View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

15

Haptic-Based 3D Carving Simulator

Gabriel Telles O'Neill, Won-Sook Lee and Jeff William
University of Ottawa
Canada

Abstract

We aim to add realistic force-feedback to the process of real-time volume removal from a 3D mesh object. We refer to this volume removal as "3D carving". 3D carving is particularly applicable to the computer simulation of surgical procedures involving bone reductions that are performed with a motorized burr tool; however the methods and algorithms presented here are generic enough to be used for other purposes, such as modeling, destructible objects & terrains in games, etc.

The system represents the volume of the virtual objects using a voxel-set during carving process. A polygonal mesh is created from this voxel-set to display a smoother rendition of the virtual object. Out system also employs the novel Dynamic Ball-Pivoting Algorithm to generate quick mesh updates when voxel-set revisions occur. The advantages of this being that minor changes to the voxel-set only require minor changes to the mesh, whereas the standard Ball-Pivoting Algorithm approach is to perform a global re-meshing.

In our haptic carving system interactions with the aforementioned voxel-set can provide output force-feedback to the system operator. A pen-based haptic tool is used to act as a 3D mouse in order to "feel" the surface of the model as well as to remove select volume segments of the object. Two collision detection schemes are presented here which allow users to feel the surface of virtual objects either using the voxels alone or by using supplemental information from the polygonal mesh. When the burr-tool has its "cutting mode" enabled, which sections of an object's volume are to be removed is decided by evaluating that volume's proximity to the center of the burr.

1. Introduction

Traditionally, the majority of computing applications have relegated their output to what can be seen or heard- modalities consistent with the hardware available on the average desktop computer. However, the advantage gained from incorporating the sense of touch to computing applications has recently increased in practicality as the hardware supporting virtual touch decreases in cost and becomes more accessible. Haptic devices are a relative new technology that provides humans a tactile interface to elements in a virtual world. Even as a fledgling field it has already shown promise in virtual-reality systems, design of computer-generated objects, medicine, robotics and gaming.

In some cases, pre-existing audio-visual systems seeking to enhance the realism of a user's experience can do so by replacing one of the system's input devices (e.g. a mouse) with one of these haptic devices. The system to be described here is an example of this.

The graphical simulation system is to demonstrate their Dynamic Ball Pivoting Algorithm (DBPA) 0. The aim was to develop a system for the real-time visualization of removing volume for an object model. In the DBPA system, an object's volume is represented using voxels, but an associated triangle mesh is also maintained for display purposes since direct visualization of voxels is unrealistic and unappealing. Tasks such as simulating the drilling or carving of bone during medical procedures, boring wood, biting food and chiselling marble can all be observed in real time by the operator of the DBPA system.

Our haptic simulation system was built on-top of the DBPA system; where control of the volume-removing element was given to a haptic device, whereas the graphical simulation system itself was mouse-driven. The result is a means to carve, drill and trim a three dimensional object stored inside the computer while observing the removal of sections of volume and receiving corresponding force feedback immediately.

While this system's general scope is that of virtual-reality, it has been designed with the ultimate goal of being used as a training environment for surgeons in order to help them accumulate experience and muscle memory useful in actual operations involving bone reductions. For example, motivational surgery around which this new system was designed was a minimally invasive procedure to correct femoroacetabular impingements (FAIs). FAIs describe a condition where there exists an overgrowth of bone on either a patient's femur neck or around the hip socket. These overgrowths often cause discomfort and chafe during normal hip exercise, producing premature cartilage damage and labral tears, which lead to further medical complications and pain. These impingements can be treated by using a motorized burr tool to grind away the excess bone and reshape the affected regions for a better fit between the femur head and hip socket. As such, a virtual burr with a spherical head, fashioned to resemble the one depicted in Fig. 1 (without the hood), is used as the default tool for volume removal inside our simulation.

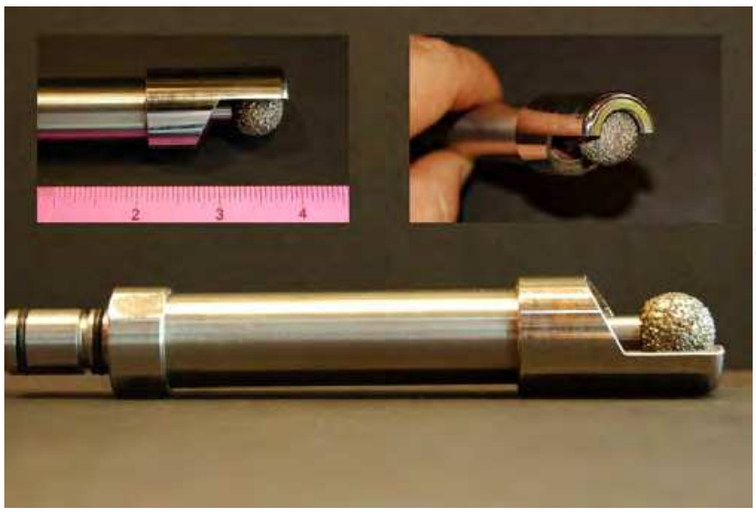


Fig. 1. An example of a motorized burr tool with a hood

In our system, the force-feedback and the intersection points between the burr and object's volume can be computed using one of two methods. The first being the voxel-set, which most directly represents to object's volume, and the second being the set of triangles that compose the polygonized mesh generated by the DBPA. Volume is removed from an object by isolating sections of voxels intersecting with the virtual burr and eliminating them from voxel-set. When voxels do get removed from the virtual model, the affected regions of the corresponding triangle mesh are updated and re-displayed.

2. Related Work

Increasing interest has developed over the past decade on developing carving simulators with faithful haptic feedback for various medical fields. Of these the most technically successful appear to be dental training systems. These systems are designed to allow dental students to practice certain procedures on a virtual set of teeth such as drilling operations and filling cavities with amalgam. In these simulators, the manner in which collision detection and the resulting haptic force-feedback is calculated is intrinsically linked to how the system's designers chose to represent the volume of their carvable objects.

In Kim and Park's dental simulator 0, their model's volume was represented using a Computational Solid Geometry (CSG) point-set. CSG expresses an object as the combination from a set of solid primitives. These primitives can be parametric equations of a quadric surface (e.g. planes, spheres, cones, cylinders, or paraboloids), or simple, regular prisms (e.g. cubes). The primitives will form the leaves of a binary tree, in which internal nodes represent rigid transformations (translations, rotations, or scalings) of the children nodes, or represent regularized Boolean set operations (union, intersection, or difference) on the left or right sub-tree. Performing volume removal (cutting, drilling, etc.) using CSG is a relatively simple matter of representing the object that will have its volume removed as a CSG sub-tree whose parent is a difference operation with the second sub-tree being the union of all instalments of volume that have been "cut". However a major drawback of this approach is that the volume to be "cut" must first be converted to a CSG representation, which is difficult and non-automated for complex objects such as human bones. A second drawback is that the method does not scale well as the number of volume removals increases. This is because the tree representing each "cut" must be added to the main tree of the object. As a result, successive cuts to an object's volume will make the tree representation grow large quickly. Using this CSG model, the two researchers chose to implement collision detection by calculating the distance from an offset field surrounding the surface of the virtual teeth to the center of the user's dental tool. If the tool's center passed the offset field, the haptic device controlling the dental tool would provide force-feedback in the direction of the field's implicit surface normal. The force's direction is used to calculate the tool's virtual contact point with the object's surface. This contact point is then used to calculate the force vector by using a spring-damper model based on Hooke's law.

Similarly, Yau et al. 0 also used a spring-damper model to calculate the force vector sent to the haptic device of their dental training system. However, instead of representing the volume of their objects using CSG, an adaptive voxel model is used. These researchers used an implicit function to define their cutting tools, which in turn were used to decide exactly what volume was to be removed from the model. As the voxels used to represent the object are of varying sizes in this scheme, if the tool comes into contact with a large voxel,

recursive subdivision must be performed on that voxel until the voxels in contact with the tool are small enough for removal. Any voxel whose volume is found to be completely "inside" the tool will subsequently be remove from the model.
A real-time haptic and visual bone dissection simulator00 was also proposed by Agus et al., aimed as being used as a training tool for temporal bone surgery. This system most closely resembles our own as it generates its object volume through the voxel discretization of 3D Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) data. In addition to bone matter, secondary visual effects such as bone dust, debris, and water are realized using a particle system to potentially heighten the realism of surgeon's experience.

3. Graphical Carving System

Graphical simulation system is necessary before detailing our own hi-fidelity haptic feedback improvements. The development goal of this graphical carving system, DBPA, is the real-time visualization of volume removal for an object model. Updates in a model's volume resulting from simulated operations such as drilling, carving, boring, biting and chiselling the object are performed and displayed back to the operator at a rate that appears instantaneous to them. This method uses voxels (of a constant size) as a volumetric representation of the object. Since renderings composed of voxels appear very blocky, they fail to provide the realistic visual interface necessary for a surgical training simulator. A further presentation step is required: from the object's voxel-set, the system generates a much nicer looking polygon mesh to visualize the object model before, during, and after cutting. Triangulation is achieved using the Dynamic Ball Pivoting Algorithm0, which is an extension of Ball Pivoting Algorithm (BPA)0.
The objects to be carved are represented by three major data structures: a voxel-set used for volume representation, a triangle mesh used for display purposes, and a modified BPA front-end used for updating the display mesh when the voxel-set is altered. Each surface voxel in the voxel-set is also linked to its corresponding vertex in the triangle mesh. The mesh is defined such that vertices are shared between adjacent edges, and edges are shared between adjacent triangles.

3.1 Initialization

To start, the system needs one of the following inputs:

1) *A solid pre-modeled as a voxel-set*

   This option is designed to facilitate the insertion of 3D CT or MRI data into the program for patient/ object specific simulation.

2) *A polygon model*

   The model provided will then subsequently be reduced to a voxel-set. Afterwards, the voxel in which each vertex of the bounding box finds itself is found in order to determine the voxel-space volume that binds the triangle. Then, for each voxel within the volume, the fast 3D triangle-box overlap test from Akenine-Möller 0. All intersecting voxels form a boundary around the mesh volume. The volume inside the boundary is then filled using a 3D scanline filling algorithm.

Either way, once the voxel representation of the solid is determined, the system then computes the corresponding polygonal mesh using the BPA. The algorithm is intended for 3D data-acquisition of real-world objects, but Williams et al 0 found it to be equally well suited for generating a triangle mesh from a voxel-set by using the centers of the voxels as the point set.

3.2 Carving Out Volume

Within the system, carving is performed by manipulating a virtual tool which resembles the motorized burrs used by medical professionals. In practice, burrs are used by surgeons to dexterously grind way at bone surfaces with a rotating, abrasive head while gripping the tool's handle. In this system, the tool is implemented as a spherical cutting head which is attached to a non-cutting handle. When the user moves the cutting tool over the object being carved, voxels are removed and the mesh is updated.

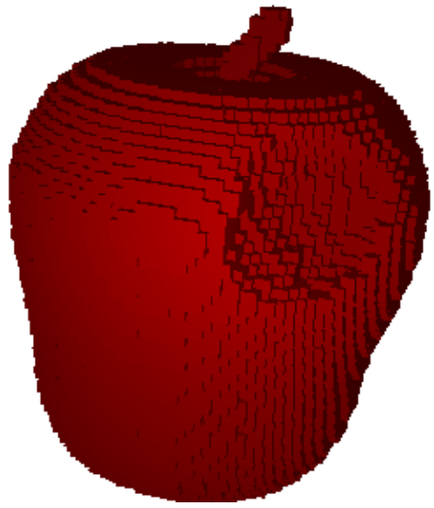


Fig. 2. Unmeshed voxelized apple with carved hole

Whenever the tool's position is changed, the system uses the geometric equation for the tool's head to determine if there are the voxels whose centers are within its boundary. If so, they are removed from the voxel-set. When voxels are removed from the set, the triangle mesh must be updated to reflect the changes. The DBPA is responsible for these updates and, for each voxel v removed from the set, performs the following procedure:

1) *If v was not a surface voxel*

   no additional steps need to be taken.

2) *Otherwise, if v was a surface voxel*

   It must have a corresponding mesh vertex with adjacent triangles. The voxel's removal indicates the removal of a vertex from the mesh, thus all triangles adjacent to that vertex become invalid and must also be removed. After removal, a new loop in the front-end is created that bounds the removed triangles. Care is taken when adding the loops to ensure that the BPA front invariant property is preserved while

iteratively removing voxels and their associated mesh triangles. Adding these loops sometimes cause undesirable adjacent edges which are removed using the BPA's "glue" operation.

In order for the DBPA to work properly, certain additional information must be stored at each edge in the front. When the BPA processes a front, it pivots a ball around each edge in that front until it strikes a point. This calculation requires not only the coordinates of the pivotal edge's vertices, but also (1) the "ball-center" (coordinates of the center of the BPA ρ-ball when it had previously struck the point that generated the triangle) and (2) the coordinates of the point (vertex) opposite to front edge in its binding triangle. In order to have this information available, newly created triangles in the mesh also store their respective ball-centers. Also, when an edge is added to the front, the front's corresponding edge will store both the ball-centre and opposite-point as additional data.

3.3 Texturing Meshes

To further enhance the visual fidelity of carvable objects, this system has the option of applying textures to the voxels. An alternative to flatly colouring the voxels, a 2D texture can be applied over the exterior of the shape while a 3D texture can be specified for the object's interior. The 2D texture is applied during the initialization stage and is attached directly to the triangle mesh that represents the whole, un-carved object. Once object carving is underway, the new triangles created through the DBPA are mapped to the contents of the 3D texture. Examples are shown in Fig. 3 and Fig. 4.

Fig. 3. Meshed apple with external texture

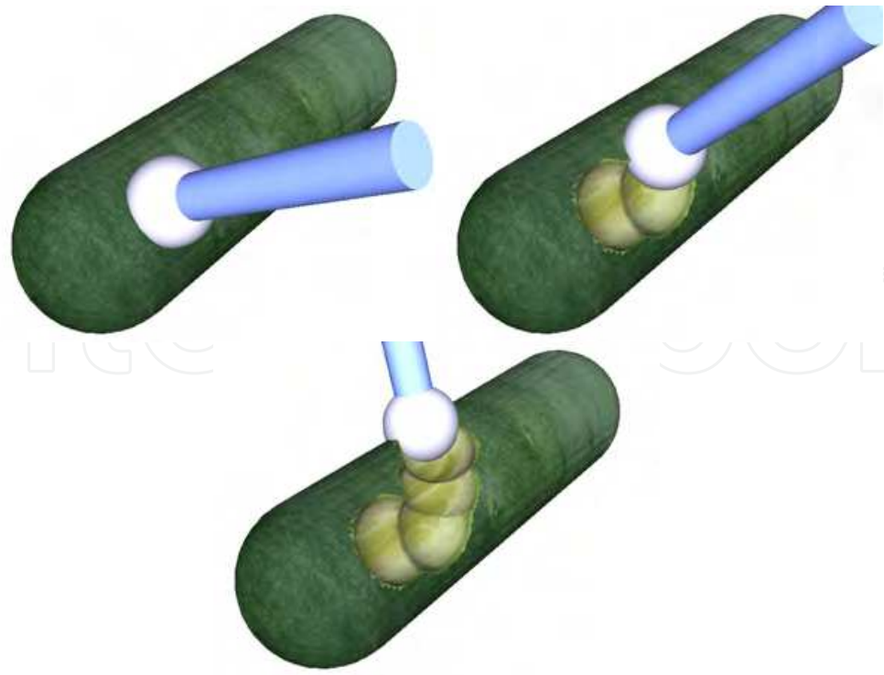


Fig. 4. Demonstration of carving an externally and internally textured object. The cucumber uses an external 2D texture as well as an internal 3D texture

4. Haptic system

While the DBPA 0 has the graphical solution using OpenSceneGraph (OSG), the haptic simulation system chooses the NOVINT Falcon™ along with its stock API to implement our conjoined haptic solution. The Falcon is a pen-based haptic device intended as a joystick or mouse substitute. It allows a user to control an application in a three dimensional space while also providing him or her with high-fidelity force-feedback. When a user holds the Falcon's grip and moves the cursor to interact with a virtual scene, motors in the device turn on and are updated at precise rate of 1000Hz, allowing the operator to feel texture, shape, weight, dimension, and dynamics.

Fig. 5. NOVINT Falcon, reprinted from the NOVINT website

Our system's interface with the Falcon was the Haptic Device Abstraction Layer (HDAL) which is NOVINT's pre-packaged API. HDAL lacks the higher-level functions that other haptic APIs (such as Sensable's OpenHaptics or H3D.org's HAPI) have, which means that most operations such as force calculations and button status lack automation and need be calculated manually. Also, because the haptic device needs to be updated a thousand times a second, all operations performed inside the HDAL's regular maintenance loop need to be performed in under 1 ms in order to maintain "high-fidelity" haptic force feedback. The implication here is that special effort must be taken in order to ensure that the haptic loop's tasks are performed as efficiently as possible to guarantee that the device operator does not sense that the haptic output feels "choppy".

4.1 System Overview

The sequence of steps used to implement this model-cutting strategy is illustrated in Fig. 6. This activity diagram serves as an overview of the new haptic system's two intercommunicating threads of execution. The two concurrent threads begin executing at runtime. There is an OSG thread responsible for what the user sees, and a haptic thread responsible for what the user feels.

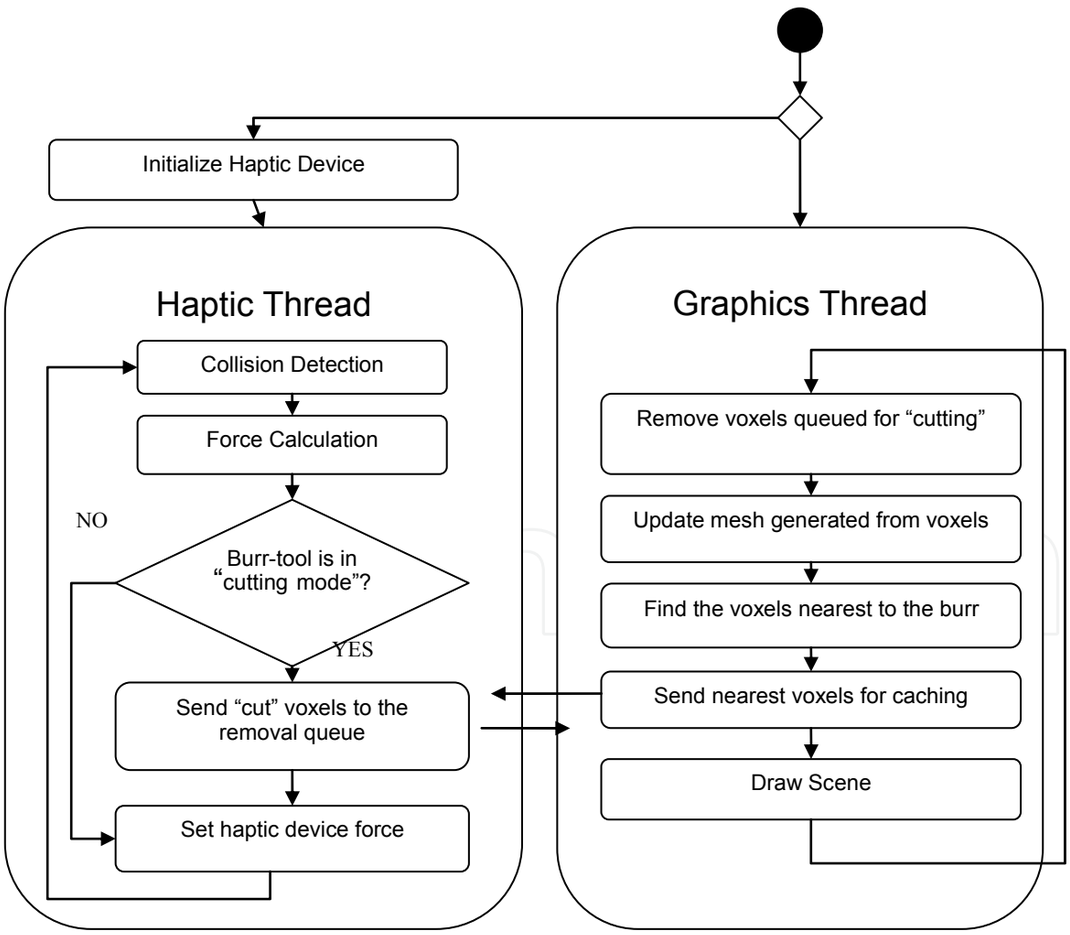


Fig. 6. Activity Diagram for the new haptic system

The haptic thread implements one of two collision detection methods which in turn provide input values for the force calculation step. The first collision detection method uses only the object's voxel representation to detect collisions and scale to force feedback while the second method takes advantage of the polygon mesh created from the voxel-set to perform these tasks. The former has shown to be useful in quickly evaluating the effects of new features to system but suffers from a somewhat "blocky" contact with the volume. The latter method provides a smoother contact force while passing over the object but in turn requires the management of a set of triangles from the mesh in addition to a cache of voxels.

4.2 Graphics Thread

Our system adds a double-ended queue to the Graphics OSG thread. Named the "removalQueue". This structure contains the coordinates of voxels which ought to be removed from the object during the next execution loop. This queue is populated by the haptic thread when it has been decided that certain sections of volume have been "cut" during the user's operation of the burr-tool. At the start of each OSG loop, the queue will be emptied and all corresponding voxels in the model will be removed. In addition, the polygonizer will be informed of any change to the voxel set so that it can re-mesh the isolated changed regions rather than re-polygonizing the whole model.

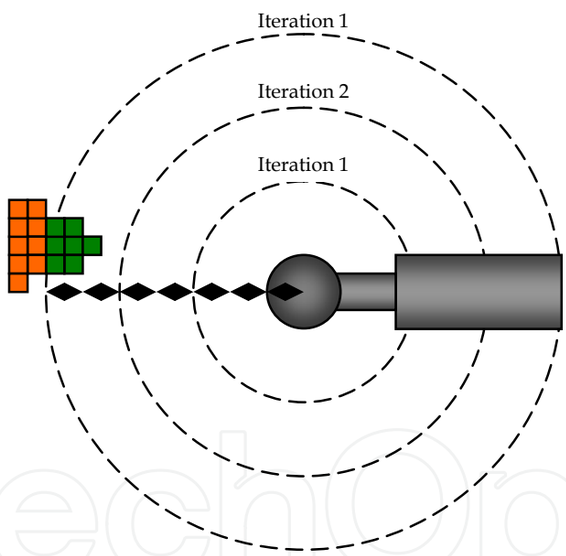


Fig. 7. A 2D slice showing how nearby voxel are found

In order to minimize the number of voxels the haptic thread needs to investigate for collisions, only the nearest voxels to the burr's head are sent for caching to the haptic class. The nearest voxels are found using an iterative, step-based approach: if the burr radius is designated the symbol *r*, the loop will first look for any voxels within 3r of the burr-head center, then 5r, then 7r, etc. until it finds at least one voxel which it can send to the haptic thread for caching. If no voxels are found within 23r of the voxel head, the searching quits. It was determined through experimentation that after 23r, the voxel set is sufficiently far enough from the burr-tool that that, even at high speeds, the tool was unlikely to come into contact with the objects in between cache updates. If the haptic thread's collision detection is

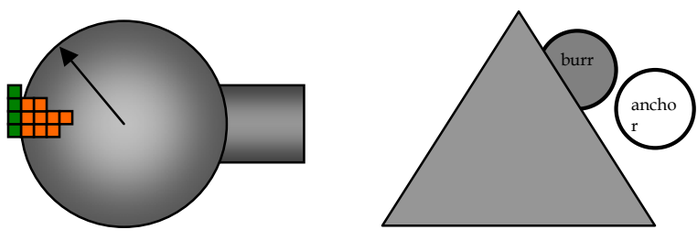
to be performed using the voxel-only method, this step ends here. Otherwise, all the triangles part of the mesh within the same "nearest distance" to the burr head are also sent to the haptic thread to be part of its cache.

4.3 Haptic Thread

It is worth mentioning that since the 3D space coordinates, roll, pitch and heading of the voxel model can be altered in the scene to get a better view of the object from all angles, conversion from local to world coordinates (and back) is required. Upon entering the haptic loop, the burr's position is converted to the model's local coordinate system to simplify the calculation of burr-to-voxel distances. Conversely, the initial force feedback direction is calculated in the voxel model's local coordinate system, so it is converted to world coordinates before being sent to the haptic device as a force command.

4.3.1 Collision Detection (with only voxels)

Detecting a collision between the burr and the voxel model (or between a prospective anchor and the voxel model) is fairly simple for a spherical burr head.

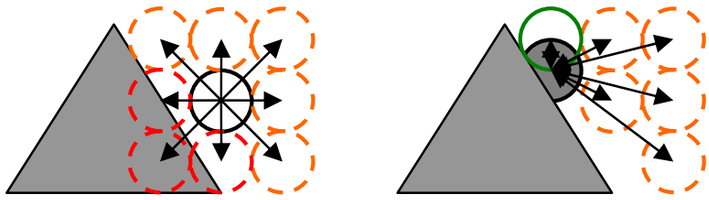


(a) (b)

Fig. 8. (a) Voxels causing a collision to be detected (b) Example positioning of a burr head and anchor point as a collision is being detected

An iterator is created to traverse the set of cached voxels. While doing so, if the distance from a voxel to the center of the burr-head is less than or equal to the burr's radius, then a collision has occurred and the iteration is halted. If the iterator has traverses the voxel cache completely, no collision has occurred. As shown in Fig. 8(a), the coloured squares represent the set of voxels cached by the haptic class, but only the voxels marked as orange squares would cause a collision to occur. If no collision has occurred for any of the nearby voxels, then the "anchor" is set to the burr head's current position. This anchor represents an approximation of closest point to the burr head that does not intersect with any voxels. This becomes important later when the next collision does occur, since this system scales the magnitude of the force feedback based on the distance between the burr center and the anchor center.

When a collision has been detected, the first thing the haptic thread will do is to determine if there is an alternate anchor point within "anchor drag" distance which is closer to the burr head than the current anchor point.



(a) (b)

Fig. 9. (a) A 2D slice of 8 new anchor candidates placed around the original anchor (b) Distances from the candidate anchors to the burr-head

Twenty-six new anchor candidates are generated by applying a fixed "anchor drag" distance in each combination of *x/y/z* direction around the old anchor. New anchor positions that collide with one or more voxel points are disqualified from being candidate anchor points (coloured red in Fig. 9(a)). The reason why the anchor is permitted to "drag" in any direction is to let the burr tool slide across the surface of the voxel model after it has collided. This allows the operator to get an impression of the model landscape. 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.
As shown in Fig. 9(b), the candidate anchor with the shortest distance to the burr-head's current position, circled here in green, becomes the new anchor point. With the anchor point established, the following two values are computed:

forceVector ← *anchor center – burr center*

linkDist ← *distance between anchor center and burr center*

4.3.2 Collision Detection (with triangle mesh)

Using the cache of nearest triangles to the burr head allows us calculate the force vector and link distance without have to evaluate candidate anchor points. First the nearest triangle to the burr head is identified from the triangle cache (coloured red in Fig. 10). The plane on which this nearest triangle lies is determined and the distance from the burr center to that plane is calculated.

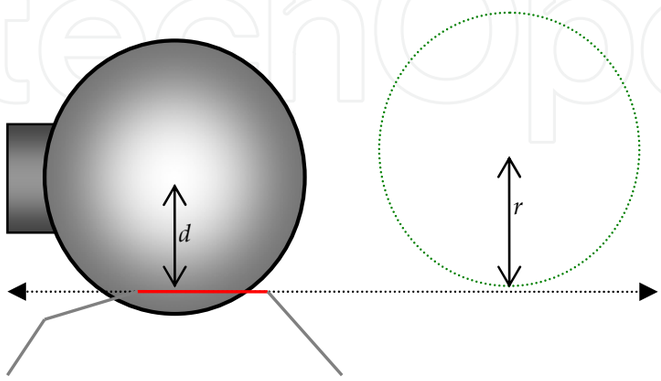


Fig. 10. A 2D slice of a collision between the burr head and some triangles from the mesh

If the distance from that burr center to the plane is greater than the burr's radius ($d > r$), no collision has occurred. Otherwise, a collision is recorded and the following two values are computed:

forceVector ← *nearest triangle's normal*

linkDist ← *burr radius – burr centre's distance from the plane*

4.3.3 Force Calculation

Finally, the actual calculation for the local force is computed using equation (1).

$$\vec{F}_t \leftarrow matUltStrength * scaleFactor * linkDist * forceVector \quad (1)$$

This above force equation is inspired by Hooke's Law ($Force = -k \times x$). However, instead of using the spring constant k, this equation uses a material's "ultimate strength". The ultimate strength of a material being defined as the "maximum stress a material can withstand". Some examples which were used as material parameters in our system can be found in Table 1.

Material	Ultimate Strength (S_u) 10^6 N/m^2
Steel (ASTM-A36)	400
Bone (limb)	170
Wood (Douglas fir)	50

Table 1. Comparative ultimate strength values0

In the previous equation, *forceVector* is the 3D directionality component along which the (local) force feedback will be aligned, *linkDistance* is the depth to which the burr has been pushed into the object's volume and is used to scale the magnitude of the force according to how strongly a user is pushing into the volume of the model, *materialUltStrength* is the ultimate strength of the material from which the model is made, and *scaleFactor* is a downscaling factor to place the final force magnitudes within range of the NOVINT Falcon's capabilities.

In order to ensure a smoother transition from one force to the next, a force-filter, as implemented by Yau et al.0, is adopted by applying a damper to our spring system. This is applied to the local force by using the method described in Equation (2) where δ is a predefined threshold for the force change.

$$\Delta\vec{F} = \vec{F}_t - \vec{F}_{t-1}$$

$$\begin{cases} \vec{F}_t' = \vec{F}_{t-1} + \delta \times \dfrac{\Delta\vec{F}}{\left\|\Delta\vec{F}\right\|} & \left\|\Delta\vec{F}\right\| > \delta \\ \vec{F}_t' = \vec{F}_t & \left\|\Delta\vec{F}\right\| \le \delta \end{cases} \quad (2)$$

Following this method, the new force is converted back into the global coordinate system, where a few force effects are added (e.g. a slight vibration in the tool tip to simulate the rotating nature of the burr) and finally, this force value is sent to the Falcon as an output force command.

4.3.4 Voxel Cutting

Any voxel whose distance to the center of the burr head is less than or equal to ¾ of the burr radius is considered "cut". To cut a voxel, first, its coordinates will be pushed to the back of the OSG removal queue. Second, its unique identifier (based on its position in the voxel model) is placed in a hash-map so that the haptic thread will no longer process that voxel for collisions while waiting for the OSG thread to send it an updated cache of the nearby and uncut voxels.

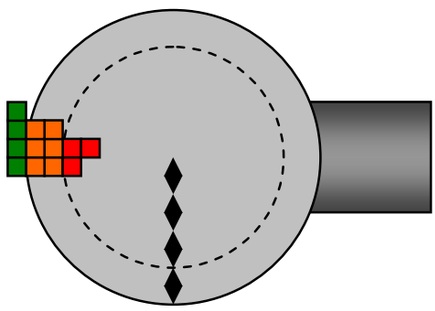


Fig. 11. Distinction between simply colliding (orange) and colliding & cut (red) voxels

5. Results

The results thus far have been promising. The surface features of our apple, femur and pelvis models can easily be felt using either collision detection method. Surface contact is definitely smoother using the mesh collision detection scheme. However, carving and, even more so, drilling operations tend to perform more reliably using the anchor based, voxel-only method at the moment. Using a material's ultimate strength has also shown to be useful in providing the user with haptic feedback on how difficult it is to cut different materials.

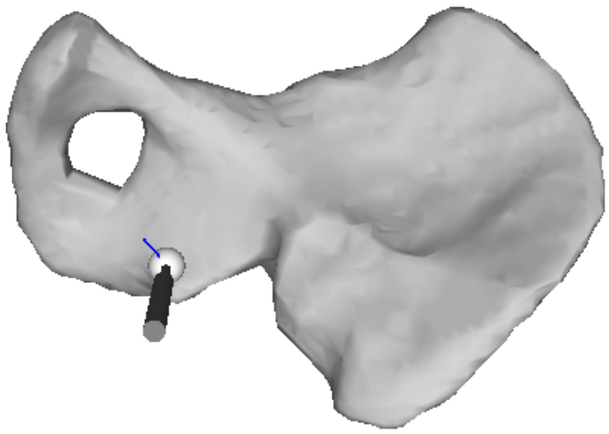


Fig. 12. A burr-tool receiving force-feedback from a polygonized pelvis model where the force (direction and strength) is displayed with a blue line

At present, users are unable to distinguish between most different types of material textures while using the voxel-only approach to collision detection. This is largely due to the discrete nature of voxels promoting a "blocky" surface contact with the spherical burr. This issue could be partially addressed by increasing the voxel density used to represent and object volume. However, this solution becomes resource demanding past a certain point. The collision detection method that exploits the mesh feels much smoother when passing over flat and rounded surfaces with the burr; however different material haptic surface textures have not yet been convincingly implemented.

6. Discussion

Both the Dynamic Ball Pivoting Algorithm and Haptic system need to mature into more robust versions of their current selves before their inherent potential can truly shine through. Also, while basing the haptic class' force equation on Hooke's law is convenient, it is also inaccurate. A more involved and realistic model would be to use a material's full stress-strain curve0 to dictate the amount of force required to remove volume from the model. However, such a change would require a means to measure to amount of force the user is exerting on the haptic device.

A question that has come up before is: why we bother with the anchor-based method for finding the force direction when we could use the nearest colliding voxel or use the summation of the direction vectors of all voxels colliding with the burr-head instead? The reason for this is that the nearest-voxel or voxel-summation methods have shown to perform erratically whenever the burr-head is placed in a tight corner or inside a pit. On the other hand, the anchor-based method has shown to perform as expected in both these situations as well as on normal surface curvatures.

7. Conclusion and Future Work

This new system adds a sense of touch to the process of removing volume from voxelized objects and is built on top of William et al.'s graphical carving simulator. Two components operate in unison in order to make this work: an OpenSceneGraph thread and a haptic thread. The former is responsible for clearing voxels queued for removal, redrawing the scene and providing the haptic thread with a subset of the object data; the voxels and triangles most likely to be relevant during collision detection are cached here. The latter deals with issuances of both the direction and magnitude of force as well as evaluating which sections of volume should be removed from the object.

There are certainly a great many directions where the haptic portion of the system can be improved and extended in the future. One area that would improve the program's use would be to have a more modular approach to the cutting tools. Tools other than a burr with a spherical head are likely to be useful to surgeons. The head may instead be an ellipsoid, conical or cylindrical. The cutting tool could also be something non-motorized such a scalpel which would require the distinction between cutting surfaces and non-cutting surfaces to be made.

At the moment, models have a global ultimate strength value meaning that all the voxel will have the same stiffness. In many cases, such as our target example; operating on human bone, this is unrealistic as their exteriors are made of dense cortical bone while their interior is composed of much softer bone marrow. Assigning each voxel its own density value is our next step. This will also allow us to examine a voxel removal strategy whereby the act of "cutting" an object will incrementally reduce the voxels density and voxels finding themselves with a density of zero are considered wholly "cut". The same idea can be extended to the mesh-based collision detection. The hope is that this will allow a user to feel a more progressive entry into an object while it is being cut.

8. References

[1] Williams J, Telles O'Neill G, Lee WS. Interactive 3d haptic carving using combined voxels and mesh. Haptic Audio visual Environments and Games, 2008. HAVE 2008; pp 108-113, DOI: 10.1109/HAVE.2008.4685308

[2] Kim L, Park SH. Haptic interaction and volume modeling techniques for realistic dental simulation. The visual Computer: International Journal of Computer Graphics. Volume 22, Issue 2, 2006; pp 90-98, DOI: 10.1007/s00371-006-0369-8

[3] Yau HT, Tsou LS, Tsai MJ. Octree-based Virtual Dental Training System with a Haptic Device. Computer-Aided Design & Applications. Volume 3, 2006; pp 415-424

[4] Agus M, Giachetti A, Gobbetti E, Zanetti G, Zorcolo A. Real-time haptic and visual simulation of bone dissection. Presence: Teleoperators and Virtual Environments; special issue: IEEE virtual reality 2002 conference; Volume 12, Issue 1, 2003; pp 110-122

[5] Agus M, Giachetti A, Gobbetti E, Zanetti G, Zorcolo A. Adaptive techniques for real-time haptic and visual simulation of bone dissection. Virtual Reality, 2003. Proceedings. IEEE; pp 102-109, DOI: 10.1109/VR.2003.1191127

[6] Bernardini F, Mittleman J, Rushmeir H, Silva C, Taubin. The ball-pivoting algorithm for surface reconstruction. Visualization and Computer Graphics, Volume 5, Issue 4, 1999; pp 349-359, DOI: 10.1109/2945.817351

[7] Akenine-Möller T. Fast 3D triangle-box overlap testing. International Conference on Computer Graphics and Interactive Techniques. ACM SIGGRAPH 2005
[8] Halliday, Resnick, Walker. Data from Table 13-1. Fundamentals of Physics, 5E, Extended, Wiley, 1997
[9] Tensile Properties. NDT Resource Center; 2005. Available: http://www.ndt-ed.org/EducationResources/CommunityCollege/Materials/Mechanical/Tensile.htm (Accessed: Tuesday, April-15-08)

IntechOpen

IntechOpen

Advances in Haptics

Edited by Mehrdad Hosseini Zadeh

ISBN 978-953-307-093-3
Hard cover, 722 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

Haptic interfaces are divided into two main categories: force feedback and tactile. Force feedback interfaces are used to explore and modify remote/virtual objects in three physical dimensions in applications including computer-aided design, computer-assisted surgery, and computer-aided assembly. Tactile interfaces deal with surface properties such as roughness, smoothness, and temperature. Haptic research is intrinsically multi-disciplinary, incorporating computer science/engineering, control, robotics, psychophysics, and human motor control. By extending the scope of research in haptics, advances can be achieved in existing applications such as computer-aided design (CAD), tele-surgery, rehabilitation, scientific visualization, robot-assisted surgery, authentication, and graphical user interfaces (GUI), to name a few. Advances in Haptics presents a number of recent contributions to the field of haptics. Authors from around the world present the results of their research on various issues in the field of haptics.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Gabriel Telles O'Neill, Won-Sook Lee and Jeff William (2010). Haptic-Based 3D Carving Simulator, Advances in Haptics, Mehrdad Hosseini Zadeh (Ed.), ISBN: 978-953-307-093-3, InTech, Available from: http://www.intechopen.com/books/advances-in-haptics/haptic-based-3d-carving-simulator

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen