Ultrasound Image Simulation with GPU-based Ray Tracing

Yuen C. Law, Sebastian Ullrich, Thomas Knott, Torsten Kuhlen

Virtual Reality Group, RWTH Aachen University
Seffenter Weg 23
52074 Aachen
E-Mail: law@vr.rwth-aachen.de

Abstract: Medical simulators are gaining importance because the experience and skills necessary to perform many of the medical procedures are difficult to obtain due to patient safety and ethical issues. With the development of graphic cards, stereographic and haptic devices, more VR-based simulators are being created. We are developing an interactive ultrasound image simulation that include deformations and needle visualization as part of a training simulator for the application of regional anesthesia. In this paper, we present a new method for rendering the simulated images directly from 3D polygonal meshes, i.e., it does not use volume data, as presented in most of the previous works. This approach will allow us to apply and render deformations with common physics-based mesh solutions. To improve interactivity, we adapt a ray tracing approach using general programming GPU (GPGPU) methods, taking advantage of parallelism in modern graphic cards. We present the results of performance measurements conducted to test the scalability of our approach.

Keywords: Ultrasound simulation, Regional Anesthesia, GPU, Ray Tracing

1 Introduction

Ultrasound (US) guided needle procedures, such as in the application of regional anesthesia (RA), require training and experience. However, these are not easy to acquire due to various reasons, e.g., patient safety, lack of opportunities with real cases, time and resource limitations, and ethical reasons. Use of simulators for training address many of these issues by providing scenarios with different levels of complexity that trainees can use without the mentioned limitations [LTCK03]. To be effective, the simulation should be able to deliver feedback at interactive rates with the necessary degree of realism and detail. Furthermore, two additional elements need to be considered when developing simulators for US guided needle procedures: deformations and needle rendering. These are required to properly identify tissue structures and needle position and direction [CJB06]. Recent studies have shown that achieving real-time US image simulation is possible with the use of the parallelization capabilities of modern graphic cards and associated general programming platforms [KSN09, RPA+09].

Existing approaches to US simulation use either CT or US images as input to generate

volumes, which are then used to render the desired 2D planes [SHN08, VHGJ08, ZMRK07]. These approaches would show a clear disadvantage if we try adding deformations to the simulation. Volume representations could present a challenge when simulating complex deformations due to discretization errors, e.g., fine anatomical structures like nerve cords, vessels or fascial tissues might not be correctly represented or could increase mesh complexity too much. In this paper we present a simulation approach that uses polygonal meshes to directly render the simulated images using a modified ray-tracing algorithm to be used in an RA training virtual environment [UGF+09]. This method will allow us to directly use common physics-based mesh deformation techniques for our simulation, e.g., finite element or mass-spring methods. Furthermore, we will be able to apply adaptive sampling to increase mesh detail without significant effects on performance and add complex deformations taking into account different tissue properties, i.e, stiffness and density. We will be able to avoid rasterization problems that can occur when voxelizing thin structures, e.g. veins, arteries and nerves. The rest of the paper is structured as follows. In Section 2 we review the related work. Then, in Section 3, we describe the methods used to simulate ultrasound properties and artifacts, i.e., intensity, shadows, reflection and attenuation. Finally, in Sections 4 and 5 we discuss some results and give an overview on future work.

2 Related Work

Many studies have been made on real-time ultrasound simulation. To our knowledge, most of these approaches either use computed tomography (CT) or real ultrasound images as input data to create volume representations that are then used to simulate the images. For example, [SHN08] and [VHGJ08] present solutions based on CT volumes, while [ZMRK07] uses US images to create syntetic textures aligned with volume data.

An ultrasound simulation with deformations is presented in [GS07]. Here, the transformations to a deformed FEM (finite element method) tesselation are computed. This information is then used to find the position of the undeformed pixels and obtain the visualization from the volume representation. The authors claim that the procedure is fast enough to allow real-time, however, the mapping time is dependent on the size of the mesh and volume. Another approach is presented in [NCQ+11], where the volume data is mapped to a proxy geometry. This geometry is then deformed with a fragment program. The authors do not present specific results on performance, though interactive user studies were conducted with apparent positive feedback. In [ZMRK07], deformations due to needle insertion are applied to the volume using a mass-spring model on a localized mesh around the needle shaft. The reference mesh must be mapped in a preprocessing step in order to apply the deformations to the pixels accordingly. In [RWE08], the volume representation is deformed directly using a modified ChainMail algorithm. One major drawback of the method is that it only achieves interactivity with low grid resolutions, i.e, at a maximum of $32 \times 32 \times 32$. This can pose a problem when rendering thin structures, e.g., nerves, veins and arteries necessary in medical imaging. Furthermore, the above methods assume the same properties,

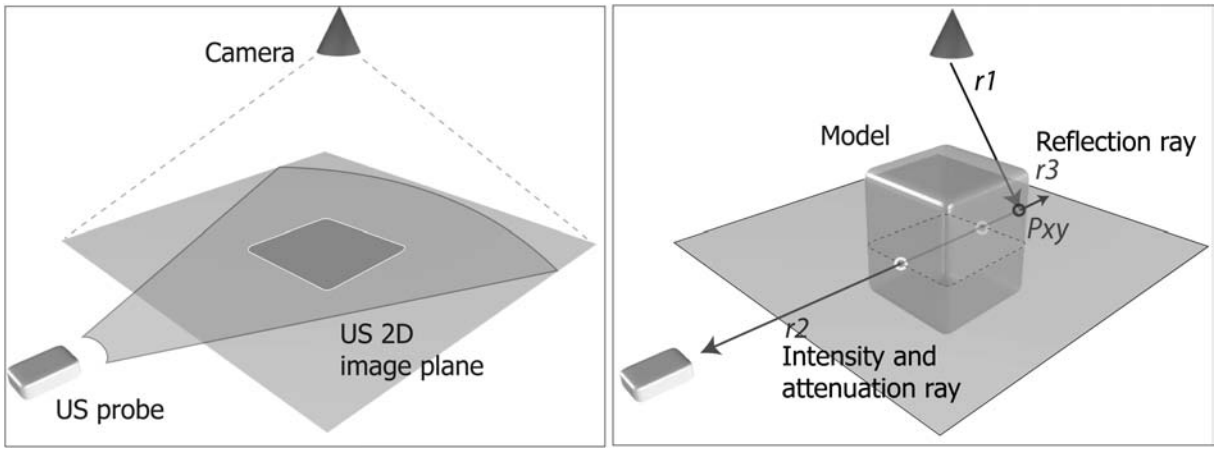

Figure 1: Left: general approach to obtain 2D US image. Right: Ray tracing method to obtain simulated US 2D plane. After intersection of *r1* with plane at *Pxy*, new rays *r2* and *r3* are created parallel to the plane.

i.e., stiffness and density, for all types of tissue, hence, more complex deformations at tissue interfaces are not simulated.

Recent approaches without deformations, such as the one presented in [KSN09] and [RPA+09], take advantage of GPU acceleration, using ray casting techniques. In both approaches, CT volume data is stored in 3D textures, which are then traversed by rays to obtain samples and create the desired image. Ultrasound physic phenomena are estimated based on the model presented in [WKC+07], with some modifications for parallel processing. The improvement of performance is clearly stated in both works. The work we present here also uses a ray-based technique on the GPU, but a key difference is that we avoid using volume data to represent body structures.

One interesting study has been recently presented in [KWN10]. The described method is also implemented on the GPU but uses a wave-based approach, as opposed to a ray-based one. The work shows promising results in terms of realism, but performance is yet to be improved in order to achieve interactivity.

To summarize, existing approaches use volume representations as input for their simulations. Most of them present realistic results but give little detail on performance. Solutions that include deformations use some kind of geometric representation to control them, assuming homogenous tissue properties and do not simulate more complex deformations at tissue interfaces. It is possible to achieve direct volume deformation at interactive times, but only with low resolution volumes, which can be a problem when rendering the thin structures required in medical imaging and training. Other authors have presented fast GPU implementations without deformations, again, using volume representations, potentially presenting the same problems as the CPU approaches when deformations are applied.

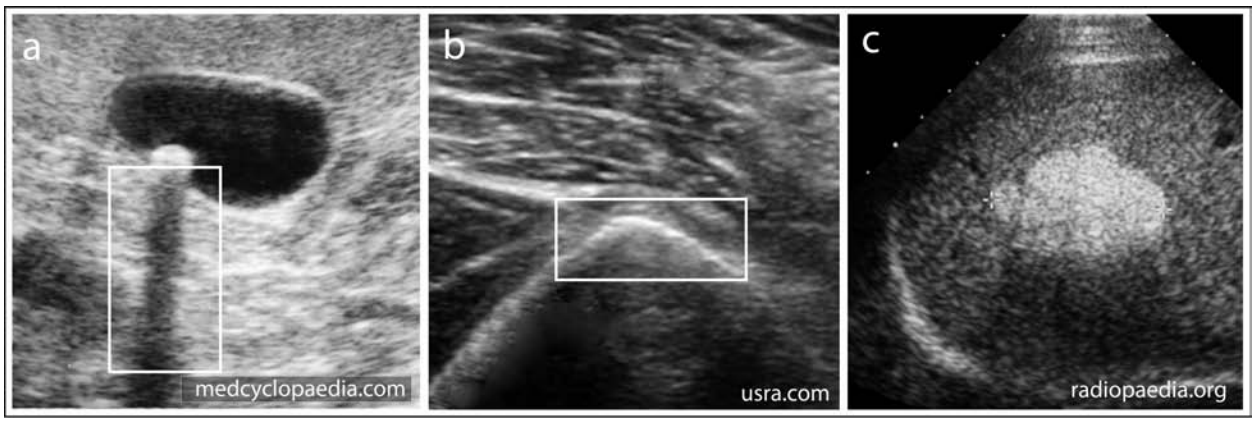

Figure 2: Sample physic phenomena in real US images. a: acoustic shadow. b: reflection due to bone-muscle interface. c: general attenuation.

3 Methods

The method we propose uses 3D polygonal meshes to render the simulated images directly, i.e., no volume data is required. Based on the position of the virtual probe, an according 2D plane is obtained. This surface is parallel to the direction of the probe (see Fig. 1, left). In reality, the ultrasound emitter and receiver are integrated in one device, and calculates pixel colors according to time differences between emission and reception of US. Here, we position the camera over the plane only to obtain pixel positions and should not be seen as the US receptor. The US probe acts, in this case, more like a light source (emitter) would, in a common ray tracing scene.

For each pixel, a ray (*r1*) is created from the viewing point (camera) to the 2D plane (see Fig. 1, right). This is only done for rendering purposes and do not take part in the simulation. From the intersection point (Pxy) on the plane, we create two new rays parallel to the plane, in the direction to the US transducer (*r2*) or opposite to it (*r3*). The rays estimate reflection, intensity and attenuation as described in the following sections. Once these values are obtained, they are combined to get the color of the corresponding pixel (Pxy). Based on the Wein model [WKC+07], we estimate physical phenomena along the rays according to material properties set to each object in the scene, i.e., acoustic impedance and absorption coefficient. Specific values of these properties are based on literature [SPL96, SBC+07].

3.1 Algorithm

As mentioned before, rays are created from the surface of the desired 2D plane to calculate the different physical phenomena that affect the ultrasound beams. Specifically, for each pixel, two rays are created: (*r2*) for the intensity transmission (I_t) and the attenuation (I_a) and (*r3*) for the reflection (I_r). The transmission and attenuation rays travel in the direction of the virtual transducer, calculating the loss of energy at each intersection in a back-to-front approach (see sections 3.2 and 3.3). The disminution of intensity produces acoustic shadows,

and a gradient through the tissues. These effects can be seen in (Fig. 2a) and (Fig. 2c).

Reflection occurs in the local area near the interface of two mediums with different acoustic impedances (see section 3.2), e.g., muscle and bone, as can be seen in (Fig. 2b). For this reason, we limit the length of the reflection ray and only calculate reflection if it reaches an intersection within this distance. The length of the ray determines the width of the reflection and can be adjusted to better match real reflections if necessary. Once the three values are obtained for the pixel, we use the following formula to combine them:

$$I_{total} = I_a \cdot I_t + I_r \tag{1}$$

A sample mesh of a vertebral bone is shown in (Fig. 3). Snapshot d) shows the combined results of the other three rendered images.

3.2 Reflection and Transmission

When an ultrasound beam passes through an interface of mediums with different acoustic impedances (measured in Ns/m^2), some of its intensity is reflected. The reflected intensity at the k-th interface I_r^k can be calculated as follows:

$$I_r^k = I_i^k \left(\frac{Z2 - Z1}{Z2 + Z1} \right)^2 \tag{2}$$

where I_i^k is the incoming intensity and $Z1$ and $Z2$ are the acoustic impedances of the current and next sampled tissues. Note that when $Z1$ and $Z2$ are equal, the formula results in 0 reflection. Therefore, calculating reflection is only necessary at medium interfaces, i.e., intersections with objects in the scene. Similarly, at each interface, the remaining energy $1 - I_r^k$ is transmitted further into the tissue and can be calculated directly as follows:

$$I_t^k = I_i^k \left(\frac{4 \cdot Z2 \cdot Z1}{Z2 + Z1} \right)^2 \tag{3}$$

Here, the incoming intensity I_i^k corresponds to the intensity of the prevoius sample I_t^{k-1}. Thus, it is recursively calculated. The initial incoming intensity I_i^0 is set to 1.

3.3 Attenuation

Attenuation of ultrasound occurs in two ways: scattering and absorption, and is dependant on the distance traveled and the frequency of the US wave. The formula for calculating the output intensity I_a^k due to attenuation of ultrasound is:

$$I_a^k = I_i^k e^{-\beta df} \tag{4}$$

where I_i^k is the incoming intensity, β is the attenuation coefficient of the medium in Np

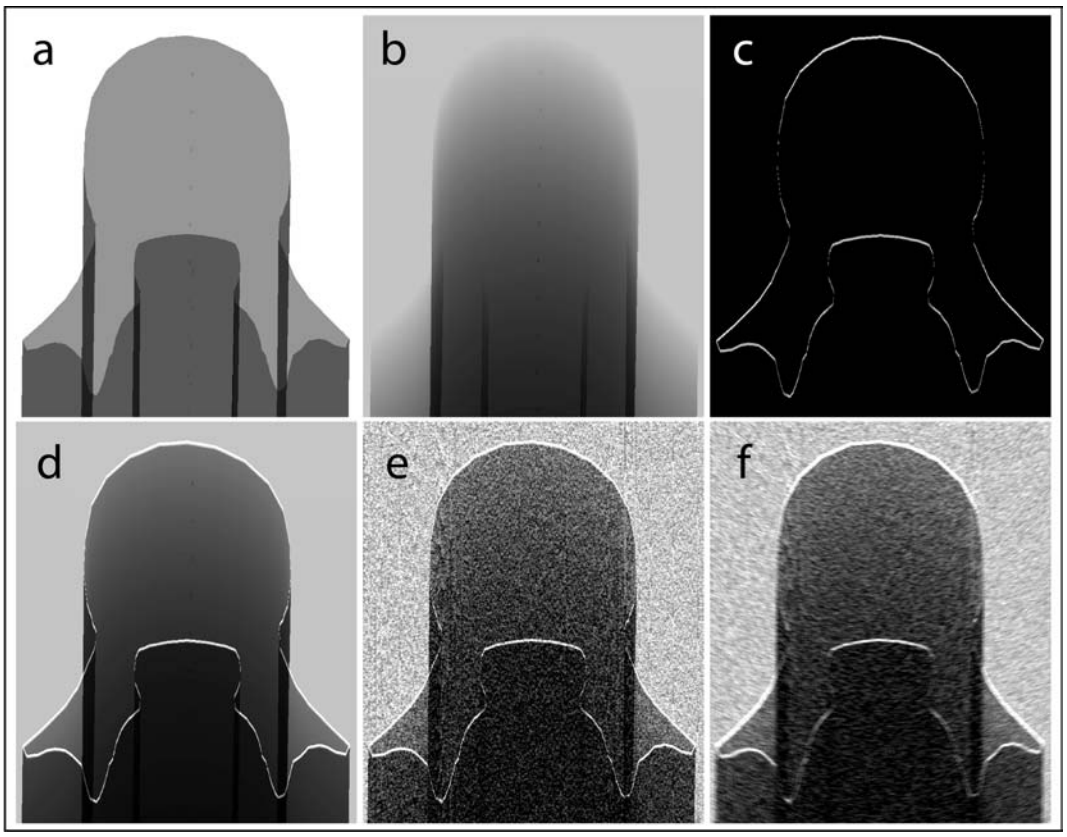

Figure 3: The different components for US image simulation, shown on a single object for better visibility of the effects: a) intensity image, b) absorption image, c) reflection image, d) combined result, e) with noise and f) with blur effect.

(Neper), d is the distance traveled in the tissue and f is the frequency of the ultrasound wave. Using the conversion formula from Np to dB (decibel):

$$\alpha = 20 \cdot log_{10}(e) \cdot \beta \tag{5}$$

This can be expressed in decibels as follows:

$$I_a^k = I_i^k \cdot 10^{-\alpha df/20} \tag{6}$$

where α is the new attenuation coefficient, now in dB.

3.4 Noise and Blurs

The sound beams in a ray-based model for ultrasound simulation behave, in many ways, similar to light rays. Sound and light beams are reflected, refracted and scattered by the medium through which they travel. This interaction creates the noise and blurs that are typical in US imagery. The noise, shown in (Fig. 3e), is added using a texture image,

randomly generated once when the application is initialized and stored in the graphics card. The radial blur effect in (Fig. 3f) is created by rotating each frame and combining it with previous ones using progressive rendering. The frames are combined as follows:

$$\left(\frac{1}{f}\cdot c\right)+\left(\frac{f-1}{f}\cdot c'\right) \tag{7}$$

where f is the number of the current rendered frame and c and c' are the values of the new color and the previously computed color. When moving the virtual probe, only the first frame is rendered. This technique allows faster rendering times while the user is interacting with the simulation.

4 Results

4.1 Performance Tests

To measure actual frame rendering times and estimate the scalabilty of our system, we performed four tests as follows. Our base scenario consists of a cube with bone properties in the middle of a muscle medium. This cube is composed of 12 triangles, the minimum possible. The used approach needs watertight meshes to render the images correctly.

The system prototype used for testing was implemented with OptiX, a general purpose ray tracing engine [PRS+10]. All tests were done rendering the image with a screen resolution of 512×384 pixels and measured the time in milliseconds to render one frame. For each scenario, we timed 1000 frames and computed an average. Tests were performed on a PC with an Intel Xeon E5540 (4 cores) 2.53 GHz processor with 12 Gb RAM and an Nvidia GeForce GTX480 graphics card with 1.5 Gb VRAM.

For tests 1) and 2), we changed the triangle count of the cube, dividing each of the faces several times to obtain 48, 192, 768, 3.072, 12.288, 49.152, 196.608 and 786.432-triangle cubes. An important scenario to evaluate is how updates of the acceleration structure would affect rendering performance (1d), since these updates will be done whenever the geometry changes due to deformations. Rendering times were obtained using each cube in the following scenarios:

1. Using a bounding volume hierarchy (BVH) acceleration structure with an axis-aligned bounding box (AABB) tree, included in the OptiX system.
    a. Complete US image simulation.
    b. US image simulation without computation of intensities, but creating all rays and detecting intersections.
    c. US image simulation without secondary rays (*r2* and *r3*).
    d. US image simulation forcing an update of the acceleration structure at every frame.

2. Without using acceleration structures.

    a. Complete US image simulation.

    b. US image simulation without computation of intensities, but creating all rays and detecting intersections.

For tests 3) and 4) we used only the 12-triangle cube, but increased the number of cubes in the scene. Test 3) measures how the number of intersections affect rendering times. We visually ordered the cubes in the direction of the ray, so that any ray originated behind the last cube would also intersect the remaining ones, which represents the worst case scenario. Furthermore, no optimization is done, i.e., the rays are not terminated after reaching a minimum intensity threshold, after which further calculations are unnecessary. Internally, all of the cubes are added to the same geometry and are seen as one object using the same material.

Test 4) measures how the number of geometries affect rendering times. This measurement is important because objects with different material properties must be separated into different geometries, which, as shown in the results, affect performance. The cubes were ordered in a way so that any ray in the scene would intersect, at most, only one cube. This is done to isolate the variable avoiding the addition of intersections to the rays' paths with every new cube.

4.2 Discussion

It can be observed from the results that, for scenearios 1.a), 1.b) and 1.c), the rendering times per frame do not show important increments as the number of triangles increases (Fig. 4 top). This shows that the level of detail of the simulated images could be improved by increasing the complexity of the geometry without greatly affecting rendering times.

For scenario 1.d), the time increases almost linearly after 768 triangles, and reaches 1 second for the 49.152-triangle cube (Fig. 4 bottom). Rebuild times of the acceleration structure could be possibly reduced with different techniques, e.g., selectively updating parts of the hierarchy over time [WBS06] or only when needed [LYMT06]. These modifications still need to be adapted for a GPU implementation.

Results of test 2), (Fig. 5), show the importance of the acceleration structure when using more complex geometries. A more efficient structure or traversal method should allow better performance in the simulation. A tradeoff between the time required to build the acceleration structure and the time to traverse it becomes necessary since usually, building structures for efficient traversal requires more computational time.

Results for tests 3) and 4), (Fig 6), show irregular increments of frame rendering times as the number of intersections in the scene increases, as well as with the number of separated geometries, but the tendence is almost linear. Although the effects on performance are not very dramatic, this information will be used to optimize the 3D models for performance.

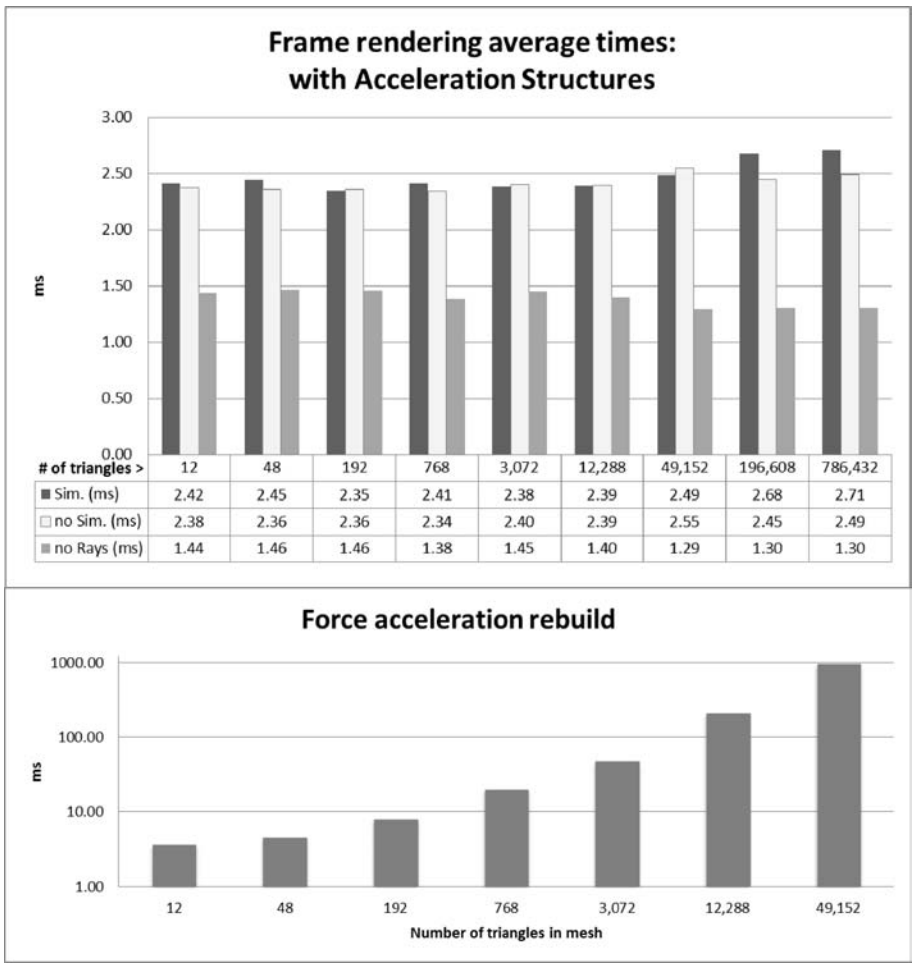

Figure 4: Top: rendering times for test 1), scenarios a, b and c. Bottom: rendering times for scenario 1.d. (log scale applied to ms axis)

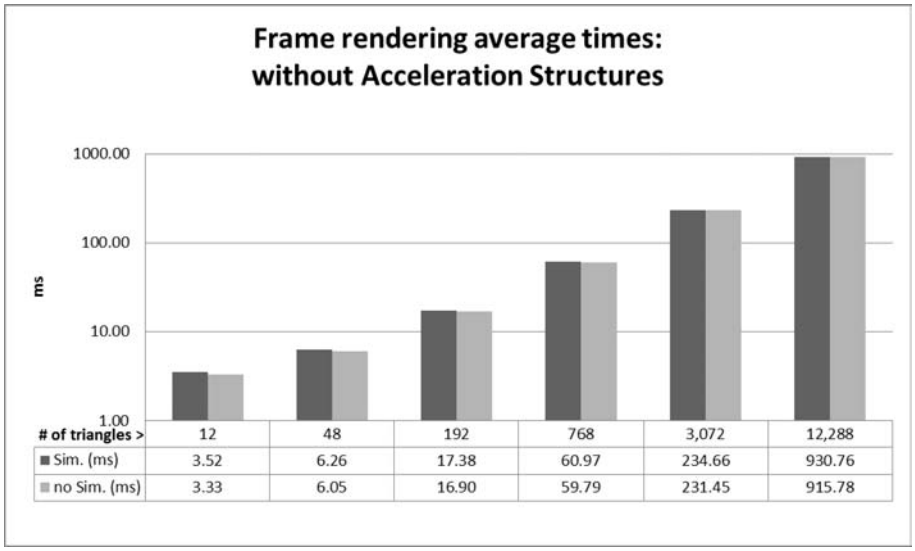

Figure 5: Rendering times for test 2), using no acceleration structures. (log scale applied to ms axis)

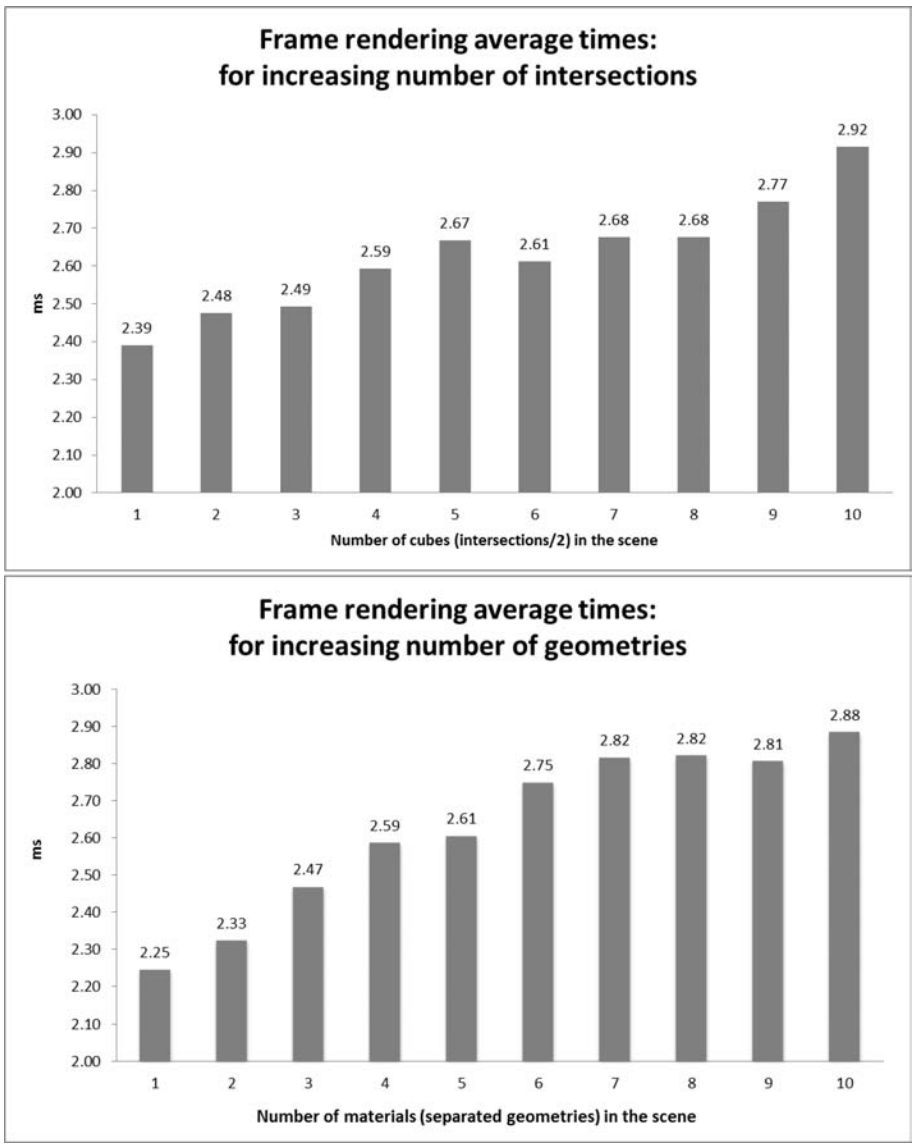

Figure 6: Rendering times for test 3) and 4).

5 Conclusions and Future Work

We present a method for simulating US images for training purposes implemented on the GPU, which in contrast to other similar solutions, does not use volume data. Instead, our ray-based algorithm uses 3D polygonal meshes directly to estimate ultrasound physic phenomena. This characteristic will allow us to incorporate mesh-based deformations, e.g., due to needle insertion, probing pressure or pulse. It is clear that the level of detail of the resulting images depends on the detail of the underlying meshes, but this is also true when using CT or US images. Furthermore, using meshes would allow us to adapt the sampling resolution, increasing it where needed, e.g., thin structures.

In future work, we will add deformations to the simulator, which is an important and necessary feature, since these are used to identify structures as well as to determine needle

position and direction. Additionaly, we will improve the detail of the anatomical models. Although desirable, comparing the quality of the images with previous works is difficult due to the subjectiveness of the measures, scarce examples and difference in the choice of simulated body regions. Therefore, the level of detail of the models, the physic properties, the US effects and the accuracy and quality of the resulting simulated images will then be evaluated with subject-matter experts. Finally, we expect to incorporate the simulation into a VR desktop training system and perform further studies.

References

[CJB06] G. A. Chapman, D. Johnson, and A. R. Bodenham. Visualisation of needle position using ultrasonography. *Anaesthesia*, 61(2):148–158, 2006.

[GS07] O. Goksel and S.E. Salcudean. Fast B-mode ultrasound image simulation of deformed tissue. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2007, pages 87–90, January 2007.

[KSN09] O. Kutter, R. Shams, and N. Navab. Visualization and GPU-accelerated simulation of medical ultrasound from CT images. *Computer methods and programs in biomedicine*, 94(3):250–66, June 2009.

[KWN10] A. Karamalis, W. Wein, and N. Navab. Fast ultrasound image simulation using the Westervelt equation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 13, pages 243–50, January 2010.

[LTCK03] A. Liu, F. Tendick, K. Cleary, and C. Kaufmann. A Survey of Surgical Simulation: Applications, Technology, and Education. *Presence: Teleoperators and Virtual Environments*, 12(6):599–614, December 2003.

[LYMT06] C. Lauterbach, S.-E. Yoon, D. Manocha, and D. Tuft. RT-DEFORM: Interactive Ray Tracing of Dynamic Scenes using BVHs. In *2006 IEEE Symposium on Interactive Ray Tracing*, pages 39–46. Ieee, September 2006.

[NCQ+11] D. Ni, W.Y. Chan, J. Qin, Y.-P. Chui, I. Qu, S.M. Ho, and P.-A. Heng. A Virtual Reality Simulator for Ultrasound-Guided Biopsy Training. *IEEE Computer Graphics and Applications*, 31(2):36–48, March 2011.

[PRS+10] S.G. Parker, A. Robison, M. Stich, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, and K. Morley. OptiX: A General Purpose Ray Tracing Engine. *ACM Transactions on Graphics*, 29(4):1, July 2010.

[RPA+09] T. Reichl, J. Passenger, O. Acosta, S. Riek, and O. Salvado. Ultrasound goes gpu: real-time simulation using cuda. In *SPIE Medical Imaging 2009*, volume 7261, pages 726116–1–10, Lake Buena Vista, Florida, USA, February 2009.

[RWE08] F. Roessler, T. Wolff, and T. Ertl. Direct GPU-based Volume Deformation. In *Proceedings of CURAC 2008*, pages 65–68, 2008.

[SBC+07] B.D. Sites, R. Brull, V.W.S. Chan, B.C. Spence, J. Gallagher, M.L. Beach, V.R. Sites, and G.S. Hartman. Artifacts and pitfall errors associated with ultrasound-guided regional anesthesia. Part I: understanding the basic principles of ultrasound physics and machine operations. *Regional anesthesia and pain medicine*, 32(5):412–8, 2007.

[SHN08] R. Shams, R. Hartley, and N. Navab. Real-time simulation of medical ultrasound from CT images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 11, pages 734–41, January 2008.

[SPL96] U. Schneider, E. Pedroni, and A. Lomax. The calibration of CT Hounsfield units for radiotherapy treatment planning. *Physics in medicine and biology*, 41(1):111–24, January 1996.

[UGF+09] S. Ullrich, O. Grottke, E. Fried, T. Frommen, W. Liao, R. Rossaint, T. Kuhlen, and T.M Deserno. An intersubject variable regional anesthesia simulator with a virtual patient architecture. *IJCARS*, 4(6):561–570, 2009.

[VHGJ08] F.P. Vidal, A.E. Healey, D.A. Gould, and N.W. John. Simulation of ultrasound guided needle puncture using patient specific data with 3D textures and volume haptics. *Computer Animation and Virtual Worlds*, 19(2):111–127, 2008.

[WBS06] I. Wald, S. Boulos, and P. Shirley. Ray tracing deformable scenes using bounding volume hierarchies. In *ACM Transactions on Graphics*, pages 1–10, 2006.

[WKC+07] W. Wein, A. Khamene, D.-A. Clevert, O. Kutter, and N. Navab. Simulation and fully automatic multimodal registration of medical ultrasound. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 10, pages 136–43, January 2007.

[ZMRK07] Y. Zhu, D. Magee, R. Ratnalingam, and D. Kessel. A training system for ultrasound-guided needle insertion procedures. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 10, pages 566–74, January 2007.