World Space Surface Pasting

Leith Kin Yip Chan lkychan@cgl.uwaterloo.ca
Stephen Mann smann@cgl.uwaterloo.ca
Richard Bartels rhbartel@cgl.uwaterloo.ca

Computer Science Department
University of Waterloo
Waterloo, Ontario, N2L 3G1 CANADA

Abstract

Surface pasting is a composition method that applies features to base surfaces to provide details on the base surfaces. The location and size of a feature are determined by the transformations of the feature's domain. By modifying the domain layout of pasted surfaces, we can manipulate the appearance of a feature interactively in a *Domain Space User Interface*. However, this domain user interface is inconvenient because the user cannot interact with the three-dimensional model directly. In this paper, we describe a *World Space User Interface* that maps actions on the three space surfaces to two-dimensional domain operations.

Résumé

Le collage de surface est une méthode de composition qui ajoute des traits à une surface de base pour augmenter le niveau de détail de cette surface. La position et la taille d'un trait est déterminer par des transformations du domaine de ce trait. En modifiant le positionnement du domaine d'une surface coller on peut modifier l'apparence d'un trait inter activement par une Interface Usager de l'Espace Domaine. Par contre, cette interface est inconvenante à utiliser parceque l'usager ne peut pas contrôler le modéle tri-dimensionnel directement. Dans cet article, nous présentons une Interface Usager de l'Espace Monde qui converti des action sur use surface tri-dimensionnel à des actions sur le domaine bi-dimensionnel.

Keywords: Surfaces, User Interfaces

1 Introduction

Tensor product B-splines are a polynomial surface representation commonly used in industry. A surface in tensor product B-spline form is defined as

$$B(u,v) = \sum_{i}^{n} \sum_{j}^{m} P_{i,j} N_i^n(u) N_j^m(v),$$

where the $P_{i,j}$ are a set of control points that define the surface, and the $N_i^n(u)$ are the B-spline basis functions [3].

To enhance the detail on a region of a B-spline tensor product surface, we can increase the number of control vertices by splitting patches (Boehm's algorithm [5] and the Oslo algorithm [3]). However, only an entire row or an entire column of patches can be split, thus adding details (patches) to a particular region requires the addition of patches across the entire surface. This results in the addition of unnecessary control vertices and computation needed to evaluate the surface. An alternative way to add detail to the surface, aimed at maintaining a low number of control vertices, was proposed by Forsey and Bartels [8]. In their approach, detail surfaces, called overlays or features, are applied to a base surface in a hierarchical fashion to create a composite surface of increased complexity. Each overlay is a normal displacement from a conveniently chosen reference point.

Barghiel [1] generalised Forsey's method, elaborated on a displacement method proposed by Bartels [4], and applied it to interactive, real-time modeling. To avoid computing the displacement of every point along the feature surface, he defined an approximate displacement mechanism that involved only the control vertices. This approximation technique is called *surface pasting*. By using surface pasting, detailed features are added to composite spline surfaces in a multi-layered fashion by means of an efficient displacement scheme. The feature orientation is arbitrary, and the underlying domains may be partially overlapping and non-linearly transformed. As

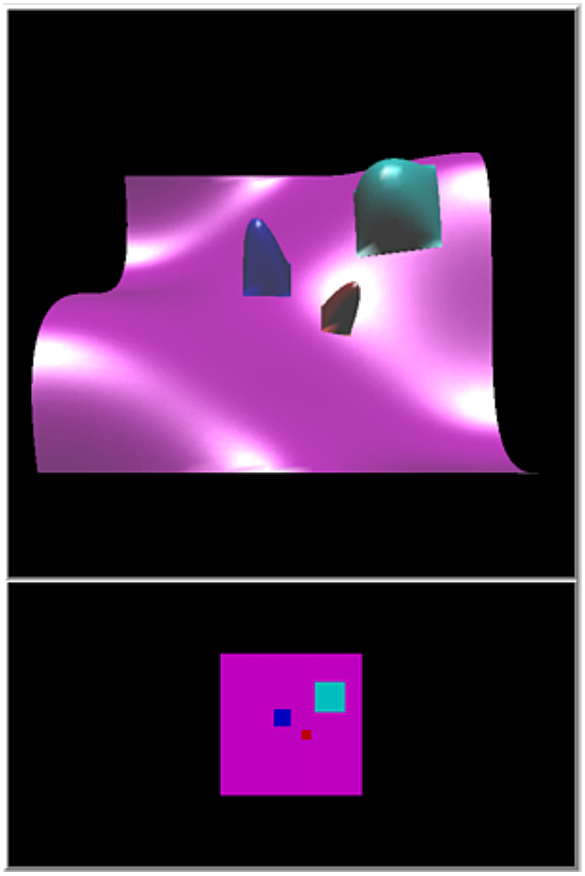

Figure 1: Domain Space User Interface

a result, this technique allows us to model surfaces by interactively changing displacements, control vertices, or the domain layout.

Moreover, Barghiel implemented a pasting editor called PasteMaker to paste features on base surfaces hierarchically. A user of PasteMaker can control the location as well as the size of a feature by manipulating the domain relationship between the base and feature surfaces (*Domain Space User Interface*). Figure 1 shows a domain space user interface. In the top window, there is a view of the world space pasted objects, and in the bottom window is a view of the domains of these pasted objects. A pasting session involves making an initial embedding of the feature domain in the base domain, and then translating, rotating, and scaling the feature domain (in the bottom window) until the corresponding 3-space feature is correctly positioned on the 3-space base surface (in the top window).

This domain space user interface is inconvenient because the user cannot interact with the three-dimensional model directly. Positioning of a feature on the base surface is a trial-and-error process because we do not know which 3D point on the base surface corresponds to a 2D point in the base domain until we have moved a feature domain onto that domain point and see the result.

In this paper, we describe *PasteInterface*, a *World Space User Interface* [6] that maps three-space user actions to two-space domain operations so that the user can manipulate the three-dimensional composite surfaces directly. This software was implemented on top of the Spline Library of the University of Waterloo [12] and OpenInventor [14]. The goal of this research was to find operations in the world 3-space that naturally map to the two-dimensional domain functions used in surface pasting. Thus, the focus of this paper is on the mathematics behind these user interface operations. The mathematical details of the pasting process itself can be found in Barghiel's thesis [1] and a related paper [2].

Other researchers have devised schemes for locally adding surface detail (see, for example, [8, 9, 10, 11, 13]). However, while these methods allow you to add local detail, they are not particularly amenable to editing and repositioning this detail. Coquillart devised a scheme that allows you to add and edit local detail by using free-form deformation to locally deform space in the area you wish to change [7]. However, this technique requires functionally composing the base surface with the feature deformation, which creates a surface that is not in standard B-spline form. The technique described in this paper can represent the final surface as a standard sequence of trimmed tensor product patches.

2 World Space User Interface

The goal in surface pasting is to position a feature on the base surface. In the Domain Space Interface of Barghiel, this positioning was done by manipulating the feature domain as a sub-domain of the base domain. The user of this system would position the feature domain in the base domain, and then look at the resulting surfaces in the world space. Unfortunately, depending on the orientation of the viewpoint in the three space, a translation of the feature domain to the right on the screen might result in a translation of the feature to the left in world space.

The purpose of the world space user interface is to hide the domains from the user so that he can directly manipulate the three-dimensional objects. However, no matter what is displayed to the user or how the user manipulates the objects, operations such as feature translation and rotation have to be performed in the domain space. Therefore, the world

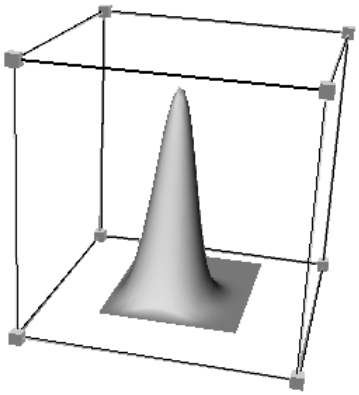

Figure 2: Transformation Box 3D Manipulator

space user interface maps all user actions into domain operations.

Surface pasting is well suited to mouse-style interactions, since mathematically, the operations on the surfaces occur in their two dimensional domains. The difficulty is in finding natural three space operations, and in mapping them to appropriate domain operations.

The following sections discuss the techniques employed to allow a user to perform operations in world space, and the conversion of these 3-space operations into corresponding domain space operations.

3 Pasting

To paste a feature on a base surface, we need to map the feature domain into the base domain. This transformation determines the location as well as the size of the pasted feature on the base surface. In PasteInterface, for the user to specify the initial pasting location, he must move and scale the feature close to the target location on the base surface. Once the user is satisfied with the position, we calculate the embedding of the feature domain in the base domain, and the feature is pasted onto the base surface. The following sections talk about the pasting process in more detail.

3.1 Moving the Unpasted Feature

The user manipulates the unpasted feature via an OpenInventor 3D manipulator called a *transformation box* [14]. This transformation box consists of handles, called *draggers*, as shown in Figure 2. There are three sets of draggers: one for translation, one for scaling and the remaining one for rotation. Their functions are shown in Table 1. Notice that manipulations of the unpasted feature are carried out solely to find an embedding of the feature domains in the base domain, as discussed in the next section. Once this embedding is performed, the three-dimensional transformation of the feature is discarded.

3.2 Projecting the Feature

After moving the feature, the user may want to preview the pasting location of the feature. This is done by projecting the four corner points of the feature onto the base surface. The projection of each corner is in the direction of the normal to the plane that best fits the four corner points of the feature. The intersection points will be the corners of the pasting position, as illustrated in Figure 3a (Figure 3b shows the feature after pasting). Note that this projecting process is similar to a process used by Alias Studio.[1] However, Alias Studio projects a dense sampling of the entire boundary of the feature, and uses root finding to find the domain pre-image of these pasted boundaries.

If the user decides to paste the feature to the preview location, we have to transform the feature domain so that the feature can paste onto the preview location. First, we find the corresponding domain points of the four preview points on the base surface, as shown in Figure 4. Then we need to transform the original domain polygon $D = F_1F_2F_3F_4$ to the projected polygon $\overline{D} = \overline{F_1}\,\overline{F_2}\,\overline{F_3}\,\overline{F_4}$. Since the target polygon is not necessarily a rectangle, a linear transformation will not work, but a *bi-linear transformation* will be sufficient. Moreover, if both polygons are convex, the bi-linear transformation has an unique inverse from $\overline{D}$ back to D (a requirement for surface pasting [1]).

By using a bi-linear transformation, all points inside $F_1F_2F_3F_4$ can be represented as:

$$\begin{aligned} p(u,v) &= l_1(u)\cdot v + l_2(u)\cdot(1-v) \quad (1)\\ \text{where}\quad l_1(u) &= F_1\cdot u + F_4\cdot(1-u)\\ l_2(u) &= F_2\cdot u + F_3\cdot(1-u)\\ 0\le u\le 1,\qquad & 0\le v\le 1 \end{aligned}$$

Similarly, all points inside $\overline{F_1}\,\overline{F_2}\,\overline{F_3}\,\overline{F_4}$ can be represented as:

$$\begin{aligned} \overline{p}(u,v) &= \overline{l_1}(u)\cdot v + \overline{l_2}(u)\cdot(1-v) \quad (2)\\ \text{where}\quad \overline{l_1}(u) &= \overline{F_1}\cdot u + \overline{F_4}\cdot(1-u)\\ \overline{l_2}(u) &= \overline{F_2}\cdot u + \overline{F_3}\cdot(1-u)\\ 0\le u\le 1,\qquad & 0\le v\le 1 \end{aligned}$$

[1] Alias Studio is a trademark of Alias|Wavefront

Draggers	*Functions*
6 faces of the box	move the object along the corresponding plane
8 small cubes at corners	scale the object uniformly
12 edges of the box	rotate the object around the axis at the center and parallel to the corresponding edge

Table 1: Transformation Box 3D Manipulator's Functions

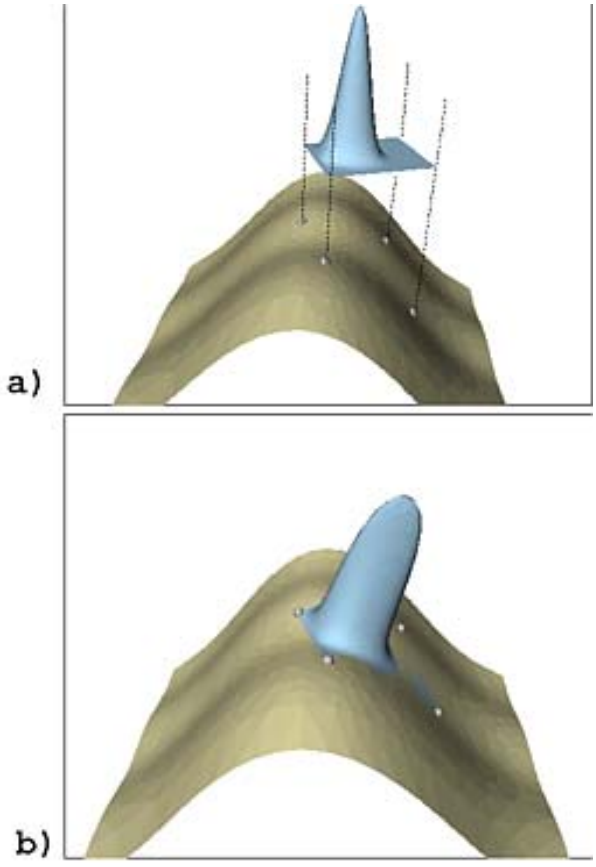

Figure 3: Projecting Feature on Base Surface and Pasting

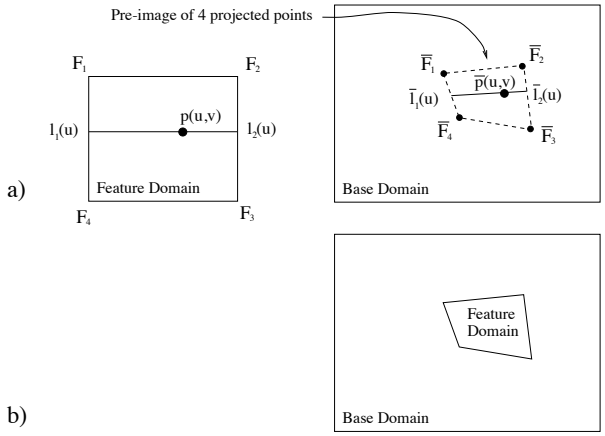

Figure 4: Pasting - Domain View

To transform any point from one polygon to the other polygon domain, we only need find the u and v values of the point inside a polygon and then evaluate the above equations to find the transformed point inside the other polygon. For example, in Figure 4a, given a point $p(u,v)$ inside $F_1F_2F_3F_4$, and the observation that $p(u,v)l_1(u)$ is parallel to $l_2(u)l_1(u)$, we have:

$$(p(u,v) - l_1(u)) \times (l_2(u) - l_1(u)) = 0 \qquad (3)$$

Solving Equation 1 and 3 gives u and v, and hence we can get $\overline{p}(u,v)$ from Equation 2.

Finding the inverse point is the same process, except that we solve for (u,v) from polygon $\overline{F_1}\,\overline{F_2}\,\overline{F_3}\,\overline{F_4}$ and evaluate $p(u,v)$ from Equation 1.

4 Translating a Pasted Feature

After pasting, the user may want to adjust the position of the feature surface on the base surface. One type of adjustment is provided by translating the feature across the base surface.

The challenge of providing translation is finding a way to move the pasted feature surface with the mouse cursor so that the user has a direct manipulation of the feature. Surprisingly, it is hard to achieve this. The following is a loose description of what is wanted: The user should manipulate the feature as if

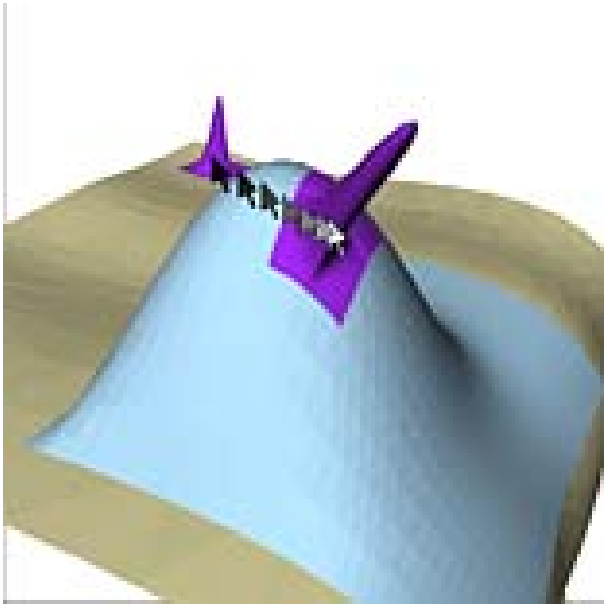

Figure 5: Moving a Feature with the Mouse

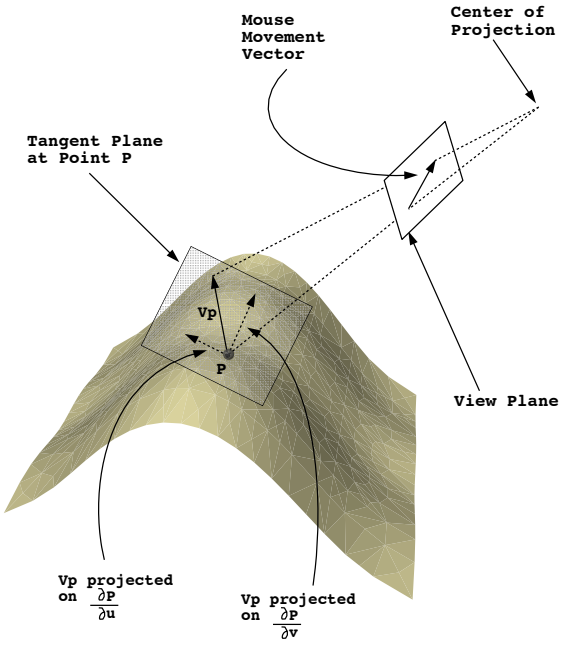

Figure 6: Projecting Mouse Movement Vector on the Tangent Plane

there is a real, three dimensional model of a feature on a base surface sitting in front of her so that she can use her hand to slide the feature back and front, left and right, or set the feature at an arbitrary position. It would be ideal if the user has that kind of feeling when she is using the mouse to translate the feature. Unfortunately, the sliding paradigm is impossible to implement with the mouse, since strong feedback is required to restrict the feature to the base surface, while the mouse is free to move anywhere on the screen.

An alternative solution is to have the feature follow the mouse as best it can. While this works well for setting the feature at an arbitrary position on the base surface, this cursor tracking does not work well when sliding the feature across the base surface. Consider the situation illustrated in Figure 5. In this figure, a user tries to move the shaded feature (the one closer to us) on the top of the bump (base surface) with the mouse cursor moving from right to left as shown. Since the feature stays under the mouse cursor as the user moves the mouse, the feature moves across the top of the bump and then suddenly jumps to the edge of the base surface (the left shaded feature). Although this movement is desired if the user wants to move the feature to the edge, it does not provide the sliding paradigm where the user wants to slide the feature to the back of the bump.

We provided two approaches for simulating the direct manipulation: projection and picking. In the latter method, the feature directly follows the mouse cursor, while in the former, the feature slides along the base surface, avoiding the problem discussed in the previous paragraph.

4.1 Projective-Translation

The main purpose of the projective-translation is to give the user the feeling of sliding a feature. When the user moves the mouse, the feature should slide on the base surface by following the direction of the mouse movement. Thus, we have to transform the mouse movement $(\Delta x, \Delta y)$ on the screen to a translation $(\Delta u, \Delta v)$ in the base domain. This $(\Delta x, \Delta y)$ translation is then used to translate the feature domain within the base domain.

We chose the following approach: given a mouse movement vector produced over a short time span, we first find the center point P of the feature to represent the location of the feature surface. Second, we find the tangent plane at that point, which is defined by the two partial derivative vectors $\frac{\partial P}{\partial u}$ and $\frac{\partial P}{\partial v}$, as in Figure 6. Then by projecting the mouse movement vector from the view plane to the tangent plane, we get the vector v_P. Finally, we get the domain displacements Δu and Δv, which are the magnitudes of vectors obtained by projecting v_P onto $\frac{\partial P}{\partial u}$ and $\frac{\partial P}{\partial v}$ respectively.

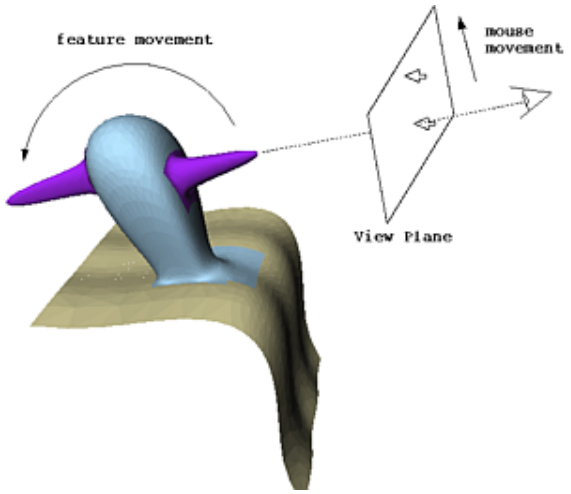

Figure 7: Sliding a Feature with a Mouse

There is a subtlety involved with this process. Consider Figure 7, which shows the desired movement of the feature resulting from the mouse movement. When the mouse moves upward, we would like the feature go up at the beginning, and then pass over the top and continue to go "forward". Initially, the feature will move upward because the tangent plane is facing the view plane. Once the feature moves across the top and to the back of the base surface, we would like the feature to move down as we continue our mouse motion. However, note that now the tangent plane has flipped so that its back faces the view plane. If the tangent plane is updated in real time (eg., if at each time interval we pick a new point P and tangent plane to the surface at P), the mouse movement vector will project to the back of the tangent plane, and moving the mouse further upward will actually move the feature backward.

To avoid this problem, we select the point P when the mouse button is pressed, and use the tangent plane at this point for all translation movement until the mouse button is released. Thus, we do not update the tangent plane in real time with the movement of feature and avoid the problem of having the feature go backward while the mouse is moved along one direction within a click-drag-release cycle.

4.2 Picking-Translation

An alternative translation method is picking. Picking lets the user pick and drop a feature to anywhere the user can see on the screen. By attaching a 3D manipulator to the feature surface, a user can drag the manipulator to pick a point on the base surface, as shown in Figure 8. Then we can find the coordinates of the point in the base domain, and translate the feature so that a designated point in the feature's domain lies on that base domain point. Currently, we allow the user to translate the domain based on either the feature's center or any of its four corners. A picking dragger for translation, which is a part of the manipulator, is shown in Figure 9.

5 Rotation and Scaling

Rotation and scaling of a pasted surface is done with a 3D manipulator. Figure 9 shows two draggers for scaling and one for rotation. The rotation dragger has one degree of freedom (the angle of rotation) and the horizontal scaling dragger has two degrees of freedom (the scaling ratios of two perpendicular directions). The values of these draggers are directly mapped to the underlying domain polygon. For example, if the rotation manipulator rotates 90 degrees, the feature domain will also make a right angle rotation. The remaining height dragger has one degree of freedom; however, unlike the above two manipulators, its scaling ratio maps to the z-component of all the feature's control vertices in the feature's local coordinate frame.

6 Fine Tuning

In order to fine tune the feature's location and appearance, an additional function is loaded into the four corner draggers (Figure 9). A user can translate a individual single corner without moving the whole feature by shift-clicking[2] a corner dragger. However, the corner is not allowed to move out of the valid region as shown in the domain view of Figure 10, because the convexity of the feature domain has to be maintained. To help the user identifying the valid region, the region's boundary is drawn on the surfaces whenever the corner dragger is activated, as shown in the model view of the same figure.

The mechanism of these draggers is the same as the picking manipulator shown in Figure 8, except they move individual corners instead of the whole feature.

7 Outline Mode

Performance issues arise when updating a grouped feature or a feature that has many dependent surfaces. For example, if a user picks a feature that has a number of dependent surfaces, the system has to update all of these surfaces while the user is translating/rotating/scaling the feature. This updating can be very slow. In order to get interactive speeds, an

[2]Clicking the dragger while pressing the shift key

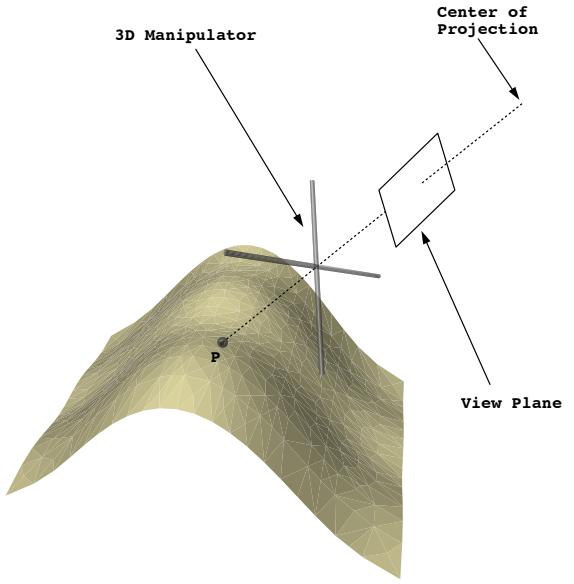

Figure 8: Using a 3D Manipulator to Pick a Point on the Surface

outline mode is provided. Under this mode, only the outline (four corners) of the selected feature is updated within a click-drag-release-cycle of projective-translation, rotation and scaling. The feature and related surfaces are updated only at the end after the user releases the button.

The performance problem is even worst for picking-translation (Section 4.2). In OpenInventor, ray-picking is a slow process. Translation would be too slow if we ray-pick during a drag-release-cycle to update the new position of the feature and show the outline to the user, and even slower if we update the feature and its dependent surfaces. Therefore, ray-picking and surface updating has to be performed at the end of the drag-release-cycle. Fortunately, the visual feedback in picking-translation is not important for this method of translation, as the paradigm is that of selecting a new location for the feature rather than dragging the feature across the base surface. Indeed, were the feature's position updated in real-time, the feature would potentially jump back and forth across the base surface, as discussed earlier.

8 Discussion and Conclusion

In general, the three space surface operations worked well for surface pasting. Translation proved the most problematic, primarily because there was no natural way to specify a translation of the pasted surface in three space. While neither of the two methods of translation we provided are ideal, together they seem to perform adequately. The picking-translation technique allows the user to coarsely specify the feature position, while the projective-translation method allows for more fine grain control.

The primary new functionality provided by the World Space User Interface is corner dragging, which enables the user to place individual corners at precise locations on the base surface, allowing the user to make changes to the feature that cannot be accomplished with only translation, rotation, and scaling. Corner dragging appears to give the user significantly more control over the shape and location of the features.

Both the World Space User Interface and the Domain Space User Interface suffer some performance problems. In particular, when multiple surfaces are pasted together, the time required to compute a new surface after translating, scaling, or rotating a pasted feature becomes too large for real-time interaction. As a compromise between speed and feedback, we chose to only display the corner points of the pasted feature when transforming it.

The performance of the World Space User Interface is superior to the Domain Space User Interface in the selecting the initial embedding of the feature domain in the base domain. In the Domain Space User Interface, the initial embedding was selecting by placing the feature domain in the base domain and seeing where the feature appeared on the base in the world space. Since this initial placement is just a guess, the user will usually have to transform the pasted feature to get it to the correct position and orientation. These adjustment transformations suffer from the performance problems discussed in the previous paragraph.

With the world space interface, the user can position the feature to roughly the correct position on the base surface in the world space. This positioning does not suffer from the performance problems faced by adjusting a pasted feature, as we do not need to compute a new composite surface at each time step. Once the user is satisfied with the approximate location of the feature, the software then determines the embedding of the domain. Thus, with the World Space User Interface, the user is able to quickly determine the initial embedding, requiring fewer transformations of the pasted feature. This is a big advantage of the World Space User Interface, given the high cost of transforming the pasted feature in both user interfaces.

As examples of more complex surfaces created using pasting, see Figures 11 and 12. Figure 11 shows a dog's face composed of 7 surfaces in a 2 level pasting hierarchy created with a domain space user interface. Figure 12 shows a dog that is composed of 10 surfaces in a pasting hierarchy that is 5 levels deep that was created with a world space user interface. While complex surfaces can be created with either interface, the world space interface is more natural to use and has better performance when creating deeper hierarchies as seen in this latter figure.

9 Future Work

Our research focused on the mathematical aspect of mapping three-dimensional operations into two-dimensional domain functions. No user interface experiments have been conducted; the look and feel of current user interface is only based on the advise of a few users. Further research should focus on human computer interactions.

Another direction of future work is in improving the surface quality. Surface pasting is only an approximation technique. It optimizes performance by trading off surface quality. Irregularities in the approximation may arise from the different tensor product alignment of the overlapping surfaces. A post-processor could take the information of pasted surfaces created from an efficient approximated modeler like PasteInterface and create high quality composite surfaces. This post-processor can optimize the surface quality rather than the performance because it does not need to be interactive.

10 Acknowledgments

We would like to thank Clara Tsang for allowing us to use Figures 1 and 11.

This research was funded by the Natural Sciences and Engineering Research Council of Canada.

References

[1] C. Barghiel. Feature oriented composition of b-spline surfaces. Master's thesis, Univeristy of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1994. (Available as Computer Science Department Technical Report CS-94-13).

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

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

[4] R. Bartels and D. Forsey. Spline overlay surfaces. Technical Report CS-92-08, Univeristy of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1991.

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

[6] L. K. Y. Chan. World space user interface for surface pasting. Master's thesis, Univeristy of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1996. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-96-32/.

[7] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):187–196, 1990.

[8] D. Forsey and R. Bartels. Hierarchical b-spline refinement. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):205–212, 1988.

[9] B. Fowler. Geometric manipulation of tensor product surfaces. *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, 25(2):101–108, 1992.

[10] W. Hsu, J. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):177–184, 1992.

[11] N. Matsuki. *An interactive shape modification method for B-spline curves and surfaces*, pages 385–397. Elsevier (North–Holland) (IFIP), 1992.

[12] A. Vermeulen and R. Bartels. C++ splines classes for prototyping. In *Curves and Surfaces in Computer Vision and Graphics II*, pages 1610:121–131, Bellingham, Washington, 1992. SPIE Proceedings, SPIE '92.

[13] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):157–166, 1992.

[14] J. Wernecke. *Programming Object-Oriented 3D Graphics with Open Inventor.* Addison-Wesley Publishing Company, 1994.

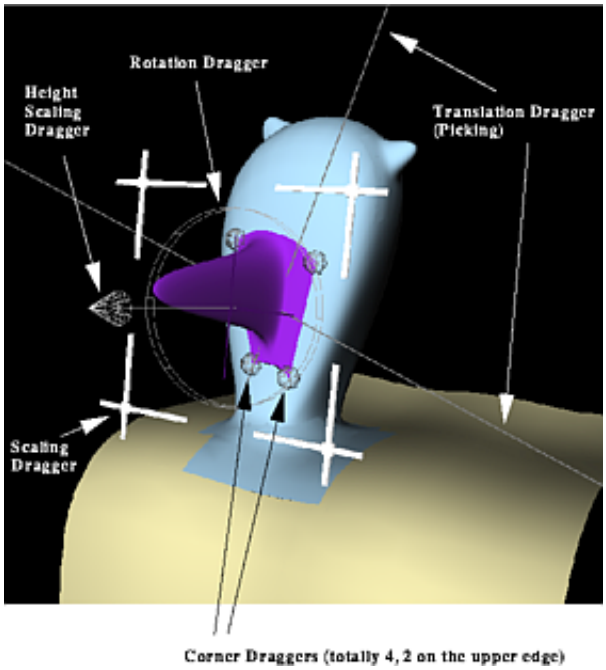

Figure 9: 3D Manipulators

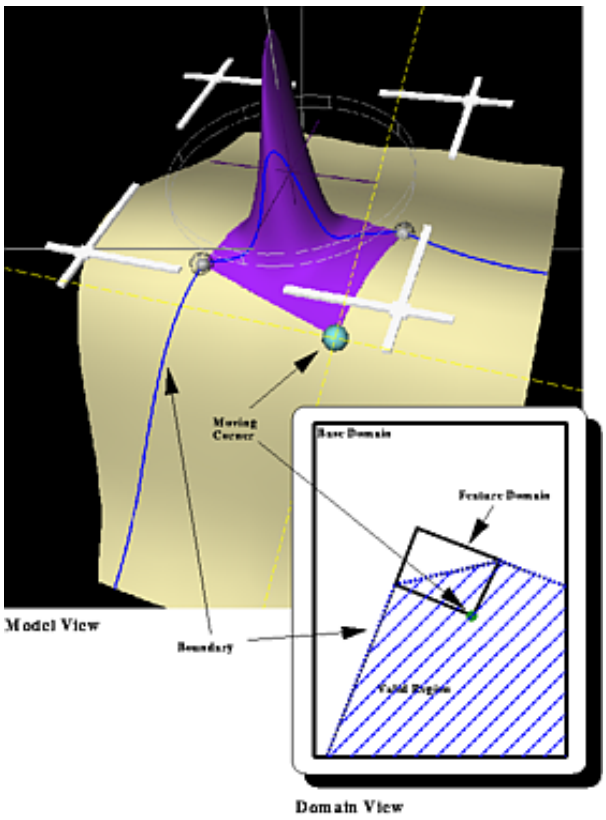

Figure 10: Translating a Corner of a Feature

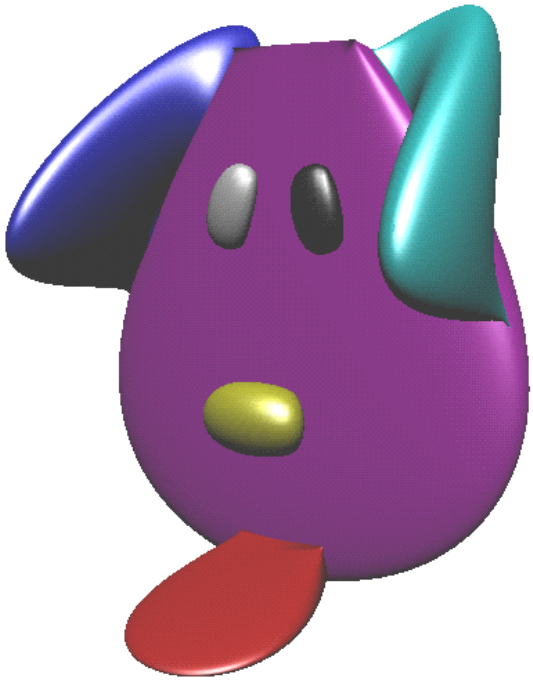

Figure 11: A pasted dog. By Clara Tsang.

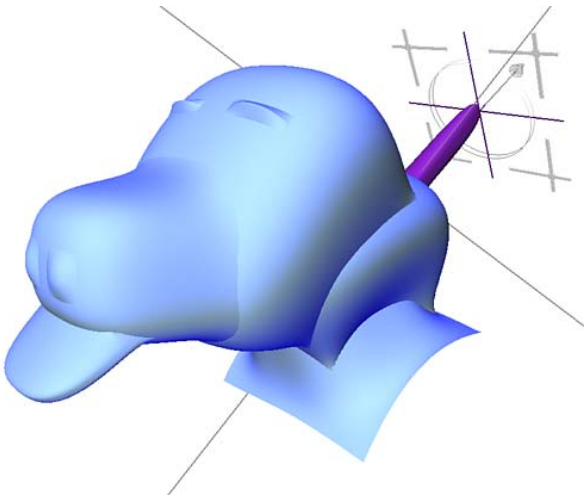

Figure 12: A pasted dog. By Leith Chan.