Segmentation and Visualization of 3D Medical Images through Haptic Rendering

Ingela Nyström*, Filip Malmberg, Erik Vidholm, and Ewert Bengtsson

Centre for Image Analysis, Uppsala University, Box 337, SE-75105, Uppsala, Sweden

E-mail: {*ingela,filip,erik,ewert*}*@cb.uu.se*

Abstract: *High-dimensional and high-resolution image data is increasingly produced by modern medical imaging equipment. As a consequence, the need for efficient interactive tools for segmentation and visualization of these medical images is also increasing. Existing software include state-of-the-art algorithms, but in most cases the interaction part is limited to 2D mouse/keyboard, despite the tasks being highly 3D oriented. This project involves interactive medical image visualization and segmentation, where true 3D interaction is obtained with stereo graphics and haptic feedback. Well-known image segmentation algorithms, e.g., fast marching, fuzzy connectedness, deformable models, and live-wire, have been implemented in a framework allowing the user to interact with the algorithms and the volumetric data in an efficient manner. The data is visualized via multi-planar reformatting, surface rendering, and hardware-accelerated volume rendering. We present a case study where liver segmentation is performed in CT images with high accuracy and precision.*

Keywords: volume haptics, live-wire, deformable simplex mesh, fast marching, volume rendering

1 INTRODUCTION

Today imaging systems provide high quality images valuable in a number of medical applications, e.g., diagnostics, treatment planning, surgical planning, and surgical simulation. The images obtained with modern computed tomography (CT) and magnetic resonance (MR) devices are 3D or sometimes 4D and the resolution is high and steadily increasing. The result is a steady flow of high-dimensional image data to visualize, analyze, and interpret.

One of the most important tasks is segmentation, i.e., separation of structures from each other and from the background. Segmentation is needed for, e.g., shape analysis, volume and area measurements, and extraction of 3D models. Lack of contrast between different tissues and shape variability of organs make automatic segmentation hard. By using interactive segmentation [1], expert knowledge is used as additional input to the algorithms and thereby facilitates the task. Interactive segmentation can be divided into recognition and delineation [2]. Recognition is the task of roughly determining object location, while delineation consists of determining the exact extent of the object. Human users outperform computers in most recognition tasks, while computers often are better at delineation. A successful interactive method combines these abilities to minimize user interaction time, while maintaining user control to guarantee correctness of the result.

Examples of softwares for interactive medical image processing and visualization are 3D Slicer [3], MeVisLab [4], and ITK-SNAP [5]. These softwares are designed mainly for use on ordinary workstations with mouse/keyboard interaction, which may become a limitation for complex, highly 3D oriented tasks. An example where it is shown how true 3D interaction can improve segmentation is the LiverPlanner [6].

Our approach is to use haptic feedback and stereo graphics in order to obtain true 3D interaction, see Fig. 1. Haptic interaction provides the possibility of simultaneous exploration and manipulation of data by providing controlled force feedback to the user. Direct volume haptics [7, 8] has shown to be useful in volume exploration [9] and for interactive medical segmentation [10]. Our work has involved development and implementation of algorithms for interactive segmentation [11, 12, 13, 14], hardware accelerated volume visualization [15], and volume haptics [16, 17]. These implemementations have been collected in a toolkit called *WISH—interactive segmentation with haptics*.

This paper presents our haptics project. Section 2 introduces haptics and haptic rendering. Section 3 describes our hardware accelerated volume visualization engine. The developed image processing and interactive segmentation methods are described in Section 4. Our toolkit is presented in Section 5. A case study for interactive liver segmentation is given in Section 6. We summarize our work in Section 7.

Fig. 1: A SenseGraphics display with a PHANToM device. Stereo graphics is rendered onto a semi-transparent mirror to obtain co-localization of haptics and graphics.

*Corresponding author.

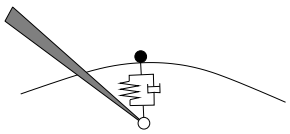

Fig. 2: Proxy-based surface haptics. The haptic probe (white) is connected to a virtual proxy (black) through a spring-damper.

2 HAPTIC INTERACTION

Haptic interaction with 3D objects is commonly performed with haptic devices that have one interaction point and three or six degrees of freedom (DOF). We use a PHANToM Omni device from Sensable[1]. The PHANToM is designed as a stylus, and the haptic feedback is given at the stylus tip, the *haptic probe*. This device has 6DOF for input and 3DOF for output, i.e., a position and an orientation for input, and a force vector for output. The device can be used with an ordinary workstation, but in our work we use a specialized haptic display from SenseGraphics[2] that allows for co-localized haptics and stereo graphics, see Fig. 1.

Haptic rendering algorithms should generate intuitive force feedback when the user moves the haptic probe so that it comes in contact with an object. Depending on the application, there are different object representations to interact with, e.g., explicit and implicit surfaces or, as in this work, volumetric data.

State-of-the art haptic rendering methods are constraint-based; the haptic probe is connected to a virtual *proxy point*. The proxy is controlled by the application and is constrained to certain movements, e.g., to stay on a surface. The connection between the probe and the proxy is made by a virtual coupling device consisting of a spring and a damper, see Fig. 2. The rendered force feedback is thereby proportional to the distance between the haptic probe and the proxy, i.e.,

$$\boldsymbol{f} = -k(\boldsymbol{x}-\boldsymbol{p}) - \gamma(\dot{\boldsymbol{x}}-\dot{\boldsymbol{p}}),$$

where $\boldsymbol{x}$ is the probe position, $\boldsymbol{p}$ the proxy position, k the stiffness of the spring-coupler, γ the damping coefficient, and $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{p}}$ the velocities of the probe and the proxy, respectively. This idea was first developed for surface haptics [18, 19] and later for volume haptics [7, 8].

In proxy-based volume haptics, the key is to choose an appropriate local reference frame (LRF) and generate constraints for proxy-movements in the LRF. Our current volume haptics implementation is based on the framework in [7]. Here, $\{\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_2\}$ denotes the LRF, $\boldsymbol{p}^t$ the proxy position at time step t, $\boldsymbol{x}^t$ the probe position, and $\boldsymbol{d} = (\boldsymbol{x}^t - \boldsymbol{p}^{t-1})$ the displacement of the probe relative to the previous proxy position. In each iteration, the proxy is moved in small steps according to certain *motion rules* for each axis in the LRF. The proxy position at time step t is computed as

$$\boldsymbol{p}^t = \boldsymbol{p}^{t-1} + \sum_{i=0}^{2} \Delta p_i \boldsymbol{e}_i,$$

[1]URL: *http://www.sensable.com/*
[2]URL: *http://www.sensegraphics.com/*

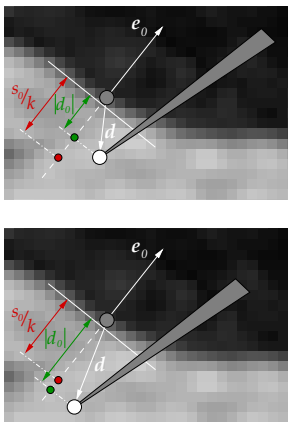

Fig. 3: Proxy-based volume haptics with a unilateral constraint for surface simulation. The gradient is used to compute the normal direction, i.e., $\boldsymbol{e}_0 = -\nabla f/||\nabla f||$. In order to move in the direction $-\boldsymbol{e}_0$, the user has to apply a force such that $|d_0| > s_0/k$. Top: $|d_0| < s_0/k$, which gives $\Delta p_0 = 0$, i.e., the proxy will not move. Bottom: $|d_0| > s_0/k$, which gives $\Delta p_0 = |d_0| - s_0/k$, i.e., the proxy will move $\Delta p_0 \boldsymbol{e}_0$.

where Δp_i is a motion rule function of the displacement $d_i = \boldsymbol{d} \cdot \boldsymbol{e}_i$. The resulting force is computed as $\boldsymbol{f}^t = -k(\boldsymbol{x}^t - \boldsymbol{p}^t)$, where k is the stiffness of the spring-coupler. In our work, we skip the damping term. The motion rule functions can be connected to haptic transfer functions to interactively tune the feedback. The motion rule function for a *unilateral* constraint along axis i is defined by

$$\Delta p_i = \begin{cases} d_i & \text{if} \quad d_i > 0 \\ \max(|d_i| - s_i/k, 0) & \text{if} \quad d_i \leq 0 \end{cases},$$

where s_i is the strength of the constraint, in this case the force threshold that the user must apply to move in the direction $-\boldsymbol{e}_i$. Along $+\boldsymbol{e}_i$ we have free motion. This is the motion rule commonly used for surface simulation with axis i being the normal direction, see Fig. 3. For a *bilateral* constraint, we have

$$\Delta p_i = \text{sign}(d_i)\max(|d_i| - s_i/k, 0),$$

which is used in surface rendering to simulate friction in the directions orthogonal to the surface normal. With a bilateral constraint for all axes of the LRF, viscosity can be simulated. In addition to these constraints (defined in [7]), we also define the motion rule function for a *directed force*:

$$\Delta p_i = s_i/k + d_i,$$

which can be used, e.g., to follow a vector field.

The strengths s_i of the involved constraints are controlled through haptic transfer functions, e.g., sigmoid functions based on the image intensity at the proxy position:

$$s_i = s_i(I(\boldsymbol{p})) = \frac{A}{1 + e^{-(I(\boldsymbol{p})-\beta)/\alpha}},$$

where A is the maximum strength, β controls the center of the function, and α controls the width of the function.

We combine certain motion rules and LRFs into four haptic modes: surface mode, viscosity mode, potential mode, and vector mode. These are described in detail in [20].

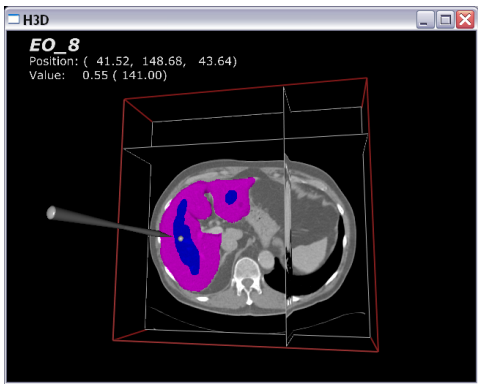

Fig. 4: A CT image displayed in the MPR viewer with an overlay of the segmented liver (magenta) and seed-region (blue).

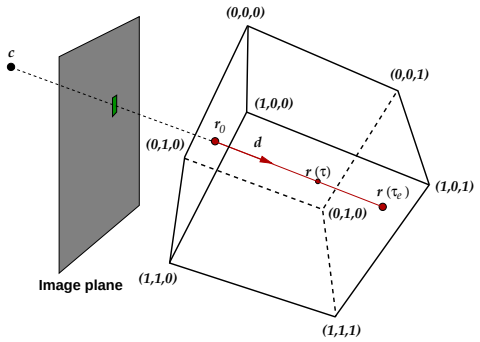

Fig. 5: Hardware-accelerated ray-casting. For each fragment generated in the rasterization of the front-facing faces of the box, the entry point r_0 and the direction d are interpolated. The value τ_e for the exit point is computed by performing ray-box intersection.

3 VISUALIZATION

The 3D image visualization tools available in WISH are multi-planar reformatting (MPR) and hardware accelerated volume rendering. The MPR consists of three orthogonal planes that are controlled with the haptic device. Adjustment of contrast and brightness can be made with a standard transfer function. The volume slicer also supports rendering of overlays. This is used in order to, e.g., display an original image and a segmentation simultaneously, see Fig. 4.

Volume rendering techniques [21, 22] are used to directly display volumetric data without first converting the data to an intermediate surface representation. Volume rendering might, e.g., be used to make important parts of a volume clearly visible while uninteresting parts are made transparent or semi-transparent. The most common approach to perform volume rendering is by using ray-casting. High-quality volume ray-casting of large datasets is computationally demanding and therefore not suitable for interactive renderings when implemented in software.

In recent years, there has been a great development of consumer graphics hardware where the fixed-function pipeline in the graphics processing unit (GPU) has been replaced by programmable vertex processors and fragment processors. These can be customized for the application by using so called shader programs. Techniques for performing direct volume rendering on GPUs have evolved from 2D texture based techniques, to slicing of 3D textures with viewplane aligned polygons [23], to multi-pass ray-casting [24], and more recently to single-pass ray-casting [25]. Our single-pass ray-casting engine [15] is implemented with the OpenGL shading language GLSL [26]. In order to generate the information needed for the ray-casting at each pixel, we use the bounding box of the volume as a *proxy geometry*. For each vertex of the box, we specify the entry position of the ray $\boldsymbol{r}_0$ and the direction $\boldsymbol{d} = \boldsymbol{r}_0 - \boldsymbol{c}$, where $\boldsymbol{c}$ is the camera position. These values are interpolated across the box faces during rasterization in order to obtain the correct value for each pixel. Per-pixel ray-casting is then performed on the GPU by sampling the 3D texture representation of the volume along the parametric ray $\boldsymbol{r}(\tau) = \boldsymbol{r}_0 + \tau \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|}$, see Fig. 5. The exit point of the ray is found by performing a quick ray-box intersection.

With the ray-casting engine, different techniques of compositing the samples along the ray are possible. In volume renderings the colours and opacities are controlled with transfer functions. We have implemented three compositing modes: (colour-correct) maximum intensity projection (MIP), front-to-back alpha blending, and iso-surface extraction with shading. For shading of iso-surfaces, normals are computed with a gradient filter and stored as colours in an RGB or RGBA texture. See Fig. 6 for a rendering example.

4 IMAGE HANDLING

Our toolkit includes several image processing and segmentation algorithms. Voxel-wise operations, Gaussian filters, bilateral filtering [27], gradient filters, gradient vector flow (GVF) [28], distance transforms, basic morphological operations, etc., are mainly used for pre- or post-processing purposes in the segmentation pipeline. The toolkit also includes sophisticated segmentation algorithms that require more interaction and tuning of parameters:

Fast marching and level set methods [29] belong to a powerful framework with partial differential equations

Fig. 6: A volume rendering with our hardware-accelerated ray-caster. Alpha composition of a CT image of the head and torso.

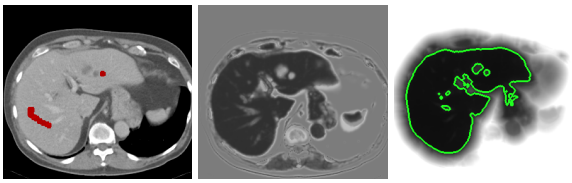

Fig. 7: Fast marching segmentation of the liver in a CT image. Left: Seed-regions placed inside the liver. Middle: The cost image. Right: Resulting time-of-arrival map with overlaid contour obtained by thresholding.

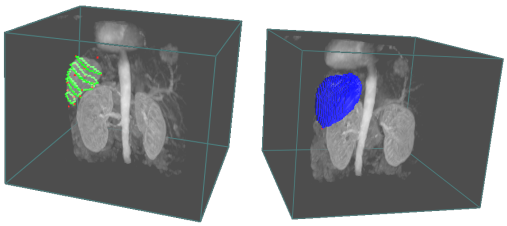

Fig. 8: Our 3D live-wire method. A number of live-wire curves (left) are connected to generate a discrete surface (right).

(PDEs). Fast marching methods are essentially efficient numerical schemes for solving the boundary value problem

$$||\nabla u(\boldsymbol{x})|| = C(\boldsymbol{x}), \quad u(\boldsymbol{x}) = 0 \text{ on } \Gamma,$$

which is called the Eikonal equation. Here, $u(\boldsymbol{x})$ is time of arrival and $C(\boldsymbol{x})$ is a "slowness" or cost function. The equation is derived by considering the closed surface Γ propagating in its normal direction with speed $1/C$. The key is to systematically construct the solution u by propagating information outward from the boundary condition, from smaller values of u to larger values. This requires "upwind" difference schemes in place of classical approximations, e.g., centered differences. The fast marching algorithm is accelerated by limiting the computational domain to a *narrow band* in the proximity of the front.

Image segmentation with fast marching methods involves the design of a cost function C, providing a set of seed-points representing the initial front, and propagation of the front until a certain arrival time is reached, see Fig. 7. The cost image C should have low values in homogeneous parts and high values at edges. The user places seed-regions inside the liver guided by the haptics. C is generated based on gray-level statistics of the seed-regions. The arrival time threshold is automatically found by analysis of the average cost of the narrow band points during propagation.

Live-wire [2, 30] is a semi-automatic segmentation method for 2D images and slice-wise segmentation of 3D images. It is based on shortest path calculation through graph-searching. For every edge in the graph, a cost is assigned to represent the "likelihood" that the edge belongs to a desired boundary in the image. To segment an object, the user places a seed-point on the object boundary. All possible minimum-cost paths from the seed-point to all other points in the image are computed via Dijkstra's algorithm [31]. As the user moves the cursor in the image, the minimum-cost path (the live-wire), from the current position of the cursor back to the seed-point is displayed in real-time. The idea is to have low cost at the desired boundary in order to make the live-wire "snap" onto it. When the user is satisfied with a live-wire segment, that segment is frozen by placing a new seed-point. The tracing then continues from the new seed-point. In this way, the entire object boundary can be traced with rather few live-wire segments.

Most 3D extensions of live-wire are based on using the standard live-wire method on a subset of 2D slices in the 3D volume, and then reconstructing the entire object using this information [32]. Even though the reconstruction algorithms might take 3D information into account, all user interaction is performed in 2D. We suggest a more direct 3D approach [13], where live-wire curves are connected to form discrete surfaces, see Fig. 8. The live-wire curves are not required to belong to a certain slice, but can be drawn freely in the volume. In order to place seed-points directly in 3D, volume haptics guide the user.

Deformable models (snakes) [33] are driven by minimization of an energy functional consisting of an internal shape regularizing term and several external terms based on image data. The key is to build the energy functional so that the desired solution coincides with the global minimum of the functional. The two main characteristics of a deformable surface model are its geometrical representation and its law of deformation. The geometrical representation sets the degrees of freedom and the topological flexibility of the model, while the law of deformation tells how the model should be deformed in order to fit the underlying image data. The concept has been extended to 3D deformable surfaces, e.g., [34].

In our work, we use a discrete simplex mesh representation [35]. In the deformation engine each vertex is regarded as a point-mass influenced by internal forces, external forces, and damping.

5 OUR TOOLKIT

The core of the WISH software consists of a stand-alone C++ class library for image analysis, visualization, and volume haptics. Hence, the algorithms can be integrated into applications written in C++ or used as command-line tools. The visualization algorithms are implemented with OpenGL, and therefore easy to include in any application where OpenGL is used for graphics rendering. We mainly use the VTK[3] file format for volumetric data and a slice se-

[3]URL: *http://www.vtk.org/*

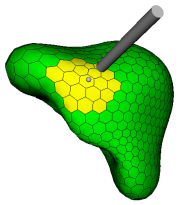

Fig. 9: A simplex mesh being deformed to a liver based on CT data. A user is interacting with the mesh by selecting mesh faces using the haptic probe.

quence reader based on the FreeImage[4] library that can read sequences of the common image file formats.

In order to realize the interactive tool we aim for, we have created an interface between the core functionality of WISH and the multi-sensory 3D visualization software H3D API [5] (version 1.5) from SenseGraphics. H3D API is a cross-platform open-source API for 3D haptics and graphics based on the X3D[6] scene-graph standard. The haptic rendering is performed with OpenHaptics from Sensable. We set up the main scene using X3D and use Python for most of the scene graph management. An advantage of using Python is the possibility to use built-in and external libraries for, e.g., file management, database handling, and user interface development.

The toolkit is availbale for download from the project webpage *http://www.cb.uu.se/research/haptics*

6 CASE STUDY

We present results of interactive liver segmentation with fast marching and deformable models [36]. Liver segmentation is of great importance in hepatic surgery planning [37] and also for monitoring liver enlargement which is correlated to disease progress for patients with liver metastases. Automated liver segmentation is a hard image analysis task due to the high anatomical variability of the liver, often higher among patients with liver tumors, and barely detectable borders between the liver and its neighboring structures in images. Our fast and robust semi-automatic fast marching segmentation method was presented in [12]. Four users independently segmented the liver in 52 abdominal contrast enhanced venous phase CT images from patients with either carcinoid or endocrine pancreas tumor. The method showed high reproducibility in pairwise comparisons of the four sets of segmented datasets. The accuracy was visually verified by a radiologist by combined examination of contour overlays and surface renderings.

In some cases, the fast marching segmentations contain leakage errors due to low contrast, especially at the boundary between the liver and the heart, see Fig. 10. Subsequently, we use the fast marching results in combination with deformable simplex meshes in order to mitigate the leakage and thereby obtain a more accurate segmentation. From the fast marching result, we compute a signed distance map with positive values outside the contour and negative values inside the contour.

We quantitatively evaluate segmentation precision and accuracy [38]. For two segmentations, the precision is computed as the fraction of intersection and union, i.e., the amount of tissue common to both segmentations divided by the amount of tissue in the union of the segmentations. For 23 of the datasets, manual delineations have been performed by two radiologists.

The mean interaction time for seeding of the fast marching method was 40 seconds per dataset. For the subsequent deformable mesh segmentation, the mean interaction time is 93 seconds. The interaction time required for the manual delineation was between 5 and 18 minutes.

[4]URL: *http://freeimage.sourceforge.net/*
[5]URL: *http://www.h3d.org/*
[6]URL: *http://www.web3d.org/*

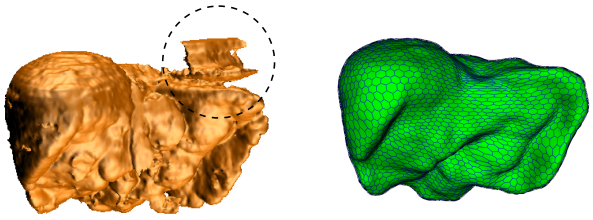

Fig. 10: Left: Surface rendering of the fast marching segmentation for an abnormally shaped liver. Note the leakage between the liver and the heart. Right: Segmentation obtained with the deformable mesh. The irregular region caused by leaking is removed thanks to the built-in shape regularization of the mesh.

For the two sets of 23 manual segmentations performed by the radiologists, we obtain a mean precision of 88.9% (CV 1.9%). The mean precision of the fast marching method is 96.9% (CV 3.8%), which is considerably higher. For the two sets of 23 simplex mesh segmentations, we obtain a mean precision of 97.8% (CV 0.5%) which indicates a high reproducibility.

For the fast marching method, the average sensitivity is 93% and the specificity is close to 100%, i.e., only a few false positive voxels. When we apply the deformable mesh segmentation, we get a sensitivity increase of about three percentage points, while the high specificity is maintained.

7 CONCLUSIONS

We have presented our project on interactive medical image segmentation and visualization in a 3D environment with haptic feedback. A number of well-known tools specially tailored and developed for our environment have been integrated into a toolkit. The software is based solely on cross-platform open-source code and is therefore easily extendable. With limited effort, new methods can be integrated by creating wrappers in the form of H3D API nodes.

In a case study, we demonstrated the performance of the interactive segmentation tools for liver segmentation from CT data. First, we used fast marching segmentation with interactive seeding in order to obtain a fairly accurate seg-

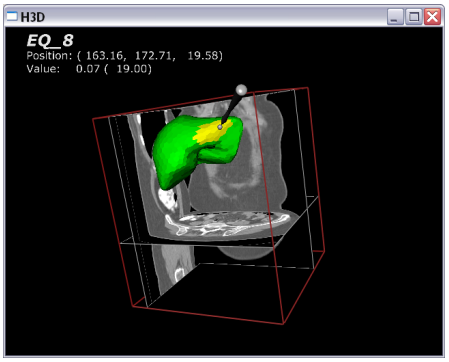

Fig. 11: Liver segmentation with a simplex mesh. The external force is based on a signed distance map computed from a fast marching segmentation. The user applies interaction forces on selected parts of the mesh.

mentation of the liver with high precision. In the subsequent step, we used our deformable simplex mesh to refine the fast marching segmentation. The results showed a considerable increase of accuracy and high precision.

The benefits of using more advanced hardware should be balanced against the increased hardware costs. Although the prices of haptic enabled 3D input devices have decreased significantly lately, they are still more expensive than traditional 2D input devices, which ought to be taken into account in evaluation of our methods. Our bottomline is however that haptic enabled 3D input devices offer many new and exciting possibilities for interactive manipulation and exploration of 3D data.

References

[1] S. D. Olabarriaga, A. W. M. Smeulders. Interaction in the segmentation of medical images: A survey. *Medical Image Analysis*, 5(2):127–142, 2001.

[2] A. X. Falcão, J. K. Udupa, *et al.* User-steered image segmentation paradigms: Live wire and Live lane. *Graphical Models and Image Processing*, 60(4):233–260, 1998.

[3] S. Pieper, M. Halle, R. Kikinis. 3D slicer. In *Proc. of IEEE Int. Symp. on Biomedical Imaging*, pp. 632–635, 2004.

[4] MeVisLab. `http://www.mevislab.de/`, 2007.

[5] P. A. Yushkevich, J. Piven, *et al.* User-guided 3D active contour segmentation of anatomical structures. *NeuroImage*, 31(3):1116–1128, 2006.

[6] B. Reitinger, A. Bornik, *et al.* Liver surgery planning using virtual reality. *IEEE Comp. Graphics and Applications*, 26(6):36–47, 2006.

[7] M. Ikits, J. D. Brederson, *et al.* A constraint-based technique for haptic volume exploration. In *Proc. of IEEE Visualization*, pp. 263–269, 2003.

[8] K. Lundin, B. Gudmundsson, A. Ynnerman. General proxy-based haptics for volume visualization. In *Proc. of World Haptics*, pp. 557–560. IEEE, 2005.

[9] K. Lundin, M. Cooper, *et al.* Enabling design and interactive selection of haptic modes. *Virtual Reality*, 2006.

[10] M. Harders, G. Székely. Enhancing human-computer interaction in medical segmentation. *Proc. of Multimodal Human Computer Interfaces*, 91(9):1430–1442, 2003.

[11] E. Vidholm, X. Tizon, *et al.* Haptic guided seeding of MRA images for semi-automatic segmentation. In *Proc. of IEEE Int. Symp. on Biomedical Imaging*, pp. 278–281, 2004.

[12] E. Vidholm, S. Nilsson, I. Nyström. Fast and robust semi-automatic liver segmentation with haptic interaction. In *Proc. of MICCAI'06*, LNCS 4191, pp. 774–781, 2006.

[13] F. Malmberg, E. Vidholm, I. Nyström. A 3D live-wire segmentation method for volume images using haptic interaction. In *Proc. of DGCI'06*, pp. 663–673, 2006.

[14] E. Vidholm, I. Nyström. Haptic interaction with deformable models for 3D liver segmentation. In *Proc. of MICCAI Workshop*, 2007.

[15] E. Vidholm, A. Mehnert, *et al.* Hardware accelerated visualization of parametrically mapped dynamic breast MRI data. In *Proc. of MICCAI Workshop*, 2007.

[16] E. Vidholm, J. Agmund. Fast surface rendering for interactive medical image segmentation with haptic feedback. In *Proc. of SIGRAD'04*, Sweden, 2004.

[17] E. Vidholm, I. Nyström. A haptic interaction technique for volume images based on gradient diffusion. In *Proc. of World Haptics*, pp. 336–341. IEEE, 2005.

[18] C. Zilles, J. Salisbury. A constraint-based god-object method for haptic display. In *Proc. of Int. Conf. on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots*, pp. 146–151, 1995.

[19] D. C. Ruspini, K. Kolarov, O. Khatib. The haptic display of complex graphical environments. In *Proc. of ACM SIGGRAPH'97*, pp. 345–352, 1997.

[20] E. Vidholm. *Visualization and Haptics for Interactive Medical Image Analysis*. PhD thesis, Uppsala University, 2008.

[21] R. A. Drebin, L. Carpenter, P. Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, 1988.

[22] M. Levoy. Display of surfaces from volume data. *IEEE Comp. Graphics & Applications*, 8(3):29–37, 1988.

[23] R. Westermann, T. Ertl. Efficiently using graphics hardware in volume rendering applications. In *Proc. of ACM SIGGRAPH'98*, 1998.

[24] J. Krüger, R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proc. of IEEE Visualization*, pp. 287–292, 2003.

[25] M. Strengert, T. Klein, *et al.* Spectral volume rendering using GPU-based raycasting. *Visual Computer*, 22(8):550–561, 2006.

[26] J. Kessenich, D. Baldwin,, R. Rost. *The OpenGL shading language*. 3Dlabs, Inc. Ltd., 2006.

[27] C. Tomasi, R. Manduchi. Bilateral filtering for gray and color images. In *Proc. of Int. Conf. on Computer Vision*, pp. 839–846, 1998.

[28] C. Xu, J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Trans. on Image Processing*, 7(3):359–369, 1998.

[29] J. A. Sethian. *Level set methods and fast marching methods*. Cambridge University Press, 1999.

[30] A. X. Falcão, J. K. Udupa, *et al.* An ultra-fast user-steered image segmentation paradigm: Live-wire on the fly. *IEEE Trans. on Medical Imaging*, 19(1):55–62, 2000.

[31] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[32] A. X. Falcão, J. K. Udupa. A 3D generalization of user-steered live-wire segmentation. *Medical Image Analysis*, 4(4):389–402, 2000.

[33] M. Kass, A. Witkin, D. Terzopoulos. Snakes: Active contour models. *Int. J. of Comp. Vision*, 1(4):321–331, 1988.

[34] J. Montagnat, H. Delingette, N. Ayache. A review of deformable surfaces: topology, geometry and deformation. *Image and Vision Computing*, 19(14):1023–1040, 2001.

[35] H. Delingette. General object reconstruction based on simplex meshes. *Int. J. of Comp. Vision*, 32(2):111–146, 1999.

[36] E. Vidholm, M. Golubovic, *et al.* Accurate and reproducible semi-automatic liver segmentation using haptic interaction. In *Proc. of SPIE Medical Imaging*, 2008.

[37] H-P. Meinzer, M. Thorn,, C. E. Cárdenas. Computerized planning of liver surgery–an overview. *Computers and Graphics*, 26:569–576, 2002.

[38] J. K. Udupa, V. R. Leblanc, *et al.* A framework for evaluating image segmentation algorithms. *Computerized Medical Imaging and Graphics*, 30(2):75–87, 2006.