

# A Haptic-Rendering Technique Based on Hybrid Surface Representation

Laehyun Kim  
Korea Institute of Science and Technology

Gaurav S. Sukhatme and Mathieu Desbrun  
University of Southern California

A haptic interface lets the user touch, explore, paint, and manipulate virtual 3D models in a natural way using a haptic display device. A haptic rendering algorithm must generate a force field to simulate the presence of these virtual objects and their surface properties (such as friction and texture), or to guide the user along a specific trajectory.

We can roughly classify haptic rendering algorithms according to the surface representation they use: geometric haptic algorithms for surface data,<sup>1,2</sup> and volumetric haptic algorithms based on volumetric data<sup>3</sup> including implicit surface representation.<sup>4</sup>

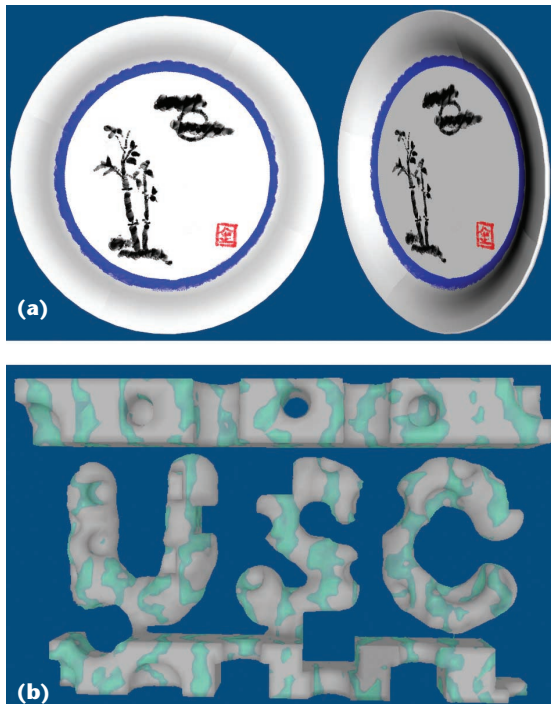
A novel haptic rendering technique using a hybrid surface representation addresses conventional limitations in haptic displays.

Our algorithm is based on a hybrid surface representation—a combination of geometric (B-rep) and implicit (V-rep) surface representations for a given 3D object, which takes advantage of both surface representations.<sup>5</sup> For the visual display, the geometric model can effectively represent the 3D model compared to volume rendering. Meanwhile, the implicit surface representation has many properties that address the limitations of current geometric haptic algorithms and introduce new haptic and visual effects. The benefits of using a hybrid representation include

- fast collision detection and contact point determination, allowing fast and precise force computation;
- avoidance of force discontinuity by interpolating local surface normals of grid points around the tool tip;
- no geometric gap due to small numerical errors in geometric models;
- stable and fast simulation of haptic texture by directly modulating potential values resulting in a new textured implicit surface;
- force computation independent of surface complexity since it's performed locally in a 3D regular grid instead of a geometric model; and
- easy offset surface computation for isosurfaces of the same potential with different isovalues. (We use an offset surface to handle thin objects and to simulate a magnetic surface.)

In addition to the basic haptic rendering technique initially presented in Kim et al.,<sup>5</sup> we discuss a haptic editing technique for decoration and local material properties based on our previous work.<sup>6</sup> Haptic decoration (see Figure 1a) lets the user paint directly on the surface, then sense the surface variation generated by the painted image (a process called *image-based haptic texturing*) based on the hybrid surface representation. Material editing enables editing of local material properties such as friction and stiffness. Our system also simulates assigned material properties in the fast haptic loop, which it saves directly into the volumetric representation rather than in the geometric representation. This

1 Examples using our haptic technique: (a) decorated Asian-style plate and (b) sculpted University of Southern California logo with mesh-based solid texture.



provides a reasonable approximation and fast computation of corresponding friction and stiffness values.

We also developed a virtual sculpting system based on a volumetric implicit surface as an alternative to existing digital sculpting implementations. For a fast visualization of the volumetric model, we use an isosurface extraction algorithm, based on adaptive polygonization, resulting in a mesh that effectively represents sharp edges with a smaller number of triangles than a uniform polygonization method. For a better visual effect, we present a mesh-based solid texturing method in which the mesh is adaptively refined according to the detail of solid texture.<sup>7</sup> Figure 1b shows an example created by our system.

Our haptic system consists of two basic parts: the visual and haptic processes run on a PC with dual Intel Xeon 2.4-GHz CPUs, 1 Gbyte of RAM, and FireGLX1 video card. The visual rendering implementation is OpenGL based and uses a 3-DOF Phantom haptic device for display.

### Implicit surface representation

Our haptic rendering algorithm uses the implicit representation of the external surface  $S$  of an object defined by the following implicit equation:  $S = \{(x, y, z) \in R^3 | f(x, y, z) = 0\}$ , where  $f$  is the implicit function (also called potential), and  $(x, y, z)$  is the coordinate of a point in 3D space.

If the potential value is 0, then the point  $(x, y, z)$  is on the surface. The set of all points for which the potential value is 0 defines the implicit surface. If the potential is positive, then the point  $(x, y, z)$  is outside the object (red points in Figure 2a). If  $f(x, y, z) < 0$ , then the point  $(x, y, z)$  is inside the surface (blue points in Figure 2a). The surface normals of an implicit surface can be obtained using the gradient of the implicit function as follows:

$$n = \nabla f / |\nabla f|$$

$$\nabla f = \left[ \frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz} \right]$$

To create a volumetric implicit surface representation from a geometric model, we use a fast closest point transform algorithm suggested by Mauch.<sup>8</sup> The algorithm computes the closest point to a surface and its distance from it by solving the Eikonal equation using the method of characteristics.

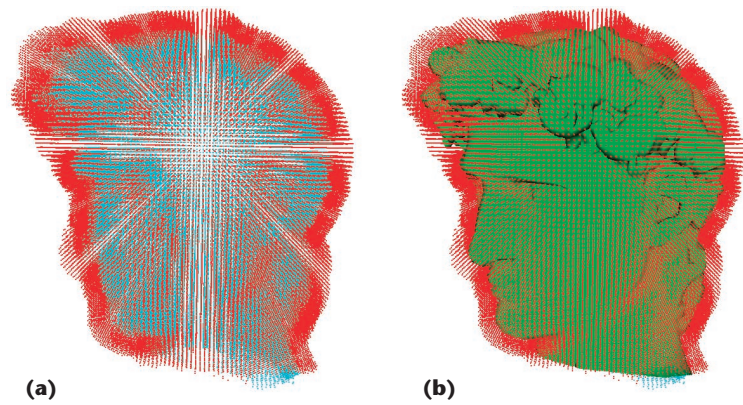
In addition to potential values to represent an implicit surface, the volumetric representation contains surface properties, and an index of the closest face on the geometric model. The face index works as a connection between the geometric model and implicit surface. We build the initial potential values and face indices offline during the conversion process from the geometric model before the haptic simulation can proceed.

### Haptic rendering model

In this section we give a detailed presentation of our basic haptic model.

#### Collision detection

In geometric haptic rendering, typically collision detection is not trivial to compute and is significantly



**2 Conversion from a geometric model to a volumetric implicit surface.**

(a) Using closest point transform. (b) Showing both geometric and implicit models.

affected by the surface complexity. McNeely, Puterbaugh, and Troy<sup>9</sup> suggested an approximate voxel-based method for 6-DOF haptic rendering that is independent of the size of geometric models. However, 3-DOF point contact haptic rendering requires the surface representation to provide even more accuracy than voxel level. Using the implicit representation, collision detection becomes trivial due to the inside/outside property discussed previously. We can obtain the proximity to the surface by interpolating the potential values of the eight neighbor points around the tool tip. If the distance becomes 0 or changes sign, a collision is detected. The computational complexity is constant regardless of the complexity of geometric models.

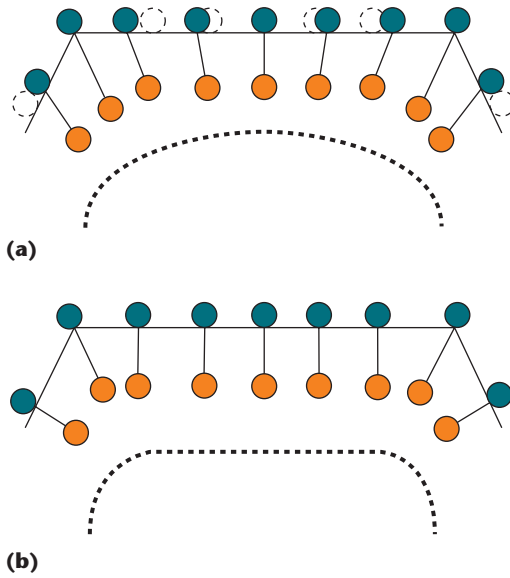
#### Frictionless model

To determine the force direction, the system first computes gradients of eight grid points around the tool tip in a regular 3D grid. Then the gradient at the tool tip position is computed by interpolating the neighbor's gradients. The resulting gradient becomes the direction of the force. The force discontinuity generally occurs when the direction and/or amount of the force suddenly changes around internal volumetric boundaries such as edges in a geometric model. Using the force-shading method, the system can avoid the force discontinuity, but it introduces a feeling of roundness on the flat surface (see Figure 3a, next page).

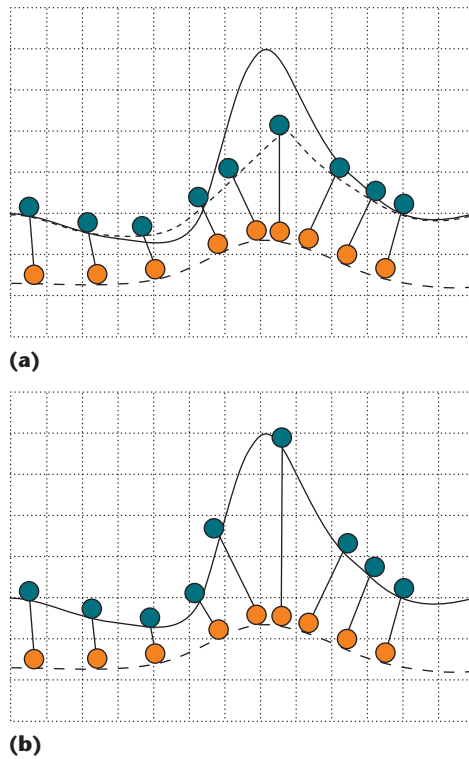
Our algorithm avoids the force discontinuity in a geometric model without a feeling of rounded surfaces (see Figure 3b), using an interpolation function in a volumetric model first introduced by Avila and Sobierajski.<sup>3</sup> Note that Avila's method is used for volumetric data. We employ this approach to simulate geometric models and improved it in terms of the force magnitude. Avila's approach approximated the force magnitude using the potential value. As a result, the surface feels smoother than it appears visually (see Figure 4a).

In our algorithm, the amount of force is proportional to the distance between the virtual contact point (VCP) and the tool tip position rather than potential values (see Figure 4b). The contact point is determined by moving the point of the tool tip toward the implicit sur-

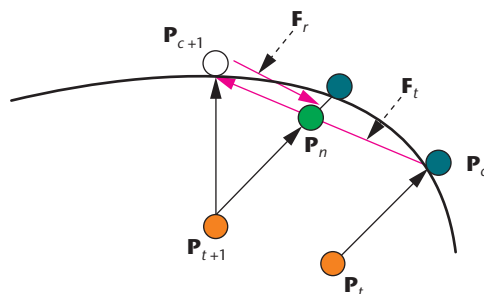
**3 Comparison of two haptic renderings without force discontinuity: (a) force shading and (b) our approach.**



**4 Force magnitude: (a) approximation in Avila's method and (b) our approach.**



**5 New virtual contact point due to friction.**



face by taking small steps along the computed force direction. Once the VCP is found, a spring-damper model computes the force vector that tries to keep the VCP and the tool tip at the same position (see Figure 5):

$$\mathbf{F} = k(p_c - p_t) - b\mathbf{V} \quad (1)$$

where  $\mathbf{F}$  is the force vector,  $k$  is stiffness,  $p_c$  is the contact point coordinate,  $p_t$  is the tool tip coordinate,  $b$  is viscosity, and  $\mathbf{V}$  is tool tip velocity (see Figure 5). Spring stiffness has a reasonably high value and viscosity prevents oscillations.

**Adding friction**

If the model has no friction (viscosity), it creates the feeling of a very slippery surface, since the direction of the force vector is almost perpendicular to the surface. Our algorithm implements friction by limiting the movement of the VCP like in the constraint-based method. The friction term takes into account a friction coefficient and the penetration depth.

$$f_v = f_c(1 + d(|p_{c+1} - p_{t+1}|)) \quad (2)$$

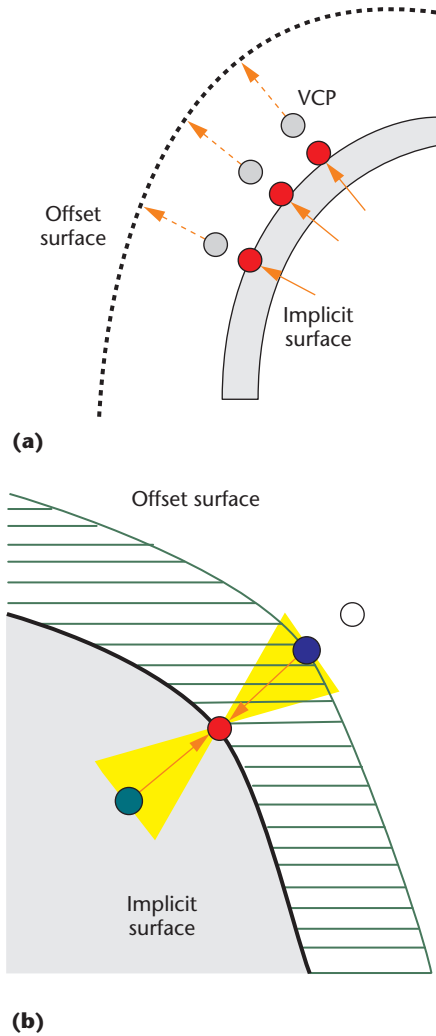
where  $f_v$  is the total friction term,  $f_c$  is the friction coefficient of material,  $d$  is a depth constant, and  $|p_{c+1} - p_{t+1}|$  is the penetration depth. Due to the depth term in Equation 2, users feel a stronger retarding force as they push the tool tip against the surface. By using the friction term  $f_v$ , we compute the retarding force  $\mathbf{F}_r$  and the new contact point as follows:

$$\begin{aligned} \mathbf{F}_t &= p_{c+1} - p_c \\ \mathbf{F}_r &= -f_v \mathbf{F}_t \\ p_n &= p_c + (\mathbf{F}_t + \mathbf{F}_v) \end{aligned}$$

where  $\mathbf{F}_t$  is the tangential force,  $p_{c+1}$  is the current contact point,  $p_c$  is the previous contact point,  $\mathbf{F}_r$  is the retarding force, and  $p_n$  is the new position of the tangential force after applying friction (the green point in Figure 5). The new position  $p_n$ , however, may not lie on the surface. We have to find the new contact point (the red point in Figure 5) on the surface that intersects with a ray along the new surface normal vector  $(p_n - p_{t+1})$ . The final force is calculated using Equation 1.

**Offset surface for thin objects**

In penalty methods, if 3D models don't have sufficient internal volume, the haptic system can't generate enough constraint force to prevent the tool tip from passing through the models. This problem can also occur in our approach. Constraint-based approaches<sup>1,2</sup> address this problem by limiting the movement of the VCP by the surface but introduce strong force discontinuity, resulting in a feeling of a bumpy surface around volume boundaries in geometric models. To give thin objects sufficient internal volume while avoiding force discontinuity, we use an offset surface that represents an iso-surface with a positive offset from the implicit surface. An additional volume between the offset surface and the original surface lets the system generate the appropriate constraint force for thin objects. Note that the

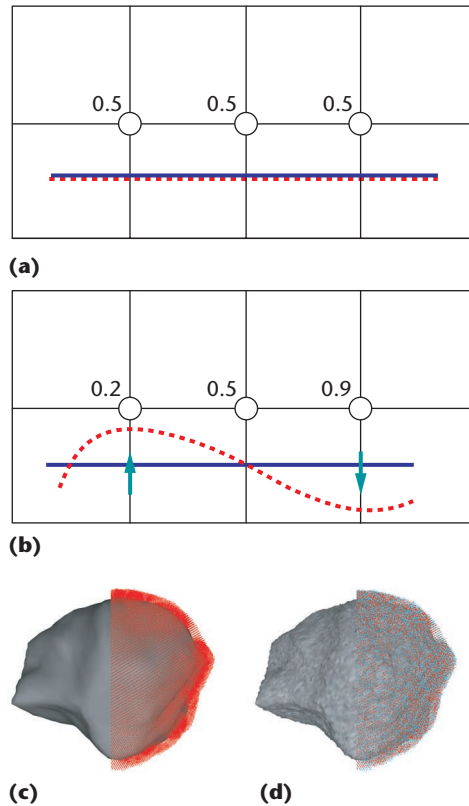


**6** Using an offset surface to (a) provide additional volume for thin objects or (b) simulate a magnetic surface.

force direction and surface properties are computed at the closest point on the original surface (red points in Figure 6a) and the visual contact point is still on the original surface. The force magnitude is proportional to the distance between the physical tool tip and the VCP on the offset surface.

### Magnetic surface

We propose a magnetic surface that attracts the tool tip to the closest point on the surface if the tool tip is within a magnetic field established between the offset surface and the original surface (the hatched area in Figure 6b). The magnetic surface forces the tool tip to stay in contact with the surface while the user explores complex 3D models. It helps users, especially if they are visually impaired, to explore the 3D model's shape without losing contact with the surface. The magnetic surface is also used to simulate the feeling of pulling for the adding operation in our haptic sculpting system. The direction of magnetic force (the red arrow within the magnetic field in Figure 6b) is determined by the surface normal at the position of the tool tip,



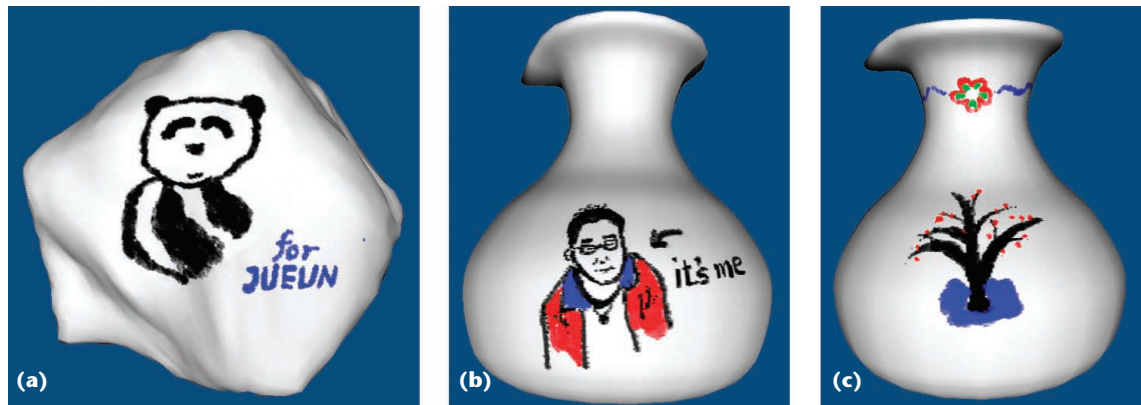
**7** Implicit haptic texturing. (a) An implicit surface (dotted red line) and geometric surface (blue line) before texturing. (b) The textured implicit surface obtained by modulating the potential values after texturing. (c) The model with smooth surface. (d) The textured model with rugged surface (light blue points indicate the grid points with textured potential values). The rugged surface is visualized using the marching cube algorithm.

and the amount (the yellow area within the magnetic field in Figure 6b) is proportional to the distance between the tool tip and the closest point on the original surface.

### Implicit haptic texturing

Earlier haptic texturing algorithms could require additional computations, might not render the surface with high stiffness, and can cause instability in the system. Furthermore, it's unclear how to extend these algorithms to support additional surface properties, such as friction. Our algorithm addresses these limitations by simulating haptic texturing, applying the texture value directly to the potential field, resulting in a new textured implicit surface. Then, the system simulates the textured implicit surface (the dotted red line in Figure 7b) rather than the geometric surface (the solid blue line in Figure 7b). We call it implicit haptic texturing. There is no need for preprocessing and additional computational cost. We demonstrate various synthetic textured models: For instance, we can simulate the rugged surface (see Figure 7d) by applying a Gaussian noise to the model with a smooth surface (see Figure 7c).

**8** Examples of haptic painting: (a) panda, (b) self-portrait, and (c) decorated pottery piece.



### Octree-based data structure

In our algorithm, the memory complexity of the volumetric representation is  $O(n^3)$ , where  $n$  is the size of one side in the 3D grid. However, only grid points around the surface are involved in the force computation. Therefore, to reduce the memory requirement, we use an octree-based data structure, avoiding the representation of empty portions of the space. In addition, you can quickly search and insert in this type of octree. The volumetric representation can be recursively decomposed into small elements with various sizes. Elements are placed at the leaves of the resulting structure.

### Haptic editing techniques

We apply haptic editing techniques for haptic decoration, material editing, and embossing/engraving.

#### Haptic decoration

Haptic decoration lets the user paint directly on the surface (haptic painting), and then to sense the surface variation of the painted image on the surface (image-based haptic texturing). When the user touches the surface to be painted, the system first finds 3D triangles within the brush volume—of which the center is the VCP on the surface—using the volumetric representation.

To find the seed face to start the rasterization, the system performs an intersection test between a line segment from the tool tip position to the VCP and faces around the VCP. The system then walks through all faces within the 3D brush volume starting from the seed face until no new face is found (this process is known as a *flood-fill* algorithm). However, this flood-fill algorithm might not find all triangles within the brush volume on the overlapped area of multiple objects or the surface with high frequency. For the volume-fill, the system checks all faces indicated by grid points within the brush volume. If it finds a new face within the brush volume, this face is used as a new seed face to perform another flood-fill (called a *volume-fill* algorithm).

The system then rasterizes corresponding 2D triangles in the texture map, one by one, using a standard scan conversion similar to inTouch.<sup>10</sup> Each texel's color is determined by a function of the brush size, brush color, fall-off rate, background color, and distance to the center of the brush volume during the triangle rasterization. The amount of applied force or a user-specified size deter-

mines the size of the brush volume. Figures 1 and 8 show some examples created by our haptic painting technique.

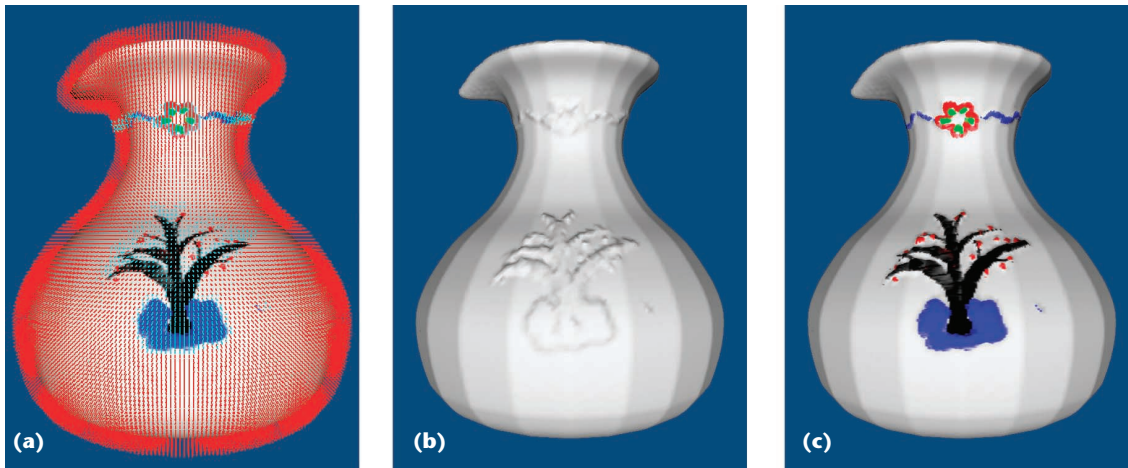
For image-based haptic texturing, we use our implicit haptic texturing method discussed previously to sense the surface variation of a 2D image. The system generates the height field to modulate potential values in 3D grid points around the image (see Figure 9a). The height value of each grid point is proportional to the gray-scale intensity of the texel determined by a barycentric coordinate. The user can correctly sense the image's surface variation based on the new geometry of the implicit surface (shown in Figure 10a). This method is independent of the shape complexity and texture mapping method unlike in previous systems.<sup>2,11</sup> The system also doesn't introduce the sudden change in the force amount and direction, since the force is generated based on the textured implicit surface instead of the geometric model (see Figure 10a). Furthermore, no additional computational cost and change in the algorithm are required.

#### Material editing

Users might expect haptically different surface properties when they touch a model with visually different material. However, in most previous haptic systems, a single surface property is applied over the whole 3D model.

We first suggest a material editing technique to edit and render the material map. The material map contains local surface properties such as stiffness and friction coefficients just as a texture map has texture values. In our method, the user performs material editing directly on the surface while doing haptic painting. During the editing of material properties, local material properties are saved at grid points within the 3D brush volume instead of vertices in geometric models (see Figure 10b).

The user also senses the assigned local surface properties on the surface immediately. As the tool tip moves on the surface, the system computes corresponding friction and stiffness values at the VCP by linearly interpolating local material properties around the VCP on the surface. This interpolation function gives a fast and reasonable approximation. Based on the volumetric representation, the material editing technique is independent of the geometric model. We can extend the material map to other haptic properties like surface roughness, for instance.



9 Image-based 3D embossing: (a) textured implicit surface, (b) embossed geometric model, and (c) embossed model with texture.

### Image-based 3D embossing and engraving

Using our haptic editing techniques, we easily create embossed and engraved 3D models. The user first generates an image on a 3D model using our haptic painting technique (see Figure 8c). We then get the textured implicit surface through image-based haptic texturing (see Figure 9a). The textured implicit surface is converted into a geometric model by the marching cube algorithm (see Figure 9b). Finally, the texture coordinate of a new geometric model is computed without surface parameterization using the texture information generated in the previous step (see Figure 9c).

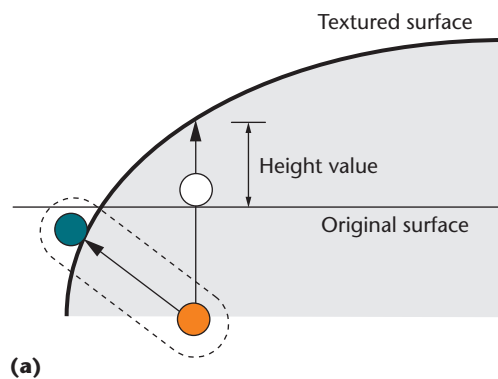
### Volume-based haptic sculpting technique

We developed a virtual sculpting system based on a volumetric implicit surface as an alternative to existing digital sculpting implementations.

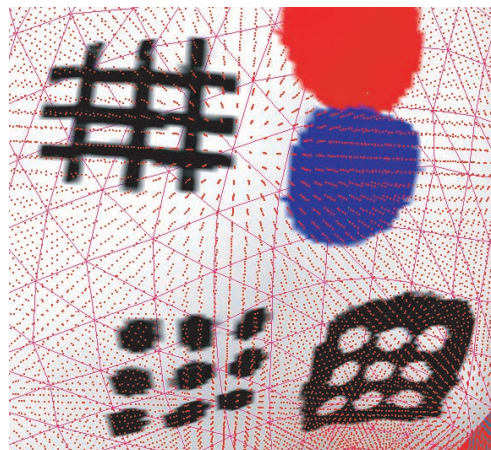
#### Polygonization method

For a fast visualization of the implicit surface, we use a polygonal rendering method instead of a ray-tracing method that would require longer rendering time. Uniform polygonization methods such as the marching cube algorithm suffers from large triangle count and a resolution limited by a fixed sampling rate, even though many sculpting systems adopted it.<sup>12,13</sup>

To address these limitations and enhance visual effects, we employ an adaptive polygonization method suggested by Velho.<sup>14</sup> This adaptive method consists of two steps. The initial polygonization step uses a uniform space decomposition to create the initial mesh that serves as the basis for adaptive refinement. The second step refines the mesh adaptively according to the surface curvature until the desired accuracy is achieved. The resulting mesh effectively represents sharp edges with a smaller number of triangles compared to uniform polygonization methods (see Figure 11d, next page). The volumetric implicit surface and generated mesh are saved into octree-based data structures to reduce the memory requirement and to dynamically manage the data similar to the scheme used by Barentzen.<sup>13</sup>



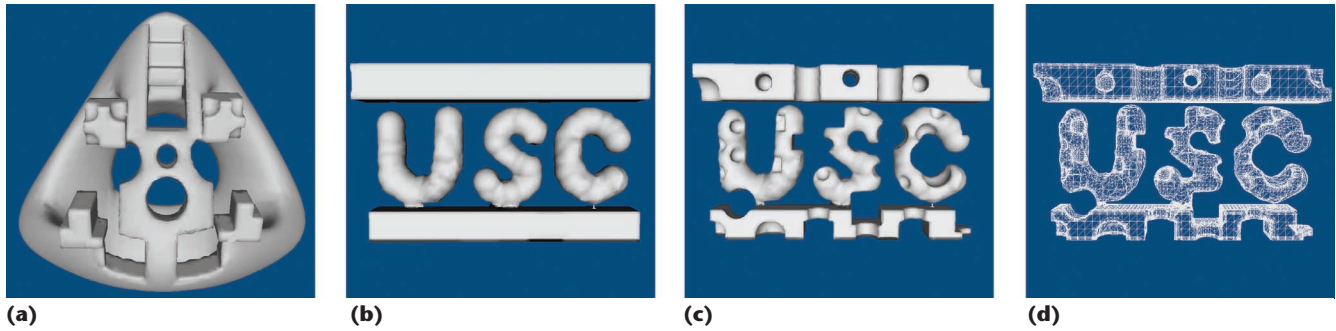
10 (a) Previous image-based haptic texturing method and our approach (dotted line). (b) Example of material editing. Painted areas in different colors have different surface properties saved in a 3D grid.



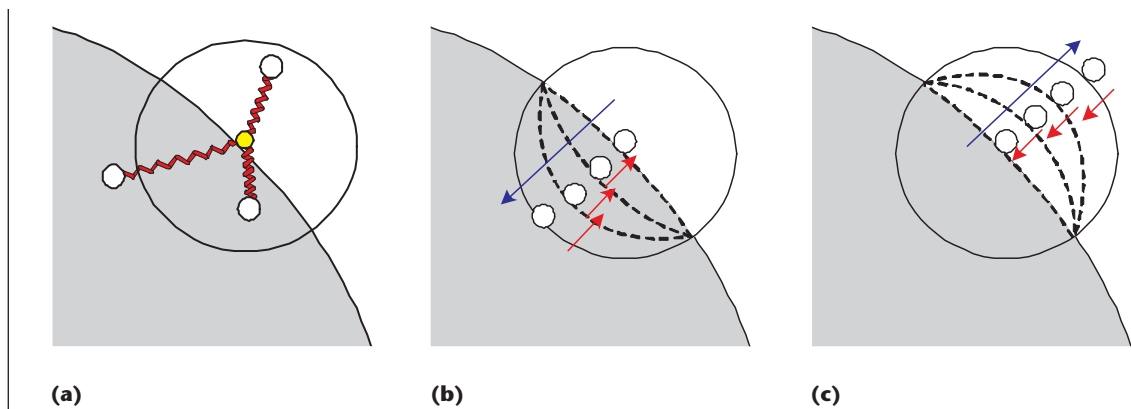
### Sculpting mode

In our system, there are two sculpting modes: haptic sculpting and block sculpting based on constructive solid geometry (CSG) Boolean operations. When we apply a sculpting tool to volumetric data, the system animates potential values around the local region, depending on the type, shape, and size of sculpting tool.

Most previous haptic sculpting systems<sup>10,15</sup> could not directly simulate the deformation of physical models since the update rate of physical model (30 Hz) is slow-



**11** 3D models created by our haptic sculpting system. (a) The model created in block sculpting mode. (b) The USC logo created in haptic sculpting mode. (c) The model after applying more sculpting tools to (b). (d) The mesh model of (c), which is adaptively polygonized according to the surface complexity.



**12** Spring-based approach and our method. (a) Spring-based sculpting. (b) Carving operation in haptic rendering (blue arrow indicates tool movement direction and red arrow the responsive force). (c) Adding operation based on magnetic surface.

er than the haptic force update rate (1 kHz). Instead, they used a spring-based force established between the initial contact point and the tool position (see Figure 12a). To bridge the performance gap between two processes, our system haptically simulates intermediate surfaces generated by interpolating potential values from the current to target values around the sculpting tool resulting in smooth force rendering (see Figures 12b and 12c). The target values are updated every graphical frame using CSG Boolean point-set operations. The animation speed of the intermediate surface is proportional to the user-applied force and inversely proportional to the material’s local stiffness. The system visually updates the physical model based on the current intermediate surface in the next graphical frame instead of the target surface.

Unlike the pushing operation, the pulling operation makes the tool tip leave the surface and loosens the contact point immediately. To simulate a feeling of pulling, we use the magnetic surface, discussed previously, which attracts the tool tip to the closest point on the implicit surface being sculpted. It lets the user intuitively perform the adding operation (see Figure 12c).

Pressing down the switch on the stylus starts deformation; releasing the switch stops this function. Figures 11b and 13c show examples created in haptic sculpting mode.

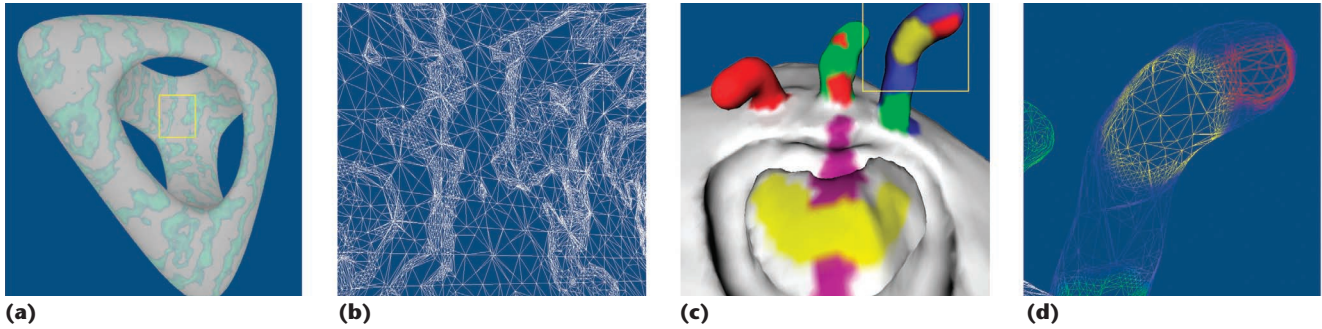
The haptic force often hinders the user from performing accurate and detailed sculpting. In block sculpting mode, the system turns off the haptic force, letting the user freely move in any direction in 3D space.

To sculpt a virtual object in this mode, the user first moves the sculpting tool onto the desired region using visual feedback. Simple wire frames showing the boundary to be sculpted represent the sculpting tools. Pressing down the switch on the stylus either adds or carves material from the surface. A CSG Boolean operation computes target potential values at grid points to be sculpted. Figure 11a and c show examples sculpted in block sculpting mode.

**Mesh-based visual effects**

We use the adaptive polygonization method to enhance visual realism for a real-time sculpting system. The mesh is adaptively refined according to the detail of color in solid texturing and painting.

To avoid texture distortion for the sculpted model, we employ the solid texture method introduced by Perlin,<sup>7</sup> which uses a procedural function to calculate the color for each point, instead of a lookup texture image. However, solid texture rendering—such as ray marching and solid mapping—requires high computational power or a large texture memory, which aren’t suitable for a real-time sculpting system. Our system applies color value,



**13** Mesh-based visual effects. (a) Model with a marble-like solid texture. (b) Mesh model within the box of (a). (c) Painted model after sculpting. (d) Mesh model within the box of (c).

generated from a noise function, to the surface by assigning per-vertex colors to speed up the solid texturing process. The problem is that the color of each vertex blends with nearby vertices, and much of the detail of the solid texture is often lost depending upon the surface complexity. To address this limitation, we use the adaptive polygonization approach, which adapts the mesh to the detail of the solid texture. If the difference between per-vertex colors along an edge is greater than a desired accuracy, this edge is split (see Figure 13b) recursively. As a result, we obtain a clear solid texture on the surface in real time without special hardware and graphical rendering methods (see Figures 1b and 13a).

The user also can paint directly on the sculpted model. The applied color is saved at grid points around the painting tool and used to determine vertex color while the mesh is adaptively polygonized in a manner similar to the mesh-based solid texturing. This makes the color boundaries clear (see Figures 13c and 13d).

### Conclusion

We plan to enhance our haptic sculpting system by adding various sculpting tools such as smoothing and stamping, modeling by sketching, and a cut-and-paste operator for fast editing. We also plan to apply this technique to various applications such as medical training

### Related Work

We discuss other researchers' approaches to haptic rendering.

#### Haptic rendering algorithms

Traditional haptic rendering methods can be classified into roughly three groups according to the surface representation used, that is, geometric, volumetric (including implicit), or parametric representation.

For geometric models, Zilles and Salisbury<sup>1</sup> introduced a constraint-based *god-object* (a virtual contact point on the surface) method that constrains the movement of a god object to stay on the object's surface. Ruspini et al.<sup>2</sup> proposed a more general constraint-based method and simulated additional surface properties such as friction and haptic texture. The constraint-based approach, however, still suffers from force discontinuity around volume boundaries such as edges and vertices and does not scale well with the complexity of the object's surface.

For volumetric data, the force field can be computed directly from the volume data without conversion to a geometric model. Avila and Sobierajski<sup>3</sup> used the gradient of the potential value to calculate the direction of the force, and the amount of force was linearly proportional to the potential value. Salisbury and Tarr<sup>4</sup> suggested a haptic rendering algorithm for implicit surfaces defined by analytic functions. This algorithm is, however, not applicable to geometric models with arbitrary shapes, such as the *David* model in Figure 2 of the main text.

Thomson et al.<sup>5</sup> introduced a haptic system to directly

render a NURBS model, taking advantage of the mathematical properties of parametric surface representation.

#### Haptic texturing

Haptic texturing enhances haptic realism just as graphical texture enriches visual realism. Minsky<sup>6</sup> first demonstrated simulation of haptic textures by lateral force fields proportional to the local gradient of the textured surface on a 3-DOF planar haptic device. Image-based haptic texturing lets the user feel the height variation of a 2D graphical texture. Ho et al.<sup>7</sup> implemented an image-based haptic texturing by perturbing the direction and magnitude of the surface normal computed by a local gradient of the height value generated using a two-stage texture mapping technique.

#### Haptic painting

A haptic painting system provides a natural painting interface, just as if the user paints on a real object. Johnson et al.<sup>8</sup> first proposed a haptic painting system, which paints directly onto a trimmed NURBS model with a haptic interface. While the user paints on a 3D model, the system updates the 2D texture map and adaptively adjusts the brush size using the surface parameterization to mitigate the texture distortion between the 2D texture image and the 3D model. Gregory et al. developed a haptic painting system (called *inTouch*) for polygonal models.<sup>9</sup> To avoid the texture distortion problem, they use a standard 2D scan

*continued on p. 74*

simulators and digital art in which realistic haptic experience is crucial. ■

### Acknowledgment

The work discussed here was supported in part by the NSF-funded Integrated Media Systems Center (EEC-9529152).

### References

1. C. Zilles and J.K. Salisbury, "A Constraint-Based God-Object Method for Haptic Display," *Proc. IEEE Int'l Conf. Intelligent Robots and Systems*, IEEE CS Press, 1995, pp. 31-46.
2. D.C. Ruspini, K. Kolarov, and O. Khatib, "The Haptic Display of a Complex Graphical Environment," *Proc. ACM Siggraph*, vol. 1, ACM Press, 1997, pp. 295-301.
3. R.S. Avila and L.M. Sobierajski, "A Haptic Interaction Method for Volume Visualization," *Proc. IEEE Visualization*, IEEE CS Press, 1996, pp. 197-204.
4. K. Salisbury and C. Tarr, "Haptic Rendering of Surfaces Defined by Implicit Functions," *Haptic Interfaces for Virtual Environment and Teleoperator Systems*, ASME, 1997, pp. 61-68.
5. L. Kim et al., "An Implicit-Based Haptic Rendering Technique," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems*, IEEE CS Press, 2002, pp. 2943-2948.
6. L. Gaurav, S. Sukhatme, and M. Desbrun, "Haptic Editing for Decoration and Material Properties," *11th Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, IEEE CS Press, 2003, pp. 213-221.
7. K. Perlin, "An Image Synthesizer," *Proc. ACM Siggraph*, vol. 19, no. 3, ACM Press, 1985, pp. 287-296.
8. S. Mauch, *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*, doctoral dissertation, Caltech Univ. Press; <http://etd.caltech.edu/etd/available/etd-05202003-170423/>.
9. W.A. McNeely, K.D. Puterbaugh, and J.J. Troy, "Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling," *Proc. ACM Siggraph*, ACM Press, 1999, pp. 401-408.
10. A.D. Gregory, S.A. Ehmann, and M.C. Lin, "In Touch: Interactive Multiresolution Modeling and 3D Painting with a Haptic Interface," *Proc. IEEE Virtual Reality*, IEEE CS Press, 2000, pp. 7-14.
11. C.-H. Ho, C. Basdogan, and M.A. Srinivasan, "Efficient Point-Based Rendering Techniques for Haptic Display of Virtual Objects," *Presence*, vol. 8, no. 5, 1999, pp. 447-491.
12. T.A. Galyean and J.F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique," *Proc. ACM Siggraph*, ACM Press, 1991, pp. 267-274.
13. J. Andreas Barentzen, "Octree-Based Volume Sculpting?" *Proc. IEEE Visualization*, IEEE CS Press, 1998, pp. 9-12.
14. L. Velho, L.H.D. Figueiredo, and J. Gomes, "A Unified Approach for Hierarchical Adaptive Tessellation of Surfaces," *ACM Trans. Graphics*, vol. 18, no. 4, 1999, pp. 329-360.
15. K.T. McDonnell, H. Qin, and R.A. Wlodarczyk, "Virtual

*continued from p. 73*

conversion in texture space. The scan conversion step updates each texel in the 2D triangle in the texture space according to a brush function based on its corresponding 3D position.

### Volume-based sculpting

Galyean and Hughes<sup>10</sup> introduced a voxel-based approach to volume sculpting that used the marching cube algorithm to display the model. Barentzen<sup>11</sup> proposed using octree-based volume sculpting to reduce the memory requirement. However, these approaches have limitations such as low resolution due to the volume's data size or the displayed surface's large number of triangles. Multiresolution<sup>12</sup> and adaptive approaches<sup>13</sup> have addressed these limitations, resulting in high image quality with fewer triangle numbers. A mouse-based computer interface in the 3D sculpting system is unnatural and inefficient. Avila<sup>3</sup> first introduced a volume-based haptic sculpting system that lets the user intuitively sculpt volumetric data using a haptic interface.

### References

1. C. Zilles and J.K. Salisbury, "A Constraint-Based God-Object Method for Haptic Display," *Proc. IEEE Int'l Conf. Intelligent Robots and Systems*, IEEE CS Press, 1995, pp. 31-46.
2. D.C. Ruspini, K. Kolarov, and O. Khatib, "The Haptic Display of a Complex Graphical Environment," *Proc. ACM Siggraph*, vol. 1, ACM Press, 1997, pp. 295-301.
3. R.S. Avila and L.M. Sobierajski, "A Haptic Interaction Method for Volume Visualization?" *Proc. IEEE Visualization*, IEEE CS Press, 1996, pp. 197-204.
4. K. Salisbury and C. Tarr, "Haptic Rendering of Surfaces Defined by Implicit Functions," *Haptic Interfaces for Virtual Environment and Teleoperator Systems*, ASME, 1997, pp. 61-68.
5. T.V. Thompson II and E. Cohen, "Direct Haptic Rendering of Complex Truncated NURBS Models," *Haptic Interfaces for Virtual Environment and Teleoperator Systems*, ASME, 1999.
6. M.D.R. Minsky, *Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display*, doctoral dissertation, MIT Press, 1995.
7. C.-H. Ho, C. Basdogan, and M.A. Srinivasan, "Efficient Point-Based Rendering Techniques for Haptic Display of Virtual Objects," *Presence*, vol. 8, no. 5, 1999, pp. 447-491.
8. D. Johnson et al., "Painting Textures with a Haptic Interface," *IEEE Virtual Reality Proc.*, IEEE CS Press, 1999, pp. 282-285.
9. A.D. Gregory, S.A. Ehmann, and M.C. Lin, "In Touch: Interactive Multiresolution Modeling and 3D Painting with a Haptic Interface," *IEEE Virtual Reality Proc.*, IEEE CS Press, 2000, pp. 7-14.
10. T.A. Galyean and J.F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique," *Proc. ACM Siggraph*, ACM Press, 1991, pp. 267-274.
11. J. Andreas Barentzen, "Octree-Based Volume Sculpting?" *Proc. IEEE Visualization*, IEEE CS Press, 1998, pp. 9-12.
12. J. Hua and H. Qin, "Haptic Sculpting of Volumetric Implicit Functions," *Proc. 9th Pacific Conf. Computer Graphics and Applications*, IEEE CS Press, 2001.
13. R.N. Perry and S.F. Frisken, "Kizamu: A System for Sculpting Digital Characters" *Proc. ACM Siggraph*, ACM Press, 2001, pp. 47-56.

Clay: A Real-Time Sculpting System with Haptic Toolkits," *Proc. ACM Symp. Interactive 3D Techniques*, ACM Press, 2001, pp. 179-190.



**Laehyun Kim** is a researcher in the systems technology division at the Korea Institute of Science and Technology. His research interests include haptics, computer graphics, and virtual reality. Kim received a BS from Hanyang University in Korea in material engineering, an MS from Yonsei University in computer science, and a PhD in computer science from the University of Southern California.



**Gaurav S. Sukhatme** is an assistant professor of computer science and electrical engineering at the University of Southern California where he directs the Robotic Embedded Systems Laboratory. His research interests include distributed robotics and

sensor-actuator networks. Sukhatme received a B.Tech. in computer science and engineering from IIT Bombay, and an MS and PhD in computer science from USC. He is a member of the AAAI, the IEEE, and the ACM and is a 2002 NSF CAREER awardee.



**Mathieu Desbrun** is an assistant professor of computer science at the University of Southern California, and a visiting associate at Caltech. His research interests include the study of discrete geometry, consisting of animation and simulation of 3D objects, processing of polygonal meshes, and theoretical work on the foundation of computations on discrete manifolds. Desbrun received a PhD in computer science from the National Polytechnic Institute of Grenoble. He received a 2001 NSF CAREER Award in 2001 and the 2003 ACM Siggraph Significant New Researcher Award.

Readers may contact Laehyun Kim at the Korea Inst. of Science and Technology, 39-1 Hawolgok-dong Seongbuk-gu Seoul, 136-791, Korea; laehyunk@kist.re.kr.

IEEE

# MultiMedia

2004

## Editorial Calendar

### January-March

Content Repurposing

With more than 600 device profiles available today for accessing online content, handcrafting content for each device, network, and usage, as well as each of their combinations is unmanageable. Content repurposing tackles this problem by taking content designed for a particular scenario and automatically repurposing it to fit another. Fundamental to this approach is the need to maintain a single copy of the content in its original form and to repurpose the content to fit the desired scenario in real time and in an automated fashion.

### April-June

Digital Multimedia on Demand

Emerging multimedia systems are expected to support a wide range of applications and integrate a wide array of data (textual, numeric, audio, video, graphics, speech, music, animation, handwriting, and so on). In many multimedia applications—such as video on demand, digital libraries, and home-based shopping—a common feature is the requirement for storing, retrieving, and transporting these data types over a network upon user request. This particular issue will target surveys and papers related to directions and advances made in the scientific and commercial fields for digital multimedia on demand associated with the multimedia user's needs.

### July-September

Multisensory Communication and Experience through Multimedia

Successful communication involves a transferral of experience. Transferring multimodal data without concern for whether this information can transcend into a consistent multisensory experience for the receiver doesn't address the full spectrum of communication. This issue focuses on real forms of communication involving all or most of our senses and on the role that multisensory experiences can play in the development of multimedia technologies and content.

### October-December

Multimedia Visions

Multimedia is unique in its applicability, both pulling from and lending itself to many fields. This issue sheds light on what multimedia is and can be, with the latest research from leading-edge developers and scientists. Whether discussing evolving standards, the impact of multimedia, or posing new avenues of thought, each article proffers a unique vision of a multimedia future.