JCSE Journal of Computing Science and Engineering

Regular Paper
Journal of Computing Science and Engineering,
Vol. 13, No. 2, June 2019, pp. 66-77

Voxel-based Haptic Rendering using Adaptive Sampling of a Local Distance Map

Kimin Kim and Jinah Park*
School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Korea
kiminkim@kaist.ac.kr, jinahpark@kaist.ac.kr

Abstract
Change of shape is an essential topic in research related to virtual reality-based sculpting, mechanical machining, and surgery training of tissue cutting. If the shape of an object in the virtual environment is modified at high temporal rates, it is very difficult to resolve contacts between objects due to the fact that the existing haptic rendering algorithm depends on expensive preprocessing strategies such as bounding volumes, bounding volume hierarchies, and distance fields. This paper investigates a haptic rendering algorithm in virtual environments with shape changes. We introduce a volumetric collision model and an on-the-fly contact normal computation method with a local distance map, which allow for shape changes of the volume model at time-critical haptic rendering. This approach significantly reduces the voxel readout, which allows the model to handle more than ten thousand voxels at haptic rates. The real-time tooth probing and cutting simulation results support the benefits of our approach.

Category: Human-Computer Interaction

Keywords: Haptic rendering; Distance map; Voxel-based simulation; Dental simulation

I. INTRODUCTION

Virtual reality-based applications for medical training mimic the physical phenomena that occur in deforming [1], cutting [2], and bleeding [3] with soft tissue as well as milling with hard tissue [4, 5]. Real-time interactions in these applications are important in enhancing medical skills and dexterity. Therefore, both visual and haptic cues should be properly provided not only to convey the deliberate intentions of the operators but also to improve their visual-spatial skills. In surgery simulation with real-time interaction, the topological changes in the physical representation are one of the essential components.

There is a large body of literature that has investigated haptic rendering algorithms for rigid and deformable bodies. Many of these studies used precomputed data structures to reduce the complexity for collision resolution since their topology was preserved [6-9]. On the other hand, some studies have challenged other bodies with topological changes caused by cutting, drilling, milling, and fluids [1, 4, 5, 10]. They derived the position and normal vector of the surface from an instantaneous configuration while the haptic feedback was calculated.

Although we primarily focus on asymmetric contact models, such as Voxmap-PointShell (VPS) [8] and its improved method using a distance field [9], some of the presented approaches can be transferred to haptic rendering algorithms with volumetric data. Specifically, the focus of our development is on an updated collision model and the contact resolution involved in shape

Open Access http://dx.doi.org/10.5626/JCSE.2019.13.2.66 http://jcse.kiise.org

Received 03 March 2019; **Accepted** 22 May 2019
*Corresponding Author

changes. We contribute to an efficient and stable contact resolution for haptic-enabled virtual environments with shape changes by employing a volumetric collision model as well as a normal estimation method based on the adaptive sampling of a local distance map. Our approach significantly reduces the voxel readout, which allows the model to handle more than ten thousand voxels at haptic rates. Moreover, the proposed method is highly parallelizable because each voxel in contact only involves local operations. Using our collision model, it is possible to generate a stable haptic display under progressive (plastic) deformation, such as in the cavity removal of a carious tooth structure in dentistry [11] or bone machining in orthopedic surgery [12, 13].

II. RELATED WORK

A. Haptic Rendering of Volumetric Data

Over the past two decades, there have been rapid developments in haptic rendering algorithms. Early studies attempted to use point interaction (3-degree-of-freedom [3-DOF] haptic rendering) to reduce the complexity of both the device and the development [14]. However, the arbitrary object-object interaction demanded a more complex collision detection and response, leading to the need for 6-DOF haptic rendering.

Direct haptic rendering with volume data was introduced by Avila and Sobierajski [15]. They used central differences with 1-byte density values to estimate the surface normal vector. One of the first studies on simulated bone surgery was conducted by Gibson et al. [16]. They used the binary volume generated from segmentation and applied smoothing (tri-linear interpolation) to reduce the unstable force feedback caused by the binary nature of the data. They concluded that the preprocessing cannot be done if the topology of the objects is actively changed, which is unfortunate as preprocessing is faster than calculations performed on the fly. Palmerius et al. [17] developed an algorithm for proxy position estimation to handle time-varying volume data with haptic interaction.

Several studies have examined the algorithm with various representations (e.g., polygonal surface, parametric surface, implicit surface, and volumetric model); the VPS, proposed by McNeely et al. [18], has been particularly widely investigated. Renz et al. [19] introduced a method for making a point shell tightly and stabilizing the collision force. Wan and McNeely [20] suggested a quasi-static approach to solve the motion of the dynamic object in place of Newtonian single-body dynamics, as it avoids the computational instability caused by the Newtonian dynamic model of the VPS.

The most recent studies on the VPS, which were conducted by Barbič and James [9], demonstrated continuous contact force and torque using the distance field. They also presented geometrically complex rigid and deformable object interaction using a reduced deformable model and a multi-resolution nested point shell. The advantage of their method is that the deformation simulation and multi-contact resolution run together in one loop at a haptic cycle; this has inspired our on-the-fly contact normal computation that allows for shape changes of the volume model at the haptic cycle.

B. Normal Estimation from a Discretized Object

The precision of the surface normals significantly affects both the image quality of visual rendering and the fidelity of haptic rendering. The existing techniques used for normal computation in a 3D discrete space can be categorized into the following two subgroups: image space approaches and object space approaches. Image space approaches derive normals from the neighborhood information of the projected images, such as depth images or the Z-buffer. Object space approaches compute normals by using neighborhood information in the voxel space.

Normal estimation of discrete objects in 3D is a common problem in rendering volume data collected by medical imaging devices such as a computed tomography (CT) scanner. Many shading methods have been investigated [21-23]; the classical method is gradient-based shading that uses neighborhood information in the image space. Gordon Reynolds [23] computed gradient vectors by applying different operators at visible points in the Z-buffer. Whenever the volume data were modified, the normals were recalculated. This method required less time to recalculate normal vectors because the normal computation was only coupled with the depth values in the Z-buffer as opposed to the representation of the volume data. However, the depth values in the Z-buffer were generated at a specific projection plane related to a viewpoint. Depending on the view position, there may be hidden surfaces and ambiguity of depth values adjacent to the object boundary. In order to reduce these issues, Gordon Reynolds [23] used a weighted average of the forward and backward differences.

Object space methods are suitable for ray tracing and haptic rendering where normal vectors are used for collision resolution in 3D space. Thürmer and his colleagues [24-26] thoroughly investigated the computation of normal vectors on binary discrete surfaces in the object space. They presented a formula that covered any neighborhood sizes and weighting factors to compute the normal vector at a point on a discrete surface. The results showed that using weighting factors reduced errors from the cubic neighborhood; the most effective weighting factor was the reciprocal of the square of the Euclidean distance. They obtained the largest error corrections when increasing the neighborhood size to three. They also proposed a local separation criterion of surfaces for preserving slope discontinuities and small details. However, object space methods require a large neighborhood size in order to ensure smoother change of the normal, because the slope changes of the surface are distributed over unit steps.

New object space approaches have also been proposed. For example, Tellier and Debled-Rennesson [27] introduced a method using a tangential line in 2D slices of a 3D image for the normal computation of digital objects.

Flin et al. [28] proposed the gradient vector field analysis of the object distance map and adaptive filtering around each surface voxel using angle symmetry criteria. Fourey and Malgouyres [29] also suggested on-surface convolution, which is an extension of a 2D digital filter for discrete surfaces.

III. COLLISION MODEL

Our volumetric collision model involves two different representations: a binary volume model for objects with shape changes and a signed distance field for tool objects. The signed distance field is also stored in a discretely sampled volume; every voxel contains the minimum distance from the surface. For the broad phase, the octree of the volume model is traversed in pre-order, and the bounding box of each node being visited is tested against the bounding volume of the tool, such as a bounding capsule, a sphere, or a box. All of the leaf (or terminal) nodes colliding with the tool object are passed to the narrow phase, which precisely identifies voxels colliding against the tool object. Given the position of the potentially colliding voxel, the distance to the tool object is computed using tri-linearly interpolated distance fields.

Our method is based on the point-contact model described by Barbič and James [9]; however, it differs from their method in that our model addresses the shape change of the contact model, which allows for haptic cutting simulation with hard material for mechanical machining and medical simulations. We use the surface voxels of the volume model as opposed to the point shell points tightly covering the surface. The pointerless full octree [30] representation is adopted to handle the topological change of the contact model rather than the Bounded Deformation Tree for elastic deformation. Moreover, we introduce an "on-the-fly" normal vector estimation method using adaptive sampling in the local space of the voxel in contact.

A. Linear Octrees

The main objective of our octree representation is to minimize the number of voxel collision tests against the distance field object. Therefore, efficient memory techniques for the octree representation, such as locational code and condensation, are not considered here. Instead, we implement an octree with additional information for child nodes in order to reduce the tree search time for collision detection and update time along with volume modification. This is inspired by the complete hyper octrees described by Yau and Srihari [31]. Each node contains the minimum and maximum values of its boundary rather than simple index encoding of the relative position from the position of its parent. Every internal (non-leaf) node also has child node status (CNS), a variable used to aggregate the existence of child nodes. The range of CNS is the set $L = \{0, 1, 2\}$, where elements 0, 1, and 2 denote no children, full of children (8 children), and one or more but fewer than 8 children (1 to 7 children), respectively. All valid voxels are stored in the leaf nodes as a compact representation: each 3D indexing variable $p(i, j, k)$ in $\mathbb{Z}^3$ is converted into one 32-bit integer variable, where i, j, and k evenly take 10 bits. In our implementation, the octree is stored in a linear array without parent and child pointers, and all nodes at the same level are contiguous within the array, i.e., it is a pointerless full octree.

B. Lazy Evaluation Scheme

The most prevalent methods for volume modification apply constructive solid geometry operations (e.g., union, difference, and intersection) between volume models. If the volume model is modified, the topology of the octree must be updated recursively to ensure proper and efficient tree search. However, global reconstruction techniques with octrees cannot easily be carried out in one haptic cycle. Therefore, we deactivate empty nodes first, i.e., the CNS becomes 0, and apply a lazy evaluation scheme that delays updating the node until it is required. Whenever an internal node with no child nodes is found, i.e., every CNS of child nodes is 0, the CNS of the internal nodes becomes 0. Note that the lazy evaluation scheme distributes a recursive update of parent nodes over the next several haptic frames.

C. Penalty Force Estimation

Once the colliding voxels have been identified, the following equation is used to calculate the contact penalty force for the i^{th} voxel in contact [9]:

$$\vec{F}_i = -k_i d_i \vec{N}_i \tag{1}$$

where k_i is the penalty force stiffness, d_i is the distance value obtained by the distance field ($d_i < 0$), and $\vec{N}_i$ is the voxel normal in the world coordinate space. Constant k_i can be derived from the mechanical properties of the workpiece such as stiffness, hardness, and strength to support physically-based haptic display and inhomogeneous haptic representation. $\vec{N}_i$ corresponds to the voxel gradient [22] or the inward normal of the point shell point [8]. We must focus on the evaluation of these values in order to achieve a stable haptic interaction with numerical integration. Both k_i and d_i determine the magnitude of the contact force while N_i decides the direction of the contact force. To prevent a sudden change in the magnitude of the contact force, we need to scale k_i down proportionally to the number of colliding voxels, as described in [20, 32]. The trilinear interpolation of the distance field keeps d_i continuous. Our primary concern is to compute $\vec{N}_i$ along with the volume change of the workpiece.

IV. VOXEL CLASSIFICATION

A. Definition

This paper follows the definitions suggested by Cohen et al. [33]. A 3D discrete space $\mathbb{Z}^3$ consists of a set of grid points (sample points) with three integer elements, which is the subset of the 3D Euclidean space $\mathbb{R}^3$. A voxel is a cube enclosing all points in $\mathbb{R}^3$ that are closer to some grid point p than to any other grid point, i.e., the Voronoi neighborhood of p. Each voxel contains a binary value that is either 1 for “foreground” objects or 0 “background”. $N_m(v)$ denotes m-neighbourhood of a voxel v. $N_6(v)$ represents 6 voxels sharing a face with v, $N_{18}(v)$ represents 18 voxels sharing a vertex or an edge with v, and $N_{26}(v)$ represents 26 voxels sharing a vertex, an edge, or a face with v. In other words, $N_6(v)$ is *6-adjacent* to v, $N_{18}(v)$ is *18-adjacent* to v, and $N_{26}(v)$ is *26-adjacent* to v. A *m-path* means a sequence of voxels $p = v_1, v_2, ..., v_n$ of $\mathbb{Z}^3$ such that consecutive pairs (v_i and v_{i+1} with $1 < i \leq n-1$) are m-adjacent. A subset Σ of $\mathbb{Z}^3$ is said to be *m-connected* in Σ if there exists an m-path between every pair of voxels in Σ.

The set of voxels in the neighborhood size (NS) of a voxel v contains all voxels such that the shortest 26-path to v is greater than 0 and lower than or equal to NS. The neighborhood is a cube with side length $2NS + 1$. The number of neighborhoods for v with NS is the same as $(2NS + 1)^3 - 1$. For example, if $NS = 1$ for v, the neighborhood voxels are the same as the ones in $N_{26}(v)$.

The discrete surface S^d is defined by the relaxed adjacency condition [24] as follows: Let S^d be a 26-connected subset of $\mathbb{Z}^3$. For each voxel v of S^d, S^d divides $N_{26}(v)$ into two disjoint components, I^d and O^d, where I^d corresponds to the inside of the surface and O^d corresponds to voxels outside of the surface. v is 6-adjacent to O^d and 26-adjacent to I^d.

B. Voxel Types for Directions of Projection

Each voxel in S_d has hidden faces according to its connectivity, which is particularly related to algorithms for drawing lines or planess in $\mathbb{Z}^3$. Debled-Rennesson and Reveilles [34] introduced algorithms for drawing a digital naive plane in $\mathbb{Z}^3$. In order to avoid computing the hidden face in a discrete plane, they identified three kinds of voxels: corner voxels, ridge voxels, and middle voxels. Three faces are involved in a corner voxel, two faces are involved in a ridge voxel, and only one face is involved in a middle voxel. For normal computation, Tellier and Debled-Rennesson [27] also divided S_d into two cases: non-regular cases and regular cases. Non-regular cases are voxels that have at least two opposite faces adjacent to the edge and corner or side voxels. Regular cases possess 1, 2, or 3 faces.

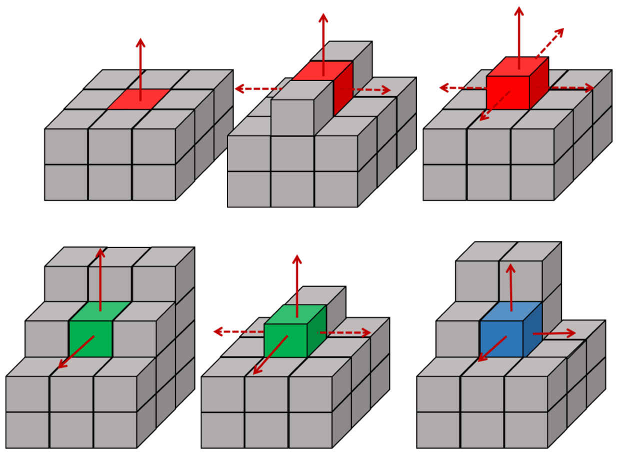
Fig. 1. Three cases of voxels according to the number of potential search directions: uni-directional (red voxels), bi-directional (green), and tri-directional (blue).

The objective of our voxel classification is to restrict the direction leading to the appropriate local distance maps without boundary artifacts. We divide S_d in $\mathbb{Z}^3$ into three cases of voxels according to the number of potential search directions: unidirectional, bi-directional, and tri-directional cases. Note that the number of visible faces of a voxel does not correspond to the number of search directions. The red voxels in Fig. 1 represent the uni-directional cases, the green voxels represent the bi-directional cases, and the blue voxels represent the tri-directional cases. The red solid arrows show the potential search directions. The red dotted arrows represent pairs of opposing directions that are not considered as potential search directions, since boundary artifacts of the distance maps could occur along those directions.

C. Encoding of Voxel Cases

Since the discrete surface S_d is 26-connected and each voxel v of S_d is 6-adjacent to the outside of the surface O_d, it is sufficient to test 6 directions from v, i.e., $-X$, $+X$, $-Y$, $+Y$, $-Z$, and $+Z$, in searching the outward directions of v. The 6 directions are the same as the six face normals of

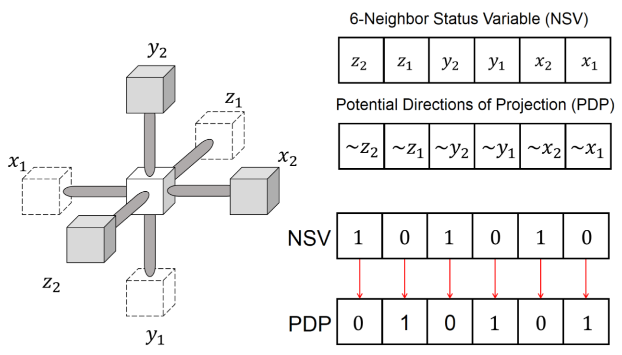
6-Neighbor Status Variable (NSV)
z_2 z_1 y_2 y_1 x_2 x_1
Potential Directions of Projection (PDP)
$\sim z_2$ $\sim z_1$ $\sim y_2$ $\sim y_1$ $\sim x_2$ $\sim x_1$
NSV 1 0 1 0 1 0
PDP 0 1 0 1 0 1
y_2 z_1 x_1 x_2 z_2 y_1

Fig. 2. Denition of NSV and PDP.

the axis-aligned bounding box of a volume model.

We propose a case table of 6-adjacent neighborhoods to determine the outward directions of v. Neighbor status variables (NSV) encode 6-adjacent cases ($2^6 = 64$) as a bit array; potential directions of projection (PDP) are complementary of NSV. Each bit of NSV indicates whether there exists a corresponding neighborhood (see Fig. 2). On the other hand, each bit of PDP indicates whether the corresponding direction from v is outward. If one of the bits in PDP is true, the local distance map (LDM) can be generated along the corresponding direction.

D. Voxel Type Reduction

Some PDP could include pairs of opposing directions. Fig. 3 shows a case containing both the positive z direction and the negative z direction at voxel v. Note that both z_1 and z_2 in NSV have the same value, i.e., one. In this case, we negate pairs of 2 bits corresponding to opposing directions in order to prevent redundant computation and boundary artifacts before building an LDM. There are 38 PDP involved in pairs of ill-posed directions (number of single pairs of ill-posed directions, 27; of two pairs, 9; and of three pairs, 2). This reduces all possible combinations of PDP from 64 cases to 26. By enumerating 64 cases, we build a table that contains 64 elements with a list of the outward directions without pairs of opposing directions. The key of the table is 6-bit variables of PDP. Using this table, we can find outward directions at p in constant time and remove pairs of opposing directions.

V. COMPUTING A NORMAL VECTOR USING A LOCAL DISTANCE MAP

A. Adaptive Local Distance Maps

In the previous section, we defined the three kinds of voxel types based on the outward direction and found a total of 26 different cases for PDP. Each of the 26 PDP values corresponded to each face normal of the rhombicuboctahedron in Fig. 4 where the normal vectors of the red, green, and blue faces (also see Fig. 1) correspond to the 6, 12, and 8 PDP values for the unidirectional case, bidirectional case and tri-diractional case, respectively. Now that the outward directions are determined at a voxel v, we can discuss a way to build an LDM along a specific direction.

Fig. 4 is an example of a 3×3 LDM construction using a set of voxels. The PDP of the center voxel corresponds to $+Y$. Let π be a plane in $\mathbb{Z}^3$ with a center point of v and a direction of $+Y$. The left LDM contains indices of elements; each element of the right LDM contains signed minimum distances from π to most outer voxels p_i ($1 \le i \le 8$) along the Y axis. p_i is a subset of S_d (blue voxels in Fig. 4). n penetrates 9 voxels: one is v and the others are the eight adjacent voxels of v, i.e. $N_8(v)$. Let q_i ($1 \le i \le 8$) be the 8-adjacent voxels of v. Using a simple voxel readout scheme, we traced each p_i from q_i. If p_i is a member of I_d or S_d, we search along the positive direction (the red dotted arrow in Fig. 4) for q_i. If it is not found, i.e., if p_i is a member of O_d, we trace q_i along the negative direction (the blue dotted arrow in Fig. 4).

The range of tracing is limited by the maximum distance value α ($\alpha > 0$). If α is 1, the range of the voxel readout corresponds to $N_{26}(v)$. Therefore, the minimum and maximum values in the LDM are α and $-\alpha$, respectively. Although the optimal α value depends on

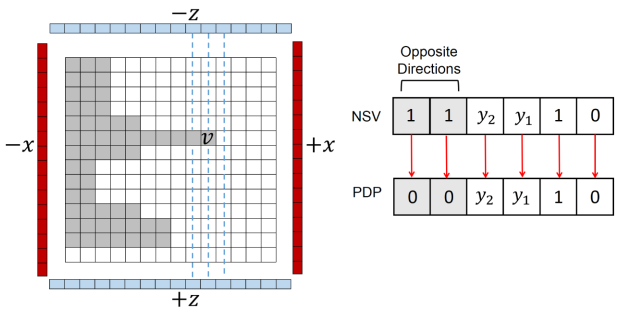

Fig. 3. Negate bits related to mutually opposite directions.

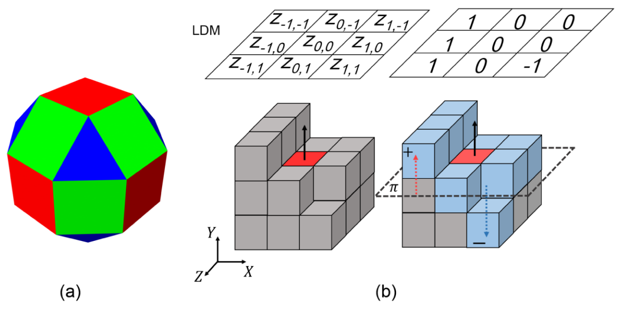
LDM
$z_{-1,-1}$ $z_{0,-1}$ $z_{1,-1}$
$z_{-1,0}$ $z_{0,0}$ $z_{1,0}$
$z_{-1,1}$ $z_{0,1}$ $z_{1,1}$
1 0 0
1 0 0
1 0 -1
+
π
−
Y
Z
X
(a)
(b)

Fig. 4. (a) Rhombicuboctahedron and (b) a 3×3 LDM construction.

the connectivity of the model, we suggest that α is between 4 and 5 voxel units, based on the findings of previous studies. Sramek and Kaufman [35] found that the optimal width of the filter is 2×1.7 voxel units for gradient estimation by central differences [35]. Flin et al. [28] also showed that $\alpha = 5$ voxel unit provided acceptable results.

In order to prevent over-smoothed results of sharp features, we build the LDM adaptively; early rejections for building the LDM are allowed according to the distance value of LDM z_i. First, we start at $NS = 1$, i.e., the size of the LDM is 3×3. If any $| z_i |$ is greater than α, a normal is computed by the central differences with $N_{26}(v)$. Otherwise, NS is increased by one, i.e., the size of the LDM is expanded to 5×5, and then the distance values of additional z_i are computed. These operations are repeated until either the criteria of early rejection are met (any $| z_i |$ is greater than a) or NS is smaller than α.

B. Computing Normal from Local Distance Maps

The summary of the normal estimation is described in Algorithm 1. Each pixel of the LDM contains the signed minimum distance from the position of a pixel to the most outer voxel along the PDP. The gradient $\vec{G}$ of the LDM corresponds to the vector with a direction from the minimum distance to the maximum one, and the magnitude of $\vec{G}$ is the corresponding change in distance at the center voxel. $\vec{G}$ is computed by a weighted summation with a neighborhood with NS. Local averaging of the operator reduces the occurrence of discretization artifacts caused by voxelization. We selected the reciprocal of the Euclidean distance to compute the weighting factors because square root computation can be avoided as follows:

$$\vec{G} = \vec{G}_u + \vec{G}_v \tag{2}$$

$$\vec{G}_u = \frac{1}{s}\sum_{j=-NS}^{NS}\sum_{i=-NS}^{NS} z_{ij}\frac{i}{i^2+j^2} \quad \text{without } i=0 \wedge j=0 \tag{3}$$

$$\vec{G}_v = \frac{1}{s}\sum_{j=-NS}^{NS}\sum_{i=-NS}^{NS} z_{ij}\frac{j}{i^2+j^2} \quad \text{without } i=0 \wedge j=0 \tag{4}$$

where s indicates the number of antipodal pairs in the LDM that is $((2NS + 1)^2 – 1)/2$. Using the unit normal vector of π, $\vec{P}$, the gradient of the LDM, $\vec{N}'$, can be computed by

$$\vec{N}' = \vec{P} - \vec{G} \tag{5}$$

Finally, normalizing $\vec{N}'$ yields the contact normal of the voxel.

Algorithm 1 Algorithm for normal estimation using a LDM

```
Input: $V, I(i,j,k) \in \mathbb{Z}^3$, $\alpha \geq 1$ and $NS_{max} \geq 1$
    V: binary volume data
    I: a voxel index
    $\alpha$: a maximum distance of LDM
    $NS_{max}$: a maximum neighborhood size of LDM
Output: $\vec{N}(x,y,z) \in \mathbb{R}^3$
    $\vec{N}$: a normal vector
 1: Compute a Neighbor Status Variables (NSV) based on 6-adjacent neighborhoods (Section IV-C)
 2: Set Potential Directions of Projection (PDP) = negation of NSV
 3: Decide a direction of projection ($\vec{P}$) of $LDM$ using the $PDP$ (Section IV-D)
    Define a LDM
 4: Set $LDM(i,j) \in \mathbb{Z}^2$ where $-NS_{max} \leq i,j \leq NS_{max}$
    LOOP process: compute a distance value at symmetric pairs of LDM.
 5: $NS = 1$
 6: while $NS \leq NS_{max}$ do
 7:   for $k = -NS$ to $NS$ do
 8:     Compute a distance value $z(k,k)$ of $LDM(k,k)$ (Section V-A)
 9:     if $z(k,k) < -\alpha$ or $z(k,k) > \alpha$ then
10:       go to line 19
11:     end if
12:     Compute a distance value $z(-k,-k)$ of $LDM(-k,-k)$
13:     if $z(-k,-k) < -\alpha$ or $z(-k,-k) > \alpha$ then
14:       go to line 19
15:     end if
16:   end for
17:   $NS = NS + 1$
18: end while
19: $NS = NS - 1$
20: if $NS < 1$ then
21:   $\vec{N} = \vec{P}$
22: else
23:   Compute a gradient $\vec{G}$ of LDM with neighbors in $NS$ (Section V-B)
24:   $\vec{N}' = \vec{P} - \vec{G}$
25:   $\vec{N} = \vec{N}'/\|\vec{N}'\|$
26: end if=0
```

VI. EXPERIMENTS

All experiments were conducted on a computer with an Intel Core i7-4790K CPU 4.0 GHz processor with 4 physical cores and 8 logical cores, an NVIDIA GeForce GTX 960 graphics card, and 16 GB of memory. In order to take advantage of the multi-core processors, the codes for collision detection and normal estimation were parallelized using Microsoft Parallel Patterns Library. Floating-point operations were performed on 32-bit variables.

For the workpiece with shape changes, we selected a human mandibular molar that was scanned using a SkyScan 1076 Micro-CT system with a voxel size of 34.8 µm. We then constructed a 512×512×512 resolution volume with binary values segmented from the Micro-CT tooth data. The volume was recursively subdivided into 8 nodes in a linear octree with a maximum height of 6. Therefore, each leaf node contained an 8×8×8 resolution volume. In order to probe or cut the volumetric tooth, a polygonal mesh model of the burr bit was created. A capsule covering the mesh model was computed for a broad-phase collision detection that extracts leaf nodes of the octree that potentially collided with the burr bit. The signed distance field for the burr bit was constructed and stored in a 128×512×128 resolution volume. Each voxel in the octree nodes extracted from the broad-phase collision detection was subject to exact collision detection with the signed distance field of the burr bit.

We conducted a tooth probing and cutting simulation using those representations. The trajectory of the haptic interface was generated for a simulation object, a polygon model of the burr bit, to touch a valley on the surface of the tooth. The movement of the haptic device was simulated at a constant velocity in the trajectory. The contact normal of the tooth was estimated using the LDM as described in Section V. The maximum distance value of the LDM α was 5. The *NS* of the LDM is 6, i.e., the size of the LDM was 13×13.

In order to assess the accuracy of the feedback force and torque, our results were compared to those computed in an offline simulation using an exact normal computation. The offline simulation first applied the exact Euclidean distance transformation [36] within the 13×13×13 local volume at every voxel in contact, and the gradient vector field was generated using a 3D Prewitt mask [37] with 1 *NS*. Adaptive distance gradient filtering with angle and symmetry criteria was used to compute the normal vectors [28]. Table 1 presents the summary statistics for contact configuration and the root mean squared error (RMSE) of the feedback force and torque. The next section provides an analysis of two experiments: probing and cutting.

Table 1. Statistics for contact configuration and feedback force and torque errors

Variables	Probing	Cutting
Average number of leaf nodes in contact	48	83
Average number of voxels in contact	10	351
Average number of voxels traversed	9333	17577
Max number of voxels in contact	69	1180
Mean haptic frame rate (µs)	92	468
RMSE if force magnitude (N)	0.13	0.15
RMSE of torque magnitude (mN·m)	5.01	6.33

A. Analysis of Probing Tooth

The first experiment evaluated the efficiency of our collision model and the accuracy of our normal estimation method under a small number of contacts. The movement of the haptic interface was simulated at a constant velocity along the direction of a positive X-axis. As shown in Fig. 5, the burr bit, i.e., the simulation object, moved along the tooth sulcus, which is a valley on the occlusal surface. As shown in Table 1, the mean number of leaf nodes in contact was 48. This suggests that the broad phase collision detection algorithm, performed between the octree and the capsule, identified 48 leaf nodes in contact on average. As shown in Table 1, the mean number of voxels traversed for feedback force computation was 9333; in other words, the mean number of distance queries for exact collision detection, performed between a tooth voxel and a distance field of the burr bit, was 9333. The results obtained from the real-time simulation with our method are summarized in the graphs in Fig. 5. From the feedback force and torque profiles, we can see that our normal estimation method presents results quite similar to those computed by the offline

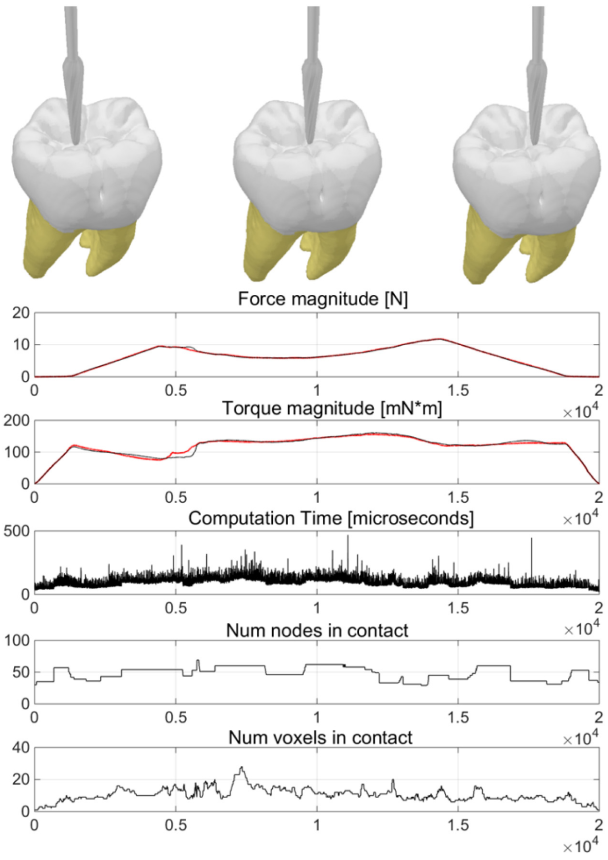

Fig. 5. Simulation results on tooth probing the tooth. Red lines indicate results from the offline simulation results. The common horizontal axis on the graph shows the haptic frames.

simulation (red line). Moreover, the mean time of haptic frames (92 μs) was about 20 times less than that of the offline simulation (1800 μs).

B. Analysis of Tooth Cutting

In the second experiment, the position of the haptic interface was followed by the reverse trajectory of the first experiment with the same speed. However, the tooth voxel modification was allowed. Fig. 6 shows the results of tooth cutting according to positional changes of the haptic interface. The volume removal rate ΔV at every voxel in contact was computed using the principle of conservation of energy [38, 39] as follows:

$$\Delta V \propto \frac{\vec{F}_t \vec{v}_r}{K} \tag{6}$$

where $\vec{F}_t$, $\vec{v}_r$, and K indicate the tangential force, relative velocity of the cutting edge, and fracture toughness, respectively. According to the grinding theory [40], the force exerted by the tooth against the burr can be

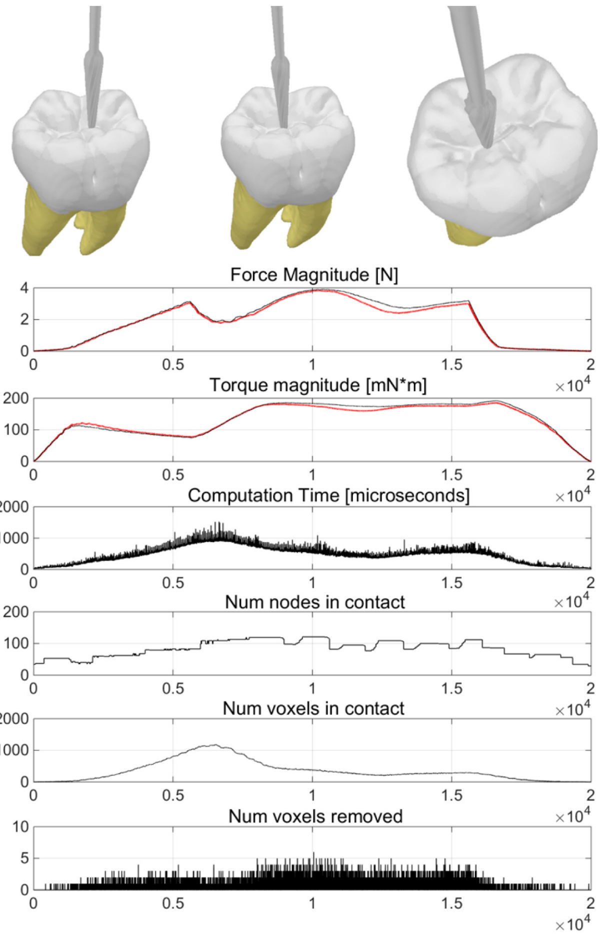

Fig. 6. Simulation results on tooth cutting.

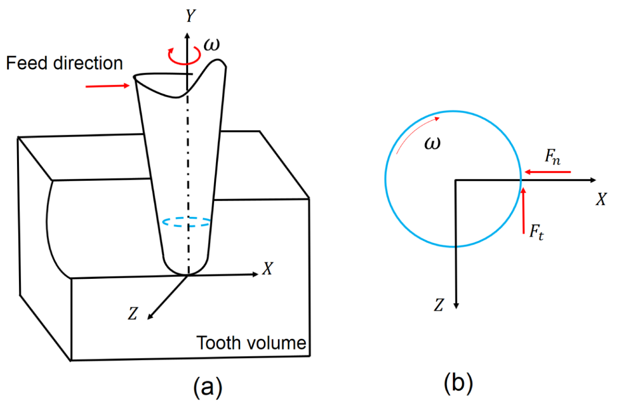

Fig. 7. (a) Force analysis for volume removal ratio. (b) Two force components: a tangential force and a normal force.

separated into a tangential component $\vec{F}_n$ and a normal component $\vec{F}_t$, as shown in Fig. 7. Since the magnitude of the two force components had a linear correlation, the tangential force was derived by

$$F_t = -\beta |F_n| \frac{v_r}{|v_r|} \tag{7}$$

where β indicates a constant related to burr properties such as rack angle and sharpness. James et al. [41] provided an in-depth analysis of the β. $\vec{v}_r$ is computed by the cross product angular velocity of the burr bit ω with a local vector from the rotational axis to the voxel in contact. ω was set for 7000 RPM. Fracture toughness is a material constant that indicates the resistance of a material to the propagation of a crack. Arola et al. [42] reported on the fracture toughness of hard tissue. Since $\vec{F}_n$ is linearly related to the depth of the cut [41], $\vec{F}_t$ can be substituted with the contact force at each voxel in contact.

A cutting simulation was performed in a haptic rendering loop, i.e., a multi-rate rendering approach [43] was not used. A ploughing and sliding simulation of the burr during cutting was not considered. The total number of voxels removed was 10103 for 20000 haptic frames. The number of voxels in contact reached its peak (1180) at the 6554th haptic frame. As a result, the haptic frame rate around that frame is decreased, sometimes to less than 1 msec. The computation time gradually decreased as the number of voxels in contact grew. This is because each voxel in contact required more computation time than in the first experiment; computing the volume removal ratio and octree updates were necessary. Although our method presents a force profile quite similar to the offline simulation results, the maximum difference between force magnitude was 0.4686 N at the 12546th frame. However, the mean time of the haptic frame (468 μs) was significantly lower than that of the offline simulation (40670 μs).

C. Accuracy and Time Complexity

An experiment was also performed to verify the precision and efficiency of our normal estimation method. A sphere was used for a quantitative comparison of the different methods because it can be simply represented as an implicit form. Sramek and Kaufman [35] also mentioned the adequacy of the sphere for comparing the performances of different algorithms due to its simple representation and constant curvature. The sphere was voxelized into a resolution volume of 256×256×256. Voxels in the discrete surface S_d are only allowed to compute normal vectors. S_d contains 108760 voxels. The accuracy and time complexity of the generated normal vectors were evaluated by comparing them with those computed by the implicit equation of the sphere. We also measured the results from the different algorithms: weighted summation (WS), Principle Component Analysis (PCA), and moving least squares (MLS). The reciprocal of the Euclidean distance was used for the weighting factor of WS [24]. PCA and MLS was implemented using the Point Cloud Library [44].

As shown in the time graph in Fig. 8, the time complexity of our method is significantly different from the others. Our method involves a time complexity of $O(n^2)$ with neighborhood size n while the other methods run in $O(n^3)$. Each pixel of the LDM requires a voxel readout of less than or equal to α where α indicates the maximum distance value. In fact, most pixels of the LDM require a voxel readout smaller than α because each voxel readout starts from a plane π generated from the center voxel. If the distance of the surface voxel from π has a uniform distribution in the LDM, we can assume that the number of the voxel readout is half of α.

The mean absolute error (MAE) graph in Fig. 8 shows that every method dropped sharply up to 3 *NS* and remained steady over 4 *NS*. The MAE of our method (LDM) reached near that of PCA or MLS when the size of the LDM was equal to or more than 7×7. The MAE of PCA, MLS, and LDM reached a low point near zero; however, WS converged to 2 degrees. This suggests that simple linear computation has a limitation in compensating errors from binary volume data. Fig. 9 shows detailed results such as error distribution, computation time, and precision. WS has the fastest performance until *NS* = 4, although its precision is the worst. PCA and MLS do not differ much in terms of precision; however, MLS is 2.7 times slower than PCA when *NS* = 6. The larger *NS*, the closer our approach (LDM) is to the precision of MLS. We can see that our method is about 3 times faster than PCA, and about 9 times faster than MLS when *NS* = 6.

To analyze the ability to sharp feature preservation, we measured the results from the different algorithms with a cube volume model, as shown in Fig. 10. WS, PCA, and MLS showed more errors as the NS increased. The errors are mainly distributed throughout the edges of the cube. However, our method showed a constant error as the NS increased.

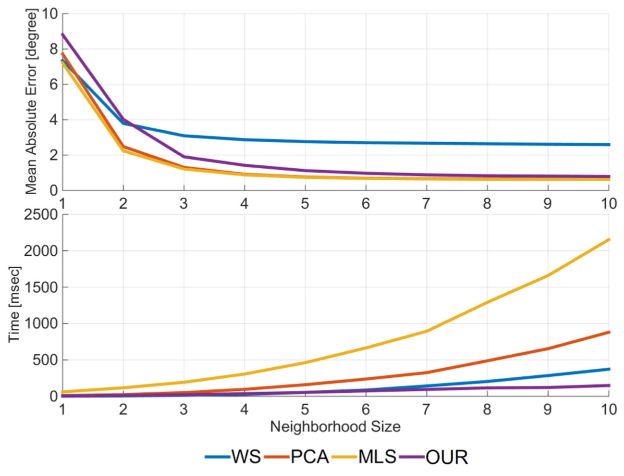

Fig. 8. (A) Mean absolute error and (B) time complexity of different methods for normal estimation.

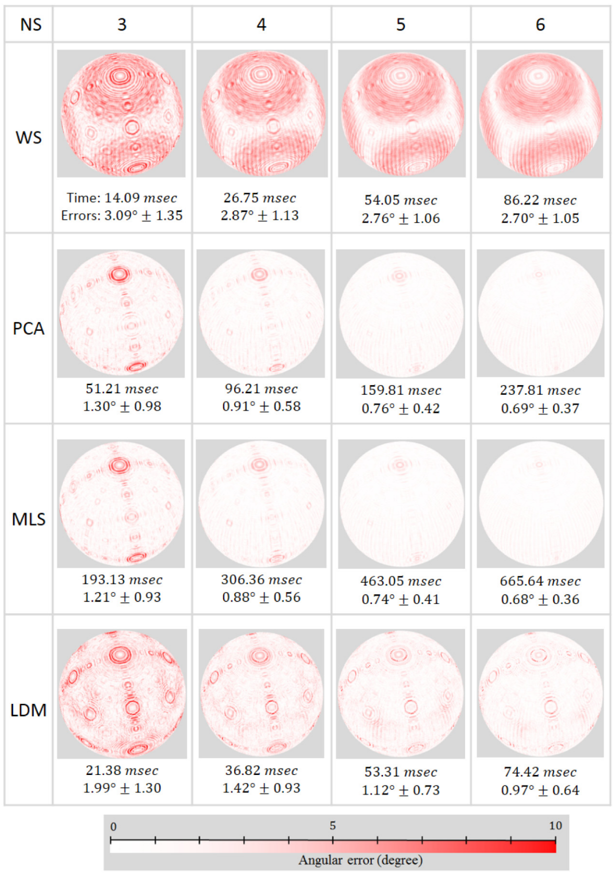

Fig. 9. Error distribution of a sphere model with dierent methods for normal estimation. The number of surface voxels is 108760. NS indicates the neighborhood size.

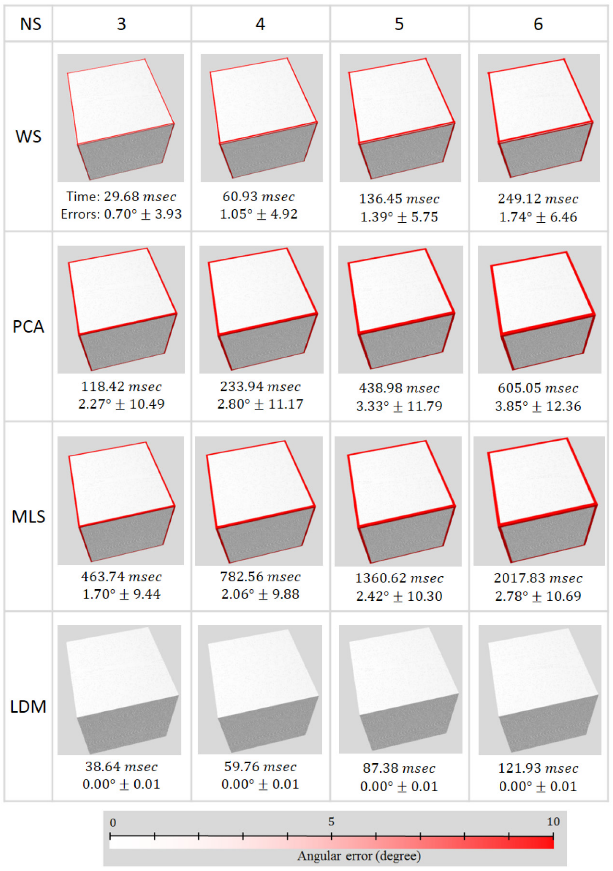

Fig. 10. Error distribution of a cube model with dierent methods for normal estimation. The number of surface voxels is 302425. NS indicates the neighborhood size.

VII. CONCLUSION

This study has shown that the voxel-based haptic rendering algorithm can be extended for use in environments involving topological changes. Our volumetric collision model adopts a binary volume model for an object with shape changes, as well as a signed distance field for a tool object. Our on-the-fly contact normal computation with a local distance map allows for the presence of shape changes in the volume model at time-critical haptic rendering. The present method involves a time complexity of $O(n^2)$ with neighborhood size n in volumetric data. Our approach significantly reduces the voxel readout, which allows for the model to handle more than ten thousand voxels at haptic rates. The results of a tooth cutting simulation confirm that our method can generate accurate and stable haptic displays under progressive shape changes of the volume model.

One important issue that merits further research is spatial coherence. Since our normal estimation is locally performed, the distance calculations of some of the pixels of the distance map may be redundant. Another possible area of investigation for future research is haptic-enabled fluid simulation. The past decade has seen the rapid development of real-time fluid dynamics. Our collision model may be applicable to provide haptic feedback for interactive applications with fluid dynamics.

REFERENCES

1. Y. Kim, L. Kim, D. Lee, S. Shin, H. Cho, F. Roy, and S. Park, "Deformable mesh simulation for virtual laparoscopic cholecystectomy training," *The Visual Computer*, vol. 31, no. 4, pp. 485-495, 2015.
2. C. J. Paulus, L. Untereiner, H. Courtecuisse, S. Cotin, and D. Cazier, "Virtual cutting of deformable objects based on efficient topological operations," *The Visual Computer*, vol. 31, no. 6-8, pp. 831-841, 2015.
3. M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross, "Interaction of fluids with deformable solids," *Computer Animation and Virtual Worlds*, vol. 15, no. 3-4, pp. 159-171, 2004.
4. M. Arbabtafti, M. Moghaddam, A. Nahvi, M. Mahvash, B. Richardson, and B. Shirinzadeh, "Physics-based haptic simulation of bone machining," *IEEE Transactions on Haptics*, vol. 4, no. 1, pp. 39-50, 2010.
5. K. Kim and J. Park, "Virtual bone drilling for dental implant surgery training," in *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, Kyoto, Japan, 2009, pp. 91-94.
6. C. H. Ho, C. Basdogan, and M. A. Srinivasan, "Efficient point-based rendering techniques for haptic display of virtual objects," *Presence*, vol. 8, no. 5, pp. 477-491, 1999.
7. M. A. Otaduy and M. C. Lin, "Sensation preserving simplification for haptic rendering," in *Proceedings of the ACM SIGGRAPH 2005 Courses*, Los Angeles, CA, 2005.
8. W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in *Proceedings of the ACM SIGGRAPH 2005 Courses*, Los Angeles, CA, 2005.
9. J. Barbic and D. L. James, "Six-DoF haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Transactions on Haptics*, vol. 1, no. 1, pp. 39-52, 2008.
10. G. Cirio, M. Marchal, S. Hillaire, and A. Lecuyer, "Six degrees-of-freedom haptic interaction with fluids, "*IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 11, pp. 1714-1727, 2010.
11. C. Osnes and A. J. Keeling, "Developing haptic caries simulation for dental education," *Journal of Surgical Simulation*, vol. 4, pp. 29-34, 2017.
12. X. Chen and J. Hu, "A review of haptic simulator for oral and maxillofacial surgery based on virtual reality," *Expert Review of Medical Devices*, vol. 15, no. 6, pp. 435-444, 2018.
13. Y. Yan, Q. Li, Q. Wang, and Y. Peng, "Real-time bone sawing interaction in orthopedic surgical simulation based on the volumetric object," *Journal of Visualization*, vol. 21, no. 2, pp. 239-252, 2018.

14. C. B. Zilles and J. K. Salisbury, "A constraint-based god-object method for haptic display," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems: Human Robot Interaction and Cooperative Robots*, Pittsburgh, PA, 1995, pp. 146-151.
15. R. S. Avila and L. M. Sobierajski, "A haptic interaction method for volume visualization," in *Proceedings of 7th Annual IEEE Visualization*, San Francisco, CA, 1996, pp. 197-204.
16. S. Gibson, J. Samosky, A. Mor, C. Fyock, E. Grimson, T. Kanade, et al., "Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback," in *CVRMed-MRCAS'97*. Heidelberg: Springer, 1997, pp. 367-378.
17. K. L. Palmerius, M. Cooper, and A. Ynnerman, "Haptic rendering of dynamic volumetric data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 263-276, 2008.
18. W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, 1999, pp. 401-408.
19. M. Renz, C. Preusche, M. Potke, H. P. Kriegel, and G. Hirzinger, "Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm," in *Proceedings of Eurohaptics*, Birmingham, UK, 2001.
20. M. Wan and W. A. McNeely, "Quasi-static approximation for 6 degrees-of-freedom haptic rendering," in *Proceedings of the 14th IEEE Visualization*, Seattle, WA, 2003, pp. 257-262.
21. L. S. Chen, G. T. Herman, R. A. Reynolds, and J. K. Udupa, "Surface shading in the Cuberille environment," *IEEE Computer Graphics and Applications*, vol. 5, no. 12, pp. 33-43, 1985.
22. U. Tiede, K. H. Hohne, M. Bomans, A. Pommert, M. Riemer, and G. Wiebecke, "Investigation of medical 3D-rendering algorithms," *IEEE Computer Graphics and Applications*, vol. 10, no. 2, pp. 41-53, 1990.
23. D. Gordon and R. A. Reynolds, "Image space shading of 3-dimensional objects," *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 3, pp. 361-376, 1985.
24. G. Thürmer and C. A. Wüthrich, "Normal computation for discrete surfaces in 3D space," *Computer Graphics Forum*, vol. 16, no. 3, pp. C15-C26, 1997.
25. G. Thürmer and C. A. Wüthrich, "Varying neighbourhood parameters for the computation of normals on surfaces in discrete space," in *Proceedings of Computer Graphics International (Cat. No. 98EX149)*, Hannover, Germany, 1998, pp. 616-625.
26. G. Thürmer, "Smoothing normal vectors on discrete surfaces while preserving slope discontinuities," *Computer Graphics Forum*, vol. 20, no. 2, pp. 103-114, 2001.
27. P. Tellier and I. Debled-Rennesson, "3D discrete normal vectors," in *Discrete Geometry for Computer Imagery.* Heidelberg: Springer, 1999, pp. 447-458.
28. F. Flin, J. B. Brzoska, D. Coeurjolly, R. A. Pieritz, B. Lesaffre, C. Coleou, O. Teytaud, and J. F. Delesse, "Adaptive estimation of normals and surface area for discrete 3-D objects: application to snow binary data from X-ray tomography," *IEEE Transactions on Image Processing*, vol. 14, no. 5, pp. 585-596, 2005.
29. S. Fourey and R. Malgouyres, "Normals estimation for digital surfaces based on convolutions," *Computers & Graphics*, vol. 33, no. 1, pp. 2-10, 2009.
30. I. Gargantini, "Linear octtrees for fast processing of three-dimensional objects," *Computer Graphics and Image Processing*, vol. 20, no. 4, pp. 365-374, 1982.
31. M. M. Yau and S. N. Srihari, "A hierarchical data structure for multidimensional digital images," *Communications of the ACM*, vol. 26, no. 7, pp. 504-515, 1983.
32. J. Barbic, "Real-time reduced large-deformation models and distributed contact for computer graphics and haptics," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 2007.
33. D. Cohen-Or and A. Kaufman, "Fundamentals of surface voxelization," *Graphical Models and Image Processing*, vol. 57, no. 6, pp. 453-461, 1995.
34. I. Debled-Rennesson and J. P. Reveilles, "New approach to digital planes," in *Vision Geometry III* (Proceedings of SPIE 2356). Bellingham, WA: International Society for Optics and Photonics, 1995, pp. 12-22.
35. M. Sramek and A. Kaufman, "Object voxelization by filtering," in *Proceedings of IEEE Symposium on Volume Visualization (Cat. No. 989EX300)*, Research Triangle Park, NC, 1998, pp. 111-118.
36. Y. H. Lee, S. J. Horng, T. W. Kao, and Y. J. Chen, "Parallel computation of the Euclidean distance transform on the mesh of trees and the hypercube computer," *Computer Vision and Image Understanding*, vol. 68, no. 1, pp. 109-119, 1997.
37. Y. J. Zhang, "Quantitative study of 3D gradient operators," *Image and Vision Computing*, vol. 11, no. 10, pp. 611-622, 1993.
38. M. Mahvash and V. Hayward, "Haptic rendering of cutting: a fracture mechanics approach," *Haptics-e*, vol. 2, no. 3, pp. 1-12, 2001.
39. M. Mahvash, L. M. Voo, D. Kim, K. Jeung, J. Wainer, and A. M. Okamura, "Modeling the forces of cutting with scissors," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 3, pp. 848-856, 2008.
40. S. Malkin and C. Guo, *Grinding Technology: Theory and Application of Machining with Abrasives*. New York, NY: Industrial Press Inc., 2008.
41. T. P. James, J. J. Pearlman, and A. Saigal, "Predictive force model for haptic feedback in bone sawing," *Medical Engineering & Physics*, vol. 35, no. 11, pp. 1638-1644, 2013.
42. D. Arola, D. Bajaj, J. Ivancik, H. Majd, and D. Zhang, "Fatigue of biomaterials: hard tissues," *International Journal of Fatigue*, vol. 32, no. 9, pp. 1400-1412, 2010.
43. T. C. Knott and T. W. Kuhlen, "Accurate and adaptive contact modeling for multi-rate multi-point haptic rendering of static and deformable environments," *Computers & Graphics*, vol. 57, pp. 68-80, 2016.
44. R. B. Rusu and S. Cousins, "Point cloud library (pcl)," in *Proceedings of 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 1-4.

Kimin Kim

Kimin Kim received the Ph.D. degree from the School of Computing at the Korea Advanced Institute of Science and Technology (KAIST), in 2016. From 2016 to 2018, he was a postdoctoral researcher in the KAIST Computer Graphics and Visualization Research Laboratory. Since 2018, he has been working at Samsung Research Korea. His research interests include VR/AR, haptics, and real-time simulations.

Jinah Park

Jinah Park is currently an Associate Professor in the School of Computing at the Korea Advanced Institute of Science and Technology (KAIST), where she leads the Computer Graphics and Visualization Research Laboratory. She received her B.S.E. degree in Electrical Engineering from Columbia University in New York, NY, and M.S.E. and Ph.D. degrees in Computer and Information Science from the University of Pennsylvania, Philadelphia, PA, USA. Her research interests include medical-image data analysis and 3D interactive visualization including computer haptics.