Virtual Sculpting with Haptic Displacement Maps

Robert Jagnow Julie Dorsey

Massachusetts Institute of Technology

Abstract

This paper presents an efficient data structure that facilitates high-speed haptic (force feedback) interaction with detailed digital models. Models are partitioned into coarse *slabs*, which collectively define a piecewise continuous vector field over a thick volumetric region surrounding the surface of the model. Within each slab, the surface is represented as a displacement map, which uses the vector field to define a relationship between points in space and corresponding points on the model's surface. This representation facilitates efficient haptic interaction without compromising the visual complexity of the scene. Furthermore, the data structure provides a basis for interactive local editing of a model's color and geometry using the haptic interface. We describe implementation details and demonstrate the use of the data structure with a variety of digital models.

Key words: Haptic, displacement map, sculpt, slab

1 Introduction

The pursuit of intuitive human-machine interfaces has led researchers to investigate the potential of haptic hardware – force-feedback devices capable of facilitating tactile interaction with digital models. This new generation of interface devices offers the promise of more immersive virtual environments that engage the tactual senses in much the same way that animation and sound engage the visual and auditory senses. But as with any fledgling technology, haptics comes with its own unique set of challenges.

Haptic devices require far faster update rates than visual output devices. For instance, the *PHANToM* system by Sensable Technologies Inc. requires updates at 1000 Hz – a constraint imposed by the inherent sensitivity of human tactile sensation. If this constraint is not met, unacceptable tactile artifacts, and possibly even hardware instability may result. Thus, the data structures that are useful for generating visually convincing scenes are often not efficient enough for haptic rendering.

The problem of generating an efficient haptic rendering system is exacerbated if we desire to modify the geometric data interactively, as it limits the amount of precomputation that we can perform on the model.

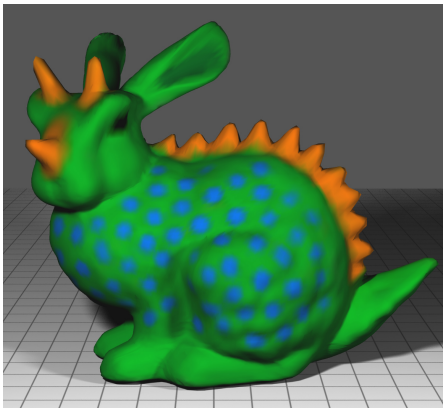
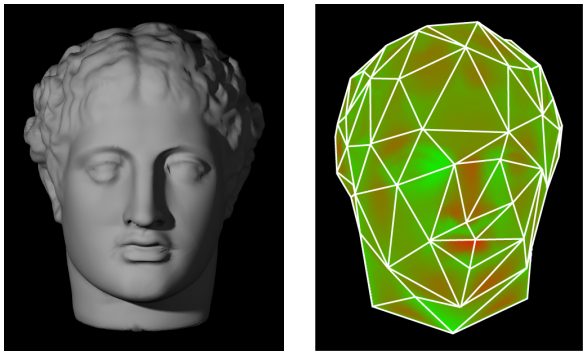


Figure 1: A familiar model, edited in a few minutes with a haptic device.

In this paper, we introduce a data structure that facilitates haptic rendering of complex scenes and accommodates local modifications to a model's surface characteristics, including, but not limited to, its geometry and color. We have implemented a system that uses the data structure for intuitive local editing of digital models.



(a) (b)

Figure 2: A geometrically detailed object (a) and its slab representation (b). Red regions in the displacement maps represent the largest displacements from the interior slab boundaries.

1.1 Related Work

This paper builds on a foundation of related research for improving the efficiency and flexibility of haptic rendering, as well as the efficiency of graphic rendering. The simplest method for decreasing the computational burden on a haptic system is to decrease the complexity of the digital model. In some instances, this can be accomplished with minimal impact on the apparent complexity of the scene. Morgenbesser [10] demonstrated that in some situations, a coarse polygonal mesh can suffice to represent the tactile feedback of a more complex geometric surface. He uses a *force shading* algorithm to provide the tactile illusion of a smoothly curved surface in much the same way that Phong shading provides a visual illusion of smoothness [11]. Normal vectors are precomputed at each vertex in a coarse mesh so that a local normal can be calculated as a weighted average of the normals at adjacent vertices. Morgenbesser performed user experiments using simple models consisting of three or fewer polygons, but did not address the haptic rendering of models of arbitrary complexity.

Other researchers considered alternative haptic data structures. McNeely et al. [9] implemented a voxel-based system to accelerate haptic collision detection in complex environments, but their geometric models were not intended for interactive modification. SensAble Technologies, Inc. [14] commercially produces a volume-based modeling system. Their system is powerful and intuitive, but exhibits common voxel rendering artifacts. Other related data representations include voxel-based systems with isosurfaces extraction [5], B-spline surfaces [3], and subdivision surfaces [6], all of which are suitable for interactive geometric editing of models with low to moderate visual complexity.

Here, we explore the use of displacement maps [2] for representing visually complex surfaces in a manner that is amenable to high-speed haptic interaction with limited geometric modification. A similar data structure was used by Lee et al. [8] for displacing subdivision surfaces, but not for the purpose of haptic rendering. The most closely related data structure is the *volumetric surface*, introduced by Dorsey et al. [4], which consists of a set of extruded quadrilateral slabs that form a thick skin of varying depth around the surface of a model. Dorsey used this representation for simulation of surface erosion rather than for haptic editing.

1.2 Goals and Contributions

In this paper, we introduce a data structure that facilitates high-speed haptic interaction with visually complex models. A model is represented as a collection of extruded triangular *slabs* – small volumetric regions in which the local geometry is expressed as an array of scalar displacements embedded between the inner and outer extents of the slab. Adjacent slabs are seamlessly stitched together to provide both visual and tactile continuity.

The advantage of this representation lies in its simplicity and flexibility. The natural hierarchical division between coarse and fine features allows for rapid computation of local surface features, making the data structure ideal for rapid collision detection for a haptic interface. Furthermore, since local features are represented by an array of scalar values, limited editing of the local geometry can be done rapidly by modifying the values in the displacement map.

In addition to the displacement values, supplementary arrays may be used to represent surface properties such as color, friction, hardness, or specularity. Other fields may represent the depth of various materials underneath or above the visible surface – materials that may be exposed or added by an edit operation.

In the remainder of this paper, we provide an overview of haptic displacement maps (Section 2), introduce a method for high-speed haptic collision detection and response (Section 3) and discuss methods for modifying the local geometry and color of the model (Sections 4 and 5). We then address graphic rendering concerns (Section 6) and describe our implementation, showing some example models (Section 7). Finally, we discuss the limitations of the algorithm and directions for future research (Section 8).

2 Haptic Displacement Maps

In the haptic displacement map data structure, slabs are arranged to completely and unambiguously enclose features of a detailed mesh while maintaining the full detail of the model by representing local features as displacements between the interior and exterior slab boundaries. This hybrid data structure offers the detail of a surface representation with the flexibility and physical intuition of a volumetric representation. Unlike many voxel representations, haptic displacement maps avoid common artifacts by orienting the slabs to coincide with the orientation of the local surface.

Figure 2 shows an object that appears to be highly tessellated with fine geometric details; however, the underlying representation is a simple slab mesh with the detailed features stored as displacement maps at each surface.

Each slab is defined as a region of space enclosed by six vertices: three on the interior of the model, and three on the exterior. The interior and exterior faces of the slab are planes defined by the three interior and exterior vertices respectively. The three other faces of the slab are bilinear patches defined by linearly interpolating be-

tween two interior and two exterior vertices. This results in a consistent definition of boundaries between slabs as shown in Figure 3.

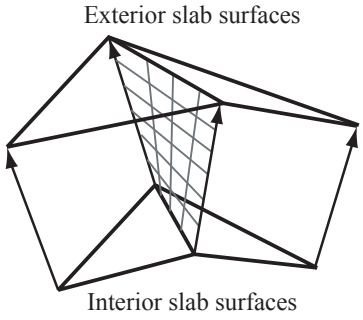


Figure 3: A bilinear patch between adjacent slabs.

Within each slab, displacement values are stored in a uniform triangular grid. In practice, this triangular grid is stored in the lower triangular region of a two-dimensional array, as shown in Figure 5c. The direction of surface projection is interpolated between the three rays at the corners of the slab, forming a detailed *submesh* as shown in Figure 4. Each displacement value along the edge of a slab is duplicated in the adjacent slab. T-vertices are disallowed in the slab mesh to insure correspondence of submesh vertices along the edges of adjacent slabs.

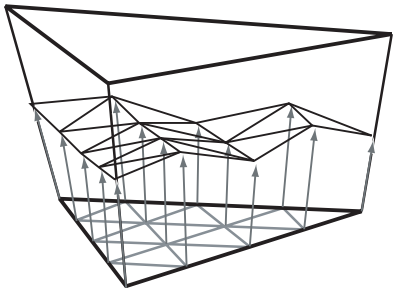


Figure 4: A detailed submesh formed by offsetting the interior slab surface.

3 Haptic Collision Detection and Response

One of the primary advantages of the displacement map data structure described in Section 2 is its efficiency for use with haptic collision detection. Within a slab, a continuous vector field directed from the interior plane to the exterior plane can be defined by linearly interpolating between the rays at the corners of the slab. At slab boundaries, this vector field remains continuous due to a consistent definition of the bilinear patches that separate adjacent regions. In this section, we demonstrate how this continuous vector field is used to define a relationship between arbitrary points in space and corresponding points on the model's surface – a mapping that can be used for efficient haptic collision detection and response.

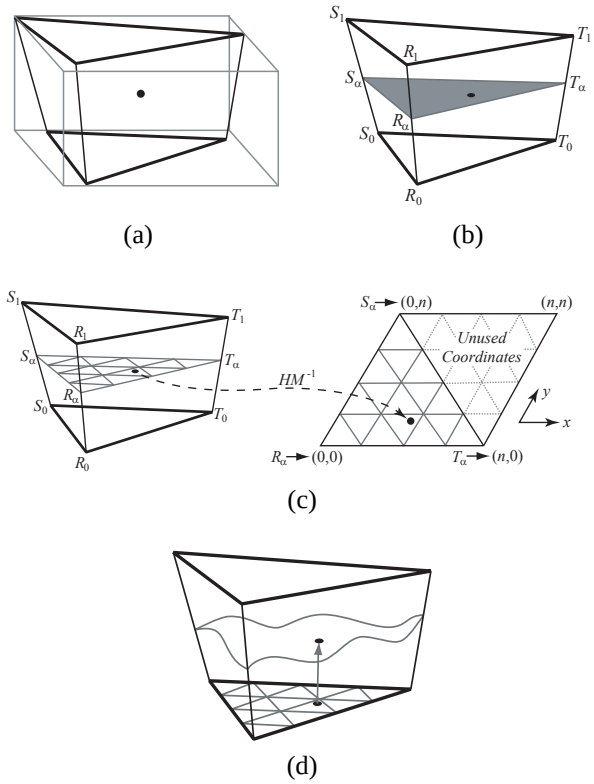


Figure 5: The haptic collision detection process: (a) Check the cursor against an axis-aligned bounding box. (b) Calculate the position of the interpolated plane containing the cursor. (c) Cast the cursor into a homogeneous coordinate space. (d) Reconstruct the surface.

3.1 Determining Slab Intersections

The first step in the haptic collision detection process is determining which slab, if any, contains the haptic cursor. Although the cursor is visualized as having volume, we treat it as a point for purposes of collision detection. The algorithm begins with a conservative check to see if the haptic cursor lies inside of an axis-aligned bounding box that encloses the slab (Figure 5a). If this succeeds, the next step is to determine how far the cursor has penetrated into the interior of the slab.

Each slab can be thought of as the region of space swept out by a triangle whose vertices are linearly interpolated from the interior slab plane to the exterior slab plane. At any distance between the interior plane, which has an interpolation value of zero, and the exterior plane, which has an interpolation value of one, this triangle de-

notes a surface of constant penetration α between zero and one. An example alpha-plane is shown in Figure 5b. To determine penetration, we solve for an α interpolation value that defines the plane at the same penetration depth as the haptic cursor.

Each alpha-plane is expressed in terms of its three corner vertices R_α, S_α, and T_α:

$$f(\alpha) = A_\alpha x + B_\alpha y + C_\alpha z + D_\alpha = 0, \qquad (1)$$

where

$$\begin{aligned}
A_\alpha &= \begin{bmatrix} 1 & R_{\alpha y} & R_{\alpha z} \\ 1 & S_{\alpha y} & S_{\alpha z} \\ 1 & T_{\alpha y} & T_{\alpha z} \end{bmatrix}, \\
B_\alpha &= \begin{bmatrix} R_{\alpha x} & 1 & R_{\alpha z} \\ S_{\alpha x} & 1 & S_{\alpha z} \\ T_{\alpha x} & 1 & T_{\alpha z} \end{bmatrix}, \\
C_\alpha &= \begin{bmatrix} R_{\alpha x} & R_{\alpha y} & 1 \\ S_{\alpha x} & S_{\alpha y} & 1 \\ T_{\alpha x} & T_{\alpha y} & 1 \end{bmatrix}, \\
D_\alpha &= -\begin{bmatrix} R_{\alpha x} & R_{\alpha y} & R_{\alpha z} \\ S_{\alpha x} & S_{\alpha y} & S_{\alpha z} \\ T_{\alpha x} & T_{\alpha y} & T_{\alpha z} \end{bmatrix}.
\end{aligned}$$

Each of R_α, S_α, and T_α are, in turn, defined as interpolants between the slab's extrema R_0, R_1, S_0, S_1, T_0, and T_1:

$$\begin{aligned}
R_\alpha &= (R_1 - R_0)\alpha + R_0 \\
S_\alpha &= (S_1 - S_0)\alpha + S_0 \\
T_\alpha &= (T_1 - T_0)\alpha + T_0.
\end{aligned}$$

When expanded, this series of equations yields quadratic expressions for A_α, B_α, and C_α, and a cubic expression for D_α:

$$\begin{aligned}
A_\alpha &= A''\alpha^2 + A'\alpha + A \\
B_\alpha &= B''\alpha^2 + B'\alpha + B \\
C_\alpha &= C''\alpha^2 + C'\alpha + C \\
D_\alpha &= D'''\alpha^3 + D''\alpha^2 + D'\alpha + D.
\end{aligned}$$

Each of the terms A, through D''' depends only on the six points that define a slab boundary. Thus, these constants can be precomputed for each slab. Looking again at expression (1), we can redistribute the terms to express the alpha-plane as a cubic function in α:

$$\begin{aligned}
f(\alpha) &= D'''\alpha^3 + [A''x + B''y + C''z + D'']\alpha^2 \\
&+ [A'x + B'y + C'z + D']\alpha \qquad (2) \\
&+ [A + B + C + D] = 0
\end{aligned}$$

Thus, given the location of the haptic cursor p, we can solve for α by using the general solution to the cubic or, more simply, by using an iterative approach.

To begin the iterative solution process, we check if the point p lies between the inner and outer extents of the slab, which have α-values zero and one respectively. If $f(0)$ and $f(1)$ have the same sign, then the haptic cursor lies outside of the slab. But if they have opposite signs, then p is bounded by the interior and exterior planes, in which case we can approximate a value for α by using a binary search algorithm. The search process repeatedly divides the search area in half, insuring that the upper and lower search bounds always yield opposite signs when entered into Equation 2. In practice, 15 iterations is sufficient, yielding an answer for α within an error of $1/2^{15}$.

Now that we have determined where the haptic cursor lies between the inner and outer slab boundaries, we can check its position against the bilinear patches at the other three slab faces. This is done by casting the point p into a homogeneous coordinate system where the boundary check becomes trivial. We begin by defining two matrices, M and H. The former transforms coordinates into a system with basis R_α S_α T_α. The latter transforms coordinates into the homogeneous displacement map coordinate system shown in Figure 5c. These matrices are defined as follows, where n is the width of the displacement map:

$$\begin{aligned}
M &= \begin{bmatrix} R_{\alpha x} & S_{\alpha x} & T_{\alpha x} \\ R_{\alpha y} & S_{\alpha y} & T_{\alpha y} \\ R_{\alpha z} & S_{\alpha z} & T_{\alpha z} \end{bmatrix} \\
\text{and} \quad H &= \begin{bmatrix} 0 & 0 & n \\ 0 & n & 0 \\ 0 & 0 & 0 \end{bmatrix}.
\end{aligned}$$

We can now use HM^{-1} to transform the point p from its position in the interpolated alpha-plane to the homogeneous coordinate system as illustrated in Figure 5c. M is invertible if R_α, S_α, and T_α are noncollinear, which should always be the case with a well-formed slab.

In the homogeneous coordinate system, a point $h = (h_x, h_y)^T$ is known to lie on the interior of the slab if it satisfies the following three conditions:

$$\begin{aligned}
h_x &\geq 0 \\
h_y &\geq 0 \\
h_x + h_y &\leq n.
\end{aligned}$$

All other points lie outside of the slab boundaries.

If no hierarchical optimization is desired, slabs are checked in sequence to determine the location of the haptic cursor, yielding a cost that scales linearly with the number of slabs in the model. Since slabs are designed to

be nonoverlapping, the algorithm can terminate collision checks as soon as any intersecting slab is found. Since the haptic cursor only moves small distances between time steps, this search can be optimized by first checking the slab found in the previous time step and its immediate neighbors.

The following sections describe how the homogeneous coordinates recovered by this process are used to reconstruct the surface of the detailed mesh and to provide appropriate haptic feedback.

3.2 Reconstructing the Surface

In the final stage of the collision detection process, we transform the haptic cursor position into a homogeneous coordinate system. The resulting 2D coordinates are used to index into the displacement map to determine the local surface displacement. Since the homogeneous coordinates of the haptic cursor are unlikely to coincide directly with integral coordinate values in the displacement map, the algorithm calculates the displacement as a weighted average of the values at the three nearest coordinates. This simple interpolation scheme yields piecewise linear connectivity between adjacent displacement map coordinates.

For a known homogeneous coordinate $h = (h_x, h_y)^T$ with a known displacement δ, the world coordinate of the corresponding point on the surface of the detailed mesh can be computed as follows. First, determine the vertices at the corners of the alpha-plane:

$$\begin{aligned} R_\delta &= (R_1 - R_0)\delta + R_0 \\ S_\delta &= (S_1 - S_0)\delta + S_0 \\ T_\delta &= (T_1 - T_0)\delta + T_0. \end{aligned}$$

Then, interpolate between these three points using the interpolation weights:

$$\begin{aligned} T_{weight} &= h_x/n \\ S_{weight} &= h_y/n \\ R_{weight} &= 1 - T_{weight} - S_{weight} \end{aligned}$$

to yield the surface point:

$$R_{weight} * R_\delta + S_{weight} * S_\delta + T_{weight} * T_\delta.$$

3.3 Calculating Cursor Penetration

Displacement values are stored as normalized scalars that indicate the relative depth of the surface between the inner and outer extents of the slab. Thus, if the α value for the cursor position (Section 3.1) is greater than the corresponding surface displacement value δ (Section 3.2), then the haptic cursor has not penetrated the surface of the detailed mesh, and no force needs to be returned to the haptic device. If, however, the α value is less than the surface displacement value, then the cursor has penetrated the mesh by a magnitude equal to the distance between the cursor position and the corresponding point on the surface of the mesh. In this case, a response force should be applied proportional to the depth of penetration, as indicated in the following section.

3.4 Applying a Response Force

To return a force to the user, a haptic system requires knowledge of the *surface contact point* (SCP), or *proxy position* – the point of interaction constrained to the surface of the model – and the surface normal at the SCP. Given this information, the system can account for surface spring and damping characteristics, as well as the effects of static and dynamic friction [12, 13].

Every small triangle in the displaced submesh of a particular slab can be considered to have its own local normal. But if the haptic feedback loop applies a response force in the direction of this local normal, at a magnitude proportional to the penetration distance, we encounter problematic behavior near major concavities and convexities in the surface of the model. Figure 6a shows five adjacent vertices of a displaced surface. As the position of the haptic cursor is moved along the dotted line, a force is applied in the direction of the local surface normal. The resulting surface feels as though it has a discontinuity at the peak. Furthermore, the ambiguous forces at the concavity can cause instabilities in the haptic device as opposing forces alternately attempt to achieve an unattainable equilibrium state.

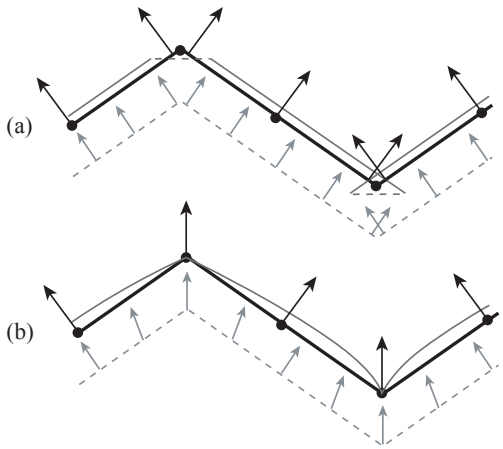


Figure 6: Surface reconstruction using (a) piecewise normals and (b) interpolated normals. Haptic cursor penetration through the model surface results in the reconstructed surface slightly above the polygons of the original mesh.

A better solution associates a normal vector with every vertex in the displaced mesh. This supplementary vector can be computed for each vertex as the average of the normals of the adjacent surfaces. At each point in the displacement map, an interpolated normal can be calculated as a weighted average of the normals of the three nearest vertices.

Using the penetration distance and the local surface normal, the collision response algorithm computes the surface contact point using a method similar to Morgenbesser's force shading algorithm. The surface contact point is offset from the haptic cursor position in the direction of the local surface normal, at a distance equal to the penetration depth. As can be seen in Figure 6b, the resulting point may not lie directly on the surface of the displaced mesh, but the magnitude and direction of the resulting force provide a convincing representation of the local geometry, as verified by Morgenbesser's user experiments [10].

4 Modifying Geometry

Since the coordinates of the displacement maps provide a uniform scaffolding that fully covers the surface of the model, they provide an ideal framework for locally modifying geometry or color attributes. The displacement maps within each slab have an inherent notion of adjacency and connectivity, and we can use precomputed adjacency relationships along slab boundaries to seamlessly span slabs as attributes are modified.

4.1 Removing Volume with Surface Clipping

3D sculpting or painting can be done by using a flood-fill method that walks from vertex to vertex within a constrained region of the surface, modifying geometry or color according to a desired function. To begin the sculpting process, our system first positions a tool that is used to modify the geometry, such as a simple sphere that clips away submesh vertices that intersect with its surface. As part of the collision detection process, the haptics algorithm already computes the local surface normal $\vec{n}$. To allow the tool to penetrate the surface by an amount proportional to the user-applied pressure, the sculpting routine positions the center of the tool as follows:

$$\dot{center} = \dot{cursor} + \vec{n} * (r + k),$$

where $\dot{cursor}$ designates the position of the haptic cursor, r is the desired tool radius, and k is the thickness of a resilient band of material at the model's surface. The resilience constant k allows the model surface to return feedback forces without being sculpted until sufficient force is applied (Figure 7).

Once the tool is positioned, the sculpting algorithm clips the geometry that falls inside of the sphere. It begins by "seeding" a recursive process at the homogeneous coordinate nearest to the haptic cursor. At each point in the submesh, the algorithm computes the surface projection line that passes through the homogeneous coordinate at the interior and exterior slab boundaries. If this line intersects the spherical sculpting tool at a point on the interior of the mesh, then the surface is clipped at the sphere boundary. The procedure continues by recursing to all adjacent displacement map coordinates that fall inside of the spherical tool.

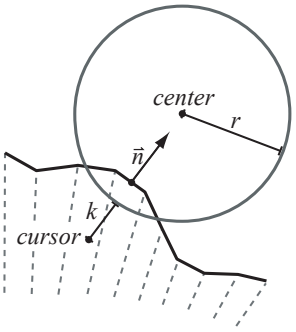


Figure 7: Positioning a spherical sculpting tool at the surface of a slab.

Since the sculpt routine adds some overhead to the basic collision-detection algorithm described in Section 3, the routine is not executed with each iteration of the haptic collision detection process. Instead, sculpting is performed at 10Hz – approximately once per every 100 iterations of the haptic loop – with minimal impact on the overall performance of the haptic rendering routine. The tactile artifacts that result from this temporal quantization can be reduced by increasing the frequency of the sculpting operations, at the expense of haptic rendering speed.

4.2 Adding Volume

By centering a spherical tool at the surface contact point, a similar routine can be used for *adding* material to the surface of the model. In this case, as the recursive algorithm marches along the surface of the mesh, a thin layer of material is deposited with a thickness linearly proportional to the distance from each submesh vertex to the center of the spherical tool. Thus, material is added in a conical shape, at a constant speed, regardless of pressure.

4.3 Limitations

Although haptic displacement maps are well-suited for limited local modification of a model's surface, the algorithms presented in this section are not intended for dramatic topological changes. For instance, slabs cannot be modified in such a way as to add a new undercut surface or to drill entirely through an object.

5 Haptic Painting

The challenge of precisely applying color to a three-dimensional model has been pursued for some time – initially using mouse-based interfaces [7], and more recently, using haptic interfaces [1, 6]. The advantage of the haptic interface is that it offers the intuitive simplicity of a paintbrush. With the haptic displacement map data structure, color can be applied to a surface by using a straightforward modification to the sculpting algorithm described in the previous section.

In our system, the virtual paintbrush is represented as a sphere, centered at the surface contact point. When the user touches the surface of the model, the paint process calculates the pressure p with which the paint should be applied. This is based on the penetration distance, normalized to a range between zero and one. Calculating the pressure as zero for small penetration distances allows the user to lightly touch the surface without modifying any surface properties.

As with the geometry modification routine, the recursive floodfill routine is "seeded" at the homogeneous coordinate nearest to the haptic cursor. At each iteration of the procedure, a new color c_n for the coordinate is calculated based on the old color c_o, the paintbrush color c_b, the pressure p, the brush radius r, and the distance δ between the center of the brush and the submesh vertex.

The color is applied with a normalized intensity k equal to $p*(1-\frac{\delta}{r})$. Thus, the most intense color is applied with greater pressure, nearest to the center of the brush, linearly decreasing out toward the edges of the sphere. The new color is computed by blending the existing color and the brush color as follows:

$$c_n = c_b * k + c_o * (1 - k).$$

The procedure continues by recursing to all adjacent texture coordinates that lie within the sphere of the paintbrush, as is done in the haptic sculpting algorithm. Like the sculpting algorithm, the paint routine is only executed 10 times per second to keep the basic collision detection routine unencumbered.

6 Graphic Rendering

Interactive visual feedback forms an important component of an effective haptic interface, so it is vital that the graphic rendering algorithm for the slab data be carefully optimized. Since the graphic and haptic routines require different rates of execution, each is run as a separate process to decouple the frame rate and haptic refresh.

Each slab maintains its own optimized display list to help accelerate graphic rendering. If the haptic process modifies a slab, a flag is set to indicate that the graphic process should generate a new display list for that particular region of geometry. Thus, the graphic process is concerned only with local updates for modified slabs while reusing display lists for unmodified regions of geometry. If graphic rendering is unacceptably slow, one alternative is to generate the visible mesh from a coarse subset of the coordinates in the displacement map.

Color is applied to the surface of each slab submesh using a texture map rather than by assigning per-vertex colors. This provides a speed advantage, and also decouples the resolutions of the geometric data and the texture data. Thus, color can be mapped to the surface of a slab at a higher density than the displacement data with negligible impact on rendering speed.

7 Implementation and Examples

We implemented our system on an SGI Octane with dual 250 MHz MIPS R10000 processors and 1.5 Gb memory, running IRIX 6.5. For the haptic interface, we used the A-model *PHANToM Premium* haptic device from SensAble Technologies, Inc. with the *GHOST* Software Development Toolkit, Version 3.0. Graphic rendering is done using OpenGL.

To initialize the slab data structure, we produced a decimated triangle mesh for each model. This mesh was grown inward and outward from the model surface to define the interior and exterior slab boundaries. For regions with thin features, like the ears in the bunny model, the resulting slabs were edited by hand to prevent overlap. The displacement maps were constructed by calculating where the projecting rays for the slab submesh intersect the original polymesh geometry.

Figures 1, 8, and 9 present polygonal models that were converted to the slab data structure and edited using the haptic sculpting system. The head and pumpkin models are formed with 200 slabs, and the bunny model with 150 slabs. Each slab stores displacement and color data with 16 values along each edge, forming a visually rendered mesh with 225 polygons per slab. All edited models were produced in approximately 15 minutes. In a typical editing session, the geometry modification routines perform 20,000 to 80,000 discrete edit operations on 1,000 to 3,000 individual slab vertices.

With the most complex of these models, the haptic system typically performs a single iteration of the collision detection and response process in under 0.65 milliseconds, thus utilizing 65% of the available processor time. The additional burden of performing sculpting or painting operations at 10Hz appears to have negligible impact on overall computational cost. With each of the example models, the graphic rendering system maintains a refresh rate of approximately 10 Hz, with minimal performance

degradation during sculpting or painting operations.

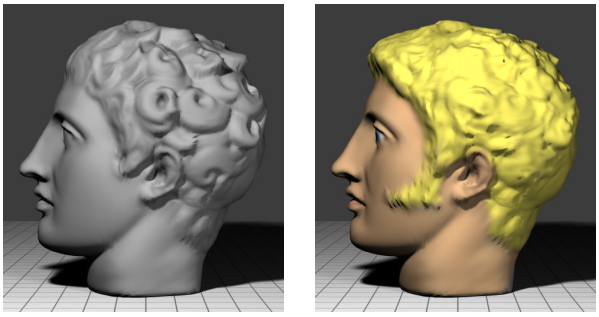


Figure 8: The head model is composed of 200 slabs, forming a visually rendered mesh with 45,000 triangles.

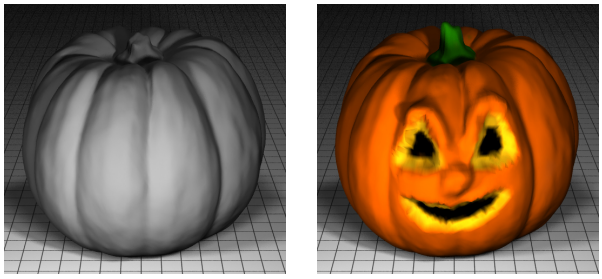


Figure 9: The pumpkin model is also formed with 200 slabs. The edited model was completed in approximately 15 minutes.

8 Conclusions and Future Work

Haptic displacement maps offer a step toward intuitive methods of interaction for rapid editing of complex models. Still, this data structure does not solve all the difficulties of digital editing. Slabs are effective for performing local modifications to a model in a thin region surrounding the surface of the initial mesh, but different techniques are needed for applying dramatic changes to the underlying topology of an object.

There are additional limits on the types of models that can be represented using haptic displacement maps. Slabs are designed to be used with "thick" models, with clear interior and exterior regions, and without thin or overlapping topological features.

Despite these restrictions, there are certain applications for which haptic displacement maps are particularly well suited. Many physical phenomena limit their influence to a thin region near the surface of a material, suggesting that a number of interesting effects might be produced via physical simulation or other forms of direct editing.

Other areas of future research include multiresolution meshes that can increase submesh detail where desired and multiple levels of detail obtained by rendering coarse subsets of submesh vertices.

Acknowledgements

We would like to thank Barb Cutler, Leonard McMillan, and Matthias Müller for helpful discussions. This work was supported by NSF grants CCR-9988535 and EIA-9802220 and by a gift from Pixar Animation Studios.

References

[1] Maneesh Agrawala, Andrew C. Beers, and Marc Levoy. "3D Painting on Scanned Surfaces," *1995 Symposium on Interactive 3D Graphics*, Monterey, CA, pp.145-150.

[2] Robert L. Cook. *Shade Trees*. In the Proceedings of *SIGGRAPH 1984*.

[3] Frank Dachille IX, Hong Qin, Arie Kaufman, and Jihad El-Sana. "Haptic Sculpting of Dynamic Surfaces," In the Proceedings of *Symposium on Interactive 3D Graphics 1999*, pp 103-110.

[4] Julie Dorsey, Alan Edelman, Henrik Wann Jensen, Justin Legakis, and Hans Kohling Pedersen. "Modeling and Rendering of Weathered Stone," In the Proceedings of *SIGGRAPH 1999*, pp 225-234.

[5] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. "Sculpture Virtuelle," In the Proceedings of *Les journées AFIG 99*, Reims, November 1999.

[6] A. Gregroy, S. Ehmann and M. C. Lin. "*inTouch*: Interactive Multiresolution Modeling and 3D Painting with a Haptic Interface," In the Proceedings of *IEEE Virtual Reality Conference 2000*.

[7] Pat Hanrahan and Paul Haeberli. "Direct WYSIWYG Painting and Texturing on 3D Shapes," In the Proceedings of *SIGGRAPH 1990*, pp.215-223.

[8] Aaron Lee, Henry Moreton, and Hugues Hoppe. "Displaced Subdivision Surfaces," In the Proceedings of *SIGGRAPH 2000*, pp 85-94.

[9] William A. Mcneely, Kevin D. Puterbaugh, and James J. Troy. "Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling," In the Proceedings of *SIGGRAPH 1999*, pp 401-408.

[10] H. B. Morgenbesser. *Force Shading for Haptic Shape Perception in Haptic Virtual Envornments*. M.Eng. Thesis, Massachusetts Institute of Technology, September 1995.

[11] B. T. Phong. "Illumination for Computer Generated Pictures," *Communications of the ACM*, 18(6), pp 311-317, June 1975.

[12] Diego C. Ruspini, Krasimir Kolarov and Oussama Khatib. "The Haptic Display of Complex Graphical Environments," In the Proceedings of *SIGGRAPH 1997*, pp 345-352.

[13] SensAble Technologies, Inc. *GHOST SDK Version 3: General Haptic Open Software Toolkit*. January, 1999.

[14] SensAble Technologies, Inc. *FreeForm Modeling System*. http://www.sensable.com/freeform. 2001.