Computer-Aided Design and Applications
© 2009 CAD Solutions, LLC
http://www.cadanda.com

A Virtual Sculpting System Based on Triple Dexel Models with Haptics

Xiaobo Peng[1] and Weihan Zhang[2]

[1]Prairie View A&M University, xipeng@pvamu.edu
[2]Glidewell Laboratories, james.zhang@glidewelldental.com

ABSTRACT

This paper presents the development of a Virtual Sculpting system and addresses the issues of interactive freeform solid modeling with haptic interface. The workpiece and the sculpting tool are both represented by triple-dexel models and a sculpting tool is controlled by a force-reflective input device. Virtual sculpting is implemented via a series of Boolean operations that computationally subtract/unite successive tool volumes from the workpiece. The haptic device PHANToM Omni is used as an input device to provide the position and orientation data of the sculpting tool, and it is also used as an output device to provide force feedback during sculpting. The freeform workpiece can be converted into triangular mesh models using a novel triple-dexel surface reconstruction method.

Keywords: virtual sculpting, haptics, triple dexel, surface reconstruction.
DOI: 10.3722/cadaps.2009.645-659

1. INTRODUCTION

Freeform models are widely employed in the aerospace, automobile, mold/die making, toy design, and many other industries. The techniques of digital freeform model generation have been developed and deployed in virtual product design, digital prototyping, NC simulation, surgery simulation, human model simulation, etc. Virtual sculpting is well suited for the design of parts with freeform geometry, especially at the conceptual design stage [19], [20]. In a virtual sculpting system, the user creates a 3D object on a computer like molding a piece of clay.

Various geometric freeform modeling techniques have been developed in support of virtual sculpting, such as polygonal meshes modeling, Spline-based surface or solid modeling, and discrete volume modeling. Detailed survey is given in the next session. In a mesh based sculpting system, the mesh generation process is generally very labor intensive. The system also encounters the challenges of maintaining the correct topology and high mesh quality mesh. A surface representation based virtual sculpting system deforms the object by manipulating the control points of a parametric surface. The operation becomes tedious, time-consuming, and non-intuitive. Furthermore, it is a major challenge to generate complex shapes in a surface based method. Considerable efforts are needed to join the parametric patches so that their boundaries are geometrically continuous. A volume based virtual sculpting system has the major impediments of low accuracy and large memory cost. Commercial system FreeForm Modeling produced by SensAble Technologies is a powerful and intuitive system [25]. However, it exhibits organic like artifacts limited by voxel rendering.

Most recently, virtual sculpting technique has been augmented by virtual reality (VR) techniques which enables the user to create, modify and manipulate 3D CAD models intuitively, and at the same time allows the user to visualize CAD models in a virtual environment immersively. Incorporating a haptic interface to the virtual sculpting system provides the user with a more realistic experience. Force feedback enables the user to feel the model creation process like actual sculpting with physical materials. To continuously provide a stable sensation of touch, an update rate of at least 1,000 Hz is required for the haptic rendering [6, 24]. Less than 1 kHz update rate will result in uncomfortable perception of friction, ridges, and general roughness.

Previously the authors have developed a virtual sculpting system based on single-dexel modeling and sweep differential equation algorithm [19], [20]. In the single-dexel model, low sampling quality occurs in regions where the surface normals are nearly perpendicular to the ray direction. This paper presents a new sculpting system with triple-dexel modeling which overcomes the drawback of previous system.

In our virtual sculpting system, the workpiece and the sculpting tool are both represented by triple-dexel models. The sculpting tool is controlled by a force-reflective input device. Virtual sculpting is implemented via a series of Boolean operations that computationally subtract/unite successive tool volumes from the workpiece. The Boolean operation is performed between the dexel representation of the tool volume and that of the workpiece by comparing the sorted depth data for each dexel in x, y and z directions. The haptic device PHANToM Omni is used as an input device to provide the position and orientation data of the sculpting tool, and it is also used as an output device to provide force feedback during sculpting. The freeform workpiece can be converted into triangular mesh models using the novel triple-dexel surface reconstruction method [31].

The sculpting system has the following merits. First, the sculpting process is simulated by removing and attaching material which is analogous to real life sculpting. Therefore, it is intuitive and easy to operate. Second, compared to voxel data based sculpting system, the developed system is more efficient in terms of computational complexity and memory cost and the sculpted models are more accurate. Furthermore, the topology ambiguity problem in the mesh generation is avoided in the system, which is generally the challenge suffered by most mesh based sculpting system. Finally, the reconstructed mesh surface model can be output to CAD/CAM/CAE systems to perform geometric design, engineering analysis, and automated manufacturing applications. For example, the reconstructed mesh surface can be stored as a STL file which is widely accepted by CAD/CAM/CAE software.

The rest of the paper is structured as follows. In Section 2, related works are reviewed. Section 3 presents details on the triple-dexel model and Boolean operations. Mesh generation techniques are presented in Section 4 and haptic rendering are discussed in Section 5. Section 6 shows the implementation results. Conclusions are drawn in Section 7.

2. Related Work

2.1 Geometric Models in Virtual Sculpting

Solid modeling technique is the kernel of a virtual sculpting system which mathematically represents the freeform models. Various solid modeling techniques have been developed in support of virtual sculpting. Three types of geometric modeling methods are mostly used: boundary representations (B-reps) based on polygonal meshes, Spline-based surface or solid, and discrete scalar fields.

Mesh based sculpting system: In Jagnow and Dorsey's virtual sculpting system, the polymesh geometry is partitioned into coarse slabs with the detailed features stored as displacement maps at each surface [11]. Nienhuys presented a surface cutting method using Delaunay Triangulation in deformable objects [18]. Knopf and Igwe [13] developed a virtual sculpting framework base on mesh models represented by a self-organizing feature map (SOFM). A dental surgery simulation was developed by Wang et al. in which the tooth and tool are modeled using triangle mesh [28]. Gunn [7] introduced a collaborative haptic virtual sculpting environment in which the deformation of virtual clay is simulated by transforming the vertices of the polygonal model.

Spline-based sculpting system: Wong et al. [29] developed a Virtual 3D Sculpting system based on a parametric control hand surface which is defined by an Open-Uniform B-Spline tensor product surface.

Developed by Dachille et al. [6], a sculpting system utilized dual representation for B-spline surfaces in both mathematical and physical space. A collaborative distributed virtual sculpting system was introduced in which the deformable object is modeled by NURBS [14]. Zheng et al. [32] presented a surface representation model for freeform surface deformation to enhance the controllability of the surface shape. The surface model combines a displacement function and a function for representing an NURBS parent surface.

Discrete modeling based sculpting system: The Freeform Modeling system is a volume-based sculpting system commercially produced by SensAble Technologies Inc [25]. Barentzen [1] proposed an octree-based volume sculpting system, which was used to accelerate ray-casting based rendering. Hua and Qin [9] presented a haptic sculpting system based on volumetric implicit functions. Perry and Frisken [21] proposed a sculpting system with Adaptively Sampled Distance Fields to preserve sharp features with an efficient refinement. Barentzen and Christensen [2] extended their volume sculpting system using Level-Set Method. Recently, Ren et al. [22] presented a method of using the triple-dexel volumetric model and force feedback for virtual prototyping and manufacturing planning.

2.2. Deformation Methods in Virtual Sculpting

One deformation method in a virtual sculpting system is analogous to hand modeling of a clay in the physical world by "pinching and pulling" the virtual clay models. The systems [5-7], [13], [29] use this deformation method to manipulate the shape of the models. The other type of deformation method is analogous to wood or stone sculpting in the physical world by "removing and attaching" materials from the models. This type of deformation methods are utilized in many systems [9], [18], [20-22], [25], [28].

In some systems, physics-based models are utilized to reflect the dynamic properties of the virtual object. In Knopf and Igwe's system [13], the action of sculpting is simulated by moving the mesh model according to a simple dynamic model based on a mass-spring system. McDonnell et al. [15] introduced a virtual clay framework which synchronizes the subdivision solids and physics-based modeling to represent both the boundary and interior material of a solid. The system developed by Dachille et al. [6] utilized the physical-based mass-spring model to manipulate the B-spline surfaces. Chen et al. [5] incorporated the dynamic simulation of mass-spring systems in the touch-enabled modeling of soft object bounded by Bezier volume lattice.

2.3 Haptic Rendering

A haptic rendering algorithm consists of using information about the virtual environment to assign forces to be displayed, given the position, velocity, etc. of the operational point of a haptic interface. Based on the type of haptic interaction, the existing haptic rendering techniques can be divided into three kinds: point-based, ray-based, and 3D-based. The point-based haptic interaction algorithm simply models the haptic interface as a point [23], [33]. For the ray-based algorithm, the haptic interface is modeled as a line segment [3], [10]. Many efforts have focused on rendering haptic interactions between two 3D models [8], [12], [16].

3. SOLID MODELING

3.1 System Overview

The schematic of the virtual sculpting system configuration is shown in Fig. 1. The goal of this system is to provide the designer with an intuitive virtual sculpting environment including stereo viewing and haptic interface capabilities such that the users can focus on the design intent. Interactive modeling is implemented with VR hardware and software to enable the user to create and modify 3D freeform objects.

The virtual sculpting system functions as follows: A virtual tool controlled by a device capable of 6-DOF positioning/orienting is used to carve a virtual workpiece to obtain the design model based on the designer's intent. The design part is obtained by performing Boolean operations of the successive tool volumes from the workpiece. The solid modeling part of the system includes a triple-dexel generation module and a Boolean operation module. The solid modeling engine represents the volume of the sculpting tool for each segment of tool by 1) generating the triple-dexel representations of the tool and the workpiece by scan-converting their boundary representations, and 2) performing Boolean operations between the dexel representation of the workpiece and that of the tool.

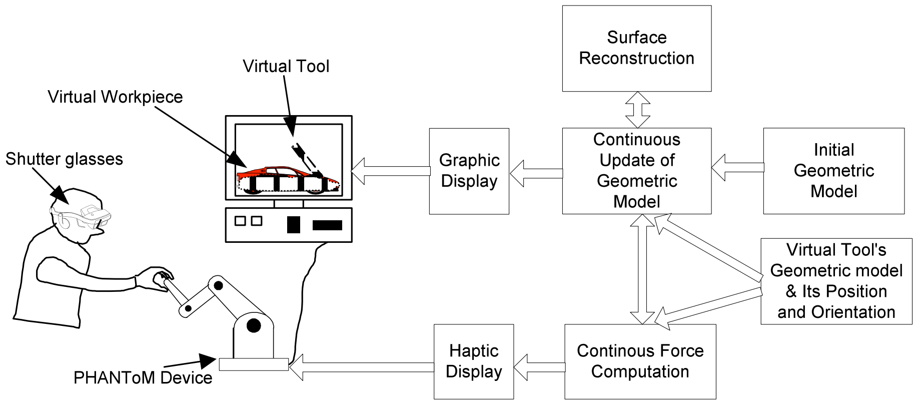
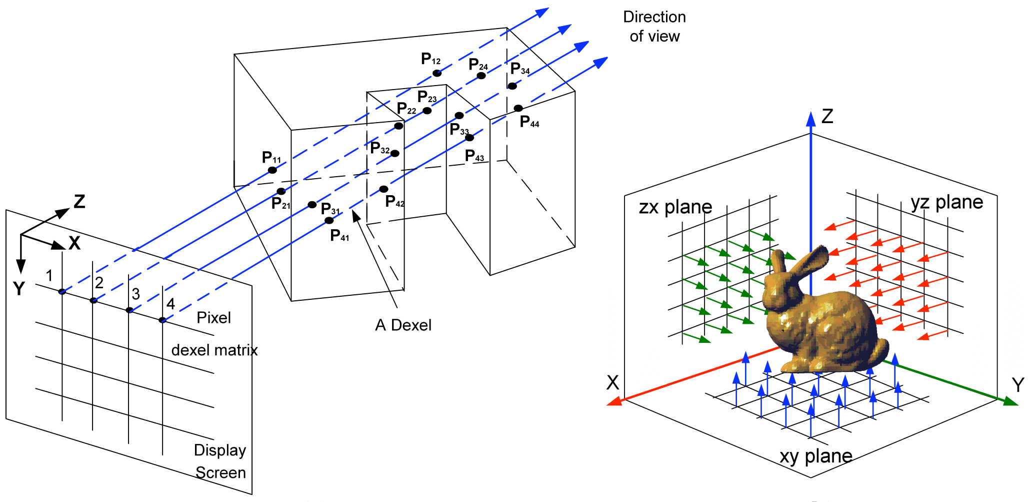


Fig. 1: Schematic of the system configuration.

3.2 Triple-dexel Representation

In keeping with the convention on the names pixel and voxel, Van Hook [27] introduced the notion of dexel as an abbreviation of 'depth element'. Single dexel representation of a solid is constructed via computing ray intersections with the solid. For a given solid, a set of parallel and equidistant rays are projected and intersected with the object as shown in Fig. 2(a). For each ray the intersection points are stored in the following manner: two points defining a line segment that is fully inside the solid make up a dexel. For example, in Fig. 2(a)., the two line segments $P_{21}P_{22}$ and $P_{23}P_{24}$ indicate that the points between them are inside the solid. However, in the single-dexel model, low sampling quality occurs in regions where the surface normals are nearly perpendicular to the ray direction. To overcome this problem, a triple-dexel modeling method developed by Zhang and Leu [31] has been integrated in the virtual sculpting system. A triple-dexel model is constructed by casting rays in three orthogonal directions (normally in x, y, and z directions) to discretize the model, as shown in Fig. 2(b). All dexels on a ray are sorted and concatenated into a dexel list. The dexel lists are organized into a dexel matrix as shown in Fig. 3. A dexel data contains the depth values of near intersection point and far intersection point and the properties of the dexel such as material, color, brightness, etc.



(a) (b)

Fig. 2: (a) Single dexel representation; (b) Triple-dexel model.

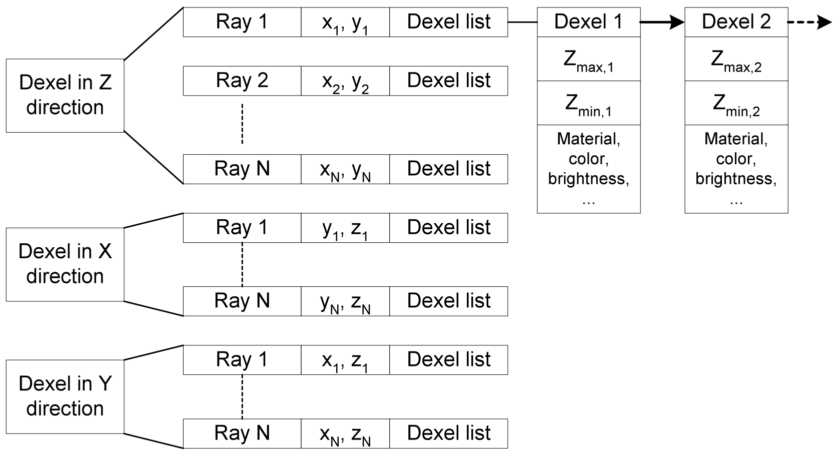


Fig. 3: Triple-dexel data structure.

3.3 Boolean Operation

Both the tool and the stock (initial workpiece) are initially represented by a polyhedral boundary representation, where the object surface is a faceted approximation composed of connected, non-overlapping triangles. The workpiece and the tool are scan-converted to obtain their triple-dexel representations in three orthogonal directions (x, y, and z directions). Boolean operations on triple-dexels are executed in x, y, z direction respectively by comparing and merging the depths value of the dexels. For example, for the dexels in z direction, the Boolean is obtained by comparing and merging the z value of the dexels. The Boolean difference and union algorithms in dexel data have been discussed in detail in the previous work [19]. Boolean difference is illustrated in this session using the dexels in z direction as example.

Using the linked list data structure illustrated in Fig. 3 simplifies the implementation of Boolean operation, which becomes comparison of one-dimensional z values between the workpiece and the tool swept volume. Since the operation is performed on dexel lists at each pixel position, x and y are invariants in the operation. The only variables that have to be considered are ($z_{\max}, z_{\min}$) of each dexel.

Fig. 4. shows six types of relationships between the ($z_{\max}, z_{\min}$) of the workpiece and that of the swept volume. The abbreviated symbols ZVN and ZVF represent the near and far z value of the tool. ZSN and ZSF represent the near and far z value of the workpiece. For each of these six cases, the system takes a different action to adjust the data structure of the workpiece as follows:

1) In the case of ‘ComeTo’, the tool doesn’t have a contact with the workpiece. No action is needed.
2) In the case of ‘GoAway', the tool also doesn’t have a contact with the workpiece. No action is needed.
3) In the case of “CutIn’, the tool cuts into the surface of the workpiece. The maximum z value of the workpiece is changed to the minimum z value of the swept volume.
4) In the case of “CutOut’, the tool moves out of the surface of the workpiece. The minimum z value of the workpiece is changed to the maximum z value of the swept volume.
5) In the case of “CutAll’, the tool cuts the workpiece dexel completely. The dexel of the workpiece is deleted.
6) In the case of “Merge’, the tool removes an inner part of the workpiece dexel. This results in two dexels for the workpiece.

4. MESH GENERATION

A novel surface reconstruction algorithm [31] is integrated into our virtual sculpting system to generate triangular mesh surface of the design model represented by triple-dexel data. Muller et al. [17] developed a triple-dexel based online milling simulation system. Ren et al. [22] developed a virtual sculpting system with haptic feedback by using the triple-dexel model. However, both of these studies did not reconstruct triangular surface models. The conversion from the triple-dexel model into triangular surface patches is an important issue. It enables designed model to be imported to any CAD/CAM/CAE systems to perform geometric design, engineering analysis, and automated manufacturing applications. Furthermore, the triangulated 3D model can be viewed in any directions as desired using standard routines of computer graphics software.

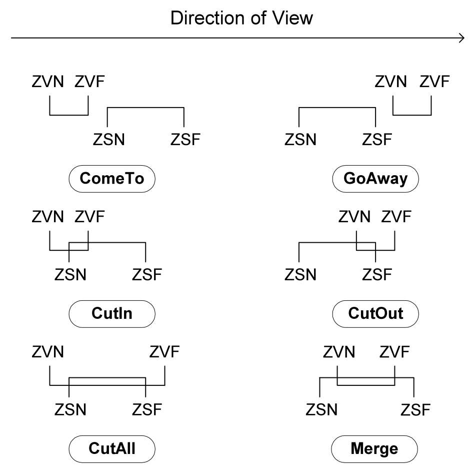


Fig. 4: Six possible relationships between dexel of the work piece and that of the tool.

A surface reconstruction from triple-dexel data algorithm is developed by first converting the triple-dexel data into contours on three sets of orthogonal slices and then generating the boundary surface in triangular facets from these contours. This method overcomes topological ambiguity problem which is suffered by most of the mesh generation algorithm. Contours on three orthogonal slices offer connectivity information among dexel points on each slice, thus not only the reconstructed surface model is topological correct and accurate, but also the reconstruct process is fast [26].

This mesh generation method consists of three steps: 1) the contour generation algorithm converts triple-dexel data into six sets of parallel contour slices in x, y, and z directions. For example, the dexel data in x direction is used to generate xy contours and xz contours. 2) the two sets of contours on the same plane generated from the first step are combined into one set of contours. Three sets of contours on xy, yz and zx planes are generated. For example, a xy contour is combined with a yx contour on the same slice to generate a contour parallel to xy plane. 3) a volume-based tiling algorithm is utilized to generate triangular facets from the three sets of contours. The schematic diagram of the method is shown in Fig. 5.

4.1 Contour Generation Algorithm

A contour generation algorithm has been developed to reconstruct contours from single-dexel data in our previous research [30]. The main challenge in defining the connection topology is to identify whether the dexel points lie on inner contours, such as point 6 and point 12 in Fig. 6., and to construct connections between the dexel points. A grouping criterion is defined which categorizes the dexels on two adjacent rays into different groups. If two dexel spaces on two adjacent rays are overlapped, they may form part of an inner contour. Therefore, the end points of the two dexel spaces are connected. For example, point 6 is connected with point 12 and point 7 is connected with point 13. Overlapped dexel spaces on every two adjacent rays separate dexels into different groups, and within each group the end points of the dexel spaces are connected. For example, along ray 3 and ray 4, dexels $d_{4,1}$, $d_{3,2}$, and $d_{3,3}$ are in one group and $d_{3,1}$ is in another group. Thus points 12 and 13, points 11 and 15, and points 14 and 16 are connected. After storing all the connections in a data structure table, the contours can be generated by traversing the table.

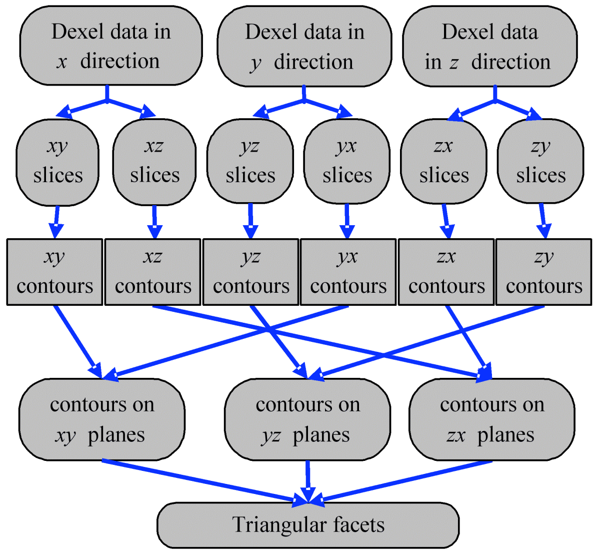


Fig. 5: Surface reconstruction from triple-dexel data.

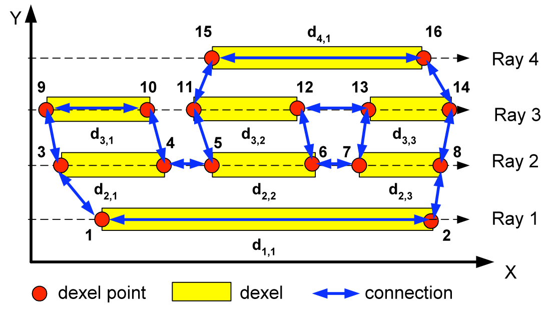
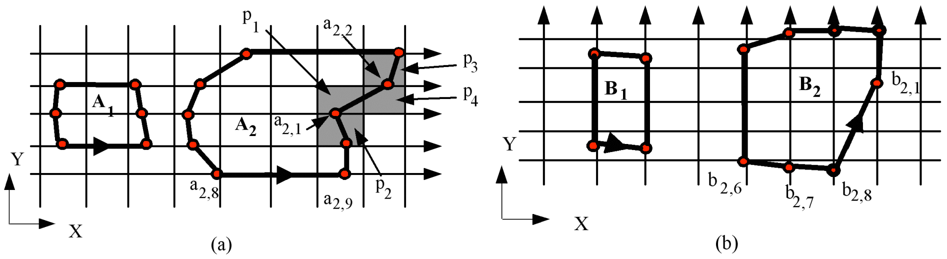


Fig. 6: Contour generation algorithm.

4.2 Contour Combination Algorithm

Two sets of planar contours are generated on each of xy, yz, and zx planes with total number of six sets after applying the contour generation algorithm. The objective of the contour combination algorithm is to more accurately represent the cross-sectional profiles of the 3D model. For example, in Fig. 7., contour A1 generated from dexel data in x direction is corresponded with contour B1 generated from dexel data in y direction to create contour C1. Likewise, contour A2 is corresponded and combined with B2 to generate contour C2.



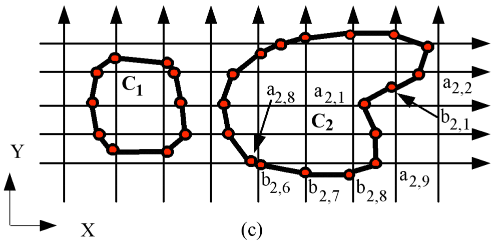


Fig. 7: (a) xy contours; (b) yx contours; (c) the combined contours.

Suppose a_i and a_{i+1} are two adjacent points on a xy contour. They are either on the same ray or on two adjacent rays as shown in Fig. 8. Thus the potential points from the corresponding yx contour between points a_i and a_{i+1} are within a localized area, as shown by the hatch area in Fig. 8. Given two correspondent contours A and B, starting from two adjacent points a_0 and a_1 in contour a, the contour b is searched to find points which fall into the localized area. The searched points, represented by b_j to b_{j+n}, are inserted to the list of points between a_0 and a_1. This process is continued until all the points in contour A have been searched for insertion of points from contour B. The completion of the process will result in a combined contour C. The details of the algorithm and pseudo codes are described in [31].

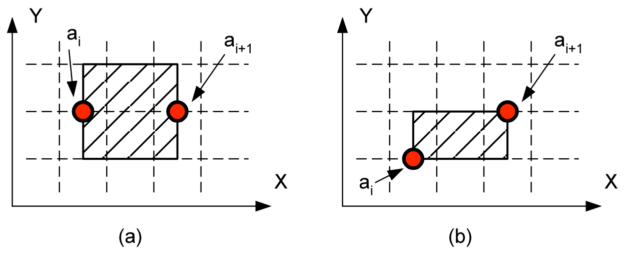


Fig. 8: Localized area between points a_i and a_{i+1}: (a) points on the same ray; (b) points on adjacent rays.

4.3 Surface Tiling

After the contour combination process, three sets of contours on orthogonal slices are generated. The volume-based tiling algorithm developed by Svitak and Skala [26] is utilized to reconstruct the boundary surface from these contours. Given a triple-dexel data with M, N, and O numbers of divisions in the x, y and z axes, respectively, the 3D space is divided into $M \times N \times O$ rectangular boxes. The algorithm first identifies the Boundary Sub-Volumes (BSVs) that are the boxes having non-null intersections between their edges and the solid's boundary surface. Second, the three orthogonal sets of contours are searched to find a close loop of vertices within each BSV. Finally, triangular facets are created within each BSV by patching these vertices.

The algorithm identifies the BSVs by searching the intersection points within the object's boundary surface along the three orthogonal sets of rays. For example, in Fig. 9(a)., the intersection point between ray R1 and the object's boundary surface is point p and thus boxes A, B, C and D are the BSVs. Within each BSV, the boundary surface of the object forms a close loop. To find the close loop of vertices within a BSV, the algorithm starts from the point on the bottom of the BSV and ends when coming back to the starting point. Taking Fig. 9(b). as an example, within BSV1, the search starts from point a on the bottom. After searching contour c2 on the xy plane, the next point found is point b. Because point b is on both contour c2 and contour c4, thus contour c4 is searched to find the next point, which is point e. The search continues to find point d and then point a, which is the starting point. Thus, within BSV1, the close loop of points a$\longrightarrow$b$\longrightarrow$e$\longrightarrow$d$\longrightarrow$a is generated and then patched into two triangular facets abe and aed. Likewise, the close loop b$\longrightarrow$c$\longrightarrow$f$\longrightarrow$e$\longrightarrow$b is generated within BSV2.

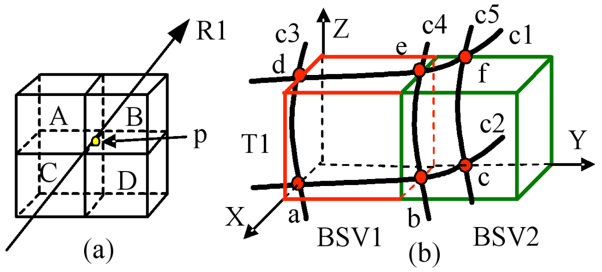


Fig. 9: (a) Identification of boundary sub-volumes; (b) generation of surface patches.

5. HAPTIC INTERFACE

Haptic interface uses an electro-mechanical feedback device which enables a user to touch, feel, and manipulate virtual objects. Incorporating a haptic interface enhances the fidelity of the virtual sculpting system. The 1000 Hz update rate of the haptic rendering must be guaranteed because it is the precondition to run the haptic device. This stringent requirement leaves very little CPU time for computing the tool-workpiece interface and reflecting the force back to the user in real time. Providing haptic interface in virtual sculpting is a challenging problem because the boundary of the workpiece keeps changing when the workpiece is undergoing sculpting. More specifically the challenge issues rely on:

1) Detect the collision between the virtual tool and the designed model in real time and calculate the realistic force to simulate the sculpting process. Most of the previous haptic rendering algorithms have been focused on the force model interacting with a static solid model [4]. How to interact a deformable object haptically is still a promising research topic.
2) Sustain the 1K Hz update rate of the haptic rendering process. Given the fact that the virtual sculpting system is a real time simulation system, pre-computed data is not available. All the calculations including collision detection and force model generation need to be accomplished in real time.
3) Maintain different update rates of various components of the virtual sculpting system. In the system, there are three main components: geometric modeling, graphics rendering and haptic rendering. The three processes have different computation complexities.

The haptic rendering algorithm in our system consists of two parts: (a) collision detection and (b) collision response. As the user manipulates the stylus of the haptic device, the new position and orientation of the stylus are obtained. Then the position and orientation of the tool are input to the collision detection process. If a collision is detected between the haptic stylus and virtual workpiece, the contact force is calculated using a force model for collision response. The force is then conveyed through the haptic device to provide the force feedback.

5.1 Collision Detection

As the tool moves, collisions between the tool and the object's surface must be checked. As described in Section 3, both the workpiece and the virtual tool are represented by triple-dexel model in the virtual sculpting system. A simple way to detect collisions is to check if the dexels of the virtual tool overlap with the dexels of the workpiece in x, y, and z directions respectively. If the dexel line segment intersects any one of the workpiece dexel in any direction then a collision occurs. This checking process can be extremely time-consuming, however, if the object consists of a large number of dexels. For example, the triple-dexel data of a workpiece consists of over 480,000 dexels with the resolution of 400 by 400. It is proven that such a time-consuming computation procedure (by checking the overlapping between all the workpiece dexels with the tool dexels) could not satisfy the 1000Hz update rate stringent requirement.

A localized collision detection technique as given below can be utilized to speed up the process. Fig. 10. illustrates the collision detection algorithm using the dexel data in z direction. First, the dexel data of the virtual tool is located which is stored in a linked-list data structure DexelT(I, J), where I and J are the index of the dexel data which can reflect the x and y values of the dexel. Then, a 2D bounding box in XY plane is calculated which encloses the virtual tool as shown in Fig. 10(a). Suppose the bounding box is represented by four corner vertices with the indices of (i, j), (i+m-1, j), (i, j+n-1) and (i+m-1, i+n-1). It is straightforward to locate the dexel data of the workpiece within the bounding box. Here the workpiece dexel data is indicated as DexelW(I, J). The z values of DexelT(I, J) and DexelW(I, J) are compared as shown in Fig. 10(b). The six relationships described in the Boolean Difference algorithm as shown in Fig. 4 are used to identify the collision. If any case of “CutIn”, “CutOut”, “CutAll” or “Merge” happens, there is a collision. Otherwise there is no collision.

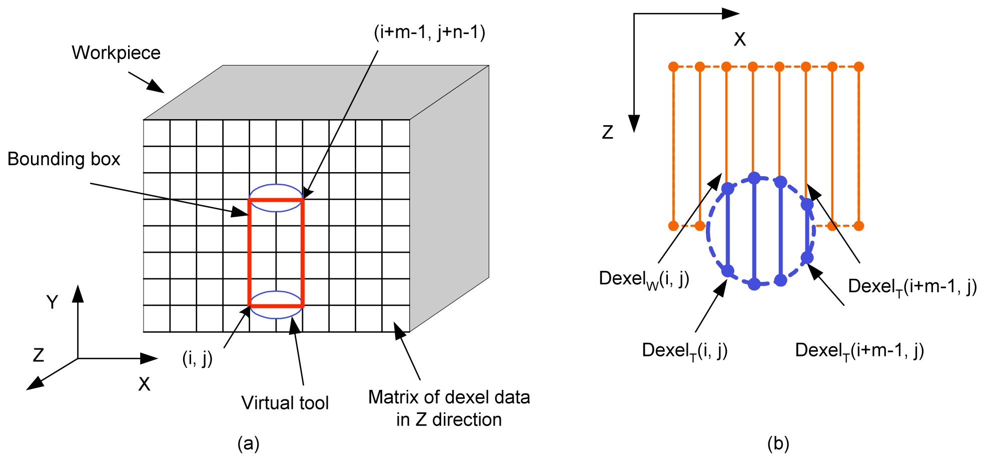


Fig. 10: Collision detection based on dexel data.

Compared to the workpiece, the virtual tool is relative small. By using the bounding box, only small amount of dexel data is searched for the collision detection purpose. The detection is accomplished by simply comparing a single depth value. Same process is executed using the dexel data in y and z directions. A fast and accurate collision detection to sustain the requirements discussed earlier in this section can be achieved.

5.2 Force Generation

The god-object concept suggested by Zilles and Salisbury [33] is adopted to haptically render the dexel model in the virtual sculpting system. Simply the virtual tool tracked by haptic device is simulated as a single point called the Haptic Interface Point (HIP). The ideal haptic interface point (IHIP) is considered to follow the trace of the HIP. The HIP can penetrate the object and the IHIP is constrained on the surface such that it doesn’t penetrate any object.

When a collision is detected, the first step of the force generation algorithm is to extract a virtual boundary surface which will be utilized to interact with the virtual tool. Due to the stringent requirement of 1K Hz update rate, it is impossible to construct the surface for the whole design model. The x and y position of the HIP can be tracked from the haptic device. As illustrated in Fig. 11(a)., the dexel points P1, P2 and P3 which contain the HIP can be located so that they form a triangle. The HIP is enclosed in this triangle called “active triangle”. As illustrated in Fig. 11(b)., a virtual surface can be modeled as an elastic element. Then the reaction force acting on the tool during the interaction will be:

$$F = k_p \Delta\vec{x} + k_d \vec{v} \tag{5.1}$$

where k_p is the stiffness coefficient, k_d is the damping coefficient, $\Delta\vec{x}$ is displacement and $\vec{v}$ is velocity. For frictionless interactions, the reaction force F is normal to surface of the object that the sculpting tool collides with. Keeping the stiffness coefficient low would make the surface perceived soft. If the virtual tool penetrates the object, the active triangle is constructed as shown in Fig. 11(a). The Ideal Haptic Interface Point (IHIP) stays on this surface. Then the nearest distance from the current HIP to the active triangle is calculated. As long as the tool moves out the covering range of the active triangle, a new triangle is formed as the new active triangle.

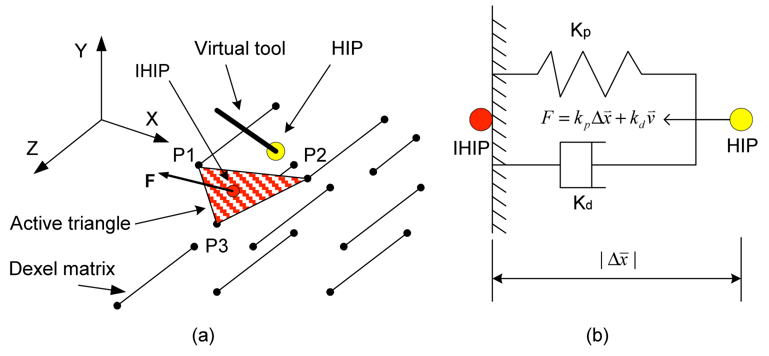


Fig. 11: Force generation algorithm.

6. IMPLEMENTATION AND RESULTS

Our system is implemented and tested on a Microsoft Windows XP workstation with a 2.8G Hz CPU, 512 MB RAM and GeForce4 MX 420 graphics card with 64MB memory. The entire system is written in Microsoft Visual C++ and the graphics rendering component is built upon OpenGL and GLUT. The haptics interface is implemented using the PHANToM Omni Device and the OpenHaptics Toolkit software available from SensAble Technologies. This device has three motors and six encoders to enable 6-DOF motion tracking and 3-DOF force feedback. The OpenHaptics is a software toolkit that enables application developers to interact with the haptic device and create a haptic environment at the object level.

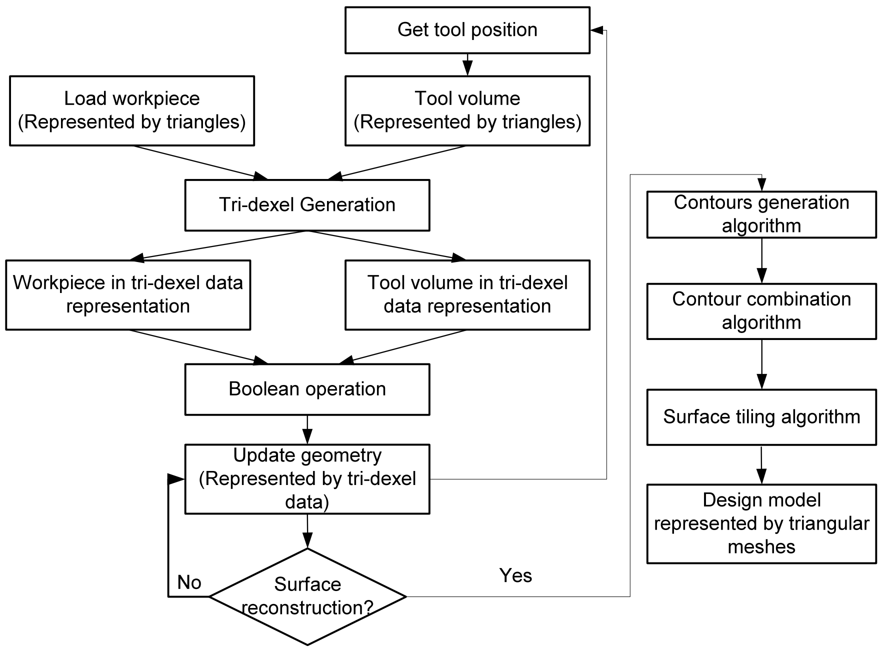

Get tool position
Load workpiece (Represented by triangles)
Tool volume (Represented by triangles)
Tri-dexel Generation
Workpiece in tri-dexel data representation
Tool volume in tri-dexel data representation
Boolean operation
Update geometry (Represented by tri-dexel data)
Surface reconstruction?
No
Yes
Contours generation algorithm
Contour combination algorithm
Surface tiling algorithm
Design model represented by triangular meshes

Fig. 12: The solid modeling diagram of the virtual sculpting system.

6.1 Solid Modeling Implementation

The geometry modeling diagram is shown in Fig. 12. Both the tool and the stock (initial workpiece) are represented by polyhedral boundary representation, where the object surface is a faceted approximation composed of connected, non-overlapping triangles. The tool location is specified by a translation and a rotation tracked by the PHANToM Omni. The workpiece and tool volumes are scan-converted to obtain their triple-dexel representations. Boolean operations on triple-dexels are obtained by comparing and merging the depth value of the involved dexels in x, y, and z direction respectively. In the process of the sculpting, the surface reconstruction module can be executed to convert the triple-dexel model to a triangular mesh model as described in Section 4.

6.2 Multithreading

There are three main components in the virtual sculpting system which are geometric modeling, graphics rendering, and haptic rendering. The haptic rendering must be guaranteed to maintain the high update rate on the order of 1K Hz. The graphics rendering needs 20~30 Hz update rate to perform smooth graphics display. The geometric rendering is the most time-consuming procedure. Obviously one single process to run those three components in sequence will not guarantee the suitable update rates for each procedure.

A multithread computation environment as shown in Fig. 13. is built which asynchronously allows maintaining suitable update rates for the various components in the run time. The triple-dexel data is shared by all the three processes and allows them to communicate with each other. There are three threads running simultaneously (but asynchronously) within our virtual sculpting system; they are:

- Solid Modeling Thread: Tasks such as creating triple-dexel data structures, performing Boolean operations and surface reconstruction are executed in this thread.
- Haptic Rendering Thread: This is the haptic thread created by the OpenHaptics functions of the PHANToM Omni manipulator. It performs collision detection and force generation.
- Graphics Rendering Thread: Graphical rendering based on the OpenGL library is executed in this thread. It is responsible for activities related to geometry display and the user interface such as Windows.

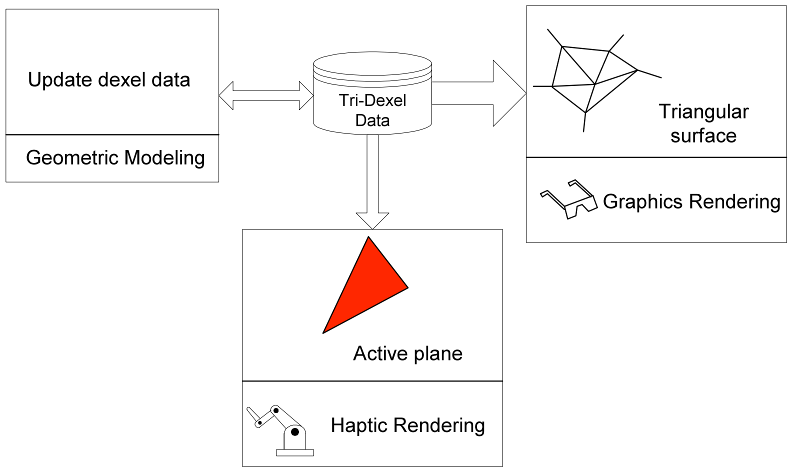


Fig. 13: Multi-threading environment in virtual sculpting system.

6.3 Designed Model Examples

Fig. 14. shows some freeform models created with the virtual sculpting system. A Chinese character is written on a circular pad using a cone shaped virtual cutter in Fig. 14(a). Haptic force is generated during the modeling process to give the user a sense of touch. With this force, the user can control the depth of the character more precisely, which provides a great help in the handwriting. At the same time, the tool can be rotated and translated in six degrees of freedom. Ball and cylinder shaped cutters are used to create the two eyes of a cat shown in Fig. 14(b)-(d). The original model of the cat's head (without eyes) is imported in STL file into the virtual sculpting system as shown in Fig. 14(b). In Fig. 14(c), eye cavities are first created by sculpting the imported model. Two eyeballs are then added and placed in the cavities by performing Boolean union of two dexel data. Finally, a tail is created by using a tool with ball shape as shown in Fig. 14(d). After applying the contour generation algorithm and tiling the contour into a triangularized patch, the modified cat model can be viewed in any directions.

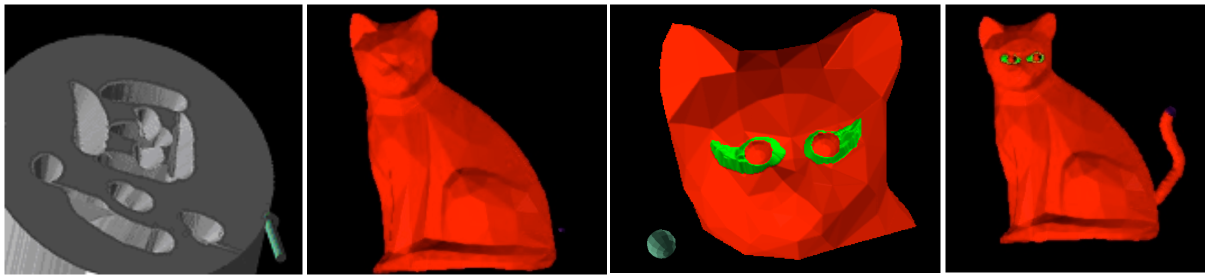


Fig. 14: (a) A virtually carved Chinese character; (b)-(d) a cat.

6.4 Surface Error Analysis

The reconstructed surface is watertight because of using the volume-based surface tiling algorithm. Each dexel point inside the boundary sub-volume is guaranteed to have connection points to form a close loop. However, the reconstructed surface is still an approximation of the original shape. To estimate the quality of the reconstructed surface, the reconstructed surface error is defined as the ratio of the Hausdorff distance between the original surface and the reconstructed surface to the diagonal length of the bounding cuboid. Hausdorff distance is the maximum distance between two non-empty sets of data. To calculate the surface error, the reconstructed surface model is sampled from three orthogonal directions and the Hausdorff distance between the sampled points and the original surface model is calculated and normalized by the diagonal length of the bounding cuboid. The normalized surface error *e* between the sampled points P and the original surface S is:

$$e = \frac{d_H(P,S)}{L} \tag{6.1}$$

where L is the diagonal length of the bounding cuboid. d_H(P, S) is the Hausdorff distance between the set of sampled points P = {p_1...p_n} of the reconstructed surface and the original surface model S. It is evaluated as the maximum of the distances between point set $p_i \in P$ and the surface S , i.e.,

$$d_H(P,S) = \max_{p_i \in P} d(p_i, S) \tag{6.2}$$

where the distance between a point in $p_i \in$ R3 and a surface S is given by:

$$d(p_i, S) = \min_{q \in S} \| p_i - q \|_2 \tag{6.3}$$

The Stanford bunny model is used to test the surface errors. The surface errors of the models reconstructed from the single-dexel data and the triple-dexel data are compared in Tab. 1., which clearly shows that the reconstructed surface from the single-dexel data has larger errors than the surface reconstructed from the triple-dexel data for the same ray resolution.

Resolution	Normalized error from the triple-dexel model	Normalized error from the single-dexel model
50*50	0.7525%	1.365%
100*100	0.4390%	0.658%

Tab. 1: Surface errors of the reconstructed bunny model.

Fig. 15. illustrates the surface quality improvement from the triple-dexel data over the single-dexel data. Fig. 15(a). and Fig. 15(c). show the results of surface reconstruction from single-dexel data, and Fig. 15(b). and Fig. 15(d). show the corresponding results of surface reconstruction from triple-dexel data. These figures clearly show that the generated surface from the triple-dexel data is more accurate than the reconstructed surface from the single-dexel data when using the same ray resolution.

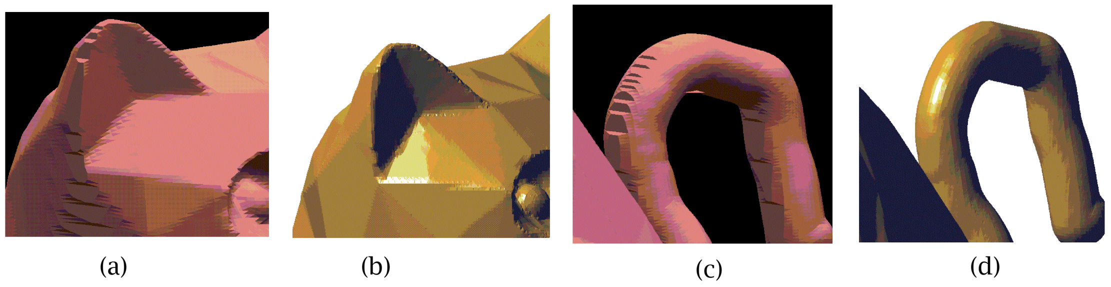


Fig. 15: Comparisons between the surfaces reconstructed from single-dexel data in (a), (c) and from triple-dexel data in (b), (d).

7. CONCLUSIONS

A virtual sculpting system has been developed in which one can use the PHANToM Omni device to manipulate a virtual tool to carve out a freeform 3D object from a virtual stock with haptics feedback. Solid modeling, computer graphics, and haptics rendering techniques are researched and integrated in a multi-threading computation environment to maintain the suitable update rate for various components of the virtual sculpting. The geometric modeling technique is based on the triple-dexel geometry representation method to compute the boundary of the tool volume and the virtual workpiece, and to perform Boolean operations between the tool volume and the virtual stock to simulate the sculpting process. A new method to convert triple-dexel data into triangular meshes has been integrated into the virtual sculpting system, to enable viewing the created model in any directions. Three sets of contours on orthogonal slices are generated from triple-dexel data by a contour generation algorithm and a contour combination algorithm. A volume-based tiling algorithm is then utilized to generate the boundary surface of the 3D object in triangular patches from these contours. Comparing with the surfaces reconstructed from single-dexel data and from voxel data with the same resolution, our triple-dexel based method has a higher surface accuracy. The haptic rendering for the sculpting process is developed with the spring-damping force model. Our haptic rendering algorithm including collision detection and force generation can achieve the high update rate of 1,000 Hz by extracting the dexel data locally.

8. REFERENCES

[1] Barentzen, J. A.: Octree-based volume sculpting, IEEE Visualization, 1998, 9-12.
[2] Barentzen, J. A.; Christensen, N. J.: Volume sculpting using the level-set method, Proceedings of Shape Modeling International, 2002, 175 – 182.
[3] Basdogan, C.; Ho, C.; Srinivasan, M. A.: A ray-based haptic rendering technique for displaying shape and texture of 3D objects in virtual environments, Proceedings of the ASME Dynamic systems and Control Division, Dallas, TX, 1997, 77-84.
[4] Basdogan, C.; Srinivasan, M. A.: Haptic rendering in virtual environments, Handbook of Virtual Environments, edited by Stanney, K. M., Lawrence Erlbaum Associates, 2001, 117-134.
[5] Chen, H.; Sun, H.; Jin, X.: Interactive haptic deformation of dynamic soft objects, Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications, 2006, 255-261.
[6] Dachille, F.; Qin, H.; Kaufman, A.: A novel haptics-based interface and sculpting system for physics-based geometric design, Computer Aided Design, 33(5), 2001, 403-420.
[7] Gunn, C.: Collaborative virtual sculpting with haptic feedback, Virtual Reality, 10(2), 2006, 73-83.
[8] Gregory, A. D.; Mascarenhas, A.; Ehmann, S.; Lin, M. C.; Manocha, D.: Six degree-of-freedom haptic display of polygonal models, IEEE Visualization, 2000, 139-146.
[9] Hua, J.; Qin, H.: Haptic sculpting of volumetric implicit functions, Proceedings of Ninth Pacific Conference on Computer Graphics and Applications, 2001, 254-264.

[10] Ho, C.; Basdogan, C.; Srinivasan, M. A.: Modeling of force and torque interactions between a line segment and triangular surfaces for haptic display of 3D convex objects in virtual and teleoperated environments, International Journal of Robotics, 19(7), 2000, 668-684.

[11] Jagnow, R.; Dorsey, J.: Virtual sculpting with haptic displacement maps, Proceedings of Graphics Interface, Calgary, Alberta, Canada, May 27-29, 2002, 125-132.

[12] Johnson, D. E.; Willemsen, P.; Cohen, E.: Six degree-of-freedom haptic rendering using spatialized normal cone search, IEEE Transactions on Visualization and Computer Graphics, 11(6), 2005, 661-670.

[13] Knopf, G. K; Igwe, P. C.: Deformable mesh for virtual shape sculpting, Robotics and Computer-Integrated Manufacturing, 21(4-5), 2005, 302-311.

[14] Li, F.; Lau, R.; Ng, F.: Collaborative distributed virtual sculpting, Proceedings of IEEE Virtual Reality Conference, 2001, 217-224.

[15] McDonnell, K. T.; Qin, H.; Wlodarczyk, R. A.: Virtual Clay: a real-time sculpting system with haptic toolkits, Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics, 2001, 179-190.

[16] McNeely, W.; Puterbaugh, K.; Troy, J.: Six degree-of-freedom haptic rendering using voxel sampling, Proceedings of the Conference on Computer Graphics and Interactive Techniques, 1999, 401-408.

[17] Muller, H.; Surmann, T.; Stautner, M.; Albersmann, F.; Weinert, K.: Online sculpting and visualization of multi-dexel volumes, Proceedings of ACM Symposium on Solid and Physical Modeling, Seattle, Washington, June 16-20, 2003, 258-261.

[18] Nienhuys, H. W.: Cutting in Deformable Objects, Ph. D. Thesis, Utrecht University, Utrecht, Netherlands, 2003.

[19] Peng, X.; Leu, M. C.: Interactive solid modeling in a virtual environment with haptic interface, Virtual and Augmented Reality Applications in Manufacturing, edited by Ong, S. K. and Nee, A. Y. C., Springer-Verlag London Limited, London, 2004, 43-61.

[20] Peng, X.; Zhang, W.; Leu, M. C.: Freeform modeling using sweep differential equation with haptic interface, Journal of Virtual and Physical Prototyping, 1(3), 2006, 183-196.

[21] Perry, R. N.; Frisken, S. F.: Kizamu: a system for sculpting digital characters, Proceedings of ACM SIGGRAPH, 2001, 9–12.

[22] Ren, Y.; Lai-Yuen, S. K.; Lee, Y. S.: Virtual prototyping and manufacturing planning by using tri-dexel models and haptic force feedback, Virtual and Physical Prototyping, 1(1), 2006, 3 – 18.

[23] Ruspini, D. C.; Kolarov, K.; Khatib, O.: Haptic interaction in virtual environments, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1997, 128-133.

[24] Salisbury, K.; Conti, F.; Barbagli, F.: Haptic rendering: introductory concepts, IEEE Computer Graphics and Applications, 24(2), 2004, 24-32.

[25] SensAble Technologies Inc.: FreeForm Modeling System. http://www.sensable.com/ products-freeform-systems.htm.

[26] Svitak, R.; Skala, V.: A robust technique for surface reconstruction from orthogonal slices, Machine Graphics & Vision, 12(3), 2004, 221-233.

[27] Van Hook, T.: Real time shaded NC milling display, Computer Graphics, 20(4), 1986, 15-20.

[28] Wang, D.; Zhang, Y.; Wang, Y.; Lee, Y. S.; Lu, P.; Wang, Y.: Cutting on triangle mesh: local model-based haptic display for dental preparation surgery simulation, IEEE Transactions on Visualization and Computer Graphics, 11(6), 2005, 671-683.

[29] Wong, J.; Lau, R.; Ma, L.: Virtual 3D sculpting, Journal of Visualization and Computer Animation, 11(3), 2000, 155-166.

[30] Zhang, W.; Peng, X.; Leu, M. C.; Zhang, W.: A novel contour generation algorithm for surface reconstruction from dexel data, ASME Journal of Computing and Information Science in Engineering, 7(3), 2007, 203-210.

[31] Zhang, W.; Leu, M. C.: Surface reconstruction using dexel data from three sets of orthogonal rays, ASME Journal of Computing and Information Science and Engineering, 9(1), 2009.

[32] Zheng, J. M.; Chan K. W.; Gibson, I.: Constrained deformation of freeform surfaces using surface features for interactive design, International Journal of Advanced Manufacturing Technology, 22(1-2), 2003, 54-67.

[33] Zilles, C. B.; Salisbury, J. K.: A constraint-based god-object method for haptic display, Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, 1995, 146-151.