Joint Virtual Reality Conference of EuroVR - EGVE (2011)
R. Blach, S. Coquillart, M. D'Cruz, A. Steed, and G. Welch (Editors)

Bimanual Haptic Simulator for Medical Training: System Architecture and Performance Measurements

Sebastian Ullrich[†1] and Dominik Rausch[1] and Torsten Kuhlen[1]

[1]Virtual Reality Group, RWTH Aachen University, Germany

Abstract

In this paper we present a simulator for two-handed haptic interaction. As an application example, we chose a medical scenario that requires simultaneous interaction with a hand and a needle on a simulated patient. The system combines bimanual haptic interaction with a physics-based soft tissue simulation. To our knowledge the combination of finite element methods for the simulation of deformable objects with haptic rendering is seldom addressed, especially with two haptic devices in a non-trivial scenario. Challenges are to find a balance between real-time constraints and high computational demands for fidelity in simulation and to synchronize data between system components. The system has been successfully implemented and tested on two different hardware platforms: one mobile on a laptop and another stationary on a semi-immersive VR system. These two platforms have been chosen to demonstrate scaleability in terms of fidelity and costs. To compare performance and estimate latency, we measured timings of update loops and logged event-based timings of several components in the software.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities, H.5.2 [Information Interfaces and Presentation]: User Interfaces—Haptic I/O, Benchmarking

1. Introduction

The haptic sense is an important modality in virtual reality (VR) and simulation. Much research has been dedicated to assess the psychological aspects of how we perceive haptics and its influence on other senses [LK09]. From a sensory point of view there are two fundamental types of haptics: kinesthetics and touch. In the following we focus on kinesthetic haptic feedback, because it is essential for many medical procedures. More specifically, in various medical procedures the view on the region of interest is occluded and thus the medical specialists have to rely on their mental image of the anatomical structures augmented by haptic perception. Despite the increasing use of simulators to learn, improve and rehearse medical skills, the adoption of skill trainers and mannequins is limited by patient variance, inaccurate representation of biological tissue, and physical wear from repeated use. However, interactive VR-based simulators are potentially valuable to overcome these constraints [GAPD08]. Based on survey papers and reports [LTCK03, UKK11] several requirements to achieve plausible training scenarios can be identified:

- realistic data sets, ideally based on medical imaging,
- deformable tissue with physics-based simulation,
- topological changes (depending on medical procedure),
- virtual representations of instruments,
- visual, auditory and haptic feedback,
- intuitive user interfaces (software and hardware), and
- real-time interaction.

In this paper we describe a haptic simulator architecture with simultaneous palpation and needle guidance that is used for a medical training system [GNU*09]. One of the biggest challenges is to address and fulfill the requirements, while finding a balance between simulation fidelity and reasonable real-time response. First, Section 2 gives a short overview of related work. Then the system architecture, that combines a physics-based soft tissue simulation with real-time haptic feedback for bimanual interaction, and its most important components are presented in Section 3. Section 4 describes two different hardware platforms that have been used for testing. To validate real-time capability, measurements and comparison of both platforms are reported in Section 5.

† email:s.ullrich@ieee.org

2. Related Work

There are already several surveys providing an excellent overview of approaches to haptic rendering [SCB04, LD07, LO08]. Additionally, haptic solutions in medical simulation are summarized in [CMJ10]. While there are needle simulators with haptic rendering, we focus here on system architectures for haptic rendering and haptic rendering of deformable objects.

A significant requirement results from research on sensory thresholds [LK09]: the servo motors of haptic devices should be updated at 1000 *Hz* [LD07]. Otherwise, vibration artifacts and unstable feedback can be perceived by the user. Usually, a virtual reality system with visual, auditory and haptic feedback contains the following elements: simulation engine, rendering algorithms, input devices and transducers and a human operator [SCB04]. There are basically two approaches with several variations to implement the haptic rendering system: sequential with virtual coupling [CSB95] or parallel with separate threads for visual rendering, simulation, collision detection and force response [AKO95]. In order to perform stable haptic rendering on deformable surfaces with multiple contact points, Barbagli et al. [BPS05] combined a multirate haptic system with coarse local representations for high update rates with virtual coupling. However, no details are provided about the deformation technique. Other deformable haptic approaches are often based on mass-spring systems [CT00, CBS06, RSK09]. Employing the long elements method with haptics basically reduces the complexity of the deformation to 2D [Bal06]. Another approach employs a linear complementary problem formulation to couple FEM-based deformable simulation with haptic rendering [SDC08]. The closest approach to our work also combines palpation with needle intervention [CGJC10]. Key differences to our work are a simulation which is mass-spring-based instead of using FEM and a focus on an user interface with an augmented reality-setup.

In summary, while there are solutions that combine haptic rendering with physics-based simulation, they are often not tested in complex medical scenarios. Furthermore, details on the non-trivial system architecture is lacking in most papers.

3. Bimanual Haptic Simulator System

The simulator system comprises several components as shown in Figure 1. The implementation is based on the ViSTA VR toolkit (http://www.sourceforge.net/projects/vistavrtoolkit), the medical simulation framework SOFA [ACF*07] for *Simulation* of hard & soft tissue, Bullet Physics Library (http://bulletphysics.org) for *Collision Detection* and OpenHaptics (http://www.sensable.com) to control the *Haptic Rendering* and haptic devices. It uses OpenSG (http://www.opensg.org) for the *Visual Rendering* component. Because the system is designed for bimanual interaction, there are two instances of

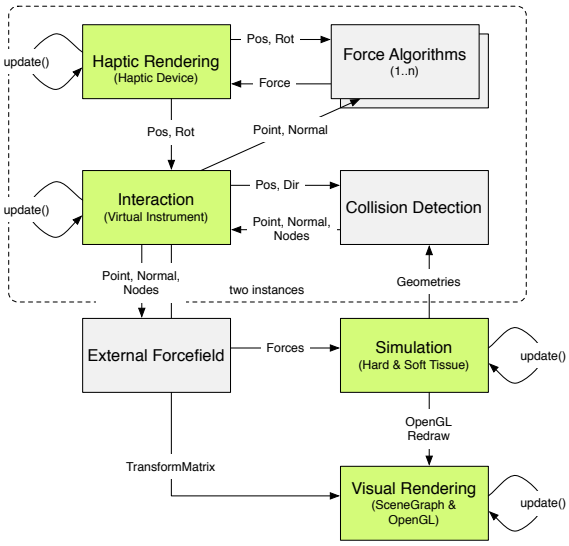

Figure 1: *System overview with components (green/darker gray = threaded) and the most important updates and data communication. The components in the dotted box exist twice: once for palpation and once for needle intervention. Thus six threads are running at different rates.*

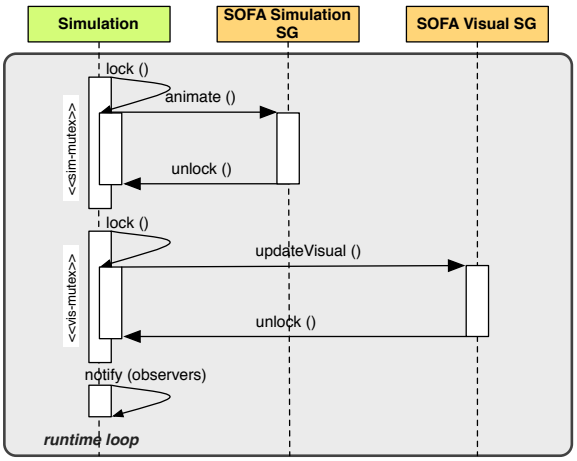

Figure 2: *Sequence diagram of the simulation loop.*

the *Interaction*, *Haptic Rendering*, *Collision Detection* and *Force Algorithms* component. One set of these components provides a virtual hand with extended fingers for palpation, while the other enables a virtual surgical instrument (e.g., a cannula or a needle) for needle intervention. In the following subsections we describe the components in more detail.

3.1. Simulation

The main task of the simulation is to provide a physics-based simulation of the hard and soft tissue of the virtual patient. It utilizes selected components of the medical simulation framework SOFA [ACF*07]. Instead of using the default single update loop of SOFA, we decouple the simulation from the visual rendering. Therefore, our simulation loop encapsulates one instance of the SOFA simulation scene graph and thus provides a dedicated thread for the simulation (see Figure 2). Furthermore, an instance of SOFA's visual scene graph can access this simulation graph under certain conditions (see Section 3.2). There are two mutexes in the simulation loop: a *simulation mutex* for a simulation update and *visualization mutex* used for a mapping update.

Simulation Update: The simulation mutex locks access to SOFA's simulation scene graph during a simulation update. Such an update increments the time stamp of the simulation and calls *animate()*, which issues the numerical solvers to compute the next simulation state, e.g., with *behavior meshes* that represent FEM volumes. Because the simulation graph is also indirectly accessed by *Interaction* component for tissue dragging (see Section 3.4), the simulation mutex is necessary for thread safety.

Mapping Update: The second mutex is a visualization mutex. It is used to temporary lock access to SOFA's visual scene graph after a simulation time step is finished. During this lock phase the call *updateVisual()* conducts changes to SOFA's visual scene graph conforming to the simulation's time step results via predefined mappings. Usually, a visual model, e.g., representing a skin surface, gets deformed by a barycentric mapping linked to a behavior mesh that is used in the simulation.

Collision Detection Update: After *updateVisual()* observers are notified and update the collision detection geometries, i.e., overwriting vertices with new positions.

Subsurface Structures: Subsurface structures are embedded in the behavior mesh either by barycentric mapping or by constraints applied to mesh nodes, e.g., fixed nodes that represent bones. Thus, internal structures like blood vessels are deformed accordingly. This can be triggered by external dragging forces from palpation acting on the skin surface or by the virtual needle puncturing or dragging tissue layers (see Section 3.4).

3.2. Visual Rendering

Visual rendering in our prototype is implemented in the ViSTA VR toolkit and utilizes OpenSG and OpenGL.

Simulation Object Rendering: To visualize the objects from our simulation loop (see Section 3.1), we have attached an OpenGL rendering node in the scene graph. This node sets SOFA-specific OpenGL states and calls the *draw()* routine of SOFA's visual graph. This call locks the *visualization mutex* in the simulation loop code (see Figure 3). During this time, the simulation can still be performed. However, updates of mappings from the simulation graph to the visual graph (*updateVisual()*-call) are blocked and have to wait until the rendering routine is completed. Usually, only the surface geometries specified by the visual models of SOFA are rendered. By switching transparency, anatomical structures can be visualized for learning and visual verification. Optionally, for debug and development purposes, it is possible to render internal meshes, mappings, forces, constraints, behavior and collision meshes as well.

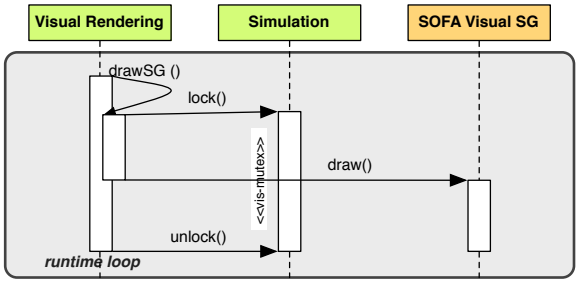

Figure 3: *Sequence diagram of the visual rendering loop.*

Instrument Rendering & Shadows: The visual models of the virtual instruments, e.g., a needle or the palpation hand, are rendered directly in the OpenSG scene graph without SOFA. They are represented by polygonal mesh geometries. To enhance the rendering of the virtual hand and arm geometry we use a simple skin shader with wrap lighting [Gre04].

In order to provide additional depth cues and to increase realism, we use shadows maps with one point light source in the scene. The shadows introduce additional render passes that decrease the visual frame rate. Especially, in a stereoscopic setting this technique divides the maximum achievable frame rate by a factor of four. Still, this sacrifice in performance is acceptable, given the fact that virtual shadows contribute to depth perception and improve positioning accuracy [HSJ04].

3.3. Collision Detection

We use two different implementations for collision detection (CD). The first one is intended to detect collisions between the interaction devices and surface mesh objects at high rates. It is a component of our system architecture and utilizes the Bullet physics library, primarily using Bullet's triangle-ray test. The second CD implementation is used for proximity search within volumes of behavior meshes and uses parts of the SOFA collision detection pipeline.

Surface Collision Detection: The CD component consists of a CD object registry, handles geometry changes and employs CD algorithms from the Bullet physics library. CD objects are registered and usually contain references

to triangular surface meshes with coordinate lists of the vertices and face indices that describes the topology. Typically, the CD objects either represent visual or collision models from the SOFA simulation environment. Each CD object has its own mutex to manage exclusive access for read operations during the actual collision detection query and write operations during geometry updates from SOFA. This mutex-per-object approach reduces locking between collision queries from interaction loops and deformation updates to CD objects, as compared to locking the whole CD with one mutex. Furthermore, the design decision to implement this decoupled CD component, instead of using the CD mechanisms provided by SOFA, is easily explained by the requirement to enable a higher update rate for the interaction loop than the simulation loop.

Proximity Search: The proximity search is used to find nodes in the data structures of the behavior meshes that are within a certain threshold of a ray. The ray represents either the virtual hand and is limited to surface collisions or it represents the needle instrument and searches for internal nodes as well (see Figure 4). The results are used for tissue dragging depending on additional parameters, as described in detail in Section 3.4.

3.4. Interaction

There are two instances of the interaction component: one for palpation and one for needle interaction.

Palpation: In short, palpation is a physical examination technique where objects, e.g., body parts or indirectly organs, are touched with fingers or hands to determine their size, shape, consistency and location. We focus on non-prehensile movements [Nap56], i.e., with no grasping or seizing motions. Consequently, the simulated soft tissue should be deformable and draggable during sliding motions depending on surface friction. A virtual hand is controlled by the user and allows for interaction with the soft tissue of a virtual patient. Dragging behavior is simulated with a physics-based approach for friction combined with some heuristics and constraints to determine stick and slip states. The resulting dragging forces are written into the *External Forcefield*. Subsequently, the forcefield influences the behavior mesh in the *Simulation* component and thereby drags nodes of the simulation mesh, deforming the tissue.

Needle Insertion: In general, needle insertion can be divided into several stages: 1) no contact, 2) deformation before rupture, and 3) deformation after rupture. These stages can be applied to multiple layers of tissue. A first penetration event can be observed at the *cutaneous layer* (skin surface) and, depending on the body region, after sliding through fatty tissue, *fascia* (strong connective tissue) surrounding musculature must be penetrated and can be sensed with a "fascia click". For the different stages, we activate specific force algorithms (see Section 3.5).

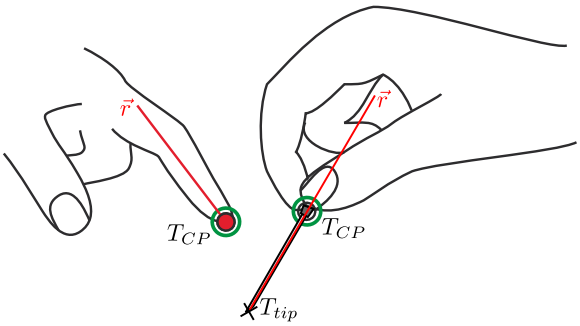

Figure 4: *Schematic overview of the tool center points T_{CP} of the haptic devices and of the rays $\vec{r}$ for collision detection on palpation hand (left) and the needle (right).*

Rupture events are triggered when reaching predefined force thresholds and lead to the next stage. Needle-tissue interaction extends the surface-based palpation dragging by internal behavior mesh nodes (see proximity search in Section 3.3) and is similar to [DS03].

3.5. Force Algorithms & Haptic Rendering

In order to support concurrent force effects and to reuse and combine effects, we use composable elemental force algorithms. Furthermore, effects can be enabled and disabled depending on interaction states and will be iterated in the haptic render loop.

Pulse Force: For palpation, the human pulse must be simulated. We approximate the pulsating effect by a rectangular wave with a duty cycle $D = 0.2$ and pulse period T between 50-160 bpm. The duty cycle is defined as the ratio of the pulse duration d and pulse period T. Similar to [UMN*08], we distribute pulse point sources along the femoral artery. When the tool center point of the haptic device $\vec{T}_{CP}$ is within range of these pulse point sources, the pulsating force is computed depending on orientation, euclidean distance and simulation time.

Multiple Surfaces Force: For palpation of *sub-dermal* structures (under the skin layer) we use a multi-object force algorithm. A collision ray $\vec{r}$ is attached to the unconstrained transformation of the input device $\vec{T}_{CP}$ (see Figure 4). On collision of $\vec{r}$ with surface or subsurface objects, constraint planes (*Force Algorithms*) are created. The approach is similar to the single surface haptic rendering in [RKK97]. However, in our approach, the proxy-object gets moved to the outermost ray-intersection, i.e, the virtual fingers stay on the skin surface. Another option would be to optimize the discrepancy by position and velocity [BRWB07]. For needle insertion this algorithm is also used in stages before rupture of layers. On rupture the according constraint plane/force algorithm is deactivated, leading to a minor loss-of-resistance or "fascia click".

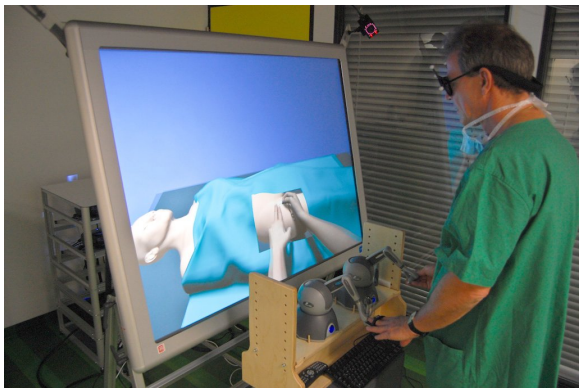

Figure 5: *Stationary platform of the bimanual haptic simulator in a semi-immersive environment with stereoscopic rendering and IR-based head tracking.*

Figure 6: *Mobile platform of the bimanual haptic simulator presented with the same haptic devices as in the stationary platform.*

Needle Force: We simulate two kinds of forces for the needle: the shaft friction and a surface correction force. The friction force is a force vector along the needle shaft. It is calculated by using the needle velocity and the current tissue type's parameters that are based on real-life forces measured by [OSO04]. The surface correction force simulates tissue resistance for lateral movements and depends on the penetration depth, deeper penetration results in stronger lateral resistance. It is a spring-based force towards the initial puncture site, dependent on lateral deviations. The force vector is perpendicular to the surface normal of the initial puncture site, i.e., approximately parallel to the surface.

Switching between constraint plane positions based on low frequency simulation update rates, as well as low-resolution surface meshes, leads to discontinuities in the force output. To resolve this issue we combine two solutions. First, we use linear interpolation between constraint planes after each position update. Furthermore, we apply a special smoothing filter for force rendering (Section 5.3 in [GME*00]).

4. Bimanual Haptic Simulator Prototype

The simulator has been successfully implemented on two different hardware setups: stationary and mobile. In both cases we set the update rates for haptics to 1000 Hz, for simulation to 25 Hz, for interaction to 120 Hz and to maximal possible visual framerate.

4.1. Stationary Platform

The *stationary semi-immersive setup* (cp. Figure 5) consists of two haptic devices (PHANTOM Omni® Haptic Device, SensAble Technologies), a passive-stereo VR-desktop system with a SXGA+ 60" rear-projection screen (flip_150, imsys) coupled with an optical IR-tracking system (TrackPack, Advanced Realtime Tracking) and is driven by a 2.4 GHz (Intel® Core™2 Quad) desktop PC with 4 GB RAM, Windows 7 operating system and a Quadro FX 4600 graphics card with 768 MB RAM (Nvidia).

Advantages of this system is the real-sized, full-scale rendering of the virtual patient with reasonable overview and natural interaction in standing or seating posture next to the slightly tilted screen, with a comparable position between practitioner and patient as in the real medical procedure.

4.2. Mobile Platform

The *mobile setup* (cp. Figure 6) consists of two haptic devices (PHANTOM Omni® Haptic Device, SensAble Technologies) and is driven by a 2.5 Ghz (Intel® Core™2 Duo) MacBook Pro 17" with 2 GB RAM, Windows XP SP3 operating system installed with Bootcamp and a GeForce 8600M GT with 512 MB RAM (Nvidia).

The small footprint makes the system easily transportable. Therefore, this configuration is well-suited for demonstrations at conferences or for short expert studies in busy hospital environments. Furthermore, the mobile system is less expensive (~6k Euro) compared to the stationary (~45k Euro).

4.3. Simulation Datasets

The whole simulation environment in SOFA can be specified in a hierarchically structured XML scene file, e.g., with rigid and deformable objects consisting of behavior meshes, visual models, collision models, mappings between representations and physics properties from continuum mechanics like Young's modulus and Poisson's ratio.

In our example, the scene contains a virtual patient on a table, covered by a blanket (4513 vertices, 8666 triangles)

with an exposed inguinal (hip) region (see Figure 5). The exposed area is simulated with SOFA and contains a behavior mesh (1915 tetrahedra) with fixed constraints under the blanket and a skin surface (2455 vertices, 4770 triangles) used for visual rendering and collision detection. A second optimized collision surface (763 vertices, 1435 triangles) is used for tissue-dragging during palpation. Internal structures are mapped to the behavior mesh and are used for multi-surface haptic rendering in palpation and needle penetration: the fascia lata which encloses the thigh musculature (1756 vertices, 3297 triangles), the femoral vein (1946 vertices, 3813 triangles) and femoral artery (1706 vertices, 3283 triangles), ligament (825 vertices, 1626 triangles), and parts of the hip (1929 vertices, 3771 triangles) and femur bone (1558 vertices, 3095 triangles).

4.4. Interaction Scenarios

A controlled environment with reproducible interaction scenarios is useful for effective benchmarks of the system and to compare performance between platforms. To achieve this goal we implemented a lightweight record and playback system. For recording it dumps the driver's sensor states during each driver update into one binary file for each device and performs the reverse procedure for playback. With this system we have recorded the following interaction scenarios:

S1) In a baseline scenario the virtual hand and instrument are just idling over the virtual patient without any collisions ($t_\Sigma = 35$ seconds).
S2) This is a palpation only scenario with some lateral movement over the skin and light pressure to find pulse sources ($t_\Sigma = 38$ seconds).
S3) This scenario features needle penetration with several angular alignments and repositioning of the needle without any palpation ($t_\Sigma = 38$ seconds).
S4) In this scenario we recorded the beginning of a medical procedure for the femoral block performed by an expert with palpation and needle insertion ($t_\Sigma = 35$ seconds).

Figure 7 and Figure 8 show the changes in force magnitude over the time-span of an interaction scenario. These magnitudes represent the length of the force vectors computed by the force rendering algorithms for each haptic device.

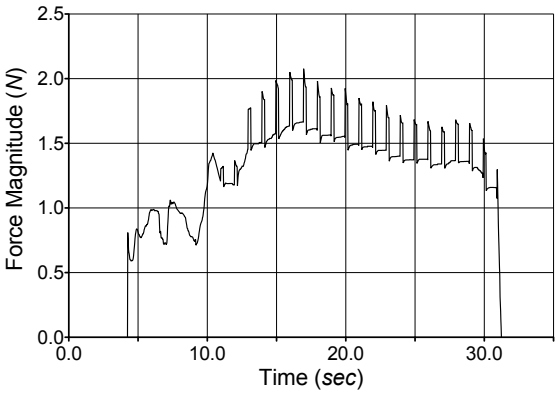

Figure 7: *Recordings of the force magnitude during palpation in interaction scenario S4 (Section 4.4). The periodic pattern from $t = 13$ to $t = 30$ depicts the simulated femoral pulse.*

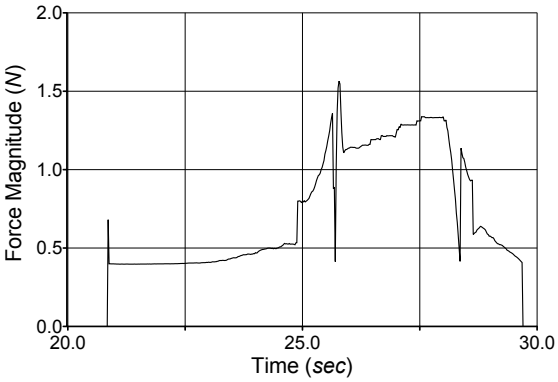

Figure 8: *Recordings of the force magnitude during needle insertion in interaction scenario S4 (Section 4.4). The force spikes and drops depict the penetration of tissue layers, e.g., skin or connective tissue.*

5. Results & Discussion

5.1. Performance Measures

To measure the performance, we manually instrumented the source code at crucial function calls. We used the *QueryPerformanceCounter* function (http://msdn.microsoft.com/en-us/library/ms644904) to get submicrosecond precision on Windows systems. The average execution time for the timer itself was 0.0229 μs on the stationary platform and 1.890 μs on the mobile platform. The timing data has been corrected at runtime accordingly. We have measured the performance for all four interaction scenarios on both hardware platforms with multiple repetitions. Table 1 as well as Figure 9 and Figure 10 summarizes the most essential performance measurements. Comparisons of S2 (palpation only) and S3 (needle only) have been omitted because their results have been very similar to S1 for the idle hand and very similar to S4 for the active hand respectively. We measured the average cumulative execution times for one haptic update frame with a dynamic number of force algorithms (up to six) with 3.76 ± 0.45 μs for the needle haptic device and 2.88 ± 1.19 μs for the palpation haptic device. End-to-end system latency for head-tracking on the stationary platform was on average 80 *ms*, as measured with the method described in [Ste08].

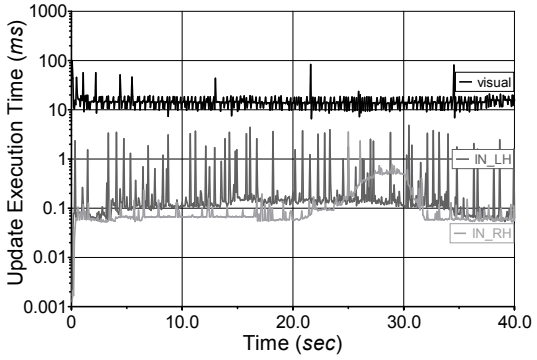

Figure 9: *Performance on the stationary platform of visual loop compared with interaction loop, including CD calls, of left palpation hand (IN_LH) and right needle hand (IN_RH) in interaction scenario S4. The maxima for IN_RH between $t = 25$ and $t = 30$ can be explained by deep penetration with an increasing number of subsurface objects and matches the events recorded in Figure 8.*

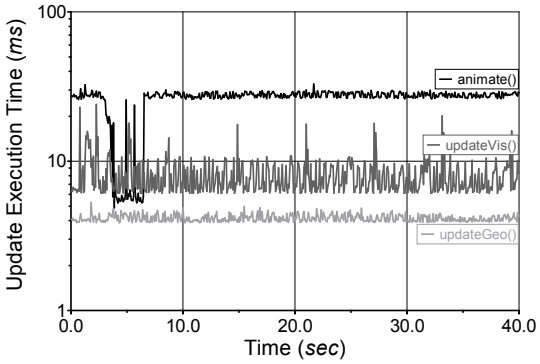

Figure 10: *Performance on the stationary platform of simulation loop divided into tissue simulation* animate() *(mean 23.7 ms), update of the mappings* updateVis() *(mean 7.7 ms) and update of collision detection geometries* updateGeo() *(mean 4.3 ms) in interaction scenario S4.*

These results show that interactive frame rates and stable haptic rendering can be achieved. Even on the less powerful mobile system with only two cores, the system is still running at reasonable speed. Spikes in Figure 9 and Figure 10 can be explained by threads waiting for a mutex. However, even the spikes are within acceptable ranges for interactivity.

5.2. Complexity of Tissue Simulation

To assess the computational complexity of the physics-based soft tissue simulation, we compared its performance between different behavior meshes. The lowest resolution was 158 tetrahedra, then 974, 1915 (which was the optimized default during the other tests), 2707, 5605, 7969 and the highest resolution with 14540 tetrahedra. This meshes have been tested with three continuous iterations of scenario S4 on both platforms (see Figure 11). A linear relationship of the mesh complexity and simulation time was found.

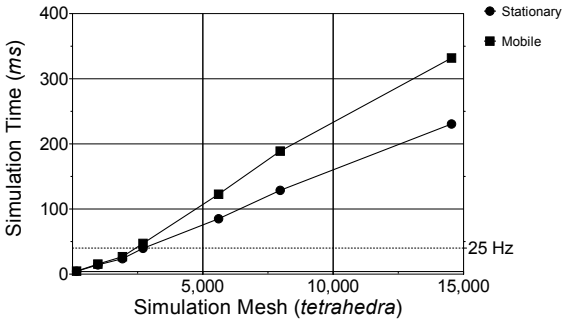

Figure 11: *Comparison of average simulation time for the* animate() *time step of the SOFA simulation between both platforms with different behavior mesh resolutions for scenario S4.*

Table 1: *Comparison of average computation times in microseconds between interaction scenarios and platforms per object (CD: query time for one object, PS: proximity search), in total for one frame (IN: total interaction handling time) and the visual frame rate (FPS) in Hz (LH: left hand/palpation, RH: right hand/needle).*

	Stationary S1		*Stationary S4*		*Mobile S4*	
Measure	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*
CD_LH	37.0	165.9	45.0	196.3	55.8	227.3
CD_RH	5.2	45.4	17.6	67.9	37.3	166.9
PS_LH	—	—	11.1	15.5	16.5	43.1
PS_RH	—	—	7.8	21.9	46.3	173.4
IN_LH	277.4	711.3	342.7	851.1	516.0	840.3
IN_RH	77.4	219.9	158.8	494.5	341.5	544.4
FPS (Hz)	63.3	24.1	74.6	23.4	61.2	5.0

6. Conclusion

In summary, real-time interaction was achieved with stable haptic rendering in a bimanual configuration in a highly dynamic environment with a physics-based simulation. The medical example application was developed in a user-centered design process, i.e., expert reviews were conducted regularly with anatomy specialists and experienced anesthetists. Additional information about the application and this simulator can be found at http://www.rasim.info. The core contribution of this paper is the detailed system description of a real-time capable haptic simulator coupled with a physics-based FEM simulation. The performance

measures indicate that even mobile systems can be successfully used for haptic real-time interaction. The system design can be abstracted and applied to similar simulators with other virtual reality toolkits and physics engines. The detailed performance measures can help to prioritize components during execution.

Future work will focus on dynamic update rates to saturate the given processing power and investigation of further system optimizations. Furthermore, the system is under evaluation with medical residents, consultants and attendings.

Acknowledgments: This work was funded by the German Research Foundation (project IDs: 19576470 and 80893132 at http://gepris.dfg.de).

References

[ACF*07] ALLARD J., COTIN S., FAURE F., BENSOUSSAN P.-J., POYER F., DURIEZ C., DELINGETTE H., GRISONI L.: SOFA – an Open Source Framework for Medical Simulation. In *Medicine Meets Virtual Reality* (2007), pp. 13–18. 2, 3

[AKO95] ADACHI Y., KUMANO T., OGINO K.: Intermediate representation for stiff virtual objects. In *Proceedings of the Virtual Reality Annual International Symposium* (1995), IEEE Press, pp. 203–210. 2

[Bal06] BALANIUK R.: A differential method for the haptic rendering of deformable objects. In *Symposium on Virtual reality software and technology* (2006), ACM Press, pp. 297–304. 2

[BPS05] BARBAGLI F., PRATTICHIZZO D., SALISBURY K.: A Multirate Approach to Haptic Interaction with Deformable Objects Single and Multipoint Contacts. *International Journal of Robotics Research 24*, 9 (September 2005), 703–715. 2

[BRWB07] BURNS E., RAZZAQUE S., WHITTON M. C., BROOKS F. P.: MACBETH: Management of Avatar Conflict by Employment of a Technique Hybrid. *The International Journal of Virtual Reality 6*, 2 (2007), 11–20. 4

[CBS06] CHEN P., BARNER K. E., STEINER K. V.: A Displacement Driven Real-Time Deformable Model For Haptic Surgery. In *International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (2006), IEEE Press, pp. 499–505. 2

[CGJC10] COLES T. R., GOULD D. A., JOHN N. W., CALDWELL D. G.: Virtual Femoral Palpation Simulation for Interventional Radiology Training (WiP Paper). In *EG UK Theory and Practice of Computer Graphics* (2010), pp. 123–126. 2

[CMJ10] COLES T., MEGLAN D., JOHN N.: The Role of Haptics in Medical Training Simulators: A Survey of the State-of-the-art. *IEEE Transactions on Haptics*, 99 (2010), 1. Early Access. 2

[CSB95] COLGATE J., STANLEY M., BROWN J.: Issues in the haptic display of tool use. In *International Conference on Intelligent Robots and Systems* (1995), vol. 3, pp. 140–145. 2

[CT00] CAVUSOGLU M. C., TENDICK F.: Multirate Simulation for High Fidelity Haptic Interaction with Deformable Objects in Virtual Environments. 2

[DS03] DIMAIO S. P., SALCUDEAN S. E.: Needle Insertion Modelling and Simulation. *IEEE Transactions on Robotics and Automation: Special Issue on Medical Robotics 19*, 5 (2003), 864–875. 4

[GAPD08] GURUSAMY K., AGGARWAL R., PALANIVELU L., DAVIDSON B. R.: Systematic review of randomized controlled trials on the effectiveness of virtual reality training for laparoscopic surgery. *British Journal Surgery 95*, 9 (Sep 2008), 1088–1097. 1

[GME*00] GREGORY A., MASCARENHAS A., EHMANN S., LIN M., MANOCHA D.: Six Degree-of-Freedom Haptic Display of Polygonal Models. In *the IEEE Visualization* (2000). 5

[GNU*09] GROTTKE O., NTOUBA A., ULLRICH S., LIAO W., FRIED E., PRESCHER A., DESERNO T. M., KUHLEN T., ROSSAINT R.: Virtual reality-based simulator for training in regional anaesthesia. *British Journal of Anaesthesia 103*, 4 (October 2009), 594–600. 1

[Gre04] GREEN S.: *GPU Gems*. Addison-Wesley, 2004, ch. Chapter 16. Real-Time Approximations to Subsurface Scattering, pp. 263–278. 3

[HSJ04] HUBONA G., SHIRAH G., JENNINGS D.: The effects of cast shadows and stereopsis on performing computer-generated spatial tasks. *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans 34*, 4 (July 2004), 483–493. 3

[LD07] LAYCOCK S., DAY A.: A Survey of Haptic Rendering Techniques. *Computer Graphics Forum 26*, 1 (March 2007), 50–65. 2

[LK09] LEDERMAN S. J., KLATZKY R. L.: Haptic perception: a tutorial. *Attenion, Perception, & Psychophysics 71*, 7 (Oct 2009), 1439–1459. 1, 2

[LO08] LIN M. C., OTADUY M.: *Haptic Rendering: Foundations, Algorithms and Applications*. A K Peters, 2008. 2

[LTCK03] LIU A., TENDICK F., CLEARY K., KAUFMANN C.: A survey of surgical simulation: applications, technology, and education. *Presence: Teleoper. Virtual Environ. 12*, 6 (2003), 599–614. 1

[Nap56] NAPIER J. R.: The prehensile movements of the human hand. *The Journal of Bone and Joint Surgery 38-B* (1956), 902–913. 4

[OSO04] OKAMURA A. M., SIMONE C., O'LEARY M. D.: Force modeling for needle insertion into soft tissue. *IEEE Trans. on Biomedical Engineering 51*, 10 (2004), 1707–1716. 5

[RKK97] RUSPINI D. C., KOLAROV K., KHATIB O.: The haptic display of complex graphical environments. In *SIGGRAPH '97* (1997), pp. 345–352. 4

[RSK09] ROMANO J. M., SAFONOVA A., KUCHENBECKER K. J.: Real-Time Graphic and Haptic Simulation of Deformable Tissue Puncture. In *Medicine Meets Virtual Reality* (2009). 2

[SCB04] SALISBURY K., CONTI F., BARBAGLI F.: Haptic rendering: introductory concepts. *IEEE Computer Graphics and Applications 24*, 2 (March 2004), 24–32. 2

[SDC08] SAUPIN G., DURIEZ C., COTIN S.: Contact Model for Haptic Medical Simulations. In *International Symposium on Computational Models for Biomedical Simulation (ISCMBS)* (2008), Springer, pp. 157–165. 2

[Ste08] STEED A.: A simple method for estimating the latency of interactive, real-time graphics simulations. In *VRST '08* (2008), ACM, pp. 123–129. 6

[UKK11] ULLRICH S., KNOTT T., KUHLEN T.: Dissecting in Silico: Towards a Taxonomy for Medical Simulators. *Studies in health technology and informatics 163* (February 2011), 677–679. 1

[UMN*08] ULLRICH S., MENDOZA J., NTOUBA A., ROSSAINT R., KUHLEN T.: Haptic Pulse Simulation for Virtual Palpation. In *Proceedings Bildverarbeitung für die Medizin 2008* (Berlin, Germany, April 2008), Springer Verlag, pp. 187–191. 4