HAVE 2008 – IEEE International Workshop on
Haptic Audio Visual Environments and their Applications
Ottawa – Canada, 18-19 October 2008

Interactive 3D Haptic Carving using Combined Voxels and Mesh

Jeff Williams[1], Gabriel Telles O'Neill [1], Won-Sook Lee[1]
[1]University of Ottawa

Abstract *– We present a system to simulate the realistic real-time carving of rigid objects. The system allows the user to move a virtual burr-like tool to carve the object using a haptic device. This tool removes the rigid volume of the object as it intersects the object, and the results are displayed in 3D. In our system the volume is represented using voxels, but is displayed using a triangle mesh.*

Keywords *– carving, haptic, interactive, 3D, computer animation*

I. INTRODUCTION

One of the goals of animation is to present to the viewers a believable representation of a physical process. When animating real-world processes it is often important to precisely capture the essences of the characteristics of the process. With this in mind, we developed a system to simulate the realistic real-time carving of rigid objects.

The system allows the user to move a virtual burr-like tool to carve the object using a haptic device. This tool removes the rigid volume of the object as it intersects the object, and the results are displayed in 3D. The methods we used, though, are applicable to many areas of sculpting animation. These include virtual sculpting simulation, modelling, games, and medical applications.

When simulating sculpting, it is necessary to maintain a representation of the remaining object volume, to render it in an accurate as well as tactically and visually appealing manner, and to update the visual and haptic representation at rates suitable for animation. In our system the volume is represented using voxels, but is displayed using a triangle mesh because direct visualization of voxels is unrealistic and unappealing. The voxel representation is used to compute the intersection of the cutting tool with the object being carved and to provide haptic feedback to the user. When voxels are removed by carving, the affected regions of the corresponding triangle mesh are updated.

The process of updating the corresponding triangle mesh due to changes in the voxel set is performed using a novel extension of the well-known Ball Pivoting Algorithm, which we call the Dynamic Ball Pivoting Algorithm.

II. RELATED WORK

Carving objects in 3D has seen quite a bit of research in the past. The work can be divided into three general areas, based on the data structure used to represent the volume: boundary representation methods, implicit function methods, and spatial partitioning methods.

Boundary representations approximate 3D objects by dividing their boundary into a number of two dimensional faces. These representations are often called meshes. There are a number of carving methods that operate directly on the object mesh. The earliest methods are implementations of Boolean set operations on 3D objects. These operations are commonly implemented using Binary Space Partitioning (BSP) Trees [1][2]. It is possible to define Boolean set operations between two BSP trees (that is, union, intersection, and difference) by calculating the intersections between the planes represented by each tree. Using BSP tree methods it is possible to represent both the object to be carved and carving tool as BSP trees and compute the difference between them at each time interval. However these methods are difficult to implement reliably in the presence of round-off errors. [3] Since carving is an iterative process where small amounts of volume are repeatedly removed, any errors introduced at one cutting step will cause further errors later on.

In [4] the authors demonstrate a method that operates on the polygon mesh boundary of the shape, and uses a Discontinuous FFD to represent the effect of the cutting blade, deforming the mesh when incising. Extra polygons are added to the mesh under the incision before deforming to help preserve the mesh shape. However this method is more applicable to simulating incision rather than carving since no actual volume is removed during a cut.

Implicit function methods have received much attention in recent years. These methods represent the 3D solid as a set of implicit functions. The boundary of the solid is taken to be an implicit surface, which is the level-sets of the implicit functions; that is, the set of all values in the function domain that yield a certain constant range value:

$$\{(x, y, z) \mid f(x, y, z) = c\}$$

The equations are calculated so that their level-set is constrained to the object's surface. Carving using these methods consists of adjusting the equations to account for the volume removal.

The classical implicit method representation is found in the computational solid geometry (CSG) representation. CSG represents the object as the combination of a set of solid primitives. A primitive can be a parametric equation of a quadric surface (a plane, sphere, cone, cylinder, or paraboloid), or some other simple regular prism such as a cube. These primitives form the leaves of a binary tree, in which internal nodes represent rigid transformations (translations, rotations, or scales) of the children nodes, or represent regularized Boolean set operations (union, intersection, or difference) on the left or right sub-tree.

Performing cutting using CSG is simply a matter of representing the solid to be cut as a CSG sub-tree whose parent is a difference operation with a second sub-tree which is the union of all the positions of the cutting tool. However a major drawback of this cutting approach is that the solid to be cut must first be converted to a CSG representation, which is difficult and non-automated for complex objects such as human bones. In addition, the method does not scale well as the number of cuts increases since the tree representing each cut must be added to the main tree of the object. This makes the solid's tree grow large quickly.

In [5] and [6] the authors blend trivariate uniform Bspline functions to represent the surface. Sculpting is done by modifying the scalar coefficient values in the mesh so that the values are inside or outside the object (higher or lower than the level-set value). In [5] the authors allow existing discrete solids (such as existing meshes or point clouds) to be used as input to the system from which an initial implicit solid is created. In [7] the authors represent the surface as the solution to a set of partial differential equations, which are represented both implicitly and parametrically. Carving is supported through the use of CSG-like operations which combine separate PDE representations.

In [8][9] the authors describe a framework for the simulation of bone surgery which involves both visual and haptic components. In order to represent the bone volume they store the bone data as voxels, but generate a triangle mesh to graphically display the bone. When voxels are removed by cutting the locally affected area is re-meshed. Their triangulation algorithm is brute force: create all possible triangles between neighbouring voxels and remove the triangles that are found to be duplicated or are below the object surface.

In [10] the authors present a system for the haptic and visual simulation of temporal bone surgery which is based on a voxel representation. In addition to bone matter, bone dust, debris, and water are represented using a particle system during the simulation. Rendering of the voxels are done using a direct volume rendering technique, in which the voxels are rendered directly using the GPU of the graphics card. The voxels use a "proxy-geometry" of object-aligned slices which are accumulated back-to-front. Shading and opacity for individual voxels is computed by shader and vertex programs on the graphics card itself. The method is fast since most of the visualization computation is offloaded to the GPU. One drawback of this visualization method is that it still suffers from terracing artifacts. In addition since three separate axis-aligned stacks of volume slices must be used so that the stack most perpendicular to the viewing direction can be chosen, when a new stack is selected for viewing when the viewpoint changes abrupt and unsightly changes to the texture can result.

A. Ball-Pivoting Algorithm

In order to visualize our volume, we use an adapted version of the well known Ball-Pivoting Algorithm (BPA) [11]. The BPA is an efficient method for computing a triangle mesh from a point cloud. Like marching triangles [13], it is a region-growing shape reconstruction algorithm. It uses a ball of fixed radius ρ, called a ρ-ball, to determine which points are surface points. The BPA begins with an initial seed triangle such that if a ρ-ball is touching all three points of the triangle it contains no other surface points. The edges of this triangle form the boundary of the expanding mesh called the *front* which is the set of edges from which the triangle mesh can expand. Each edge in the front has an associated ball-centre which is the position of the centre of the ρ-ball when the ball had touched the edge's vertices.

While there are edges in the front one is picked and pivoted around by the ρ-ball. Beginning from the ball-centre that is associated with the edge, the ball is rotated such that the edge is the axis of rotation. The first or earliest point that the ball touches along it's trajectory that is not processed, along with the pivoting edge, form a triangle that is added to the mesh. The front is updated by removing the edge we pivoted about, and adding the two new edges just created. If the ball is pivoted and no point is touched then the pivoting edge is marked as a boundary edge and is no longer considered as a candidate for pivoting around.

The front begins as a simple cycle of edges, but as edges are added the topology can become complex. The front will eventually consist of more than one cycle of edges (these cycles are referred to as loops in [11]). The BPA uses two topological operators to maintain the front: join and glue. A join is performed when the pivoting ball touches a new point and removes the pivoting edge from the front and adds the two new edges. A glue is performed when the join operation causes coincident edges in the front are removes the coincident edges in such a manner as to ensure the front is a set of cycles.

III. APPROACH

Carving is performed using a virtual carving tool which is controlled by the Novint Falcon haptic input device. The virtual carving tool is implemented as a spherical cutting head with an attached non-cutting handle. Each time the user moves the cutting tool over the object being carved, voxels are removed and the mesh updated. At each new tool position the voxels whose centers are within a defined range of the center of the tool's cutting head are removed from the voxel set.

B. Initialization

Our system is designed to operate on input 3D solids represented as voxels or as meshes. The voxel format is suitable when the input is medical data, such as images from CT or MRI scans which are often composited as a voxel dataset. Before carving, an initialization stage must be performed in which the input data is converted to voxels if necessary, and in which the initial triangle mesh representation for the voxel set is calculated.

If the input object is a mesh, it is converted to voxels. To generate the voxel representation the bounding box of each triangle in the mesh is computed. The voxels that the corners of the bounding box fall within are determined to find the voxel-space volume that bounds the triangle. Then for each voxel within the volume the fast 3D triangle-box overlap test deleloped by Akenine-Möller is applied to the voxel and the triangle [12]. All intersecting voxels form a boundary around the mesh volume. The volume inside the boundary is then filled using a 3D scanline filling algorithm so that solid voxels represent the object.

Once the voxel representation of the solid is computed, our system then computes the corresponding mesh used for display using the BPA. The algorithm is intended for 3D data-acquisition of real-world objects, but we have found it to be equally well suited to generating a triangle mesh from a set of voxels – the point set used as input to the algorithm is the set of points representing the centers of the voxels in the voxel set.

C. Haptic Control During Carving

When the user moves the virtual carving tool voxels are removed from the object, and force feedback is provided to the haptic device. To accurately model the effect of the user manipulating the carving tool we detection collision by checking whether the intersection of the tool head and the object is non-empty, and if so then move the tool head to a nearby position outside of the object and calculate the amount of resistive force that should be expressed to the user based on the impact. The process is described in more detail below. Each time the cutting tool is moved, the system detects whether there are any object voxels within the boundary of the carving tool's head. If not then the tool head is completely outside the object and the tool head's position is saved as the *anchor point*. If the tool is moved and there are voxels within the carving tool's head then the tool's head has collided with the object between the tool's last position and the current position. In this case 26 candidate spheres with the same radius as the cutting head are computed around the last anchor point in an axis-aligned a 3D grid each spaced a fixed *anchor-drag* distance apart. The candidate sphere that is closest to the object boundary while not inside of it is taken as the force measurement sphere. The magnitude of resistive force to express to the user through the haptic device is approximately calculated using Hooke's law with the displacement being the distance from the force measurement sphere to the current head position:

$$F = -x \cdot k$$

where F is the resistive force, $-x$ is the displacement and k is a scaled version of the material's ultimate strength.

Using an anchor point and candidate spheres allows the tool to slide along the surface of the object after it has collided. If a drag feature were not implemented, the burr-tool would be virtually glued to a spot on the model where it collided and would only relinquish its spot when the burr tool was fully pulled away from the model.

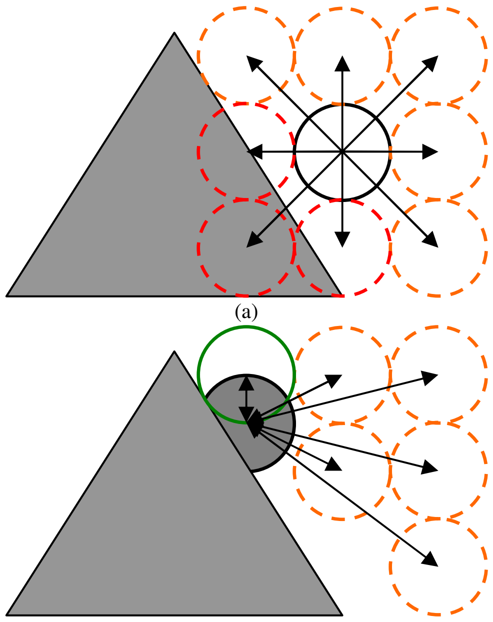

Figure 1: Method for computing the haptic feedback force. In (a) candidate spheres generated around the solid anchor point are used to determine an approximate position for the cutting head entirely outside the object. In (b) the difference between the closest outside candidate point (solid green) and the current cutting head (solid grey) is used as the displacement vector in Hooke's Law.

This use of Hooke's Law is analogous to the engineering stress-strain relationship, since the colliding head has put an amount of stress on the object approximately equal to the head displacement resulting in a proportional amount of stress. The force is applied outwards in the direction from the collided head position to the approximate surface head position. Approximating the force in this way is done in grinding simulation [14].

The voxels actually removed from the object are those that were within a distance less than ¾ of the radius away from the center of the cutting head.

D. Visualization During Carving

For visualization we use three major data structures to represent the object being carved: a voxel set used for volume calculations, a triangle mesh used for display, and a BPA front used when updating the display mesh from the voxel set. Each surface voxel in the voxel set has a link to the corresponding mesh vertex that it supports. The triangle mesh is defined such that vertices are shared between adjacent edges, and edges shared between adjacent triangles.

When voxels are removed from the voxel set the corresponding triangle mesh used for visualization is updated using a dynamic version of the BPA algorithm (DBPA).
The DBPA proceeds in two steps: the first step is to create a new front that bounds the mesh elements affected by the voxel removal, and the second is to use the standard BPA

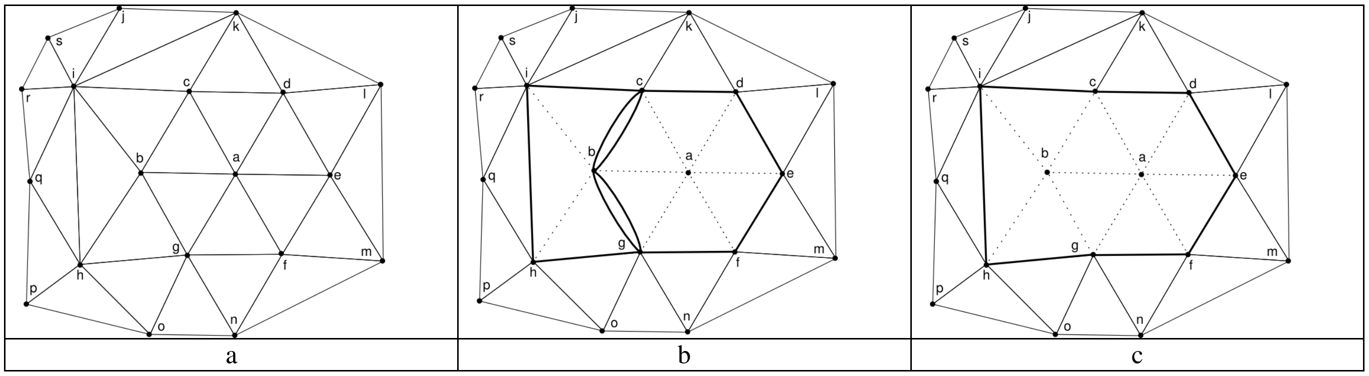

Figure 2: Steps taken to create a new front when the vertices a and b are to be removed from a sample mesh. (a) a section of a triangle mesh is shown. (b) the vertices a and b have been removed in that order. The adjacent triangles (abc, acd, bci, bih, etc.), the adjacent edges (ab, ac, bi, bh, etc.) are removed from the mesh. Removed edges are shown as dotted lines. Two new loops are created in the front: bcdefgb and bcihgb (shown in bold) are created. By adding the new cycle, the front contains a coincident pair of edges for bc between the two cycles, and a coincident pair for bg. (c) the coincident edges are removed by applying glue to each coincident pair in turn, resulting in a single cycle.

algorithm to triangulate and re-close the surface using this front. The standard BPA algorithm begins with a front that bounds a single triangle and grows outwards from that front to cover the entire surface. Likewise in the second step of DBPA we will begin BPA with a more complex front and grow triangles to close the front, ultimately replacing deleted triangles with new triangles in the existing mesh. Figure 2 shows the process of creating the new front for a simple mesh when two surface voxels are to be removed.

In the first step, for each voxel v removed, if v was not a surface voxel then no additional steps need to be taken. If v was a surface voxel it has a corresponding mesh vertex and adjacent triangles. Since the voxel is removed so must the adjacent mesh triangles and the edges that are allowed to be removed. When these triangles are removed, a new loop in the front is created that bounds the removed triangles. We call the set of remaining edges that bound the mesh triangles removed by removing a voxel the rim of the removed voxel. The rim must be ordered so that each edge in the rim is adjacent to the last.

When creating the rim, however, we must be careful that we only remove edges that are not already part of another loop that was added to the front by removing a voxel previously in the step. For example, in Fig. 2 the voxel corresponding to the mesh vertex a was removed first. When b was later removed the correct rim includes bc and bg, not ac and ag like expected since those edges have already been deleted and would cause overlapping loops in the front which violates the pre-conditions of BPA.

Instead we create coincident edges in the front. In the standard BPA these duplicate coincident edges may occur after a join operation, and they are removed using the BPA glue operation, which merges adjacent loops together as appropriate. We use the same glue operation to remove our coincident edges.

In order for the second step of the DBPA process to work, certain information must be stored on each edge in the front. When the BPA processes a front, it pivots a ball around each edge in the front until it strikes a point. To perform this calculation it must use not only the coordinates of the edge's vertices that it is pivoting around, but also the ball-center and opposite point of the existing triangle that bounds the front edge. The ball-center is the coordinate of the center of the ρ-ball then it had previously struck the point that generated the triangle.

Thus, for BPA to operate on the front that was created in the first step of DBPA, the front edges must have the correct ball-center and opposite point associated. In the classic BPA algorithm this is trivial: each front edge will only have one associated ball-center, being the one that was used to generate the front edge, and one opposite point. However since we are generating a front from an existing mesh we must choose the ball-center and opposite point for the mesh triangle that is adjacent to the edge we are creating, and has not been removed. In order to have this information available, each time a triangle is created in the mesh the ball-centre is stored with the triangle.

When each edge in a rim is added to the front the front edge created for that rim edge is created with both the ball-centre and opposite point from the adjacent triangle that has not been deleted. Figure 3 demonstrates the process.

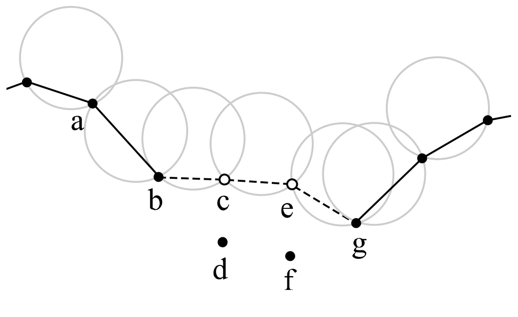

Figure 3: Ball-pivoting from a new front, in 2D. Only surface points are shown. After the vertices c and e are removed, pivoting occurs about "front edges" b and g to close the hole. For b, the ball-centre used to begin pivoting is the circle stored with ab, and the opposite point used is a.

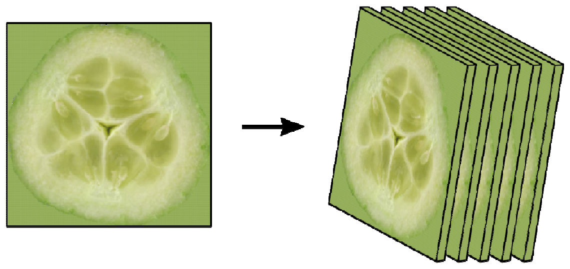

Figure 4: To create a 3D texture a 2D image is duplicated along the third dimension to create a 3D array of pixels.

E. Texturing

To provide realism and visual appeal the objects being carved display texture. Our method allows the specification of a 2D external texture and 3D internal texture for the object being carved.

The output of the initialization stage of our method is a voxel set and an associated triangle mesh. To apply the 2D texture to the object being carved the triangle mesh can be imported into a separate 3D modeling tool such as 3D studio max and have a 2D texture applied to it and then be re-imported into the system. To apply a 3D texture to the object a 3D texture consisting of a number of 2D image slices is created separately such that the axes of the texture are aligned with the object. Figure 4 shows this process.

With the 2D texture mapped to the initial mesh and the 3D texture available, our system can then initialize with the mesh and textures for carving. Initially the object appears with only the 2D texture visible since it was mapped to the external faces of the object being carved.

When carving is performed, however, each triangle created by applying the DBPA is instead mapped using the 3D texture. When the mesh being cut is first loaded the maximum axis-aligned dimensions of the mesh are stored.

When a new triangle is created using DBPA the physical coordinates are divided by the maximum initial dimensions of the mesh, giving relative texture coordinates. These relative texture coordinates are then scaled by the axis-aligned size of the 3D texture to obtain the actual texture coordinates.

For our experiments we created simple 3D textures by layering multiple copies of an image on top of itself. The 2D image was loaded and repeated as 2D slices of a 3D array. The 3D array is used as a 3D texture in the graphic library, with each image being a one "pixel" slice of the 3D texture.

IV. RESULTS

Our experiments were performed on a 2.8 GHz Pentium D processor machine. The OpenSceneGraph graphics library was used for rendering. Qualitative results can be seen in Figures 7 and 8.

We present quantitative results from a hip dataset. The hip object was created from a voxel space of dimension 120×120×120, and is shown in Fig. 6. A cutting tool with a spherical head was used to carve the surface. The performance results are shown in Table 1 and graphed in Figure 5.

Table 1: Carving Performance

Voxels Cut	Front	BPA	Total	FPS
42.8	6.7	5.1	11.8	84.7
148.8	23.4	15.9	39.3	25.4
382.3	26.8	38.4	65.2	15.3
680.3	44.2	59.1	103.3	9.7

In Table 1 each row is the average of ten sample cuts using the same cutting tool. Each cut is performed using a sphere centered on the hip surface. Voxels Cut is the average number of voxels removed per cut. Front is the average time in milliseconds to remove the voxels and mesh triangles, and compute the new front. BPA is the average time to re-mesh the front using BPA. Total is the sum of Front and BPA. FPS is the number of frames per second that can be achieved based on the time taken to cut the voxels.

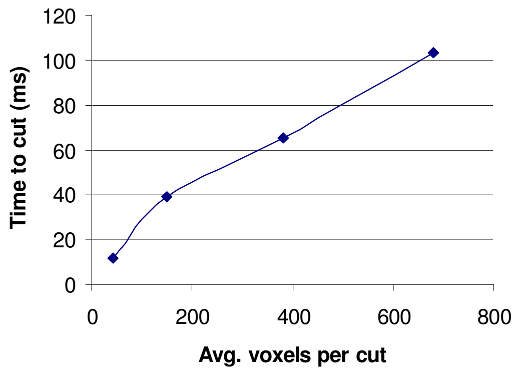

Figure 5: Graph of average time to complete a cut of varying numbers of voxels

The results are promising. Our system can achieve real-time rates on a large data set when removing below about 14 voxels per cut, which is approximately a volume of 5x5x5 voxels removed each cut.

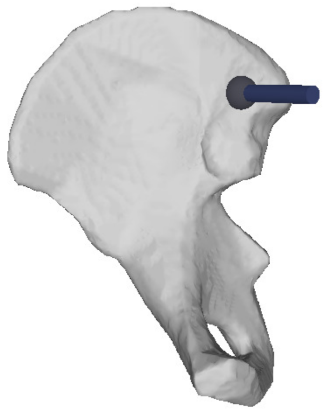

Figure 6: Hip dataset used for quantitative analysis.

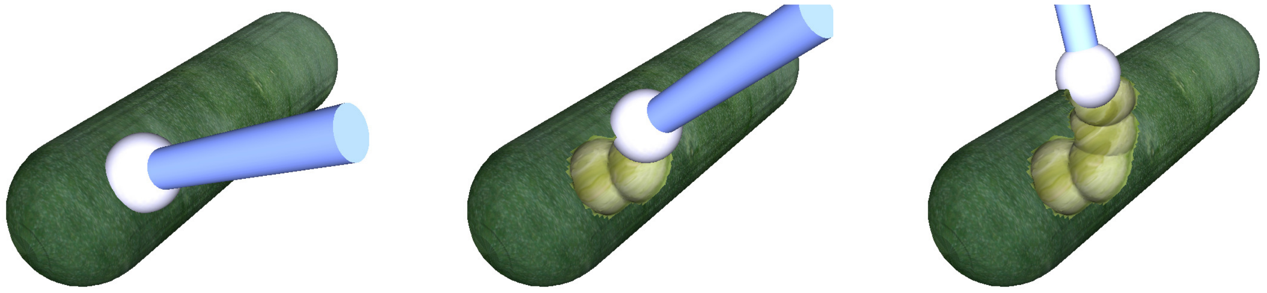

Figure 7: Demonstration of carving an externally and internally textured object. The cucumber uses an external 2D texture as well as an internal 3D texture

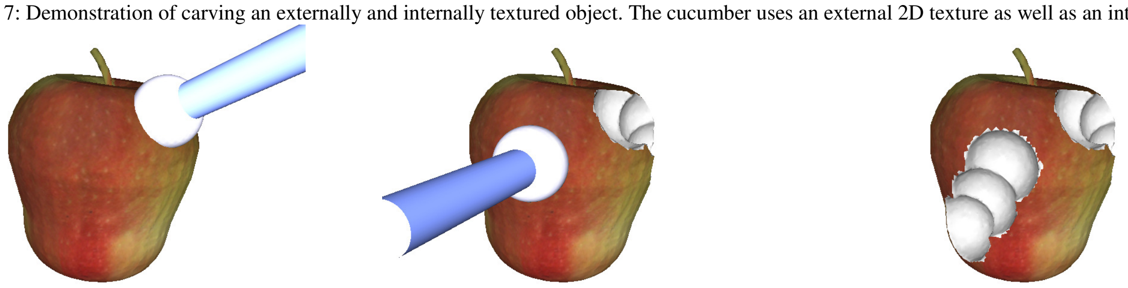

Figure 8: Demonstration of cutting an externally textured object. The apple uses an external 2D texture and a simple white internal colour.

The graph in Fig. 5 demonstrates that though it does take longer to regenerate the triangle mesh after each cut as the number of voxels affected by the cut increases, the time taken increases generally linearly. In Table 1 we see that the time taken to generate a new front increases slower than the time taken to regenerate the mesh using BPA. This is intuitive since generating the new front logically involves processing less surface area than generating the new triangles for the cut area; in the worst case an area roughly equal to the diameter of the cutting tool would be processed in the former, while in the latter an area roughly equal to the outer surface area of the tool head would be processed.

V. CONCLUSION

We have presented an interactive 3D carving system that animates carving in real time using feedback from a haptic input device. By representing the object being cut internally as voxels but rendering the results to the display as a triangle mesh, the system achieves quick computation of the carving effects while providing a pleasing and realistic 3D view of the object. The DBPA and texturing techniques presented in the paper are suitably generic that they can be used for various carving purposes from medical simulation to games.

Avenues for future work include smoothing the results of individual cuts, and using a mathematical model for grinding rather than one for the stress-strain relationship to determine the amount of resistive force.

VI. REFERENCES

[1] William C. Thibault and Bruce F. Naylor. “Set operations on polyhedra using binary space partitioning trees,” *SIGGRAPH Comput. Graph.*, 21(4):153–162, 1987.

[2] Bruce Naylor, John Amanatides, and William Thibault. “Merging bsp trees yields polyhedral set operations,” *SIGGRAPH Comput. Graph.*, 24(4):115–124, 1990.

[3] J. Rossignac and A. Requicha, “Solid modeling,” In J. Webster, editor, Encyclopedia of Electrical and Electronics Engineering. John Wiley and Sons, 1999.

[4] Guy Sela, Sagi Schein, and Gershon Elber. “Real-time incision simulation using discontinuous free form deformation,” Lecture Notes in Computer Science, 3078/2004:114–123, 2004.

[5] Jing Hua, Hong Qin, "Haptics-Based Dynamic Implicit Solid Modeling," IEEE Transactions on Visualization and Computer Graphics, vol. 10, no. 5, pp. 574-586, Sept/Oct, 2004.

[6] A. Raviv and G. Elber. “Three Dimensional Freeform Sculpting via Zero Sets of Scalar Trivariate Functions,” Proc. Fifth ACM Symp. Solid Modeling and Applications, pp. 246-257, 1999.

[7] Haixia Du, Hong Qin, "Free-Form Geometric Modeling by Integrating Parametric and Implicit PDEs," IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 3, pp. 549-561, May/Jun, 2007

[8] D. Morris, C. Sewell, N. Blevins, F. Barbagli, and K Salisbury. “A collaborative virtual environment for the simulation of temporal bone surgery,” In Proceedings of MICCAI. Springer-Verlag, 2004.

[9] D. Morris, S. Girod, F. Barbagli, and K Salisbury. “An interactive simulation environment for craniofacial surgical procedures.” In Proceedings of MMVR (Medicine Meets Virtual Reality), 2005.

[10] Agus M, Giachetti A, Gobbetti E, Zanetti G, Zorcolo A, Picasso B, Sellari Franceschini S. "A haptic model of a bone-cutting burr," Stud. Health Technol. Inform. 2003; 94:4-10.

[11] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics, 5(4):349–359, 1999.

[12] Tomas Akenine-Möller. “Fast 3d triangle-box overlap testing,” In SIGGRAPH '05: ACM SIGGRAPH 2005Courses, page 8, New York, NY, USA, 2005. ACM.

[13] S. Akkouche and E. Galin. “Adaptive implicit surface polygonization using marching triangles”. In *Computer Graphics Forum*, volume 20(2), pages 67–80, 2001.

[14] Ren, X. y.; Mueller and H.; Kuhlenkoetter, B., “Surfel-based surface modeling for robotic belt grinding simulation”, Journal – Zhejiang University Science A. Vol. 7, no. 7, pp. 1215–1224, 2006.