Haptic Rendering of Volume Data with Collision Determination Guarantee Using Ray Casting and Implicit Surface Representation

Roman Vlasov rvlasov@gdv.uni-hannover.de
Karl-Ingo Friese kif@gdv.uni-hannover.de
Franz-Erich Wolter few@gdv.uni-hannover.de
Institute of Man-Machine-Interaction, Leibniz Universität Hannover, Germany

***Abstract*—Haptic exploration adds an additional dimension to working with 3D data: a sense of touch. This is especially useful in areas such as medical simulation, training and pre-surgical planning, as well as in museum display, sculpting, CAD, military applications, assistive technology for blind and visually impaired, entertainment and others. There exist different surface- and voxel-based haptic rendering methods. Unaddressed practical problems for almost all of them are that no guarantees for collision detection could be given and/or that a special topological structure of objects is required. Here we present a novel and robust approach based on employing the ray casting technique to collision detection, which does not have the aforementioned drawbacks while guaranteeing nearly constant time complexity independent of data resolution. This is especially important for such delicate procedures as pre-operation planning. A collision response in the presented prototype system is rigid and operates on voxel data, and no precalculation is needed. Additionally, our collision response uses an implicit surface representation ”on the fly”, which can be used with dynamically changing objects.**

***Keywords*-haptics; haptic rendering; collision detection; collision response; collision resolution; ray casting; implicit surface**

I. Introduction

With the evolution of medical scanning devices, especially Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), 3D volume data is nowadays widely used in modern medicine. These modalities have become an integral part of a clinical practice. Resulting 3D images are used for diagnosis, therapy planning, interventional guidance, and follow-up. 3D volume data is also in use in geology, CAD-applications, entertainment and other areas.

In order to significantly increase usability and effectivity of work with 3D data, an additional dimension could be added to a virtual system - a sense of touch. This could be done using a haptic device. With a haptic device a user can both manipulate a virtual object and feel force feedback reactions. Source data could be in different representations (triangulated surface, hexahedrons, volumetric, ...), but we focus on a volumetric one, since it is a direct output from the scanning devices. Other data types could be transformed to this one, if necessary.

There exist many haptic rendering methods, but almost all of them have drawbacks that (1) ”thin” obstacles could be skipped or an interaction point could go inside them and/or (2) a certain topological structure of objects is needed, such as connectivity or number of holes. The last is also an important issue, since the real medical data we work with can have any structure, especially if segmentation has been done automatically. Here we present a novel approach that does not have these problems by employing ray casting for the collision detection and a ”sliding along a surface” model for the collision response. Additionally, no precalculations or explicit surface representations are needed. This means that a virtual scene may be both dynamic or static. To our best knowledge, the use of ray casting in haptic rendering is a novel interdisciplinary approach being on the cutting edge of visualization and haptic rendering research areas. Our method was implemented and tested within the framework provided by the YaDiV platform [1] – a powerful virtual system for working with 3D volume data. This allows us to combine novel haptic rendering methods for exploration of medical data with high-quality visualization. Our approach has nearly constant time complexity independent of data resolution and is very fast – up to 750 points could be simulated at haptic update rates (1 kHz) for the collision detection only and up to 150 points for the collision detection and collision response (both values are given for a moderate end-user PC). This allows to perform object-object collision detection at a sufficient speed.

II. Definitions

A. Volume Data

In the field of medical visualization volume data, also called *volumetric data*, is usually acquired with CT or MRI. The result of such an acquisition process is a data set consisting of pairs $< coordinates, intensity\_value >$, where the intensity value is a scalar measured by a scanning device (e.g. the value of unabsorpted X-rays) [2]. One can take a look for a detailed description of volume data and related terms in [3].

As far as scanned data has no color or tissue information, a segmentation step could further be needed. **Segmentation** is a process to extract certain structures from a volume data set. In medical context this can be anatomical organs, like kidney or bones, or pathological structures, like tumors. The segmentation process is a large field of research, and

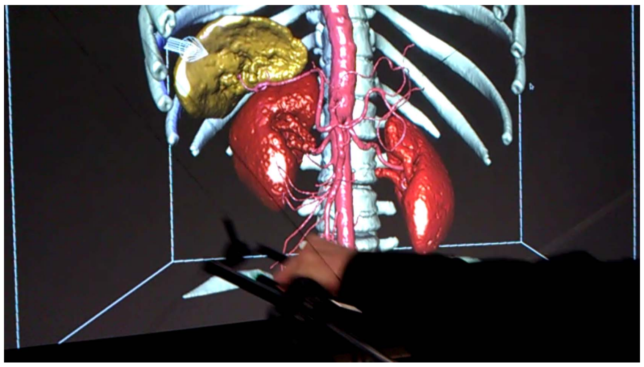

Figure 1. Haptic rendering of the Torso data set using INCA 6D device

different approaches for different purposes have already been proposed (see e.g. [2] and [4] for an overview and suggested methods). We assume that 3D data is already segmented, i.e. that a set of segments (a set of scene objects) is provided. We use a bit cube representation of segments (see [4] for details), though other representations are possible.

B. Haptics

The term **haptic** (from the Greek *haptesthai*, meaning ”to touch”) is an adjective used to describe something relating to or based on the sense of touch. Haptic is to touching as visual is to seeing and as auditory is to hearing.

The definitions below will be used in the rest of the paper.
A **haptic device (or haptic display)** is capable of tracking its own position and/or orientation and stimulates the kinesthetic sense of the user via a programmable force feedback.
A **probe (or end-effector) (of a haptic display)** is the part of the device the position/orientation is tracked for (passive Degrees-of-Freedom/DoF) and a force feedback is applied to (active DoF).
A **tool (in a virtual world)** is an object in the virtual world the user manipulates via the probe. A particular case is the **(haptic) interaction point** (if the object is a 3D point).
A **handle (in a virtual world)** is a grasped part of the tool.

III. Overview of Related Work

A haptic rendering application should solve three main tasks: contact determination (also called collision detection), collision response and generation of force feedback (see figure 2). All stages are often tightly integrated in order to effectively use a solution of one task for solving others.

There are two ways of controlling a haptic display:

1) admittance control (a user applies a force to the device, and the application moves the probe according to the simulation)
2) impedance control (a user moves the probe of the device, and the application produces forces).

According to [5] and [6], the impedance control scheme is cheaper and easier to construct and is usually used nowadays. It is also assumed in our work.

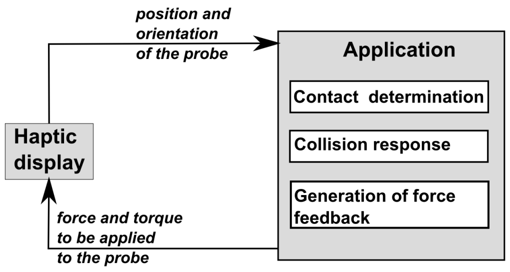
position and orientation of the probe
Application
Contact determination
Collision response
Generation of force feedback
Haptic display
force and torque to be applied to the probe

Figure 2. Haptic rendering pipeline

The quality of haptic rendering can be measured in terms of a dynamic range of impedances that can be simulated in a stable manner: a perceived impedance (i.e. force) should be very low for movements in free space and high for contacts between a rigid tool and rigid objects. So, the probe should stop quickly if rigid contact between the tool and a virtual obstacle occurs. But because of existence of sample and latency phenomena, unstable behavior of the probe could arise in such cases. This instability is directly perceived by a user in a form of disturbing oscillations. It is even possible that the tool passes through an obstacle because of a fast movement, or it appears at different sides of it in succeeding frames. In connection with these issues Colgate et al. showed in [7] that a computation of feedback forces at a high update rate is important for haptic rendering stability. According to Brooks et al. [8] it should be at least 0.5 - 1 kHz.

To address the aforementioned requirement, there exist two main techniques of handle manipulation:

1) direct rendering (apply manipulations with the probe directly to the handle)
2) virtual coupling (connect the haptic probe to the handle through a virtual spring-damper connection).

Direct rendering is useful if a haptic rendering method can perform all the stages at an update rate sufficient for a stable user interaction (1 kHz). The virtual coupling is good in rest cases, e.g. for multirate approaches, when force feedback is generated at 1 kHz, but physics simulation, say, at 30 Hz. Virtual coupling was firstly proposed by Colgate et al. [7].

We should also note that a force update rate of 1 kHz is generally not sufficient for stable haptic rendering. It means that other issues, such as too fast object movement or too strong forces, should be addressed in the system.

There are different haptic rendering approaches, starting with those very early, like [9] in 1995 and [10]. In the following we will give an overview of existing methods.

Zilles and Salisbury [11] proposed a god-object non-penetration approach for 3-Degrees-of-Freedom (3-DoF). Later it was extended to 6-DoF by Ortega et al. [12]. An extension of the god-object idea in 3-DoF is a ”virtual proxy” [13]. At each frame, the position of the probe in a virtual environment is set as a goal for the tool. Then possible constraint surfaces are identified using the ray between the old position of the virtual proxy (the tool) and the goal position. After that a quadratic optimization problem is solved and a subgoal position is found. This process is repeated until the subgoal position could not be closer to the goal. An extended 6-DoF technique of virtual proxy was used in [14].

McNeely et al. [15] developed a 6-DoF haptic rendering method for the Boeing Company. They proposed to use a voxmap (spatial occupancy map) and object pointshells. Volumetric data and penalty pre-contact forces were used. Later this approach was significantly improved in the next works of McNeely and others [16] and [17]. Basing on the above method, Barbic et al. [18], [19] proposed their own approach, which supports a contact between a rigid and a reduced deformable models, both with complex geometry. A distributed contact between objects is allowed, i.e. an interaction with potentially several simultaneous contact sites. A pointshell-based hierarchical representation was used for the deformable object and a signed-distance field for the rest of the scene. Later in [20] the distance field was made parametrically deformable.

A completely different haptic rendering approach was suggested by Otaduy, Lin et al. [21], [22]. Their method allows haptic rendering of interaction between "haptically textured" triangulated models (with fine surface details stored as a height field). Collision detection between low-resolution meshes is based on sensation-preserving contact levels of detail from [23], [24]. An evolution of the method are works [25] of Otaduy and Gross, where environmental objects are deformable (represented by tetrahedral meshes), and [26]. In the last work, a layered representation of objects is employed: a low-resolution mesh for the collision detection and haptic interaction, a deformable tetrahedral mesh for deformation computations and a detailed surface mesh for the deformable skin simulation. Further on, in [27] Garre and Otaduy presented a method, where both the tool and environmental objects could be deformable.

Another interesting approach was suggested by Johnson and Willemsen [28], [29]. They used spatialized normal cone hierarchies for fast collision detection between the tool and an environmental object. Weller and Zachmann [30] presented inner sphere trees – a structure which bounds an object from inside with a set of non-overlapping bounding volumes – and employed it for haptic rendering.

Duriez et al. [31] proposed a method using Signorini's contact model for deformable objects in haptic simulations with a focus on contact response. It belongs to approaches with non-penetration constraints and is independent from a collision/proximity detection. In the later work [32] the authors incorporated friction into the simulation model.

Debunne et al. [33] presented a method for animating dynamic deformations of a visco-elastic object with a guaranteed frame-rate, built into a 6-DoFs haptic rendering framework. An object is represented via a tetrahedral mesh, and the proposed physical simulation approach belongs to physics-based continuous models. Kuroda et al. [34] presented a simulation framework, where the manipulating point pushes a deformable object, which is in contact with another one. A work of Basdogan et al. [35] is devoted to 6-DoF haptics in minimally invasive surgical simulation and training. One more method for a "physically realistic" virtual surgery was suggested by De et al. [36]. They used the Point-Associated Finite Field (PAFF) approach. The idea is to discretize the computational domain (e.g. an organ) using a scattered set of points ("nodes") with spherical influence zone and defined nodal shape function for it. Maciel et al. [37] also presented a haptic rendering method for physics-based virtual surgery, but using NVIDIA's PhysX physics liblary, which is GPU accellerated. The method supports 6-DoF. Luciano et al. [38] devoted their work to a local elastic point-based deformation around a contact point in 3-DoF haptic rendering.Chang et al. [39] proposed a 6-DoF haptic rendering method using the mass-spring simulation model. Vidal et al. [40] made a simulation of ultrasound guided needle puncture and proposed a proxy-based surface/volume haptic rendering for that. Palmerius et al. [41] have shown in their work how subdivision of proxy movements can improve precision of volume haptic rendering. Kim and others [42] presented a method, which uses implicit surface representations and requires some preprocessing and a certain topology. An approach devoted to haptic rendering of volume-embedded isosurfaces was suggested by Chan et al. [43]. Another haptic rendering method, which uses isosurfaces defined by interpolating on tetrahedral meshes, was recently proposed by Corenthy et al. [44]. In [45], [46] Boettcher et al. suggested a kinesthetic haptic rendering of virtual fabrics grasped by two fingers. The fingers are represented via spherical tools manipulated by two 3-DoFs probes. The simulation of tactile perception of the fabrics was proposed by Allerkamp et al. [47], [48]. Later on, in [49] Boettcher et al. described a generalized multi-rate coupling scheme of physical simulations for haptic interaction with deformable objects.

As was mentioned at the beginning of this work, almost all methods can not give collision detection and non-penetration guarantees. Additionally, many of them require a special topological structure of objects. In the next sections we present our approach, which does not have these drawbacks.

IV. Our approach

A. Ray Casting

A key technique for the collision detection in our haptic rendering pipeline is ray casting in volumetric data. This technique has its roots in computer graphics.

In more detail, as far as the source data for haptic rendering is volumetric, we employed the very popular [50] ray casting visualization technique (see e.g. [50]–[55]) and adapted it to our needs.

The idea of ray casting in visualization is to numerically evaluate the volume rendering integral in a straightforward manner. The rendering integral $I_\lambda(x,r)$, i.e. the amount of the light of wavelength λ coming from a ray direction r that is received at location x on the image plane, is:

$$I_\lambda(x,r) = \int_0^L C_\lambda(s)\mu(s)e^{-\int_0^s \mu(t)dt} ds, \qquad (1)$$

where L – the length of the ray r; μ – absorption (extinction) coefficient at the specified position on the ray r; C_λ – amount of the light of wavelength λ emitted at the specified position on the ray r.

The used optical model in the method is "Absorption and Emission" (the volume consists of particles that absorb and emit light, see [56]) with the restriction, that only a directional light parallel to the viewing direction is allowed. For each pixel of the image a ray is cast into the scene. Along the cast ray the intensity values of the volumetric data are resampled at equidistant intervals, usually using trilinear interpolation. After the resampling an approximation of the volume rendering integral along the ray in either back-to-front or front-to-back order is computed. In this process the mapping of the $< coordinates, scalar\_value >$ pairs for the resampled points to colors and opacities according to a previously chosen transfer function is used.

B. Collision Detection

The method was given in short in our work [57]. In this subsection we present it in more detail.

As was mentioned above, the original method returns a numerical evaluation of the volume rendering integral along a given ray. In our case, for the interaction point (IP) following the position of the manipulator, we perform ray casting from its last position to the current one. In more detail, we are going along the ray with 1-voxel step. If the value of any bit cube representing an obstacle at the sampled point is *true* then *true* and a collision information is returned by the collision detection procedure. *False* is returned otherwise. See the figure 3 for details. We use 1-voxel step, because a minimum possible thickness of an object is also one voxel. By performing the ray casting we can always find the exact collision, if it happened between the haptic rendering updates, and react to it accordingly. To our best knowledge, there exists only one method (see [12]), which provides the same collision detection guarantees as ours, but it only works with triangulated objects and not with volumetric / voxel based data.

In order to have even higher precision for collision detection, ray casting at sub-voxel resolution or sampling once between each pair of consecutive intersections of the ray and a grid plane could be used. Though we found that 1-voxel step is quite enough for our experimental data.

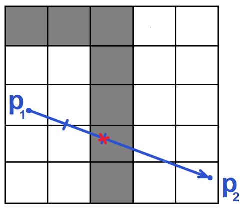

Figure 3. The ray from the previous position p_1 to the current one p_2 is cast with 1-voxel step until an obstacle is found or p_2 is reached

To further speed up the computations, we firstly create a list of objects that are determined as *collision candidates*. For that, we check if the ray from the last position to the current one collides with the Axis-Aligned-Bounding-Box (AABB) of each object. If so, then the object is a candidate. The detailed collision detection is performed for these candidates only.

Additionally, we impose a reasonable upper limit on the maximal movement of the IP between two haptic frames. This allows us to perform localized and therefore faster ray casting using the cached information from the previous frame and avoid possible haptic rendering instabilities (the last is also done in [19]).

If all data has been already loaded then the time complexity of the method is

$$O\left(N_{obj} \cdot \frac{w_{max}}{step}\right), \qquad (2)$$

where N_{obj} – number of objects in the scene; w_{max} – maximum path length per frame, in voxels; $step$ – the sampling step of ray casting (chosen as 1).

Indeed, in the worst case all objects in the scene could become the collision candidates and be checked all the way from the previous position of the IP to the current one.

C. Collision Response

The collision detection method described above is used in our joint collision detection and response stage of the haptic rendering pipeline. The method is based on the god object/proxy paradigm. It works directly with volumetric data and has no limitations.

As the IP should not go inside any object or pass through it, we make it slide over the surface. The surface is calculated locally "on the fly". The IP can encounter multiple surfaces on its way. It is connected with the actual position of the device's manipulator via a virtual spring. This approach was made to test the capacities and speed of our collision detection method and as a base for further experiments.

We denote the position of the IP from the last frame as p_1 and the one to be calculated as p_2. For the device's manipulator, we denote its last position as d_1 and the current one as d_2. The IP always moves in the direction of d_2. *Empty-space border voxels* below are the voxels, which are empty but have at least one non-empty N_{26}-neighbour.

The algorithm deals with different obstacles at the same time and looks as follows:

```
1: Get $p_1$, $d_1$, $d_2$
2: $p_2$ := $p_1$ // Initialize $p_2$
3: $p_{2last}$ := $p_2$ - (1,1,1) // make it unequal to $p_2$
4: $w$ := 0 // Path length travelled by the IP at this frame
5: while ($p_2 \neq d_2$ and $w < w_{max}$ and $p_{2last} \neq p_2$) do
6:    $p_{2last}$ := $p_2$
7:    Make the collision test from $p_2$ to $d_2$
8:    if (no collision) then
9:       Move $p_2$ towards $d_2$ for the distance min($||d_2$-$p_2||_2$, $w_{max} - w$)
10:      $w$ := $w$ + (the above movement of $p_2$)
11:      break
12:   else
13:      Move $p_2$ towards the collision point $p_{col}$ so that it is at the given $\epsilon < 1$ before $p_{col}$, or for the distance ($w_{max} - w$) from $p_2$ in case the last is shorter
14:      $w$ := $w$ + (the above movement of $p_2$)
15:      // Slide over the obstacle in the direction of $d_2$:
16:      while $w < w_{max}$ and $p_2 \neq d_2$ do
17:         // Is it shorter just to move from $p_2$ towards $d_2$
18:         // without following the surface?
19:         if ($p_2$ will not be inside any obstacle if moved by 1 voxel towards $d_2$) then
20:            // We will move directly to $d_2$ at the beginning
21:            // of the next iteration of the outer loop
22:            break
23:         end if
24:         Locate neighbour empty-sp. border voxels for $p_2$
25:         Select a voxel with the biggest dot product of (voxel-$p_2$) and ($d_2$-$p_1$)
26:         if (the biggest dot product $\leq 0$) then
27:            break
28:         end if
29:         Move $p_2$ towards the selected voxel for the distance min($||$voxel-$p_2||_2$, $w_{max} - w$)
30:         if ($p_2$ is inside another obstacle) then
31:            Cancel the above movement of $p_2$
32:            break
33:         end if
34:         $w$ := $w$ + (the above movement of $p_2$)
35:      end while
36:   end if
37: end while
```

Note: If the empty-space border voxels are precalculated for each segment at the preprocessing step then it gives 25% speed-up. All the frame rates in our paper are given for the case without preprocessing.

D. Force Feedback

The specificity of our force feedback generation is that we do not use surface normals, because we do not employ an explicit surface representation. The total force transferred to a user via the haptic manipulator is $F = F_c + F_{fr}$, where F_c is a coupling force and F_{fr} is a friction force. If F exceeds a maximum for a given haptic device then we scale it as to fit to the device limitations. A calculation of F_c yields

$$F_c = -\frac{d_2 - p_2}{\|d_2 - p_2\|_2} \cdot (\|d_2 - p_2\|_2 \cdot k) = (p_2 - d_2) \cdot k, \quad (3)$$

while for F_{fr} we obtain

$$F_{fr} = -\frac{p_1 - p_2}{\|p_1 - p_2\|_2} \cdot |F_c \cdot n| \cdot \mu \cdot \frac{N_{bv}}{w}, \quad (4)$$

where k is the coefficient of the spring; n – a normal vector, which is perpendicular to p_2-p_1 and lies on the plane defined by vectors p_2-p_1 and d_2-p_2; μ – the friction coefficient; N_{bv} – number of the border empty-space voxels, which the IP moved through in the algorithm above at this haptic frame; w – the total path length at this frame, also from the algorithm above.

We would like to note that for easier calculations $|F_c \cdot n|$ could be rewritten as $\|F_c\|_2 - \left|F_c \cdot \frac{p_1 - p_2}{\|p_1 - p_2\|_2}\right|$.

We use the given expression for F_{fr} because at the end of a haptic frame the IP is moved from p_1 to p_2, and therefore it is logical to turn the friction force to the opposite direction. Also we make it proportional to the part of F_c, which is perpendicular to p_2-p_1 in analogy to the normal force for a dry friction. Finally, we ensure it to be proportional to N_{bv}, i.e. the path length that the interaction point actually slid over a surface. We would like to note that making the forces related to physical properties of certain materials was not our goal on this stage of research.

V. Implementation Details

As it was already mentioned before, our interactive VR system is based on the YaDiV Open-Source platform [1]. The main features include reading of input data in the DICOM format and offering modules for 2D Visualization, 3D Volume Visualization (2D-Texturing and 3D-Texturing), 3D Segmentation and 3D Segment- Visualization and Registration. The platform was successfully employed for teaching and educational purposes and extended by many student projects. It is also currently used by physicians at Hannover Medical School (MHH) in various research projects.

Our prototype system is structurally a plug-in for YaDiV. In order to allow absolute platform independence, YaDiV was developed using the Java platform. This is the case for our system, too. Only the device dependent part was developed using C++. The system is independent from a haptic display, so that a wide range of devices can be used, including Phantom Omni, High-end Phantom Premium 1.5 6-DOF and INCA 6D with a very large workspace of approx. 2*1*1.6m (figure 1). The size of the virtual workspace can be scaled and varies from case to case.

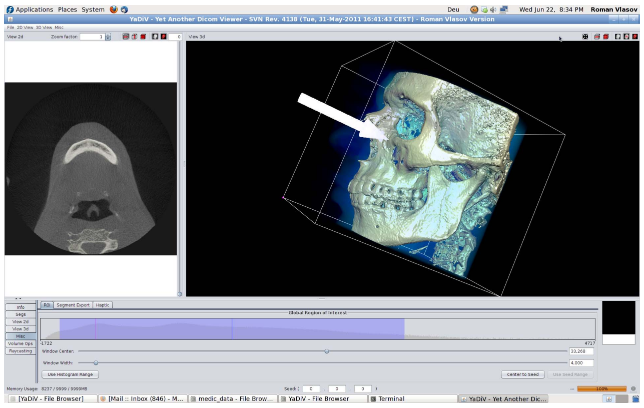
Figure 4. Working with the data set of the head Head$_{big}$

VI. Results

For tests real medical tomography data sets were used, including Torso (520x512x512, fig. 1), Head$_{big}$ (464x532x532, fig. 4) and Head$_{small}$ (113x256x256, fig. 4).

For the point-object collisions only, the haptic update rate during the *peak load* is about 750 kHz on our moderate high-end user PC (8 x Intel Xeon CPU W5580 @ 3.20GHz, 24 GB RAM, NVIDIA Quadro FX 5800). For the joint collision detection and response approach the value is about 160-170 kHz. Both values exceed the minimum requirement for real-time haptics by orders of magnitude. This efficiency and the conceptual clarity of our approach contrasts most triangle-based approaches, where millions of triangles would be generated and complex speeding-up traversing structures are required for the fast and precise collision detection. The values were obtained for the virtual haptic device, which is simulated in Java. For real devices, Java-C++ communication (transferring of the device transformations and forces) is required. We have measured the timings and found out that because of these communication costs the resulting update rate is a little lower – about 150 kHz. The values for the data sets for the joint collision detection and response approach are shown in table I. **Triangles** denotes number of triangles in the scene for the graphics rendering as a reference. Triangulation was extracted from the volumetric data using a modified marching cubes algorithm. **Update Rate** is given for real devices and during the peak load.

Our prototype system was tested under Microsoft Windows, as well as under Linux. Under Linux it was also run using the stereo graphics mode. We found the last one especially useful for an intuitive interaction with 3D data comparing to the normal graphics mode.

VII. Summary and Future Work

We presented a new haptic rendering approach employing a novel collision detection technique based on ray casting concepts known from computer graphics. The method gives

Table I
Resulting update rates

Data	Size	Triangles	Update Rate
Head$_{small}$	113x256x256	690K	152 kHz
Torso	520x512x512	2,222 Mi	138 kHz
Head$_{big}$	464x532x532	6,136 Mi	146 kHz

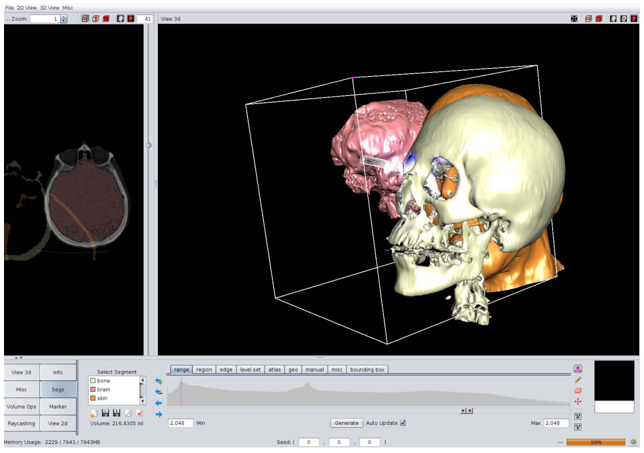
Figure 5. The data set Head$_{small}$

collision detection guarantees that a manipulated object does not pass through ”thin” obstacles and is never inside any of them while not requiring any special topological object structure. The collision detection was extensively tested with a new ”slide along a surface” approach using an implicit surface representation ”on the fly”. The results confirm our approach to be a viable alternative to existing techniques avoiding most common drawbacks. The prototype was implemented as a plug-in of the YaDiV VR system and tested on several haptic devices.

Based on the approach presented in this paper, we will add support for object-object collisions.

We are also looking into the possibility of drastically speeding up the calculations by employing GPUs. As was shown e.g. in [51], [55], ray casting could be efficiently parallelized using GPUs and/or multi-processor systems. We plan to conduct the tests on the hardware which we already have at our Institute. It includes the high-end Tesla cluster granted by NVIDIA in the context of a Professor Partnership Program, modern graphics hardware including NVIDIA Fermi (GF 480), multi-core processor systems and an IBM Cell Cluster.

For the advanced contact resolution we will focus on Finite-Elements-Models (FEMs). It is also planned to use FEM-based approaches for simulation of elastic tissues. Simplified FEMs were used e.g. in [32], [25], [20], but for those works objects were triangulated compared to our volumetric data.

The typical use case of our VR system will be assembling a fractured bone, which is especially important for pre-operation planning in facial surgery. It is planned that it will be assessed by physicians from MHH.

Acknowledgements

This research is sponsored by a grant provided by Siemens/DAAD Postgraduate Programme (DAAD - German Academic Exchange Service). The authors would like to thank P.Blanke, A.Vais and B.Berger for helpful comments.

References

[1] K.-I. Friese, P. Blanke, and F.-E. Wolter, "Yadiv – an open platform for 3d visualization and 3d segmentation of medical data," *The Visual Computer*, vol. 27, pp. 129–139, 2011.

[2] M. Chen, C. Correa, S. Islam, M. Jones, P.-Y. Shen, D. Silver, S. J. Walton, and P. J. Willis, "Manipulating, deforming and animating sampled object representations," *Computer Graphics Forum*, vol. 26(4), pp. 824–852, 2007.

[3] A. Kaufman, D. Cohen, and R. Yagel, "Volume graphics," *IEEE Computer*, vol. 26(7), pp. 51–64, July 2007.

[4] K.-I. Friese, "Entwicklung einer plattform zur 3d-visualisierung und -segmentierung medizinischer daten," Ph.D. dissertation, Leibniz Universitat Hannover, Germany, 2010.

[5] M. Glencross, A. G. Chalmers, M. C. Lin, M. A. Otaduy, and D. Gutierrez, "Exploiting perception in high-fidelity virtual environments," *ACM SIGGRAPH 2006 Courses*, July 2006.

[6] M. A. Otaduy Tristan, "6-dof haptic rendering using contact levels of detail and haptic textures," Ph.D. dissertation, University of North Carolina at Chapel Hill, 2004.

[7] J. E. Colgate, M. C. Stanley, and J. M. Brown, "Issues in the haptic display of tool use," *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems*, pp. 140–145, 1995.

[8] F. P. Brooks Jr., M. Ouh-Young, J. J. Batter, and P. J. Kilpatrick, "Project grope - haptic displays for scientific visualization," *ACM SIGGRAPH Computer Graphics*, vol. 24(4), pp. 177–185, August 1990.

[9] Y. Adachi, T. Kumano, and K. Ogino, "Intermediate representation for stiff virtual objects," *Virtual Reality Annual International Symposium*, pp. 203–210, 1995.

[10] W. R. Mark, S. C. Randolph, M. Finch, J. M. Van, V. Russell, and M. Taylor II, "Adding force feedback to graphics systems: issues and solutions," *Proc. of the 23rd annual conf. on Comp. graphics and interactive techniques*, pp. 447–452, 1996.

[11] C. B. Zilles and J. K. Salisbury, "A constraint-based god-object method for haptic display," *Proc. of the Int. Conf. on Intelligent Robots and Systems*, vol. 3, pp. 31–46, 1995.

[12] M. Ortega, S. Redon, and S. Coquillart, "A six degree-of-freedom god-object method for haptic display of rigid bodies with surface properties," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13(3), pp. 458–469, May 2007.

[13] D. C. Ruspini, K. Kolarov, and O. Khatib, "The haptic display of complex graphical environments," *Proc. of the 24th ann. conf. on Comp. gr. and interact. techn.*, pp. 345–352, 1997.

[14] A. Gregory, A. Mascarenhas, S. Ehmann, M. Lin, and D. Manocha, "Six degree-of-freedom haptic display of polygonal models," *Proc. of the conf. on Vis.'00*, pp. 139–146, 2000.

[15] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 401–408, July 1999.

[16] M. Wan and W. A. McNeely, "Quasi-static approximation for 6 degrees-of-freedom haptic rendering," *Proc. of the 14th IEEE Visualization Conference (VIS03)*, pp. 257–262, 2003.

[17] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Voxel-based 6-dof haptic rendering improvements," *Journal of Haptics-e*, vol. 3(7), 2006.

[18] J. Barbic and D. James, "Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models," *Proc. of the '07 ACM SIGGRAPH/Eurogr. symp. on Comp. animation*, pp. 171–180, 2007.

[19] J. Barbic, "Real-time reduced large-deformation models and distributed contact for computer graphics and haptics," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, 2007.

[20] J. Barbic and D. L. James, "Six-dof haptic rendering of contact between geometrically complex reduced deformable models," *IEEE Trans. on Haptics*, vol. 1(1), pp. 39–52, 2008.

[21] M. A. Otaduy, N. Jain, A. Sud, and M. C. Lin, "Haptic display of interaction between textured models," *Proceedings of the conference on Visualization '04*, pp. 297–304, October 2004.

[22] M. A. Otaduy and M. C. Lin, "A perceptually-inspired force model for haptic texture rendering," *Proc. of the 1st Symp. on App. perception in graphics and vis.*, pp. 123–126, 2004.

[23] M. A. Otaduy and M. C. Lin, "Stable and responsive six-degree-of-freedom haptic manipulation using implicit integration," *Proc. of the 1st Joint Eurohaptics Conf. and Symp. on Hapt. Interf. for Virt. Env. and Tel. Syst.*, pp. 247–256, 2005.

[24] M. A. Otaduy and M. C. Lin, "A modular haptic rendering algorithm for stable and transparent 6-dof manipulation," *IEEE Trans. on Robotics*, vol. 22(4), pp. 751–762, 2006.

[25] M. A. Otaduy and M. Gross, "Transparent rendering of tool contact with compliant environments," *Proc. of the 2nd Joint EuroHaptics Conf. and Symp. on Haptic Interfaces for Virt. Env. and Teleoperator Systems*, pp. 225–230, 2007.

[26] N. Galoppo, M. A. Otaduy, S. Tekin, M. Gross, and M. C. Lin, "Interactive haptic rendering of high-resolution deformable objects," *Proceedings of the 2nd international conference on Virtual reality*, pp. 215–223, 2007.

[27] C. Garre and M. A. Otaduy, "Haptic rendering of complex deformations through handle-space force linearization," *In the Proc. of the World Haptics Conference*, pp. 422–427, 2009.

[28] D. E. Johnson and P. Willemsen, "Six degree-of-freedom haptic rendering of complex polygonal models," *Proc. of the 11th Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, pp. 229–235, 2003.

[29] D. E. Johnson, P. Willemsen, and E. Cohen, "Six degree-of-freedom haptic rendering using spatialized normal cone search," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11(6), pp. 661–670, November 2005.

[30] R. Weller and G. Zachmann, "A unified approach for physically-based simulations and haptic rendering," *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, pp. 151–160, August 2009.

[31] C. Duriez, C. Andriot, and A. Kheddar, "Signorini's contact model for deformable objects in haptic simulations," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2004*, pp. 32–37, 2004.

[32] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12(1), pp. 36–47, January 2006.

[33] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr, "Dynamic real-time deformations using space & time adaptive sampling," *Proc. of the 28th annual conference on Computer graphics and interactive techniques*, pp. 31–36, 2001.

[34] Y. Kuroda, M. Nakao, S. Hacker, T. Kuroda, H. Oyama, M. Komori, T. Matsuda, and T. Takahashi, "Haptic force feedback with an interaction model between multiple deformable objects for surgical simulations," *Proceedings of Eurohaptics2002*, pp. 116–121, July 2002.

[35] C. Basdogan, S. De, J. Kim, M. Muniyandi, H. Kim, and M. A. Srinivasan, "Haptics in minimally invasive surgical simulation and training," *IEEE Computer Graphics and Applications*, vol. 24(2), pp. 56–64, March 2004.

[36] S. De, Y.-J. Lim, M. Manivannan, and M. A. Srinivasan, "Physically realistic virtual surgery using the point-associated finite field (paff) approach," *Presence: Teleoperators and Virtual Environments*, vol. 15(3), pp. 294–308, June 2006.

[37] A. Maciel, T. Halic, Z. Lu, L. P. Nedel, and S. De, "Using the physx engine for physics-based virtual surgery with force feedback," *In Int. Journal of Medical Robotics and Computer Assisted Surgery*, vol. 5(3), pp. 341–353, September 2009.

[38] C. J. Luciano, P. Banerjee, and S. H. R. Rizzi, "Gpu-based elastic-object deformation for enhancement of existing haptic applications," *Proc. of the 3rd Annual IEEE Conf. on Automation Science and Engineering*, pp. 146–151, 2007.

[39] Y.-H. Chang, Y.-T. Chen, C.-W. Chang, and C.-L. Lin, "Development scheme of haptic-based system for interactive deformable simulation," *Computer-Aided Design*, vol. 42(5), pp. 414–424, May 2010.

[40] F. Vidal, N. John, A. Healey, and D. Gould, "Simulation of ultrasound guided needle puncture using patient specific data with 3d textures and volume haptics," *Journal of Visualization and Computer Animation*, vol. 19, pp. 111–127, 2008.

[41] K. Palmerius and G. Baravdish, "Higher precision in volume haptics through subdivision of proxy movements," *Proc. of EuroHaptics '08*, pp. 694–699, 2008.

[42] L. Kim, A. Kyrikou, M. Desbrun, and G. Sukhatme, "An implicit-based haptic rendering technique," *In Proc. of the IEEE/RSJ International Conf. on Intelligent Robots*, 2002.

[43] S. Chan, F. Conti, N. Blevins, and K. Salisbury, "Constraint-based six degree-of-freedom haptic rendering of volume-embedded isosurfaces," *W. Hapt. Conf.'11*, pp. 89–94, 2011.

[44] L. Corenthy, J. S. Martin, M. Otaduy, and M. Garcia, "Volume haptic rendering with dynamically extracted isosurface," *Proceedings of Haptics Symposium 2012*, pp. 133–139, 2012.

[45] G. Boettcher, *Haptic Interaction with Deformable Objects*. Springer, 2011.

[46] G. Boettcher, D. Allerkamp, D. Gloeckner, and F.-E. Wolter, "Haptic two-finger contact with textiles," *Visual Computer*, vol. 24, no. 10, pp. 911–922, September 2008.

[47] D. Allerkamp, G. Boettcher, F.-E. Wolter, A. C. Brady, J. Qu, and I. R. Summers, "A vibrotactile approach to tactile rendering," *Visual Computer*, vol. 23, no. 2, pp. 97–108, 2007.

[48] D. Allerkamp, *Tactile Perception of Textiles in a Virtual-Reality System*. Springer, 2011.

[49] G. Boettcher, D. Allerkamp, and F.-E. Wolter, "Multi-rate coupling of physical simulations for haptic interaction with deformable objects," *Visual Computer*, vol. 26, no. 6-8, pp. 903–914, January 2010.

[50] M. Hadwiger, P. Ljung, C. R. Salama, and T. Ropinski, "Advanced illumination techniques for gpu-based volume raycasting," *ACM SIGGRAPH 2009 Courses*, 2009.

[51] J. Kruger and R. Westermann, "Acceleration techniques for gpu-based volume rendering," *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, pp. 287–292, October 2003.

[52] M. Levoy, "Efficient ray tracing of volume data," *ACM Transactions on Graphics*, vol. 9(3), pp. 245–261, July 1990.

[53] J. Mensmann, T. Ropinski, and K. Hinrichs, "Accelerating volume raycasting using occlusion frustums," *In IEEE/EG Int. Symp. on Vol. and Point-Based Graphics*, pp. 147–154, 2008.

[54] K. Engel, M. Hadwiger, J. M. Kniss, A. E. Lefohn, C. R. Salama, and D. Weiskopf, "Real-time volume graphics," *ACM SIGGRAPH 2004 Course Notes*, 2004.

[55] T. Ropinski, J. Kasten, and K. H. Hinrichs, "Efficient shadows for gpu-based volume raycasting," *Proc. of the 16th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG08)*, pp. 17–24, 2008.

[56] N. Max, "Optical models for direct volume rendering," *IEEE Tr. on Vis. and Comp. Graphics*, vol. 1(2), pp. 99–108, 1995.

[57] R. Vlasov, K.-I. Friese, and F.-E. Wolter, "Ray casting for collision detection in haptic rendering of volume data," *I3D '12 Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, p. 215, March 2012.