USING PHYSX FOR SIMULATION-BASED ENDOSCOPIC HARDWARE DESIGN

Felix Dingeldey, Karsten Isaković and Ilja Teiche
Fraunhofer Institute for Computer Architecture and Software Technology (FIRST)
Kekuléstraße 7, 12489 Berlin, Germany

Keywords: Surgical simulation, Simulation-based hardware design, PhysX, Physics engine integration.

Abstract: Computer-assistance becomes increasingly important in minimally invasive surgery. Automation and image-processing techniques are employed for assisting surgeons with their highly skilled tasks. Our research is aimed at developing a novel system for laparoscopic surgery consisting of a new type of endoscope and augmented reality components. In order to facilitate the design of the hardware and the algorithms, we developed a virtual endoscopy simulator. It utilizes the capabilities of NVIDIA's physics engine "PhysX" for simulating the physical behavior of soft tissue, instruments, and smoke. In this paper, we present our proposed solutions for modeling the objects using PhysX and discuss possible problems specific to the domain of medical simulation.

1 INTRODUCTION

Over the last decades, minimally invasive surgery (MIS) has become more and more important, as it requires smaller incisions and causes less pain to the patient than open surgeries. However, in MIS surgeons have to deal with limited orientation and difficult navigation. The 2D camera image of the endoscope provides the only visual feedback. Additionally, instead of being able to use their hands, surgeons have to use long instruments that only give indirect sensation of the movements and contacts.

The advances in medical imaging and computer technology let computer-assisted surgery (CAS) find its way into the operating rooms. Especially for unexperienced surgeons, CAS has the potential to support the navigation in novel manners by utilizing techniques from image processing, computer vision, and virtual or augmented reality (VR/AR).

The "Endoguide" project (Endoguide, 2010) aims to develop a novel computer-assisted system for laparoscopic surgery, consisting of two major parts: a new type of endoscope with variable viewing direction and a processing unit for offering VR/AR support and intuitive user input paradigms. The CAS unit will be able to automatically capture and stitch panoramic overview scans that can be augmented with additional information, such as the current viewing rectangle or the area already inspected. By tracking the position and orientation of all instruments and the endoscope, the system will allow augmenting the camera output with navigation aids, such as indicators that simplify the localization of the instruments with the camera. In addition, the tracking data will allow to overlay information from patient-specific imaging data, such as segmented organs, CT/MRI slices, or annotations that have been added during pre-surgical planning.

We developed a virtual simulator in order to simplify the development of both the new endoscope hardware and VR/AR algorithms. The simulator supports representative laparoscopic surgery scenarios, such as navigating in the abdomen or lifting up tissue. The simulation lets us evaluate hardware concepts before actually realizing, and lets us assess the robustness, accuracy, and speed of the algorithms being developed.

The use of simulated camera output and tracking data allows to start designing the algorithms very early, even before data from the real physical system is available. For detecting possible instabilities of the image processing due to poor visibility or variations and movements in the scene, the simulator has to provide a high degree of visual realism and, additionally, model the physical behavior of scene objects, such as instruments, soft tissue, or smoke.

In order to maintain interactive frame rates, we use the PhysX real time physics engine (Corp., 2009a) and integrate it into our rendering framework. PhysX

Dingeldey F., Isaković K. and Teiche I. (2010).
USING PHYSX FOR SIMULATION-BASED ENDOSCOPIC HARDWARE DESIGN.
In *Proceedings of the International Conference on Computer Graphics Theory and Applications*, pages 358-363
DOI: 10.5220/0002840103580363

is a production-ready and well-established physics engine that has been used in various video games, such as “Batman: Arkham Asylum” or“Unreal Tournament 3” (Corp., 2009b). In fact, the Microsoft Robotics Development Studio also uses PhysX for physics simulations (Morgan, 2008).

After reviewing related work in section 2 and giving a very brief overview of laparoscopic interventions in section 3, we discuss how to model the different aspects of the laparoscopic simulation using PhysX in sections 4 and 5. We show results in section 6 before drawing a conclusion and giving directions for future work in section 7.

2 RELATED WORK

Numerous highly specialized and advanced commercial systems are available for practicing various minimally invasive interventions, e.g. (Immersion, 2009) (Simbionix, 2009). (Çakmak et al., 2005) showed how to use their virtual trainer for designing and testing instruments, such as graspers. Training systems generally reach a very high degree of visual and physical realism. However, as proprietary solutions they are not suitable for our purposes. (Reichenbach, 2009) uses PhysX for simulating and testing the design of humanoid robots, particularly the rigid body mechanics. In addition, (Reichenbach, 2009) discusses how to combine real and virtual sensors by communicating sensor feedback between the actual robot and the simulator. (Rieffel et al., 2009) demonstrate how to utilize PhysX for simulating soft-bodied robot designs and gaits — where the high deformability usually requires computationally very complex models — by empirically determining behavioral parameters for the soft body system in PhysX.

In the field of medical simulation, (Ermisoglu et al., 2009) present a first study for using PhysX to practice the scooping procedure during cervical disc replacement surgeries. A thorough discussion of how to utilize PhysX for a virtual laparoscopic training simulator with haptic feedback is given by (Maciel et al., 2009). One focus of their work is how to handle the different update rates required for the haptic feedback ($\sim$1 kHz) and those achieved with PhysX ($\sim$20 Hz in their implementation).

(Ott et al., 2007) use PhysX’s rigid body engine in conjunction with an advanced VR haptic workstation with two data gloves for manipulating physically animated objects. The haptic feedback is computed from the deviation of the tracking data of the gloves and the simulated position of the PhysX rigid body actors that model the hands. Deviations occur if the actors are blocked by other virtual objects in the simulated scene.

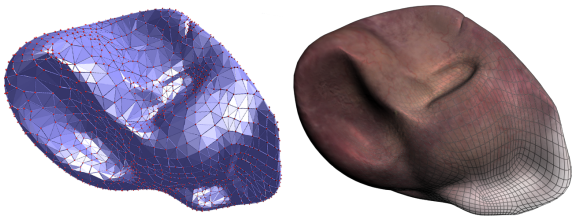

Figure 1: Example of a tetrahedral mesh used in PhysX (left) and its graphical representation (right).

3 BRIEF OVERVIEW OF LAPAROSCOPIC INTERVENTIONS

At the beginning of a minimally invasive laparoscopic intervention, several small incisions are made for inserting trocars (basically tubes that seal the cut and allow inserting instruments or the endoscope). Afterwards, the abdomen is insufflated using CO_2 in order to create a working space between the abdominal wall and the organs. Next, the instruments and the endoscope with the camera are inserted through the trocars.

During the intervention, the surgeon might have to perform various tasks using different instruments, such as: grasping, lifting, and cauterizing tissue; clipping arteries; sucking out blood; or suturing lesions using curved needles.

4 SIMULATION OF ORGANS AND TISSUE

We simulate all organs and soft tissue using soft bodies, which are simulated on GPU by PhysX. The engine uses a volumetric model for describing the deformation of soft bodies. The edge constraints between mass vertices are organized in tetraheda. For detecting collisions, a collision sphere is placed around each vertex. The radius of the spheres (the *particle radius*) can be specified per soft body object. Figure 1 shows an example of a tetrahedral soft body mesh and the corresponding graphical mesh.

After each simulation step, we update the positions of the graphical vertices according to the current deformation computed by PhysX. For this, each vertex of the graphics mesh is “linked” to the enclosing (or closest) tetrahedron. The barycentric coordinates of the vertex within the tetrahedron define the

influence (i.e. the weight) of each of the four tetrahedron's mass vertices on the graphical vertex. The update can be formulated as: $v' = \sum\_i = 0^3 b\_i \cdot t\_i$, where v' denotes the graphical vertex being updated, b_i the barycentric coordinate (weight), and t_i the position of the i'th tetrahedron's mass vertex of the soft body. In order to increase the rendering performance, we update the normals only if the total displacement has been larger than a user-definable threshold.

Our 3D model of the human abdomen consists of all important inner organs, bones and muscles. In reality, most abdominal organs are embedded by the peritoneum (a thin membrane) and connected to the abdominal wall. In the simulation, however, each organ is represented by a separate soft body and, hence, the embedding is disregarded. In consequence, the soft bodies would fall down once the simulation loop starts. Thus, we attach some of the mass vertices at the back of the organs at their initial position in world space. The attachments efficiently keep the soft bodies at their original position, but still allow deformations of the organs.

4.1 Problems and Drawbacks

A major problem when using PhysX is the lack of *two-way* soft body interaction. In the current release (version 2.8.1), collision detection between two separate soft body objects is not supported. However, in our simulator we need to model different organs that are likely to collide with each other (e.g. the liver and the gall bladder).

A common approach to tackle this is to introduce rigid body actors that are placed and attached within the soft body, as it is also shown in the PhysX sample applications. During the simulation, each soft body then collides with the rigid bodies attached to other soft body objects. However, for the complex (concave) geometry of the organs many small rigid bodies would need to be attached, which makes the parameterization extremely challenging if not impossible, and results in a highly instable simulation.

Our solution to this problem relies on the self-collision supported by PhysX (i.e. the collision detection between the object's own particles) and the fact that soft body objects may consist of totally disjoint groups of tetrahedra. In the simulator, we merge neighboring organs into a single soft body and enable self-collision. Although the merged organs then have the same parameterization, we found this solution to work very well for our purposes.

Another problem arises from the fact that collision detection is only performed at the soft body particles and not at the outer faces of the tetraheda. If the particle radius is too small, thin soft body structures tend to cut or fall through themselves. If the particle radius is quite large, overlapping structures seem to hover on top of each other, since the mass vertices cannot touch.

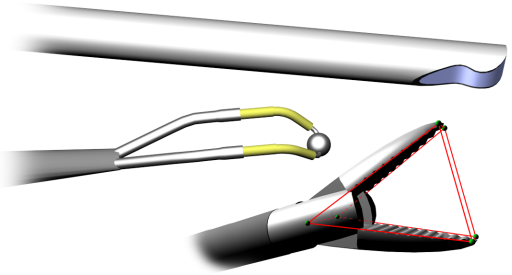

Figure 2: The tips of the instruments available in our simulator. From top to bottom: The endoscope, the cauterizer, and the grasper. The red lines depict the virtual wedge we use for detecting soft body vertices between the two jaws of the grasper.

For us, these effects have been of particular importance when modeling the small intestine. The lack of continuous collision detection forced us to partly connect the tetraedra of thin touching structures.

5 SIMULATION OF INSTRUMENTS

Our research is motivated by developing a new type of computer-assisted endoscopy system. At the current stage, the simulator supports two kinds of instruments: A grasper for lifting tissue and a cauterizer for burning tissue. These two actions (lifting and burning) significantly alter the view and are, therefore, very suitable for assessing our image processing algorithms (e.g. autofocus control or instrument tracking).

Figure 2 shows the graphical meshes of the two instruments and of an endoscope prototype being evaluated. The physical simulation relies on kinematic rigid bodies, whose positions are set explicitly according to the user input.

As discussed in section 2, (Ott et al., 2007) use a pair of kinematic and dynamic actors for calculating haptic feedback based on the deviation of the actors. We plan to utilize this method for calculating haptic feedback for our instruments by connecting additional dynamic actors to the instruments' kinematic actors.

5.1 The Endoscope

The geometry of the endoscope consists of a cylindrical shaft with a curved cut-out at the tip, at which a ro-

tating prism is located (see figure 2, top). For the endoscope, the graphical simulation is much more complex than the physical simulation. In order to be able to evaluate the concept of the rotating prism camera optics, we need to model the camera behavior as accurate as possible. For example, we model the complex movement of the view frustum caused by the internal reflections in the prism, and apply radial lens distortion, depth-of-field, and chip noise as post-processing effects.

5.2 The Grasper

Our grasper is composed of a shaft and two jaws that open and close (figure 2, bottom). In PhysX, the instrument is represented by a capsule for the shaft and two boxes for the jaws. When opening or closing the grasper, we request PhysX to rotate the boxes of the jaws around their base at the tip of the shaft.

Pinching can be realized by attaching soft body vertices that lie between the jaws. Unfortunately, PhysX does not report contacts between rigid bodies and soft bodies, which made it necessary to implement the pinching by hand.

Finding soft body vertices between the jaws could be implemented by casting rays from the jaws against the soft bodies. However, it turned out that the ray casting routine in PhysX is computationally too expensive and significantly drops the frame rate. As shown in figure 2 we span a virtual wedge between the jaws. This wedge defines five planes with which we basically perform a view-frustum culling operation in order to find all vertices inside the wedge. Once these vertices are detected, they are attached to the grasper and follow it in the continuing simulation.

For the grasping, we attach the vertices as *tearable*, which allows the attachment to break as soon as the force acting on it becomes large enough. In order to model forceful grasping when the jaws are almost closed, the tear factor depends on the opening angle of the jaws. Once the user opens the grasper, we detach all attached vertices and thereby release the soft body from the grasper.

5.3 The Cauterizer

The geometry of the cauterizer consists of a tip with a loop at which a sphere is attached with which the surgeon can cauterize (figure 2, middle). When pushing the tip against an organ, the tissue is burned and smoke rises. In PhysX, smoke can be modeled using its GPU-based fluid simulation, which implements particle-based smoothed-particle hydrodynamics (SPH) (Müller et al., 2003). During rendering, each particle can be rendered as textured billboard, using the positions from the fluid simulation.

In our simulator, we attach a PhysX fluid emitter at the tip of the cauterizer. As soon as the user activates the cauterizer, we start to check if any soft body vertex is in immediate proximity of the tip, in which case we slowly start emitting particles and let the emitter's flow rate increase constantly from near-zero to a maximum. If no soft body vertex is close to the cauterizer tip, we switch off the emitter again. Additionally, we define a fade-out time in order to avoid sudden disappearance of smoke particles.

6 RESULTS

We implemented our simulator in C++ using DirectX for rendering and input handling and PhysX version 2.8.1 for the physical simulation. Figure 3 shows a screenshot of the simulator, which consists of two windows. On the left, the control window renders with an overview camera. On the right, the output of the simulated endoscope camera is presented. The position and orientation of this camera is controlled when moving the endoscope geometry. As can be seen, the simulation takes into account various properties of the camera, such as field of view, lens distortion, chip noise, or the image circle that does not cover the entire chip depending on the zooming level. For testing our image-processing algorithms, we direct the simulation window onto the secondary monitor port and grab the video on a second computer, as it would be done with the video stream coming from the actual endoscope camera.

Our testing platform uses one core of an Intel Core2 Duo CPU at 3 GHz and 3.2 GB main memory. For rendering and the PhysX simulation, an NVIDIA GeForce 9600 GT graphics card with 512 MB RAM is used. For a scene with roughly 540,000 polygons and 4,300 tetraheda, the application renders at about 40 frames per second. The physics simulation runs at about 20 frames per second. Note that we run PhysX asynchronously for better performance.

As shown in figure 4, our grasper allows to lift organs and inspect the underlying areas. Depending on the placement of the camera, this can significantly alter the image. In the same way, the implemented cauterizer and smoke (see figure 5) also helps testing the robustness of image-processing algorithms and their reaction to fast-changing views.

We have already been able to evaluate different aspects of the endoscope prototype with our simulator. For example, we found that the rotation of the endopscope camera has to use at least eight steps, and

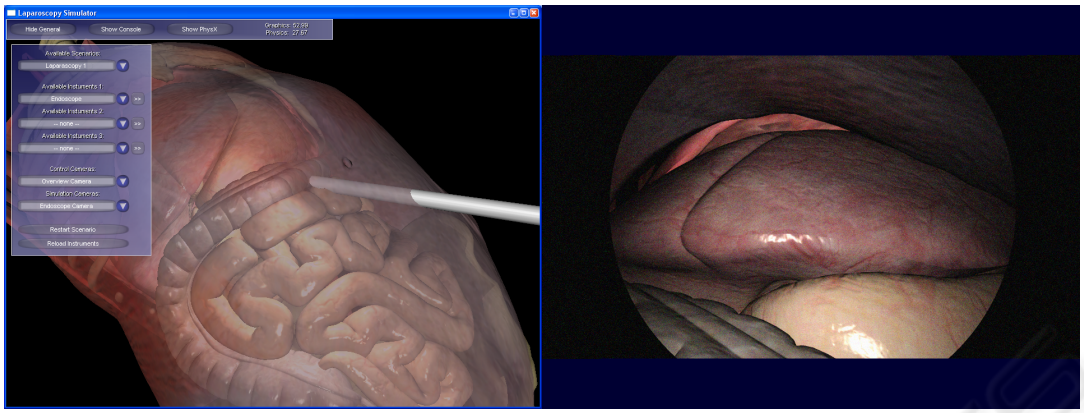

Figure 3: Screenshot of our simulator showing the control window (left) and the simulated output of the endoscope camera (right).

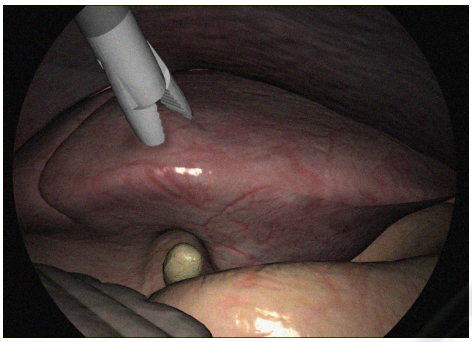

Figure 4: Screenshot of our simulator demonstrating the use of the grasper for lifting organs. The camera view corresponds to figure 3 (right).

should ideally be continuous. Otherwise, the surgeon might lose orientation. In contrast, for zooming it is sufficient to provide only a few levels, which will simplify the design of the electric motors.

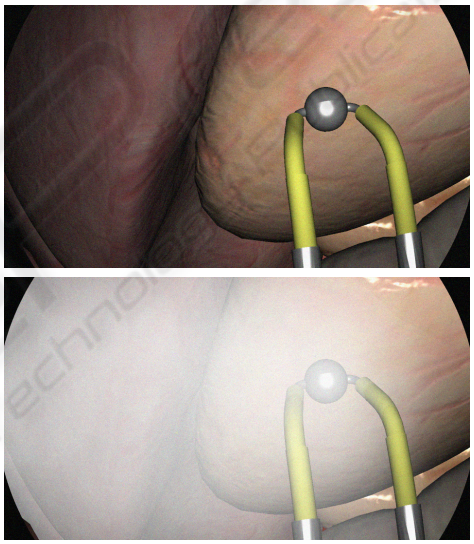

Figure 5: Screenshot of our simulator demonstrating the use of the cauterizer and the rising smoke.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we showed how to use NVIDIA's physics engine PhysX for realizing a laparoscopic simulator that we use for evaluating hardware concepts and algorithms for a novel computer-assisted endoscopy system. We utilize the real time capabilities of PhysX for simulating soft bodies (organs), rigid bodies (instruments and the endoscope), and fluids (rising smoke).

The major drawback of the current release of PhysX (version 2.8.1) is the lack of both collision detection between different soft body objects (two-way interaction) and collision detection on outer faces of the soft bodies. In consequence, we were forced to implement several workarounds. For our purposes of evaluating the endoscope hardware, the entailed reduction of physical plausibility is not very critical. For a medical training simulator, however, we assume the current limitations to be much more important.

In the future, we will evaluate alternatives to PhysX, such as "Bullet Physics" (Coumans, 2009) or the "SOFA" framework (Allard et al., 2007). Furthermore, we want to improve our simulator and increase its realism by including other aspects of laparoscopic interventions, such as cutting tissue. We also plan to model bleedings caused by the cutting using SPH fluids, as proposed by (van der Laan et al., 2009). Besides this, we will integrate haptic feedback into our application that applies forces on the input device when pushing against objects in the scene.

ACKNOWLEDGEMENTS

The "Endoguide" project is funded by the Federal Ministry of Education and Research, Germany (promotion reference 01IM08005D).

REFERENCES

Allard, J., Cotin, S., Faure, F., Bensoussan, P.-J., Poyer, F., Duriez, C., Delingette, H., and Grisoni, L. (2007). SOFA – an open source framework for medical simulation. In *Medicine Meets Virtual Reality (MMVR'15)*, Long Beach, USA.

Çakmak, H. K., Maaß, H., and Kühnapfel, U. (2005). VSOne, a virtual reality simulator for laparoscopic surgery. *Minimally Invasive Therapy and Allied Technologies*, 14(3):134–144.

Corp., N. (2009a). Physx. Retrieved 10/29/2009, from http://www.nvidia.com/object/physx_new.html.

Corp., N. (2009b). Physx games list. Retrieved 10/29/2009, from http://www.nzone.com/object/nzone_physxgames_home.html.

Coumans, E. (2009). *Physics Simulation Forum - View topic - Bullet 2.75 beta1: GPU, SPH fluids preview, new constraints*. Retrieved 11/15/2009, from http://bulletphysics.org/Bullet/phpBB3/viewtopic.php?f=18&t=3625&start=0.

Endoguide (2010). Endoguide. Retrieved 02/03/2010, from http://www.ia-vt.de/index.php?id=20.

Ermisoglu, E., Sen, F., Kockara, S., Halic, T., Bayrak, C., and Rowe, R. (2009). A scooping simulation framework for artificial cervical disk replacement surgery. In *Proceedings of SMC '09*. IEEE.

Immersion, C. (2009). Surgical simulator: The laparoscopyvr virtual-reality system. Retrieved 10/29/2009, from http://www.immersion.com/markets/medical/products/laparoscopy.

Maciel, A., Halic, T., Lu, Z., Nedel, L. P., and De, S. (2009). Using the physx engine for physics-based virtual surgery with force feedback. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 5:341–353.

Morgan, S. (2008). Simulating the world with Microsoft robotics studio. *MSDN Magazine*, 6.

Müller, M., Charypar, D., and Gross, M. (2003). Particle-based fluid simulation for interactive applications. In *Proceedings of SCA '03*, pages 154–159, Aire-la-Ville, Switzerland. Eurographics Association.

Ott, R., De Perrot, V., Thalmann, D., and Vexo, F. (2007). MHaptic: a haptic manipulation library for generic virtual environments. In *Proceedings of CW '07*, pages 338–345, Washington, DC, USA. IEEE.

Reichenbach, T. (2009). A dynamic simulator for humanoid robots. *Artificial Life and Robotics*, 13:561–565.

Rieffel, J., Saunders, F., Nadimpalli, S., Zhou, H., Hassoun, S., Rife, J., and Trimmer, B. (2009). Evolving soft robotic locomotion in PhysX. In *Proceedings of GECCO '09*, pages 2499–2504, New York, NY, USA. ACM.

Simbionix, L. (2009). Lap mentor laparoscopic surgery simulator for general surgery, gynecology and urology. Retrieved 10/29/2009, from http://www.simbionix.com/LAP_Mentor.html.

van der Laan, W. J., Green, S., and Sainz, M. (2009). Screen space fluid rendering with curvature flow. In *Proceedings of I3D '09*, pages 91–98, New York, NY, USA. ACM.