NRC Publications Archive Archives des publications du CNRC

Haptic collision handling for simulation of transnasal surgery

Neubauer, André; Brooks, Rupert; Brouwer, Iman; Debergue, Patricia;
Laroche, Denis

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version. /
La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version
acceptée du manuscrit ou la version de l'éditeur.

For the publisher's version, please access the DOI link below. / Pour consulter la version de l'éditeur, utilisez le lien
DOI ci-dessous.

Publisher's version / Version de l'éditeur:

<https://doi.org/10.1002/cav.1489>

Computer Animation and Virtual Worlds, 24, 2, pp. 127-141, 2012-10-23

NRC Publications Record / Notice d'Archives des publications de CNRC:

<https://nrc-publications.canada.ca/eng/view/object/?id=e466f389-50b0-4629-9e54-2c80164317d1>

<https://publications-cnrc.canada.ca/fra/voir/objet/?id=e466f389-50b0-4629-9e54-2c80164317d1>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at

<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site

<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at

PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the
first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la
première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez
pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

National Research
Council Canada

Conseil national de
recherches Canada

Canada

Haptic Collision Handling for Simulation of Transnasal Surgery

Abstract—Simulation of endoscopic navigation in the narrow nasal cavity poses important challenges to the computation of adequate and near-realistic collision response and haptic feedback, since extensive multi-directional contact and massive tissue deformation are inevitable. We present a virtual coupling algorithm that provides stable collision response and correct and smooth haptic interaction in all phases of the simulation. The surface of the virtual patient anatomy is modeled as a point shell, the endoscope as a cylinder. In each iteration, continuous collision detection allows rolling back the instrument trajectory to the first collision. Then a relaxation process locally optimizes instrument position and orientation. The contact solver communicates with the FEM-based deformation engine by transmitting suggestions of where colliding vertices should be moved. This paper describes the algorithm and presents results for both a simple test case and a virtual transnasal surgery scenario.

Index Terms—surgical simulation, endoscopy, collision handling, virtual coupling.

I. INTRODUCTION

DUE to its minimal invasiveness, endoscopic transnasal surgery is one of the techniques of choice for the removal of tumors of the pituitary gland. In comparison to more traditional procedures, the endoscopic technique confronts the surgeon with additional difficulties due to the non-trivial endoscopic navigation, the indirect view, the optic distortion and the lack of a three-dimensional image. Surgical simulators are gaining more and more importance as training tools since they allow novices to repeatedly practice crucial parts of a procedure without having to fear catastrophic consequences of mistakes. In recent years a lot of effort has been put into the development of simulators for transnasal surgery [1]–[4].

The computation of correct tissue-tool interaction and haptics is an essential [5] but challenging component of transnasal endoscopic simulation. Endonasal anatomy is complex and the passages are narrow. During the operation, both the tip and the shaft of the surgical instrument are almost constantly in multi-directional contact with the patient anatomy (see Figure 1). To allow insertion of the cylinder-shaped rigid endoscope, structures have to be deformed and pressure has to be maintained. Furthermore, cavities in the skull are often separated by very thin bony walls, so the contact handling model must robustly deal with fine structures.

Collision handling for simulated rigid endoscopes received some attention in the last decade. In 1999, Bockholt et al. [1] used constraint-based collision response in a nasal endoscopy simulator. However, their user controlled instrument did not provide haptic feedback. Neubauer et al. [6] developed a simulator for transnasal surgery in 2005. It provided haptic force feedback by pushing penetrating parts of the endoscope towards their points of entry into the tissue. Explicit tissue deformation was not computed, limiting the realism of the

Fig. 1. The endoscope (green) inside the nasal cavity during simulation of transnasal surgery, viewed from the side (small image) and from above with some interior structures visible (large image). The anatomy is complex and challenging for computation of tissue-tool interaction.

endoscope trajectory. In 2009, Çakmak et al. [7] presented an endoscopic ventriculostomy simulator including tissue deformation. Accommodating the requirements of a training tool for this type of surgery, haptic feedback was limited to certain deformable organ structures and disabled for the static head model. The sinus surgery simulator introduced by Parikh et al. [3] in 2009 provided haptic feedback for the tip of the endoscope but did not simulate tissue deformation. In 2010, Perez-Gutiérrez et al. [4] demonstrated the use of a simplified mechanical model in an endonasal surgery simulator. The endoscope was represented by a lever fixed at a point inside the nostril and providing 4 degrees of freedom of interaction. The paranasal sinus surgery simulator presented by Tolsdorff et al. [8] in 2010 models patient anatomy in raw data. The authors did not indicate simulation of tissue deformation.

Collision handling consists of collision detection and collision response, i.e., the computation of post-collision mechanics. Existing collision handling algorithms can be classified to be either *penalty-based* or *constraint-based*.

Penalty-based methods allow interpenetrations of objects, but apply forces proportional to the penetration depth to separate them [9], [10].

Constraint-based approaches prevent penetration of the instrument into the tissue. They achieve this by either anticipating and preventing collisions [11] or applying post-collision measures, e.g., instant deformations moving triangles and vertices out of the space occupied by the user-controlled instrument [12]–[15]. A negative consequence of using the anticipation strategy is that somewhat counterintuitive haptic forces are exerted *before* a real collision occurs. Techniques employing post-collision measures often separate interpenetrating objects

by applying tissue deformations that lack a sound mechanical foundation and might therefore be unrealistic.

The *virtual coupling/god-object* algorithm combines the advantages of penalty-based and constraint-based approaches. The term *virtual coupling* was introduced by Colgate et al. [16] and defined as the connection between the haptic device position and the virtual reality tool via a stiffness and damping. At the same time, Zilles and Salisbury [17] proposed to use the term *god-object* to define a virtual location of the haptic interface which is attached to the actual haptic position via a spring force, but subject to non-penetration constraints. They also presented an algorithm to calculate the optimal position of a point-shaped god-object by using Lagrange multipliers. McNeely et al. [18] used a six degrees-of-freedom spring-damper system to manage god-objects of arbitrarily complex shape contacting a rigid object whose surface is defined by a point shell. This approach was extended to haptic rendering of deformable objects by Barbič and James [19] who also presented a way to move the position and orientation of the god-object in direction of declining contact force and torque.

Otaduy and Gross [20] presented an algorithm that efficiently implements implicit integration of the rigid body dynamics of a tool described by its mass distribution.

Saupin et al. [21] described a technique that approximates a solution of the linear complementary problem of finding the exact contact forces needed to zero out the interpenetration. Since this does not always prevent penetrations, a god-object algorithm is used on top.

This paper introduces a new algorithm providing realistic computation of tool-tissue interaction applicable to a wide range of haptic interactions with deformable or static objects, including the simulation of transnasal surgery. It is a virtual coupling/god-object technique inspired by the one published by Barbič and James [19].

The remainder of the paper is structured as follows: Section II provides a brief overview of the architecture of the simulator into which the new algorithm was implemented. The algorithm is described in Section III. Section IV presents techniques for acceleration. Section V provides results.

II. SIMULATION ARCHITECTURE

This paper presents work done within the scope of the development of a simulator as a training tool for endoscopic navigation in the nasal cavity. Concerning the interaction between tissues and the virtual instrument, the simulator has to fulfill two major tasks: haptic solving (including collision handling and the computation of force feedback) and tissue deformation. Since material deformation of the required accuracy and complexity cannot easily be simulated at the speed required for haptic stability, it is common practice to decouple the haptic solver from the rest of the simulation [22]. As it has previously been proposed by various authors [21], [23], we use multi-rate haptic rendering, i.e., haptic interaction and tissue deformation are computed in two separate threads with different target iteration frequencies. The two threads communicate whenever the deformation thread is ready to provide a new set of vertex positions. In return, the haptic thread provides contact information to trigger tissue deformations.

Similarly to the model explained by Çavuşoğlu and Tendick [24], the force feedback provided by the haptic device is a sum of two components, a *penalty force* and a *deformation force*. The penalty force is calculated by the haptic solver. It is a spring force aiming at resolving penetrations of the user-controlled instrument into the tissue. The deformation force is the force resulting from tissue resistance to deformation. It is a function of mechanical tissue properties and generated by the deformation solver.

Patient anatomy is modeled with a tetrahedral mesh. Deformations are simulated using the finite element method in continuum solid mechanics. As proposed by other authors [18], [19], the simulated surgical instrument collides with a point shell, more precisely the set of vertices constituting the deforming surface of the tetrahedral mesh. It is therefore important to provide a surface mesh of sufficient density.

Surgical instruments are modeled as combinations of simple geometries like spheres or cylinders. For the description of the algorithm we will assume that the instrument is a single cylinder, as is the case, for example, with an endoscope. However, unless stated otherwise, the presented algorithms are applicable to instruments of more complex shapes.

III. COLLISION HANDLING

Collisions result from movement of the virtual instrument and/or tissue deformation. For the sake of simplicity, the two sources of collisions are treated consecutively. In every haptic iteration, a test is performed to know whether the instrument has collided with the (static) mesh. Analogously, after each update of vertex positions, it is tested whether any of the vertices has collided with the (static) instrument. More precisely, a haptic iteration consists of the following steps:

- **Position Update:** The haptic system is queried for a new position and orientation for the virtual instrument.
- **Collision Detection:** Vertices located inside the space traversed by the virtual instrument from its previous position to the new one are blocking its trajectory and considered *colliding*. In the collision detection step, the set of colliding vertices is identified and deformation requests to move them out of the way are prepared.
- **Instrument Motion:** The position of the virtual instrument is updated, but only the portion of the trajectory preceding the first collision with a vertex is applied.
- **Contact Relaxation:** The contact relaxation step is carried out only if collisions were detected. It is a local optimization of the virtual instrument position and orientation moving it as close as possible to the position and orientation acquired from the haptic system without violating non-penetration constraints.
- **Vertex update:** In case the deformation solver is ready to communicate, an update of vertex positions is acquired. The vertex trajectories are intersected with the virtual instrument to detect collisions. In return, the deformation requests collected in the collision detection step are passed to the deformation solver.
- **Force Feedback:** A force is generated on the haptic system indicating the occurrence of collisions to the user.

Fig. 2. The god-object technique: Left: in unconstrained movement the god-object is equal to the haptic position. Right: when the haptic position penetrates, the god-object stays on the surface.

After a brief explanation of the virtual coupling strategy used, each of these steps will be covered in more detail.

A. Virtual Coupling

The proposed algorithm makes use of the god-object technique (see Figure 2). The god-object is mechanically coupled to the haptic position by the means of an elastic link. When the haptic position penetrates a surface (i.e., a collision occurs), the god-object stays outside. This means that there are two potentially different transformation matrices describing the current location and orientation of the virtual instrument. Let $\mathbf{H}$ be the transformation matrix suggested by the haptic system (the *haptic position*), and $\mathbf{G}$ the transformation matrix of the god-object. $\mathbf{H}$ and $\mathbf{G}$ are equal in unconstrained movement. When a collision occurs, $\mathbf{G}$ and $\mathbf{H}$ separate.

Note that penetration of the god-object into the deformable tissue is not fully prevented. However, it is kept superficial, i.e., surface voxels may be located inside the god-object, but the god-object is kept from sinking deeper into the tissue.

B. Collision Detection

In the collision detection step, the trajectory of the virtual instrument (the *collision volume*, see Figure 3) is examined: The start point of the trajectory is $\mathbf{G}$, the target point is $\mathbf{H}$. For each mesh vertex we test whether it is located inside the collision volume, i.e., blocking the trajectory. Since the collision volume is in general a complex sweep structure, an approximation similar to one proposed by Forest et al. [12] is used: Instead of a moving virtual instrument and fixed vertices, the collision test is performed assuming a fixed virtual instrument and moving vertices. Each vertex position v is transformed into the canonical coordinate system of the cylinder representing the instrument, and the relative trajectory of v from the start to the target is given by the line segment $p(t)$:

$$p(t) = \mathbf{G}^{-1}v + t(\mathbf{H}^{-1}v - \mathbf{G}^{-1}v) \mid 0 \leq t \leq 1. \quad (1)$$

Note that we use 4×4 matrices and homogeneous coordinates for points and vectors.

An intersection test between the relative vertex trajectory $p(t)$ and the cylinder $\mathcal{C}$ in its canonical coordinate system is carried out. The vertex collides if $p(t)$ penetrates $\mathcal{C}$, i.e., if

$$\exists \lambda \mid p(\lambda) \in \mathcal{C}. \quad (2)$$

This linearization of the relative vertex trajectory leads to false positives in the collision detection if $\mathbf{G}$ and $\mathbf{H}$ are significantly rotated with respect to each other. Our experiments

showed that errors for angles up to 45° are insignificant. Therefore we defined a threshold angle $\alpha_{max} = 45^\circ$ and use an interpolation between $\mathbf{G}$ and $\mathbf{H}$ in place of $\mathbf{H}$ in cases where this threshold is exceeded. Interpolations of transformations are performed by interpolating both the rotational part (in quaternion space) and the translational part using the same interpolation parameter.

For each vertex v colliding the cylinder, an adequate deformation request to the deformation solver is defined in the form of an *avoidance vector*. The avoidance vector is the shortest vector, equidirectional with the inverse surface normal $\vec{n}$, by which the vertex would need to move to avoid collision with the cylinder (see Figure 3, left). In the vertex update step (see Section III-E), the set of avoidance vectors is passed to the deformation engine as contact constraints.

The computation of the avoidance vector a is also carried out in the canonical coordinate system of the cylinder. Let $p'(t)$ be the trajectory, relative to the cylinder, of the point $v' = v + a$, i.e., the suggested target position of vertex v :

$$p'(t) = \mathbf{G}^{-1}v' + t(\mathbf{H}^{-1}v' - \mathbf{G}^{-1}v') \mid 0 \leq t \leq 1. \quad (3)$$

The vector a must be long enough such that $p'(t)$ does not intersect $\mathcal{C}$, i.e.:

$$\nexists \lambda \mid p'(\lambda) \in \mathcal{C}. \quad (4)$$

Since a is equidirectional with $-\vec{n}$, we can write $a = -\xi \cdot \vec{n}$ with a scalar $\xi > 0$. If ξ is set to zero and then continuously increased, the event marking the point at which $p'(t)$ finally exits $\mathcal{C}$ is of one of three possible types, as illustrated in Figure 4: (a) one of the end-points of the trajectory exits the cylinder (left), (b) the trajectory intersects the base (or top) circle of the cylinder (center), or (c) the projection of the trajectory to the base plane exits the base circle (right). The avoidance vector is computed by identifying all events of these types and their corresponding values of ξ . The highest value ξ_{max} found defines the correct avoidance vector:

$$a = -\xi_{max} \cdot \vec{n}. \quad (5)$$

C. Instrument Motion

In case the cylinder does not collide with any vertex, the new god-object transformation can be set equal to $\mathbf{H}$. With the virtual endoscope inside the patient's nasal cavity, however, there will be collisions in most haptic iterations. The minimum (over all colliding mesh vertices) λ_{min} satisfying Equation 2 is a good estimate of the fraction of the cylinder trajectory $\mathbf{H}\mathbf{G}^{-1}$ that precedes the first collision. Therefore the new transformation of the god-object is obtained by interpolating between $\mathbf{G}$ and $\mathbf{H}$ with the interpolation parameter λ_{min} (see Figure 3, center). Since the interpolation parameter is only an estimate (due to the linearization of relative vertex trajectories), the moved god-object may penetrate the tissue, though only superficially.

Vertices might also be located inside the god-object at the start of the collision detection step, in which case λ_{min} will be equal to 0 and the god-object blocked from moving. This behavior is undesired in case the user is actually trying to move the instrument away from the tissue. Therefore, a vertex

Fig. 3. The collision detection algorithm. Left: computation of avoidance vectors; Center: new placement of the god-object; Right: only vertices whose surface normals point towards the god-object are colliding.

Fig. 4. As the vertex trajectory travels in direction of the inverse surface normal, three types of events can cause it to lose intersection with $\mathcal{C}$. For simplification, only one surface normal per example is depicted, i.e., it is assumed that $nG^{-1} = nH^{-1}$, which is in general not the case.

v with a surface normal $\vec{n}$ is considered *non-blocking*, if the instrument moves in direction of $\vec{n}$. Transferred to the coordinate system of the instrument (the cylinder), this means that v moves in direction of $-G^{-1}\vec{n}$. More precisely, v is non-blocking if

$$G^{-1}\vec{n} \cdot (H^{-1}v - G^{-1}v) < 0. \quad (6)$$

Non-blocking vertices located inside the god-object are still considered colliding and the deformation-engine will be asked to move them out. However, vertices outside the god-object satisfying Equation 6 can be entirely disregarded, as they face away from the god-object (see Figure 3, right).

D. Contact Relaxation

Using the algorithm as described so far, the user will be unable to slide the instrument along tissue surfaces which will therefore appear extremely sticky. The reason is that whenever the user applies some force against the tissue surface, the haptic position penetrates and the god-object, only able to move into direction of the haptic position, will instantly be blocked by the surface vertices. To overcome this problem, contact relaxation is performed.

Contact relaxation is a local optimization process approaching the god-object towards the haptic position in terms of position and orientation while at the same time preventing non-superficial penetrations. A technique similar to one proposed by Barbič and James [19] is used. They suggested to move the god-object such that an equilibrium between repelling contact forces of penetrating vertices and the attracting *virtual coupling force* pulling the god-object towards the haptic position

is achieved. By using this approach as a separate step aiming solely at contact relaxation, we make use of its benefits while at the same time maintaining the advantages of the algorithm described so far, including the continuous collision detection and the straightforward definition of the desired tissue reaction using avoidance vectors.

The relaxation step, in contrast, uses discrete collision detection, i.e., a vertex is considered in contact if it is located inside the god-object.

In each relaxation substep, the current force $\vec{F}$ and torque $\vec{T}$ acting on the god-object are computed: $\vec{F} = k_C \vec{F}_C + k_{VC} \vec{F}_{VC}$ and, analogously, $\vec{T} = k_C \vec{T}_C + k_{VC} \vec{T}_{VC}$, the subscript C denoting forces and torques pushing the god-object away from contacting vertices, whereas the subscript VC identifies the virtual coupling force and torque. The stiffness parameters k_C and k_{VC} control the weight of each source of forces and torques with respect to the other, and usually $k_{VC} \ll k_C$.

The gradients of force and torque with respect to changes in position x and orientation ω are computed as 3×3 matrices:

$$\frac{\partial \vec{F}}{\partial x}, \frac{\partial \vec{F}}{\partial \omega}, \frac{\partial \vec{T}}{\partial x}, \frac{\partial \vec{T}}{\partial \omega}. \quad (7)$$

These gradients are assembled in a 6×6 system of equations zeroing out $\vec{F}$ and $\vec{T}$. The system is solved for Δx and $\Delta \omega$ and the god-object is moved accordingly. Then forces, torques and gradients are recalculated and another relaxation substep can be carried out.

Figure 5 gives an example of contact relaxation. Four substeps are depicted: There is no penetration at the beginning, so in the first substep only $\vec{F}_{VC}$ is relevant, pulling the god-object towards the haptic position. Suddenly a vertex is contacting, so $\vec{F}_C$ pushes the god-object away from the surface. In the following substep, only $\vec{F}_{VC}$ is active, and so on. Due to $\vec{T}_{VC}$, the orientation of the god-object gradually aligns with the one suggested by the haptic system. Eventually, the god-object will come to rest at an equilibrium of the rivaling forces and torques. As a consequence, some penetration is expected at the end of contact relaxation. However, because of the in comparison large stiffness k_C , penetration depth is kept small.

Due to the long, thin shape of the virtual endoscope and the discrete collision detection, it is hard to prevent the god-object from missing fine structures during relaxation, unless the movement speed is kept small. We therefore limit the translation length $|\Delta x|$ in each substep to a configurable value t_{max} , and the angle θ of the rotation $\Delta \omega$ to a value r_{max} . These values depend on the cylinder radius, inter-vertex

Fig. 5. Four substeps of contact relaxation: $\vec{F}_{VC}$ pulls the god-object towards the haptic position, while $\vec{F}_C$ prevents it from penetrating the surface. The cylinder slides along a surface.

Fig. 6. To transform the dark green into the light green cylinder (left), two rotations are performed, the axial rotation R_a and the rotation of the cylinder orientation, R_o . Interpolation by scaling the angle of the combined rotation $R_a R_o$ yields a counterintuitive result (grey cylinder, center), separating the axial part from the orientational part and interpolating both independently gives the desired result (grey cylinder, right).

distances and the desired friction. As $\Delta\omega$ is in general a combination of an axial rotation R_a of the cylinder and the rotation R_o from the old to the new cylinder orientation, one can separate R_a from R_o and scale the respective angles to avoid counterintuitive results (see Figure 6).

If the virtual instrument consists of only one cylinder, like the virtual endoscope, axial rotation does not influence the occurrence of collisions and vice versa. Therefore the axial rotation suggested by the haptic system can be directly applied to the god-object before starting to compute the relaxation. So R_a will always be the identity matrix and the separation does not need to be done.

The number of relaxation substeps performed influences the perceived surface friction and can be configured according to the simulation case.

E. Vertex Update

Whenever the haptic solver and the deformation engine communicate, an update of vertex positions is received. During a simulation, tissue deformation frequently moves vertices past the surface of the god-object, e.g., when tissue bounces back after a user-triggered deformation. One solution to this problem is prevention. The haptic solver is not allowed to impose deformation by moving vertices. Thus only the deformation solver could prevent such events. However, leaving this responsibility to the deformation solver would mean sharing the control over tissue-tool interaction between the two solvers, which would increase complexity, interdependence and communication overhead. Therefore we suggest to have the haptic solver process all collisions, including those caused by tissue deformation.

After an update of vertex positions, the trajectory $v \rightarrow v + \Delta v$ of each vertex v is intersected with the surface of the god-object. If an intersection occurs, the vertex is marked *watched* (see Figure 7, left). In subsequent collision detection steps, watched vertices are treated as if they were located at the surface of the god-object, i.e., instead of using the actual vertex coordinates v for the collision detection, we use the point $r(t_{exit})$ where the ray $r(t) = v - t \cdot \vec{n} \mid t \geq 0$ exits the god-object, $\vec{n}$ being the surface normal of v . We do this to be sure that the vertex ends up inside the collision volume, even if it has fully traversed the god-object. An avoidance vector a is computed as outlined in Section III-B, still using $r(t_{exit})$ in place of v . The already assumed displacement $a_0 = r(t_{exit}) - v$ is then added to a (see Figure 7, center and right).

In the contact relaxation step, *watched* vertices are considered to be inside the god-object. So they will contribute to the force F_c described in Section III-D. The resolution of collisions caused by tissue deformation is thus achieved not only by pushing the tissue back (using avoidance vectors), but also by movement of the virtual instrument (through contact relaxation).

A vertex is removed from the *watched*-list, when $r(t)$ does not intersect the god-object.

This strategy works very well in simulation of endonasal surgery, where tissue deformations are subtle and resulting collisions are quickly resolved. However, one can easily think of other simulation cases where vertices move faster, such that it is conceivable that whole surface patches are moved right across the god-object from one iteration to the next. Resolving a collision like this may take a considerable amount of haptic iterations. In the meantime, parts of the god-object that are inside the tissue are not blocked by surface vertices and can move rather freely. The consequence will be that too many vertices are removed from the *watched*-list too early and the collision is never resolved. For this reason, whenever a massive penetration like this is detected, the instrument motion step (see Section III-C) must be skipped until the god-object is back at the surface. There are several ways of defining a criterion for a penetration being considered massive. In our implementation, the instrument motion step is temporarily blocked, if a vertex and all of its direct neighbors in the mesh fully traverse the god-object.

During the communication with the deformation thread, the haptic solver also receives an updated deformation force for each contacting vertex. In return, the new set of avoidance

Fig. 7. Collision handling after mesh update: Left: three vertex trajectories intersect the god-object; Center: in a following collision detection step, the vertices are assumed to be located on the surface of the god-object, so they are inside the collision volume. The assumed displacement a_0 is added to the avoidance vector; Right: the haptic position moves away from the penetrating surface. Therefore $a = a_0$ and the vertices are pushed out of the god-object, towards their assumed positions.

vectors collected in the collision detection step is sent to the deformation solver as contact constraints.

F. Computation of force feedback

At the end of each haptic iteration, the haptic feedback is computed. A penalty force and torque are generated whenever the transformation provided by the haptic system, H , is different from the transformation of the god-object, G , with the translation and rotation portions of the matrix GH^{-1} defining the exerted force and torque, respectively. The deformation forces (acquired from the deformation engine in the vertex update step) of all contacting vertices and the respective torques are added to produce the final haptic feedback.

IV. ACCELERATION

To ensure stable haptic interaction, haptic update frequency must be kept high. A frame-rate of 1kHz is desirable, 300Hz is acceptable for soft deformable objects [15]. Since the simulation of endonasal surgery requires the modeling of a large portion of the patient's head and the cylinder-vs.-vertex collision strategy demands a high density of vertices, we have to deal with large point shells. Some parts of the described algorithm require iteration over all vertices. An octree is used to accelerate these computations by reducing the set of vertices processed to those that are located close to the virtual instrument. Octrees partition the three-dimensional definition space by recursively subdividing it into eight subspaces. They are widely used in collision detection [25].

A. Octree construction

The octree is constructed once, at the start of the simulation. The coordinates of all surface mesh vertices are averaged to obtain the center point which defines the axis-aligned planes separating the eight subvolumes. An analogous strategy is applied in each subvolume. We create, including the root, four levels, i.e., there are 512 subspaces in the lowest level. Each of them keeps a list of the vertices that it contains. Whenever vertex positions change, the octree must be updated. Since the overall distribution of vertices in space does not change significantly during a simulation of transnasal surgery,

Fig. 8. The god-object and the cylinder in haptic position are covered by axis-aligned bounding boxes. Then, each box is combined with the respective box in the other cylinder. Here the first two and the last combinations are highlighted.

the octree structure is not recalculated. Instead, only the vertex lists are updated. As an octree update is required only whenever the haptic thread and the deformation engine communicate (ideally about every 10^{th} haptic iteration), the added computational effort for updating the octree is small compared to its benefits.

B. Vertex culling

The octree is used to obtain the set of vertices located in a certain neighborhood. First, this neighborhood is defined as a set of axis-aligned bounding boxes. Then the octree is queried for the set of vertices located inside any of the subvolumes that intersect any of the provided bounding boxes.

Three steps of the presented collision handling algorithm - collision detection, contact relaxation and vertex update - can be accelerated by using the octree to reduce the number of processed vertices.

In the collision detection step (see Section III-B), both the god-object and the cylinder in the position suggested by the haptic system are covered by the same number of axis-aligned bounding boxes (see Figure 8). Each bounding box is then combined with the respective bounding box in the other cylinder. This new set of bounding boxes which covers the entire collision volume is used to query the octree for the set of vertices to be processed.

For the discrete collision detection in contact relaxation (see Section III-D), it is sufficient to query the octree with a set of bounding boxes covering the god-object. This must be done once during each relaxation substep. The resulting set of vertex coordinates must then be transformed into the coordinate system of the cylinder. Since these vertex transformations still take a lot of time in sum, we use a non-axis-aligned bounding box around the cylinder to further shrink the set of vertices to be checked.

In the vertex update step (see Section III-E), the octree is updated right after receiving the new vertex positions. The up-to-date octree is then immediately used to reduce the number of vertex trajectories for which an intersection test is carried out, i.e., only the trajectories that end in vicinity of the god-object are processed. This is not without danger, because vertices could theoretically be moved to arbitrary places after traversing the god-object. However, one can analyze maximum movement speeds and adjust the size of the bounding boxes accordingly. For simulation of transnasal surgery, covering a

Fig. 9. The vertex mesh of the artificial funnel-structure used for assessment of the algorithm.

cylinder whose radius is twice the one of the endoscope is more than sufficient.

C. Multi-threading

Even after vertex culling using the octree and non-axis-aligned bounding boxes, some parts of the algorithm still perform transformations and collision tests for large numbers of vertices. These computations are easily parallelizable. To make use of the highly parallel architectures of modern desktop computers, we bifurcate the haptic thread into subthreads for the respective parts of the algorithm.

V. RESULTS

The algorithm was tested on a Dual Intel Xeon system (16 cores), which the haptic thread shared with the deformation engine and the graphical user interface of the simulator. We used two different haptic devices with very different properties: A Phantom Omni by SensAble Technologies (<http://www.sensable.com>), and a device conceived for transnasal surgery simulation by the company Quanser (<http://www.quanser.com>). The latter provides a much larger force spectrum and five degrees of freedom (DOF) of haptic feedback, while the Omni does not exert torques and is therefore limited to three DOF of haptic feedback.

Two different test cases were created for the assessment of the algorithm. The first one consists of a block with an opening shaped like a funnel whose cylindrical shaft has a diameter only slightly bigger than that of the endoscope (3.6mm vs. 3mm). The funnel-structure is depicted in Figure 9. The model consists of about 17 000 surface vertices.

The task was to slide the endoscope into the opening, insert it all the way to the back wall of the funnel shaft, exert some pressure on the back wall and slide it back out again. It was inspired by the peg-in-hole benchmark for rigid haptics described by Ortega et al. [26]. Two different materials were simulated, one being very soft (with a Young's modulus of 0.005 MPa) and the other stiff (0.2 MPa).

Figure 10 illustrates the tool trajectories. The observed stability was very satisfying. The insertion of the endoscope into the funnel was easy and smooth. The sliding of the endoscope produced highly intuitive haptic feedback with a realistic force-discontinuity while sliding over the edge at the entrance to the funnel shaft. When using the Omni device on the stiff material, the lack of torques got evident - when the endoscope was not accurately aligned with the funnel shaft,

Fig. 10. Tool trajectory for the endoscope-in-a-funnel task using both the deformable (top) and the stiff (bottom) material. Left: sliding into the opening; middle: pushing against the back wall; right: sliding out of the opening. The green cylinder is the god-object, the grey cylinder depicts the haptic position.

somewhat counterintuitive forces pushing the endoscope out of the funnel were observed.

The second simulation case was endoscopic navigation in the nasal cavity. A model was generated by segmentation of images acquired from computed tomography. The much bigger model size (about 190 000 surface vertices) and structural complexity significantly increased the challenges on contact stability and the computational load.

The task in this simulation case was to insert an endoscope with a diameter of 3mm into one of the patient's nostrils, advance it up to the *sphenoid ostium* (a small opening at the back of the nasal cavity), retract the endoscope and repeat the process in the other nostril. This task is difficult for people with limited experience and knowledge of the complex anatomy. If the endoscope is inserted in the wrong direction and a wrong trajectory is taken, the *ostium* will not be found. Figure 11 shows the trajectory of the instrument controlled by an experienced user.

A haptic frame rate of 1kHz was maintained while only a small portion of the endoscope was inserted. As the endoscope was advanced further, speed decreased progressively. The frame rate briefly dropped to its minimum of about 500Hz when the endoscope was fully inserted. Yet, during the entire simulation, haptic interaction was very smooth and the feedback realistic. The number of relaxation substeps per iteration had been set to 3, the maximum step size was $t_{max} = 0.08mm$ and the maximum rotation angle per substep was $r_{max} = 0.5^\circ$ (see Section III-D).

The system in its current state was demonstrated at a meeting with 15 neurosurgeons who were very satisfied with the contact mechanics and haptic feel. Even the "light" version of the system, employing the Omni haptic system, was accepted as a suitable tool for practicing endoscopic navigation.

VI. CONCLUSION

The contact handling algorithm presented in this paper provides satisfying results simulating haptic interaction in

Fig. 11. Simulation of endonasal surgery through the right nostril (top) and the left nostril (bottom): insertion of the endoscope (left), passing the middle nasal turbinate (center), finding the *sphenoid ostium* (right). For improved visibility, only the god-object (green) is displayed. As the graphics show, the passages are very narrow and access to the *sphenoid ostium* is possible only by means of massive deformation.

transnasal endoscopic surgery. Future developments will include reworking the configuration interface, such that the relaxation behavior can be described in terms of established physical parameters, like the friction coefficient. We are also working on further improving the speed of the algorithm in order to be able to insert a second virtual instrument besides the endoscope. This will be the next step towards the goal of providing a realistic simulator for the entire surgical procedure.

REFERENCES

- [1] U. Bockholt, W. Müller, G. Voss, U. Ecke, and L. Klimek, "Real-time simulation of tissue deformation for the nasal endoscopy simulator (nes)," *Comput Aided Surgery*, vol. 4, no. 5, pp. 281–285, 1999.
- [2] S. Wolfsberger, A. Neubauer, K. Bühler, R. Wegenkittl, T. Czech, S. Gentzsch, H.-G. Böcher-Schwarz, and E. Knosp, "Advanced virtual endoscopy for endoscopic transsphenoidal pituitary surgery," *Neurosurgery*, vol. 59, no. 5, pp. 1009–1019, 2006.
- [3] S. S. Parikh, S. Chan, S. K. Agrawal, P. H. Hwang, C. M. Salisbury, B. Y. Rafii, G. Varma, K. J. Salisbury, and N. H. Blevins, "Integration of patient-specific paranasal sinus computed tomographic data into a virtual surgical environment," *American Journal of Rhinology & Allergy*, vol. 23, no. 4, pp. 442–447, 2009.
- [4] B. Perez-Gutierrez, D. M. Martinez, and O. E. Rojas, "Endoscopic endonasal haptic surgery simulator prototype: A rigid endoscope model," in *Proc. of IEEE Virtual Reality*, 2010, pp. 297–298.
- [5] M. Caversaccio, A. Eichenberger, and R. Hausler, "Virtual simulator as a training tool for endonasal surgery," *American Journal of Rhinology*, vol. 17, no. 5, pp. 283–290, 2003.
- [6] A. Neubauer, S. Wolfsberger, M.-T. Forster, L. Mroz, R. Wegenkittl, and K. Bühler, "Advanced virtual endoscopic pituitary surgery," in *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 11, no. 5, 2005, pp. 497–507.
- [7] H. Çakmak, H. Maaß, C. Trantakis, G. Strauß, E. Nowatius, and U. Kühnapfel, "Haptic ventriculostomy simulation in a grid environment," *Comput. Animat. Virtual Worlds*, vol. 20, no. 1, pp. 25–38, 2009.
- [8] B. Tolsdorff, A. Pommert, K. H. Höhne, A. Petersik, B. Pflesser, U. Tiede, and R. Leuwer, "Virtual reality: A new paranasal sinus surgery simulator," *The Laryngoscope*, vol. 120, no. 2, pp. 420–426, 2010.
- [9] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," in *Proc. of ACM SIGGRAPH*, 1988, pp. 289–298.
- [10] M. Mahvash and V. Hayward, "Haptic simulation of a tool in contact with a nonlinear deformable body," in *IS4TM'03: Proc. of the 2003 International Conference on Surgery Simulation and Soft Tissue Modeling*, 2003, pp. 311–320.
- [11] D. E. Johnson and P. Willemsen, "Six degree-of-freedom haptic rendering of complex polygonal models," in *Proc. of International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2003, pp. 229–235.
- [12] C. Forest, H. Delingette, and N. Ayache, "Surface contact and reaction force models for laparoscopic simulation," in *Proc. of International Symposium on Medical Simulation*, 2004.
- [13] V. García-Pérez, E. Muñoz Moreno, S. Aja-Fernández, and C. Alberola-López, "A 3-d collision handling algorithm for surgery simulation based on feedback fuzzy logic," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 4, pp. 451–457, 2009.
- [14] R. Keiser, M. Müller, B. Heidelberger, M. Teschner, and M. H. Gross, "Contact handling for deformable point-based objects," in *Proc. of Vision Modeling Visualization (VMV)*, 2004, pp. 315–322.
- [15] G. Picinbono, J.-C. Lombardo, H. Delingette, and N. Ayache, "Improving realism of a surgery simulator: linear anisotropic elasticity, complex interactions and force extrapolation," *The Journal of Visualization and Computer Animation*, vol. 13, no. 3, pp. 147–167, 2002.
- [16] J. E. Colgate, M. C. Stanley, and J. M. Brown, "Issues in haptic display of tool use," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995, pp. 140–145.
- [17] C. B. Zilles and J. K. Salisbury, "A constraint-based god-object method for haptic display," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995, pp. 146–151.
- [18] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in *Proc. of ACM SIGGRAPH*, 1999, pp. 401–408.
- [19] J. Barbič and D. L. James, "Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models," in *Proc. of 2007 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, Aug. 2007.
- [20] M. A. Otaduy and M. Gross, "Transparent rendering of tool contact with compliant environments," in *Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2007, pp. 225–230.
- [21] G. Saupin, C. Duriez, and S. Cotin, "Contact model for haptic medical simulations," in *Proc. of International Symposium on Biomedical Simulation (ISBMS)*, 2008, pp. 157–165.
- [22] K. Salisbury, F. Conti, and F. Barbagli, "Haptic rendering: Introductory concepts," *IEEE Computer Graphics and Applications*, vol. 24, no. 2, pp. 24–32, 2004.
- [23] F. Barbagli, K. Salisbury, and D. Prattichizzo, "Dynamic local models for stable multi-contact haptic interaction with deformable objects," in *HAPTICS '03: Proc. of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, 2003, pp. 109–116.
- [24] M. C. Çavuşoğlu and F. Tendick, "Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2000, pp. 2458–2465.
- [25] T. He and A. Kaufman, "Collision detection for volumetric objects," in *Proc. of IEEE Visualization*, 1997, pp. 27–34.
- [26] M. Ortega, S. Redon, and S. Coquillart, "A six degree-of-freedom god-object method for haptic display of rigid bodies," in *Proc. of IEEE Virtual Reality*, 2006, pp. 197–204.