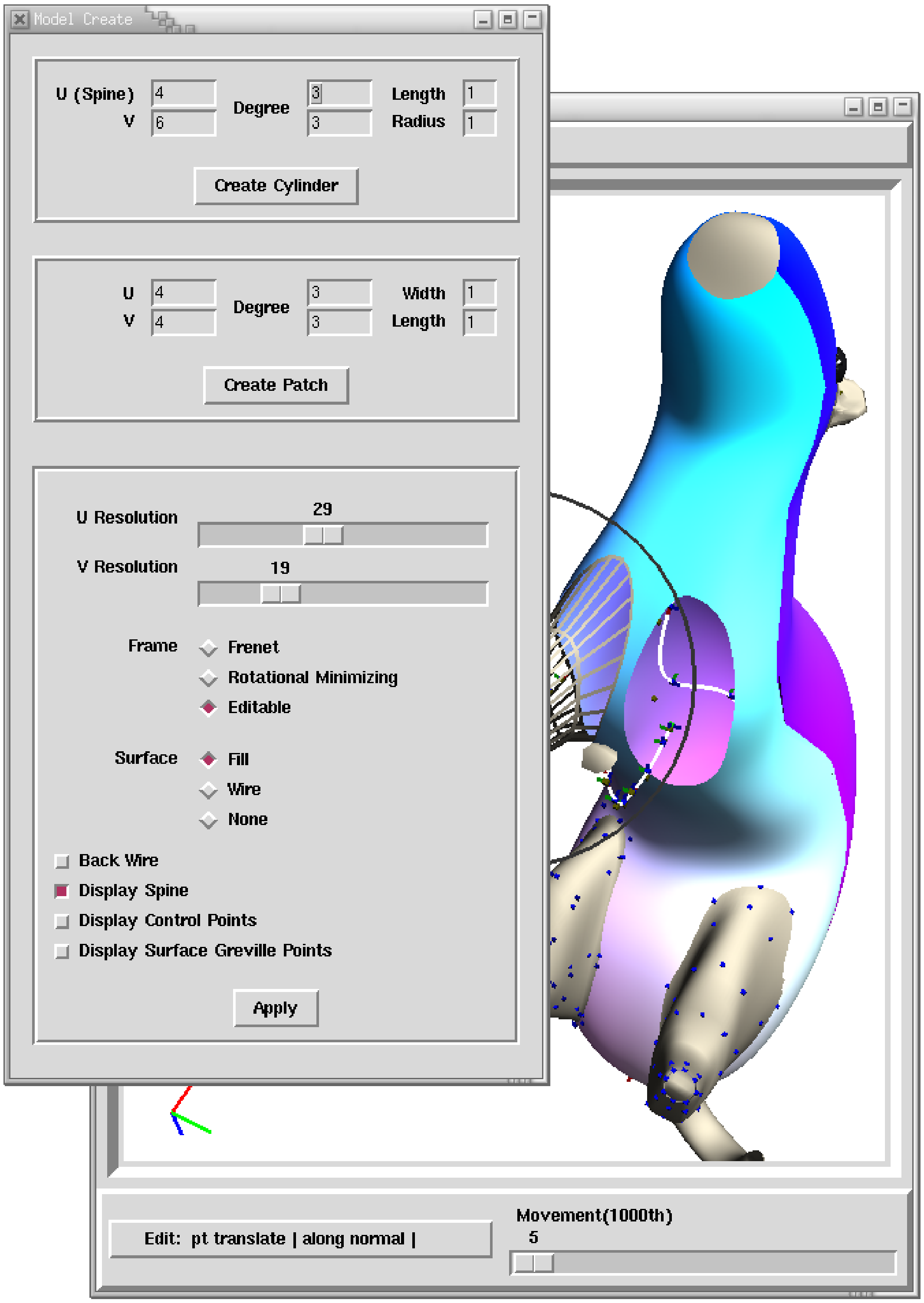
The **Movement** slider in the bottom right corner adjusts the amount of translation per pixel of mouse movement.

A.2 Modelling

Figure A.2 shows the **Model Create** dialog, with various options set on the surfaces that make up the ferret in the viewport. The top two rectangles consist of options for creating patches and cylinders. The bottom rectangle consists of display and editing options.

The top rectangle allows the user to specify parameters for creating a tensor product B-spline cylinder. **U (spine)** specifies the number of control points for the spine of the cylinder, and **V** specifies the number of visible control points for the cross sections of the cylinder. The number of duplicate control points needed for the cross sections is calculated and maintained by the program internally, and is dependent on the **Degree** of the surface. The user can also specified the **Length** of the cylinder, and the distance of the control points from the spine (the **Radius**). The user creates the cylinder by clicking on the **Create Cylinder** button.

The middle rectangle contains the parameters for creating a patch. These are similar to the parameters for a creating a cylinder, except that there are no hidden control points for a patch.



Model Create

U (Spine) 4 Degree 3 Length 1
V 6 Degree 3 Radius 1

Create Cylinder

U 4 Degree 3 Width 1
V 4 Degree 3 Length 1

Create Patch

U Resolution 29
V Resolution 19

Frame: Frenet / Rotational Minimizing / [✓ Editable]

Surface: [✓ Fill] / Wire / None

Back Wire
[✓ Display Spine]
Display Control Points
Display Surface Greville Points

Apply

Edit: pt translate | along normal |

Movement(1000th) 5

Figure A.2: The Model Create dialog.

The bottom rectangle contains mostly display and editing options. The rectangle is changed from a relief to a groove when exactly one surface is selected, and the value for each option reflects the value for the selected surface. A surface can be selected by clicking on it with the left mouse button in **View**, **Surface**, or **Edit** modes.

U/V Resolution controls how fine the OpenGL spline surface is drawn. The larger the number, the smoother the surface.

Frame specifies what type of frame is used in a cylinder. The default frame is **Editable**, which displays an addition spline (Figure A.3) next to the spine. This option is discussed in Section 3.1.1. Options for **Frenet** frames and **Rotational Minimizing** frames are discussed in Section 2.4.

Surface specifies how the surface is drawn. **Fill** draws it as a solid surface, **Wire** displays it in wireframe, and **Fill** hides the surface. The last option is useful for editing the spine of a cylinder and accessing otherwise hidden parts..

Back Wire displays the back of the spline surface in wire frame rather than solid.

Display Spine shows the spine.

Display Control Points shows the control points.

Display Surface Greville Points shows the surface Greville points.

Hitting the **Apply** button updates the selected surfaces with the given options. If no surfaces are selected, all surfaces are updated to the new options.

Users can clicked on the surface Greville points and control points to select them. Clicking on a control point with the middle button will select all control points on the same surface that have the same parametric u value. Clicking on a control point with the right button will select all control points on the same surface that have the same parametric v value. Selecting a surface Greville point also selects its corresponding control point, and vice versa. Surface points are shown in blue, spine points in yellow, and selected points in red. Selected points can be translated in the view plane with the left mouse button in **Edit** mode. The middle mouse button will translate each selected control point in the direction of the surface normal of its surface Greville point. The normals are shown in Figure A.3 on the selected boundary points.

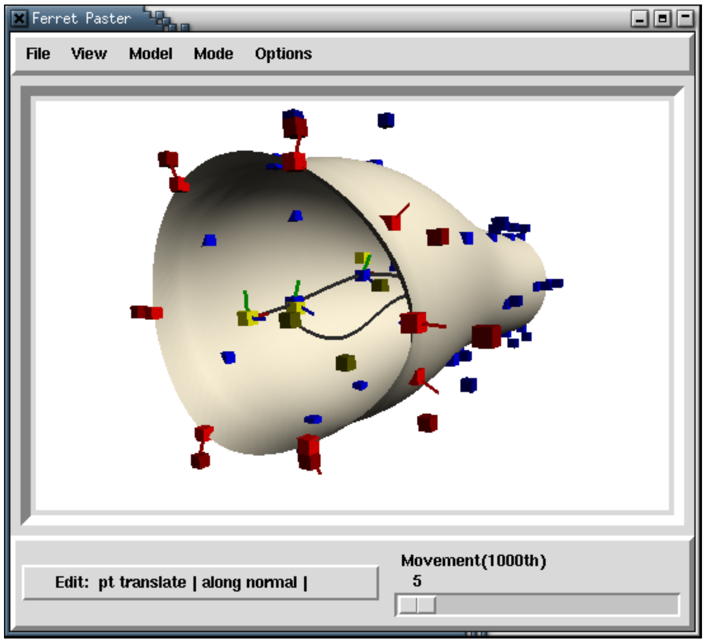


Figure A.3: A Cylinder with a double spine. The selected boundary points show the surface normal for each surface Greville point.

A.3 Domains

Figure A.4 shows the **Model Domain** dialog and menu layout. When a single surface is selected, its domain, along with any trim curves for its features, are displayed in the domain window. Users can translate in x and y using the left mouse button and zoom in and out with the middle mouse button. The lines overlaying the domain show the position of the knot values with respect to the valid range. Knot insertion in u is done by clicking in a valid region with the middle mouse button. Knot insertion in v is similar except with the right mouse button.

In the menu, **Reset** will revert to the original domain position and zoom level. **Double U knots** will insert a knot halfway between every knot in the valid u region. **Double V knots** will do it in v.

A.4 Pasting

When pasting, the user first selects a paste point on the base surface by clicking with the left mouse button. The feature is selected by clicking with the middle

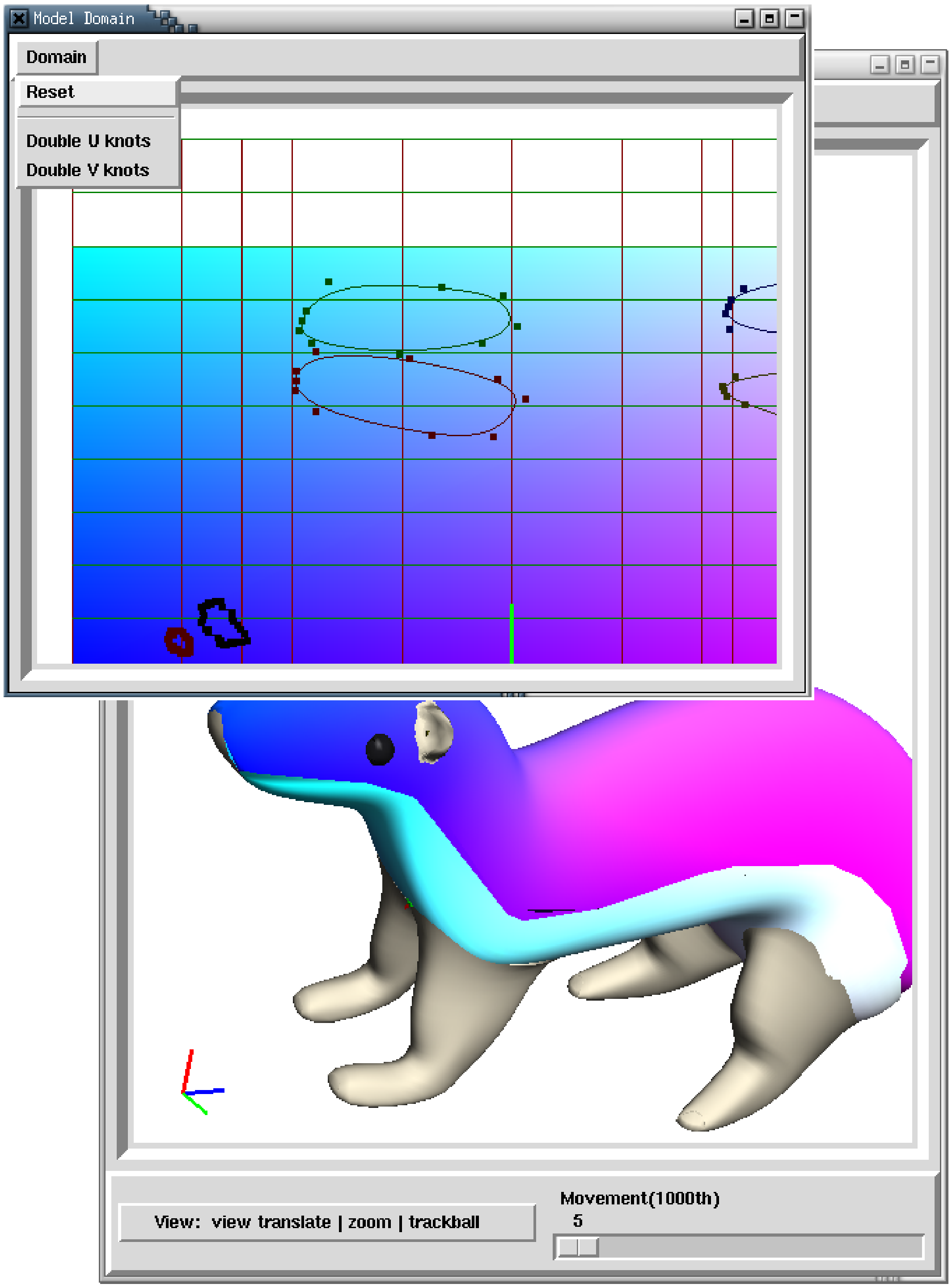


Figure A.4: The Model Domain dialog.

button any where on the surface if the feature is a patch, or by clicking close to one of the ends if the feature is a cylinder. The default paste frame is constructed with the paste direction normal to the surface, and this can be modified by rotating the paste frame via a trackball with the right mouse button.

Once a feature is selected, the **Paste Options** dialog becomes visible. Figure A.5 shows the layout of the **Paste Options** dialog. Behind the dialog, you can see the paste point, displayed in green, on the body of the ferret. The lines extruding from the boundary surface Greville points indicate the direction the feature is projected from. The user can bring up the dialog anytime by middle clicking on a pasted feature. The paste point is only displayed for the current paste feature.

The top rectangle shows the options for pasting cylinders and patches.

Translate sets the distance the end of a cylinder's spine or the patch's centroid is placed from the paste point.

Continuity determines whether it is a C^0 or C^1 paste.

Paste surface Greville point is checked if the user wants to paste the surface Greville points rather than the control points.

Keep initial paste UV is checked if the user wants to be able to reorient the paste frame without repasting the boundary of the pasted feature.

Trim allows the user to specify if there is trimming, and if the inside or the outside of the base surface is trimmed.

The following options only apply to patch pasting.

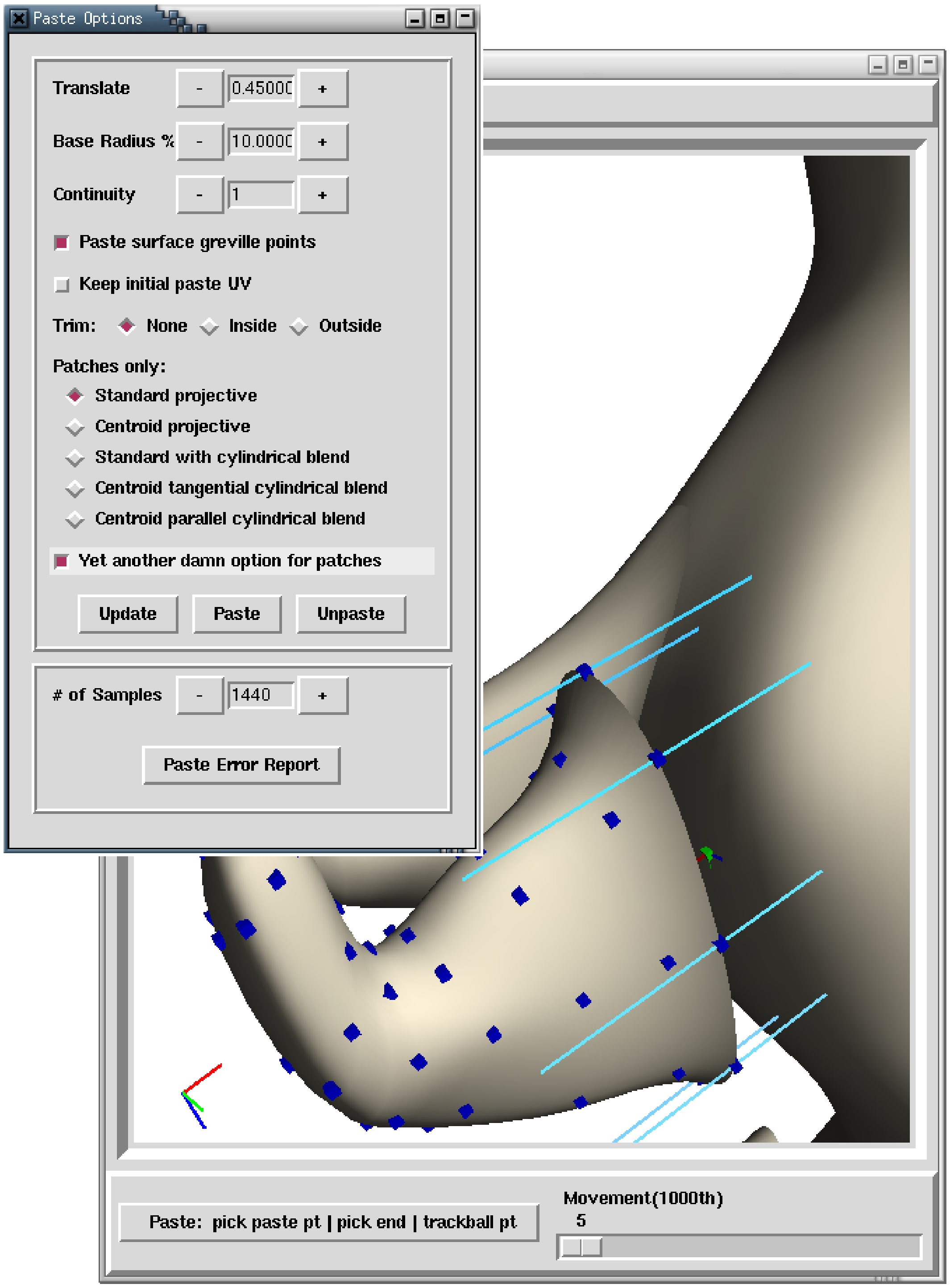
Standard projective specifies a bilinear projective paste.

Centroid projective specifies a centroid projective paste.

Standard with cylindrical blend specifies a bilinear projective trimmed paste with a tangential cylindrical blend.

Centroid tangential cylindrical blend specifies a centroid projective trimmed paste with a tangential cylindrical blend.

Centroid parallel cylindrical blend specifies a centroid projective trimmed paste with a parallel cylindrical blend.


Paste Options

Translate - 0.45000 +

Base Radius % - 10.0000 +

Continuity - 1 +

[✓ Paste surface greville points]

Keep initial paste UV

Trim: [✓ None] Inside Outside

Patches only:

[✓ Standard projective]
Centroid projective
Standard with cylindrical blend
Centroid tangential cylindrical blend
Centroid parallel cylindrical blend

[✓ Yet another damn option for patches]

# of Samples - 1440 +

Movement(1000th) 5

Figure A.5: The Paste Options dialog. The settings are shown for the right foreleg of the ferret.

Yet another damn option for patches determines how the boundary is extended for a trimmed patch paste. If this option is checked, the boundary is extended via the range space (Figure 3.18, right), otherwise it is extended in the domain (Figure 3.18, left). This option is only used for trimmed features pasted with C^1 continuity.

Base Radius sets the how far the boundary curve is extended for the above option.

The **Update** button allows the user to preview the paste via projected lines. The **Paste** button pastes the actual feature. **Unpaste** removes the feature from the base surface and deletes the paste point.

The bottom rectangle allows the user for specifying the number of samples taken when creating an error report for the boundaries.

Bibliography

[1] Shreeram S. Abhyankar. *Algebraic geometry for scientists and engineers*. Providence, R.I. : American Mathematical Society, 1990.

[2] Cristin Barghiel. Feature oriented composition of B-spline surfaces. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1994. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-94-13/.

[3] Cristin Barghiel, Richard Bartels, and David Forsey. Pasting spline surfaces. In M. Daehlen, T. Lyche, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces: Ulvik, Norway*, pages 31–40, Nashville, TN., 1994. Vanderbilt University Press.

[4] Richard Bartels and John Beatty. A technique for the direct manipulation of spline curves. In *Proceedings of Graphics Interface*, pages 33–39, 1989.

[5] Richard Bartels and David Forsey. Spline overlay surfaces. Technical Report CS-92-08, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1991.

[6] Jules Bloomenthal. Calculation of reference frames along a space curve. In Andrew S. Glassner, editor, *Graphics gems*, pages 567–571. Academic Press Professional, 1990.

[7] Jules Bloomenthal, editor. *Introduction to implicit surfaces*. Morgan Kaufmann Publishers, Inc., 1997.

[8] Leith Kin Yip Chan. World space user interface for surface pasting. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1996. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-96-32/.

[9] Blair Conrad. Better pasting through quasi-interpolation. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1999. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-99-14/.

[10] Blair Conrad and Stephen Mann. Better pasting via quasi-interpolation. In Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker, editors, *Curve and Surface Design: Saint-Malo, 1999*, pages 27–36, Nashville, TN., 2000. Vanderbilt University Press.

[11] Carl De Boor. *A practical guide to splines*. Springer-Verlag, 1978.

[12] Gerald Farin. *Curves and Surfaces for CAGD*. Academic Press, fourth edition, 1996.

[13] David Forsey and Richard Bartels. Hierarchical B-Spline refinement. In *Computer Graphics*, volume 22, August 1988.

[14] James G. Fox, editor. *Biology and Diseases of the Ferret*. Williams and Wilkins, second edition, 1998.

[15] David Kahaner, Cleve Moler, and Stephen Nash. *Numerical Methods and Software*. Prentice Hall, 1989.

[16] James T. Kajiya. Ray tracing parametric patches. In *Computer Graphics*, volume 16,, pages 245–254. ACM Siggraph, 1982.

[17] Kwansik Kim and Gershon Elber. New approaches to freeform surface fillets. In *The Journal of Visualization and Computer Animation*, volume 8, pages 69–80. 1997.

[18] Rick Leung and Stephen Mann. Distortion minimization and continuity preservation in surface pasting. In *Proc. Graphics Interface*, pages 193–200, May 2003.

[19] Ricky Leung. Distortion minimization and continuity preservation in surface pasting. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 2002.

[20] Marryat Ma. The direct manipulation of pasted surfaces. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 2000. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-200015/.

[21] Marryat Ma and Stephen Mann. Multiresolution editing of pasted surfaces. In *Mathematical Methods for Curves and Surfaces, Oslo, 2000*, 2001.

[22] Stephen Mann. *Course Notes for CS 679: Splines and their uses in computer graphics*. University of Waterloo, winter 2001 edition.

[23] Stephen Mann and Tony DeRose. Computing values and derivatives of Bézier and B-spline tensor products. In *Computer Aided Geometric Design*, volume 12, pages 107–110, February 1995.

[24] Stephen Mann and Teresa Yeung. Cylindrical surface pasting. Technical report, University of Waterloo, Waterloo, Ontario, Canada N2l 3G1, 1999. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-99-13/.

[25] Stephen Mann and Teresa Yeung. Cylindrical surface pasting. In *Geometric Modeling*, pages 233–248. Springer-Wein, 2001.

[26] Don P. Mitchell. Robust ray intersection with interval arithmetic. In *Graphics Interface '90*, pages 75–82, May 1990.

[27] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, second edition, 1998.

[28] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*, pages 279–282. Cambridge University Press, 1988.

[29] Kim Schilling. *Ferrets for dummies*. IDG Books Worldwide, 2000.

[30] Side Effects Software Inc., Toronto, Ontario, Canada M5V 3E7. *Houdini 5 Reference*, 2002.

[31] Joe Warren and Henrik Weimer. *Subdivision methods for geometric design: a constructive approach*. Morgan Kaufmann Publishers, San Francisco, USA, 2002.