Interactive Editing of Segmented Volumetric Datasets in a Hybrid 2D/3D Virtual Environment

Alexander Bornik
bornik@icg.tu-graz.ac.at

Reinhard Beichel
beichel@icg.tu-graz.ac.at

Dieter Schmalstieg
schmalstieg@icg.tu-graz.ac.at

Institute for Computer Graphics and Vision
Graz University of Technology
Inffeldgasse 16, A-8010 Graz, Austria

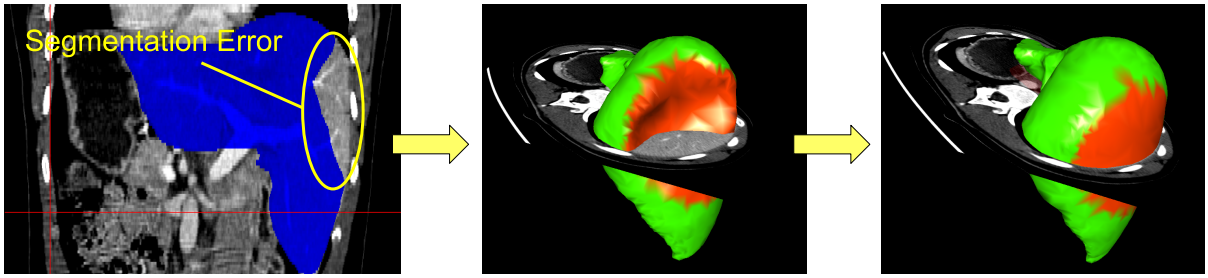


Figure 1: The automatic segmentation of a human liver (blue) on the left suffers from severe, but locally limited under-segmentation. Given appropriate 3D editing tools, the effort to manually correct the 3D model (middle) reconstructed from the segmentation is much smaller than the effort to manually segment the whole dataset from scratch.

ABSTRACT

In this paper we present a novel system for segmentation refinement, which allows for interactive correction of surface models generated from imperfect automatic segmentations of arbitrary volumetric data. The proposed approach is based on a deformable surface model allowing interactive manipulation with a hybrid user interface consisting of an immersive stereoscopic display and a Tablet PC. The user interface features visualization methods and manipulation tools specifically designed for quick inspection and correction of typical defects resulting from automated segmentation of medical datasets. A number of experiments show that typical segmentation problems can be fixed within a few minutes using the system, while maintaining real-time responsiveness of the system.

Categories and Subject Descriptors

I.3.6 [**Computer Graphics**]: Methodology and Techniques; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; C.2.4 [**Distributed System**]: Distributed Applications; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

Keywords

Segmentation Refinement, Interactive Segmentation, Virtual Reality, Hybrid User Interfaces, 3D User Interfaces

VRST'06 November 1–3, 2006, Limassol, Cyprus.

1. INTRODUCTION

Medical imaging facilities like X-ray computed tomography (CT) or magnetic resonance tomography (MR) as well as measurements from industries such as oil and gas or mechanical engineering produce high resolution volumetric datasets. Many applications require the extraction of geometric objects from the volume data through segmentation. Techniques for automatic segmentation have been extensively studied in computer vision, but certain problems remain. For example, noisy CT data makes it hard to extract exact object boundaries in particular for non-rigid objects such as human tissue.

We are developing a system for computer-aided liver surgery planning [6]. The surgeon is concerned with pathological structures (tumors) and the liver vasculature. These structures need to be accurately segmented to study the patient specific anatomy, and derive quantitative indices such as the volume of healthy tissue. Segmentation errors are not tolerable since they will lead to misunderstanding of the anatomical relations and to wrong treatment decisions with severe effects for the patient.

Current automatic segmentation approaches for soft tissue organs are highly problem specific. Some organs like the heart can be addressed using model-based approaches, since the degree of shape variation is low. For organs with a high degree of shape variation, like the liver, such approaches fail, and segmentation algorithms have to rely on evidence in the image such as gray value and gradient information. Unfortunately, the required information cannot always be extracted from the images using known algorithms, for example when adjacent organs share the same gray values in the CT image. The resulting incorrect segmentation is useless for surgery planning, and therefore the current clinical practice relies on manual segmentation by drawing segmentation outlines in stacks of 2D images. This is time consuming at best and impractical for high resolution scans with hundreds of slices. Moreover, even trained radiologists tend to misinterpret complex 3D anatomy when only

viewing 2D cross-sections.

It must be emphasized that automatic segmentation algorithms are capable of recovering major parts of the required objects structure correctly in many medical applications including liver segmentation. However, certain cases defeat attempts at fully automated segmentation. These defects are often locally bound, such as – in case of the liver – leakage into the heart and stomach region, or problems with vena cava inferior. These observations have led us to believe that computer-aided interactive editing of the segmented surface generated by an automatic algorithm – *segmentation refinement* – is a suitable compromise, which is much less time consuming than manual segmentation, yet yields high quality results directly usable in surgery planning. This paper presents a system for segmentation refinement of the liver, but the presented techniques are generally applicable and potentially useful for any kind of application relying on segmentation. Figure 1 shows the idea of segmentation refinement.

At a first glance, it may appear that segmentation refinement is similar to free-form modeling for digital content creation. However, the workflow of segmentation refinement is completely determined by evidence in the original volumetric dataset, while digital content creation relies on the artistic imagination of the designer. The objective of the radiologist working on segmentation refinement is to produce a segmentation (model) matching the shape of anatomical object, which can be arbitrarily complex and can often not be reconstructed accurately with typical free-form modeling tools such as parametric surfaces.

The contribution of this paper is therefore a Virtual Reality (VR) system for interactive segmentation refinement. The system uses a hybrid 2D/3D user interface (Section 3) for efficient yet accurate manipulation of the segmented dataset. It emphasizes the description of specialized editing (Section 4), visualization (Section 5) and deformation (Section 6) tools, which attempt to maximize the use of contextual information that can be directly or indirectly derived from the original volumetric dataset in order to assist the radiologist. We present preliminary results (Section 7) of several test cases evaluated by medical students and physicians, which show that many typical segmentation defects can be solved within minutes, providing first evidence of practical relevance for the time-critical clinical work flow.

2. RELATED WORK

Medical Virtual Reality is an interdisciplinary venture, drawing its inspirations from a variety of fields. In this section, we concentrate on selected work in computer graphics, geometric modeling, computer vision and medical applications that we found most influential for our approach.

2.1 Computer-aided Surgical Planning

Segmentation refinement is part of a liver surgery planning system [6] relying on computer graphics and virtual reality methods. There are many examples of surgical planning systems, which are typically highly specialized towards the medical procedure they support. A good general overview of literature on medical computer graphics and virtual reality can be found in [10]. Liver surgery planning in particular has mainly been investigated by four other groups: The German cancer research center DKFZ, the Center for Medical Diagnostic Systems and Visualization (MeVis), the Zuse Institute Berlin (ZIB), and INRIA. Work at DKFZ [24] on a computer-aided planning system for liver surgery focuses mainly on automatic segmentation. MeVis also focused on segmentation, and also modeling of liver structures [13]. Reserach work by ZIB included model-based techniques for liver segmentation in [22]. INRIA was concerned with model based segmentation [32] and later tissue simulation. While these projects share a similar application goal, none of them attempts to provide interactive segmentation refinement in a way comparable to our approach.

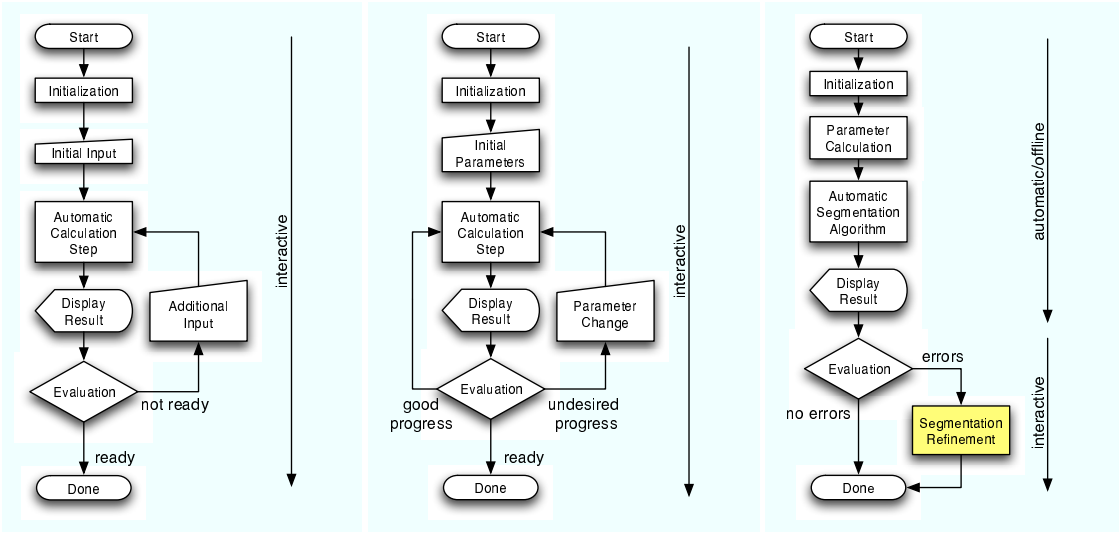
2.2 Interactive Segmentation and Editing

Interactive segmentation of three-dimensional medical datasets can be used to create models for objects, that cannot be segmented automatically. A great number of such techniques and tools can be found in literature. According to *Foo*, such techniques can be classified into three main groups according to how and when user interaction has to be performed [12].

On the one hand, there are *user-steered* methods like painting tools similar to [15]. Also more sophisticated algorithms like the live-wire approach [1], which can be used to speed up the task by exploiting structural information contained in the dataset, belong to this category. The original live-wire approach working on 2D slices of the dataset has been extended to 3D in [11]. Such methods require frequent user interaction throughout the modeling process as shown in Figure 2(a).

The second group, *user intervened* techniques are autonomous techniques operating automatically. They can be interrupted by the user and provided with supplementary input affecting the automatic process at any time (see Figure 2(b)). Such techniques are often interactive applications of deformable models like in [35] or [23].

The third group of interactive segmentation describes tools for correction or manipulation of previous results instead of assisting the segmentation from scratch, so called *segmentation refinement* tools (see Figure 2(c)). One of the rare examples in literature is reported in work by *Jackowski et al.* [19] and [3]. This work uses Rational Gaussian (RaG) Surfaces to represent segmented objects. Segmentation errors can be corrected manipulating surface control points using a 2D desktop interface for the 3D task. Another system to correct segmentations was reported in [20]. The boundary of the segmented object can be altered using a variety of tools. The most effective one from the functional point of view, spline interpolation based on user interaction, turned out to be difficult to use due to user interface issues according to [12]. In fact the system uses 2D interaction and visualization techniques. There have been attempts towards 3D segmentation environments based on VR technology e.g. [30], mostly leading to a proof of concept. A practically more useful example, an interactive vessel segmentation tool exploiting tactile feedback, has been introduced in [14]. These tools fit to the group of *user steered* or *user intervened* techniques. The proposed approach is, to our knowledge, the first interactive *segmentation refinement* system utilizing both 2D and 3D interaction techniques in a immersive environment.



(a) (b) (c)

Figure 2: Classification of interactive segmentation techniques: (a) User-steered segmentation requires frequent user input throughout the segmentation process. (b) In user-intervened segmentation the segmentation process still requires permanent user attention, although interaction is only needed occasionally. (c) Segmentation refinement is performed after a fully automated offline segmentation process.

2.3 Rendering Techniques

In our implementation, we use a number of computer graphics techniques that draw from the rich pool of work on real-time graphics. For example, highlighting of contours outlines and similar features also appears in the area of non-photorealistic rendering, where related hardware-accelerated rendering techniques are used. *Nienhaus et al.*, for instance, exploit intermediate programmable graphics hardware to detect edges in image spaces for *blueprint rendering* or simply edge enhancement [27, 28]. However, in this work contours of arbitrary polygonal geometry on a clipping plane need to be rendered, which makes a difference, looking at the problem in more detail. Efficient rendering of changing polygonal models has been mainly studied in conjunction with level of detail (LOD) techniques. *Hoppe* reported a strategy to minimize data transfers between main memory and the CPU, using indexed triangle strips and transparent vertex caching [17], which is in some sense similar to our approach. For real-time rendering of deformable models, *Kry et al.* [21] exploit programmable vertex units on the GPU to achieve real-time character skinning for large models.

2.4 Deformable models

Deformable models are used in many scientific fields like computer vision, where they are used for e.g. image segmentation, where deformable models have been applied for automatic segmentation as well as interactive tools. In computer graphics, pioneering work was done by *Terzopoulos et al.* [34]. For modeling physical properties of 3D objects, volumetric models and finite elements are employed as in recent work by *Teran et al.* [33]. In segmentation refinement the boundary of segmented structures should be altered, which affords deformable surface models. A good overview of such models can be found in [25].

3. SYSTEM DESCRIPTION

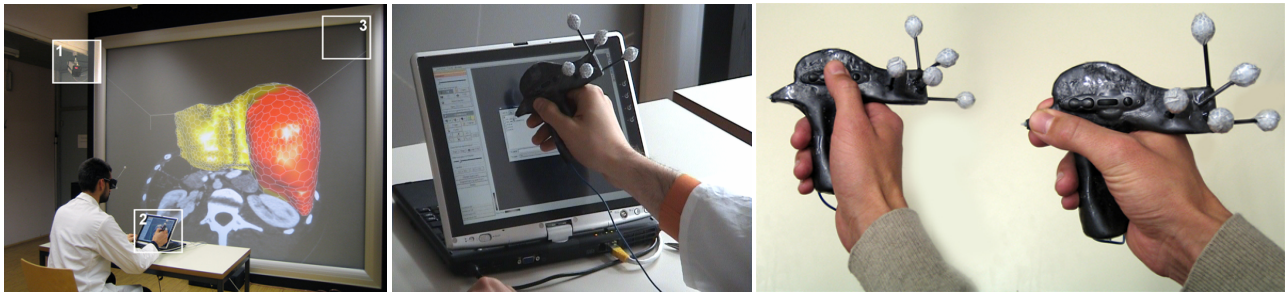
After developing several prototypes and discussions with end users, the current user interface design for segmentation refinement is based on a hybrid approach, combining aspects of VR and desktop workstations [5]. It is motivated by the observation that the editing task requires both intuitive 3D inspection/manipulation and fine-grained control over the editing. Since there are inherent limits to the precision of free-handed operation, a combination of an immersive stereo display using a free-handed 6-DOF input device for direct manipulation and a Tablet PC for extremely accurate 2D input was adopted. A special input device, the *Eye of Ra* (Figure 3(c)) was built for simultaneous operation of the 2D and 3D interface. For more details and results of the system from an HCI perspective refer to [5].

The immersive display is a large stereo wall (stereoscopic backprojection system, 375cm diameter, 1280x1024 pixels) driven by a Barco Galaxy projector and viewed with shutter glasses. The stereo wall is driven by a PC workstation (dual 3GHz Xeon, NVidia Quadro FX 3400). Optical tracking of the user's head and the input device is done using a 4-camera infrared system from Advanced Realtime Tracking. The Tablet PC interface (Toshiba Portégé M200, 1.8 GHz CPU, GeForce Go 5200 graphics card, 12-inch TFT touch screen at 1400x1050 pixels) is placed on a desk approximately 1.5m in front of the stereo wall, tilted at approximately 60 degrees for convenient viewing. The user is seated at the desk so that both stereo wall and Tablet PC are within the field of view as shown in Figure 3(a).

All user interaction can either be performed on the Tablet PC, or by 3D direct manipulation in the VR environment. The exception are system control tasks, such as tool selection or menu settings, which are done exclusively on the Tablet PC. The two software components of the system, for VR and Tablet PC, share most of the code based on the *Studierstube* VR framework and synchronize using a distributed shared scene graph [16].

4. REFINEMENT TOOLS

There are two different ways how segmentation refinement is addressed by the presented system. The final segmentation can be improved exploiting temporary data produced in the automatic stage, so called segmentation chunks. The chunks themselves are directly



(a) (b) (c)

Figure 3: (a) System setup: camera of the optical tracking system (1), Tablet PC and *Eye of Ra* (2), stereoscopic large screen projection system (3). (b) Closeup of the Tablet PC 2D user interface. (c) Eye of Ra - Input device for the hybrid user interface: The tip contains a conventional Tablet PC stylus tip for 2D interaction. Two buttons, a scroll wheel and tracking targets are necessary for 3D interaction. The device can be grasped like a remote control or like a stylus.

derived from regions of different likelihood to be part of the final segmentation, as produced by the used multi-object fuzzy connectedness approach [2]. Algorithms like graph cuts can be used to produce similar data [7]. Furthermore segmentation refinement based on a deformable surface model can be done. It is reconstructed from the selected segmentation chunks. The overall procedure can be split into five tasks:

1. **Chunk Inspection** - The user tries to locate errors in the automatic segmentation result visualized based on the selected chunks
2. **Chunk Selection** - The set of chunks approximating the target structure most accurately is found by adding or removing individual chunks.
3. **Segmentation Surface Inspection** - The user tries to locate errors in the surface model by comparing raw CT data to the boundary of the segmentation surface.
4. **Error Marking** - Regions of the surface model that were found erroneous during the inspection step are marked for further processing. This allows to restrict the following correction step to the erroneous regions, and avoids accidentally modifying correct regions.
5. **Error Correction** - Marked regions are fixed using special correction tools based on direct surface model deformation or template shape based modeling.

There is no strict order for these tasks. In general the best strategy is to perform steps 1 and 2 first, resulting in an optimal initial surface model. In the sequel steps 3 to 5 are performed repeatedly until the model is corrected. Figure 4 gives an overview of the refinement workflow.

In the reminder of this section will give an overview of the segmentation refinement tools, mostly from the user's point of view. Technical details on the implementation will be given Sections 5 and 6.

4.1 Inspection

The first step in both refinement stages, inspection, can be performed on the Tablet PC screen using the *Eye of Ra's* tip for interaction with the rotation, movement and scaling controls in the 2D user interface. In VR the model can be moved and rotated by pressing the scroll-wheel button on the input device, which fixes the model to the input device, while moving the device.

Volume data is visualized on a 2D cutting plane that can be arbitrarily placed inside the scan volume. On the Tablet PC the plane can be manipulated by dragging 3D control widgets provided by the scene graph library. In the VR system the cutting plane, visualized as a rectangle attached to the input device, is set by dragging in 3D with a specific button pressed. The user may also configure several visualization parameters. Especially in the second stage, surface model based refinement, the user can choose to visualize the surface model in any combination of wireframe, Gouraud shading and textured with the volume data. Optional clipping of the model above the cutting plane with contour highlighting allows to inspect the surface model near the clipping plane.

4.2 Chunk Selection

Chunk selection is facilitated by allowing the user to click on the screen in regions, where chunks should be added or removed, if using the Tablet PC as input device. In 3D the same procedure can be done by pressing a button on the input device at the desired location. Clipping the chunks above the cutting plane gives more insight and allows to perform the selection in the 2D plane. Selected chunks are visualized transparently in different colors based on point rendering techniques. Figure 5 shows an example.

4.3 Error Marking

For efficient organization of the surface based correction procedure, the user marks regions of the surface according to the type of observed error by painting a "traffic light" color code - green, yellow or red. Green indicates that a portion of the surface is correct and will be immutable by subsequent correction operations. Yellow indicates that the surface should approximate the boundary of the binary segmentation. Finally, red indicates the surface is incorrect. It does not obey the segmentation boundary any more and may be drastically altered by the error correction tools.

4.4 Error Correction

The presented system allows for correction of segmentation errors using a number of different tools for interactively deforming the surface representation of the object.

The *sphere deformation tool* consists of a sphere of user-defined radius which can be interactively placed in the datasets. In the VR system this is realized by moving the input device, while the tool position in the desktop setup is calculated as the position on the cutting plane corresponding to the 2D position of the cursor. Triggering sphere deformation causes object surface parts located within the sphere shape to be successively moved out of the sphere on the shortest possible path. Therefore, placing the sphere tool

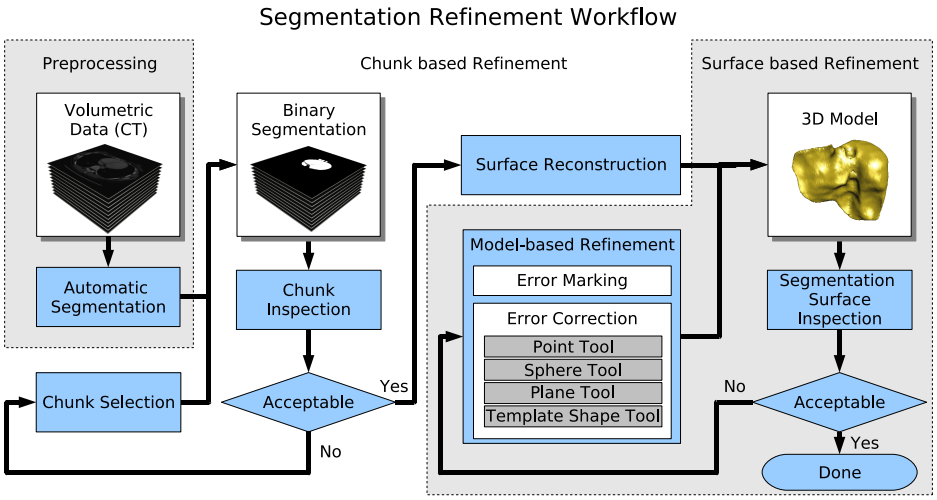


Figure 4: Segmentation Refinement Workflow: The initial segmentation calculated in an automatic preprocessing step can be optimized in the first refinement stage through chunk selection. A surface model reconstructed from the set of selected chunks can be altered using model based refinement tools in stage two.

so that most parts of it are outside the object, causes its surface to move inwards, while outward movement is achieved by placing the sphere mostly inside the object. Moving the input device, while the deformation tool is active, causes the tool to respond, just as if one was deforming a piece of clay using a real world modeling tool.

The *plane deformation tool* is much like the sphere deformation tools, except that its behavior is similar to modeling using a scraper. It can be used to flatten the object's surface. In the VR system the position and orientation of the tools is directly determined by the input device, while cutting plane and the pen stroke direction define the tool's behavior in the desktop setup.

Fine grained deformation can be achieved using the *point dragging tool*, which can be used to pick individual surface vertices and move them directly to the desired location, while the surface deforms like a rubber sheet in the vicinity.

Figure 6 shows a sequence of images taken during segmentation refinement using the plane deformation tool.

The above mentioned direct surface deformation tools are useful for correcting small local problems in the segmentation, but not efficient if a larger deviation from the true surface is identified. Modifying a surface patch by repeated deformation followed by comparison of the resulting segmentation to the original volumetric dataset becomes increasingly cumbersome with growing patch area and distance to the intended surface.

Therefore we have designed an indirect surface deformation method based on the idea of specifying a three-dimensional correction surface, the *template shape tool*. The idea is to interactively specify a correct surface patch for a single erroneous region of the object surface. First, the desired region is specified using the marking tool. Next, a small number of contour polylines are drawn by the user. Each polyline is drawn on the cutting plane after positioning the cutting plane appropriately in 3D. The template shape is then computed by interpolating the polylines. If the user is satisfied with the template shape, the mesh deformation can be triggered to locally approximate the template shape. A segmentation refinement example using the template shape tool is given in Figure 7(b). For technical details on template shapes refer to Section 6.5.

The task of specifying contours for the template shape refinement tool shows the potential of the hybrid setup. The specification of the cutting plane is far more easily performed in the VR part of the system [5], while contour drawing is easy using the Tablet PC. The Eye of Ra allows to seamlessly switch between both 2D and 3D, so that they are perceived as a single integrated user interface.

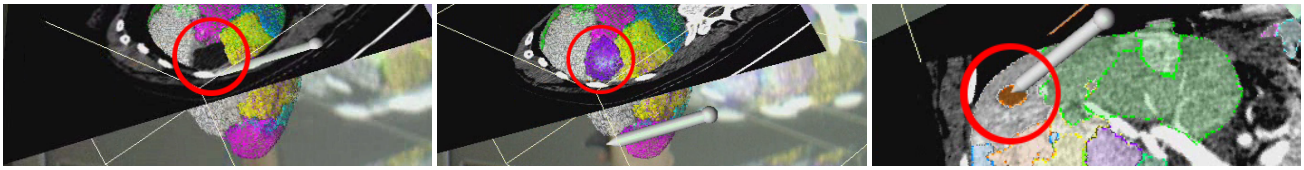
5. VISUALIZATION TOOLS

5.1 Context Data Rendering

When performing segmentation refinement, the most important information to be presented to the user is context data from the original volumetric dataset. We have experimented with the use of texture-based direct volume rendering, but the results were visually confusing and did not turn out to be useful. Instead, we found a clearer way of presenting context data as a textured cutting plane. Radiologists are used to 2D high resolution views and are pleased by the prospect of being able to interactively inspect arbitrary 2D views on such a cutting plane.

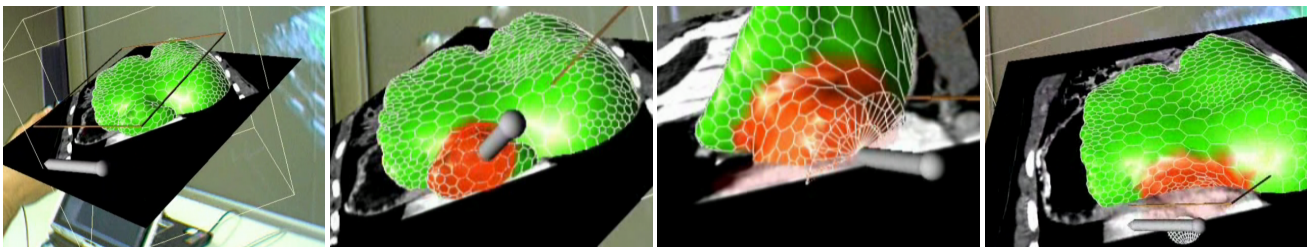
Due to graphics memory constraints on the Tablet PC, a two level approach was chosen. A downsampled volumetric dataset is loaded into the graphics memory and used to render a 3D textured polygon at interactive rates, while the user is moving the cutting plane. A background thread creates a high quality 2D texture by sampling the dataset with tri-cubic interpolation as soon as the user releases the cutting plane positioning tool. Once the 2D texture becomes available (typically after 500 ms), the visualization switches to the high resolution texture.

Besides the cutting plane, texturing the segmented surface directly with the 3D volumetric texture was also found to be useful. The user can switch the rendering style of the segmented surface between wireframe mode, smooth shading showing the local curvature, and 3D textured shading showing the correspondence between volumetric data and segmented surface. All of these rendering modes are frequently used. Figure 8 gives an overview. All texture data is stored in graphics memory at full 16 bit radiometric resolution, thus no time-critical data downloads to the graphics memory are necessary when the transfer function affecting the visible range of values is changed. Transfer functions are evaluated on the fly with a Cg fragment shader program.



(a) (b) (c)

Figure 5: Segmentation refinement using the chunk selection tool: (a) Segmentation error caused by a tumor at the liver boundary. (b) Adding a chunk using 3D interaction solves the problem. (c) Chunk selection with clipping enabled. – All interaction could alternatively be performed on the Tablet PC.



(a) (b) (c) (d)

Figure 6: Segmentation refinement using the plane tool: (a) Model inspection in the VR environment. (b) Marking the erroneous region by painting it red. (c) Interactive model correction using the plane tool. (d) Resulting model. The cutting plane is used for a final inspection.

5.2 Contour Rendering

An essential tool for the verification of the segmented model is the 2D contour resulting from the intersection of the segmented model with a textured cutting plane showing an oriented slice from the volume dataset, as explained in Section 5.1. Rendering the textured cutting plane together with a shaded or wireframe model of the segmented surface is inadequate for inspecting the exact local approximation of the desired structure by the segmented surface (see Figure 8, especially the magnified detail).

Since both clipping plane and geometry of the segmented model are subject to frequent changes, the conventional strategy of calculating the contour explicitly becomes expensive. We developed a two-pass image based approach towards rendering the contour. The algorithm utilizes the OpenGL framebuffer object (FBO) extension to render the contour information directly to a texture in the first pass, and a Cg fragment program to superimpose the final contour on top of the volume texture slice in the second pass.

In the first pass, the object is rendered into the FBO with lighting turned off. Black color is assigned to all front-facing fragments, while backfacing fragments are rendered white. The z-buffer and the z-test work normally using the renderbuffer extension. A clipping plane coincident with the cutting plane is turned on. The resulting image in the FBO shows all inner parts of the model, as visible through the clipping plane.

In the second pass, the cutting plane is rendered using a fragment program that evaluates an edge detection filter over the FBO, which is now used as texture source, accessed using screen coordinates. Fragments located on the boundary of the object's image in the FBO are highlighted. Moreover, since it is an image space method integrated into the graphics pipeline, the contour cannot deviate from its correct location due to numerical inaccuracies. The additional computations to be performed on the GPU depend on the screen size of the cutting plane polygon. For performance details refer to Section 7.1.

5.3 Rendering of the Deformable Mesh

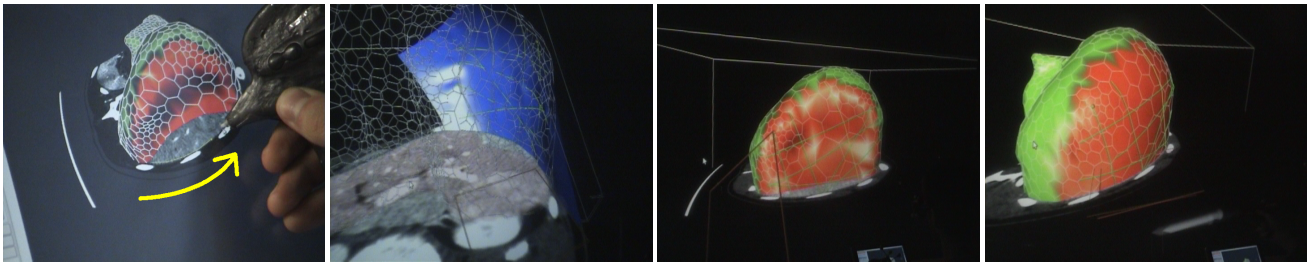
Editing of three-dimensional models in Virtual Reality requires high frame rates, but the constantly deforming nature of the segmented mesh defeats conventional real-time rendering optimization techniques. Simple techniques typically used for animated geometry – immediate mode rendering or vertex arrays – are precluded, since an accurately segmented model can become too large for continuous retransmission from CPU to GPU in every frame. Display lists are not helpful either, because they would have to be frequently re-generated as well.

We have therefore adopted a combination of an OpenGL vertex buffer object (VBO) and locally bound display elements to render the mesh structure efficiently. The deformable mesh is stored in main memory, and only incremental updates to the actually modified portions of the mesh are transmitted to the GPU as needed. We further exploit the fact that modifications are typically local, and mostly modify only vertex geometry, but not mesh topology. The mesh topology is only modified when extreme deformation demands adaptive generation of new vertices.

On the GPU we maintain a single vertex buffer for all vertices and a list of element buffers, each storing the polygon indices for a small number of facets. On the CPU, we store the deformable mesh data structure itself together with a change history: The invalid vertex set (IVS) contains the indices of all vertices changed since the last update from CPU and GPU. The invalid polygon set (IPS) contains indices of all polygons with modified topology.

The update routine maps the VBO to main memory and updates the GPU's copy of all vertex data listed in the IVS. In the rare case that too many new vertices have been added and the VBO's capacity is exceeded, the VBO must be re-allocated.

Element buffers containing polygons referenced in the IPS are rebuilt in graphics memory. Since local modifications of mesh topology occur with low frequency and affect only a few polygons, the number of updated element buffers is small. The element buffers contain only a small fixed number of n polygons, and are initialized with locally coherent polygon data. The mesh type used in our application has an average of six vertices per polygon, but often the element buffer will not be completely filled with n polygons, and can accept a few extra polygons after splits without the



(a) (b) (c) (d)

Figure 7: Segmentation refinement using the template shape tool: (a) Contour drawing on the Tablet PC in the erroneous region of the model. (b) The contours define a template shape (blue). Usually a small number of contours is sufficient. (c) The model deforms towards the template shape. (d) After some seconds the model matches the template shape.

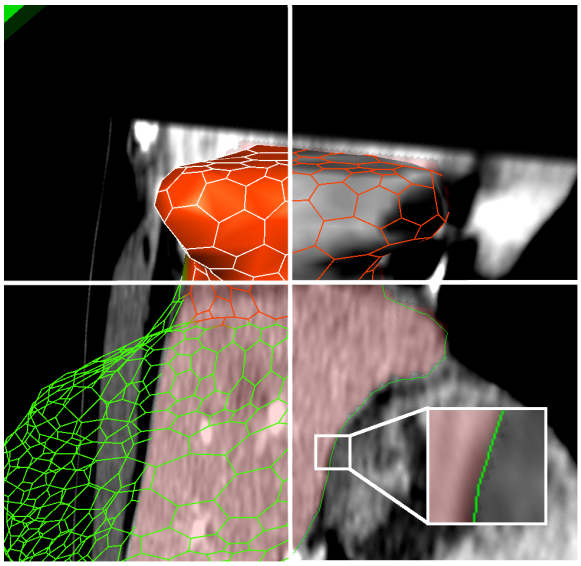


Figure 8: Rendering modes: The segmented object can be displayed as a colored surface, a 3D textured surface, in wire frame model or clipped above the cutting plane showing CT data plus the automatic segmentation result (reddish). In the lower right the object contour is highlighted using a Cg fragment program. Note: Small difference between the voxel-based initial segmentation and the surface model contour are due to the different model nature.

overhead of re-allocation.

Once all buffers have been updated, rendering is done issuing `glMultiDrawElement` calls. Mesh deformation happens at a lower frame rate than rendering, sometimes requiring a few hundred milliseconds, and consequently these two task are decoupled into two separate threads. Mutually exclusive access to vertex data is granted through the use of locking mechanisms.

Updating the data on the GPU is the responsibility of the rendering task. In order to avoid stalling the rendering while waiting for access to the deformed mesh data locked by the deformation tasks, the rendering thread only waits for a maximum of half the target frame time. After this timeout, which was empirically determined, the unmodified data on the GPU is rendered again until the lock can be obtained.

6. DEFORMATION TOOLS

6.1 Simplex Meshes

Segmentation refinement is based on simplex meshes, a deformable model based on a discrete mesh and a force framework based on a Newtonian law of motion involving internal forces F_{int} and external forces F_{ext} [8]. Internal forces F_{int} regularize the mesh. External forces F_{ext} are the most important parameter for obtaining an initial simplex mesh based on a binary segmentation of the original volumetric dataset, and they also form the basis for all segmentation refinement tools. More details on external force calculation are given in Section 6.3.

6.2 Real-time Deformation

When interactively editing the mesh, immediate feedback is essential. The initial deformation of a detailed mesh with around 100K vertices to fit the automatically pre-computed binary segmentation of the volumetric dataset requires in the order of one second to compute. After the initial deformation, we achieve real-time editing performance by exploiting local coherence. A set of active vertices is kept throughout all iterations, and force calculation is limited to this set. Vertices enter the active set if they are not classified as final (red or yellow code in the error marking procedure) and come under the influence of an error correction tool, or if a large force exceeding a certain threshold affects a neighboring vertex. When using the template shape based refinement tool, all vertices marked red or yellow are activated. Removal from the active set occurs, when the forces calculated for a particular vertex are negligible in magnitude over several iterations, or on user request (e. g., the user paints the vertex green).

6.3 External Force Formulation

Forces for initial deformation are determined by geometric constraints. Assuming a generic deformable mesh is available (for example, starting with a tessellated sphere or extracting a coarse iso-surface from the volumetric data directly), the purpose of the initial deformation is to approximate a shape given by the binary segmentations obtained through automatic segmentation of the volumetric. The binary segmentation can later be altered using chunk selection requiring updates of the surface model, which are performed similarly.

For every vertex of the deformable mesh, a force vector towards a corresponding target point on the surface of the segmented volumetric dataset needs to be computed. We use the normal vectors of the deformable and search for the segmentation boundary in its direction using a 3D Bresenham algorithm in order to determine the force vectors.

Forces for sphere refinement tool are directed towards the sphere boundary on the shortest path, for all vertices located within the

tool.

Forces for plane deformation tool are calculated for all vertices within a cylinder built on top of the tool center. All affected vertices are pushed to the bottom of the cylinder, actually representing the deformation plane.

When individual vertices are altered directly, besides moving to their new location, they are removed from the active set and therefore colored green. This prevents them from moving further.

Forces for the template shape refinement tool are calculated using the location of each affected simplex mesh vertex in the parameter space of the template shape. The actual force target point is the mesh surface point corresponding to the parameter value.

6.4 Mesh Restructuring

During refinement, the mesh surface may become subject to substantial changes, which cannot be expressed by just altering vertex positions. To avoid strongly varying polygon distribution or even self intersection due to welding, the mesh structure must be adapted to the shape.

Restructuring simplex meshes has been studied in great depth by *Delingette et al.* [8], who propose *face merge* and *face split* operations conceptually similar to edge collapse/vertex split operations known from mesh simplification [18]. These operations are used to simplify or refine the mesh. An *edge swapping* operation is used to equalize the vertex count of all polygons adjoining an edge. The proposed system relies on these operations.

The criteria we employ to control mesh restructuring are both global – enforce roughly uniform polygon area, suppress degenerate triangles via a maximum elongation measure – and based on local curvature. These criteria are recalculated every few iterations.

Real-time performance of the mesh editing demands to use large iterations steps (β 0.45), which often leads to welding and self-intersections despite these mesh restructuring efforts. An additional *welding criterion* avoids problem, while still allowing for aggressive iteration steps. It is calculated by least squares fitting a plane through each non-planar polygon. If the normal vector dot product of two adjacent polygons is negative, they are joined.

6.5 Thin Plate Spline Template Shapes

The template shape based refinement tool was already introduced in Section 4.4. It uses thin plate splines (TPS), a popular tool for all kinds of interpolation purposes in computer vision [4]. We employ thin plate splines to create an interpolating surface through all contour polygon support points and all vertices adjoining the region (border points) marked red using the error marking tools.

TPS template shapes fit the problem of segmentation refinement well. They are interpolating splines, thus control points are located on the surface and can be directly put in place by the user. Another advantage of TPS is the fact that the control points need not be specified over a regular control grids. Points can be specified where needed, minimizing their overall number, and along with that, the amount of user interaction required. The fact that the control points have global impact always leads to smooth solution as well as computational costs for large numbers of points. In practice the number of points is small (below 100), since complex deformations are better addressed using direct deformation tools, so that approaches with finite support or faster approximation techniques as for example described in [9] need not be applied.

7. RESULTS

7.1 Rendering Visualization

Overall frame rates were always greater than 50 Hz on the VR setup, even during model refinement and with features such as clipping contour rendering turned on. Rendering times for the deformable simplex meshes were below 1 ms in the VR system, only rendering of the cutting plane covering almost the whole screen with contour highlighting turned on took 2 ms. On the Tablet PC the overall frame rates were clearly lower, but still interactive (> 5 fps). Rendering times for the simplex mesh from the liver dataset ranged from 1 ms to 5 ms, depending on the size on the projected screen size. Texture mapping and the fragment program used for contour highlighting were costly on the GeForce Go 5200 of the Tablet PC. We recorded rendering times of 2 to 42 ms for the cutting plane without contour highlighting, which doubles the frame times, then ranging from 4 to 98 ms. The times correlate with the number of visible fragments of the cutting plane. The number of active vertices in the deformation process did not impact the rendering performance in the VR system. Due to the proposed vertex buffer object based rendering technique, the impact of mesh iterations was hardly noticeable on the single CPU tablet PC.

7.2 Segmentation Refinement Performance

The number of iterations per second depends on the number of active vertices and type of force calculations used. During initialization, when the mesh is deforming towards the initial segmentation by looking for target points using the ray casting approach, we measured 92 ms for the large liver dataset containing about $12,000$ vertices on the VR system's workstation, and 180 ms on the Tablet PC. For the smaller spleen model of around $5,000$ vertices the numbers are 47 ms versus 80 ms. If smaller portions of the mesh were active, iteration time decreased accordingly.

7.3 Segmentation Refinement Results

The performance of the system has been studied on seven test cases, based on a CT scan of the abdominal region. For the dataset under investigation, manual reference segmentations of different structures generated by an expert radiologist were given. Then the datasets were automatically segmented using different methods. We collected a set of erroneous segmentations showing errors in a variety of surface regions with different shape. Some datasets suffer from undersegmentation, others from oversegmentation (leaking).

In order to avoid including the surface reconstruction errors introduced during mesh fitting to the manual segmentation, the initial meshes were voxelized using the technique described in [29]. The resulting binary volumetric datasets were used as reference datasets for the comparison with resulting meshes of the refinement process, which also is voxelized.

For performance measurement the error measure calculated was the relative volume error (rVE). It is based on the number of voxels in the refined dataset differing from the reference dataset. In order to obtain rVE we compare the number of differing voxels with the number of voxels representing the structure under investigation in the reference dataset.

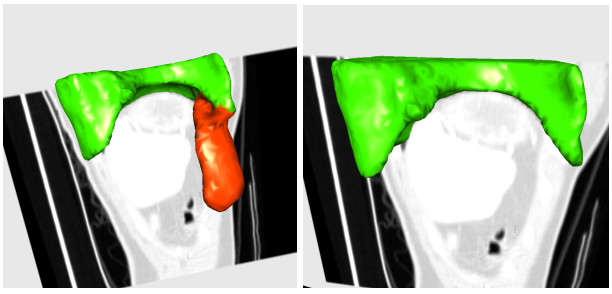
The three test subjects participating in the study consisted of an experienced radiologist, an experienced surgeon and a student of radiology. All of them had used a VR system before, though not very often. Before working on the test cases the functionality of the system was explained to them by the instructor, who also solved a refinement task while explaining all the tools to them. Afterwards, each subject was asked to try out the system by refining an erroneous segmentation result outside the seven cases used for the study. During this process the subjects were explicitly taught how to solve the task efficiently.

#	dataset	error location	rVE_1	rVE_2	$\mu(t)$
1a	liver	left lobe; overseg.	15.3%	2.7%	10.3’
1b		right lobe; underseg.	15.3%	2.7%	12.0’
2	liver	leakage into heart	6.7%	2.5%	13.7’
3	liver	leakage into stomach	9.2%	2.6%	11.3’
4	liver	lower right lobe	4.1%	2.7%	11.7’
5	spleen	leakage into kidney	64.2%	1.9%	10.0’
6	kidney	undersegmentation	8.4%	0.5 %	8.7’
7	lung	leakage into stomach	38.9%	1.8%	10.0’

Table 1: Dataset overview and evaluation results:relative volume error in percent relative to object volume before (rVE_1) and after refinement(rVE_2); average timings for the refinement task.

After the introduction the users were given unlimited time to solve the seven refinement problems as accurately as possible. During the test they were allowed to ask questions, in case of problems, which were answered by the instructor. The time needed for each individual dataset was recorded in addition to the resulting segmentation. Figure 1 shows an example from the study. Other examples from the study are shown in Figure 6 (case 2) and Figure 9 (case 7).

Table 1 gives an overview of the datasets used, the prominent errors and the results obtained in the user study. The relative volume error could be significantly decreased to a level around 2.5 percent, which seems to be a limit imposed by the conversion from the surface model to a volumetric dataset matching the low resolution of the reference dataset. Timings obtained from the rather inexperienced test users are ranging from around eight to fifteen minutes per problem - about eleven minutes on average. For comparison, live-wire based semiautomatic segmentation of the liver CT dataset by experts radiologists took about 40 to 60 minutes. Using a fully automated segmentation method in combination with our segmentation refinement approach reduces the time needed for user interaction significantly. Thus, our approach can make segmentation of medical volumetric data feasible in clinical routine. There was no significant difference in the performance of the users. However, we observed, that the radiologist used the Tablet PC more often, while the surgeon and the students performed all tasks except for contour drawing in the VR system.



(a) (b)

Figure 9: Segmentation refinement of a lung dataset: (a) Lung model before refinement – The region marked red indicates, where severe leakage into the stomach happens. (b) The same model after approximately 10 minutes of interactive editing using the presented tool set – The error is largely resolved.

Results of a study focussing on the usability of the proposed system, based of a small subset of tools but a significantly larger number of users can be found in [5].

8. CONCLUSIONS AND FUTURE WORK

The novel segmentation refinement system presented in this paper combines automatic and user-guided methods for producing high quality three-dimensional models of structures present in volumetric datasets. It aims to strike a useful compromise between the high quality results demanded in medical applications, and the time available for segmentation tasks from expert radiologist. By providing the user with a toolset specifically designed to optimally exploit the data obtained from the automated segmentation and correcting remaining errors, high quality results with a very low error can be created in a fraction of the time required for manual segmentation. The proposed system enables physicians to unleash the full potential of 3D imaging facilities like CT or MRI for surgical planning. Our evaluations with physicians confirm this hypothesis, and informal feedback is also very encouraging in terms of user satisfaction.

Although the tools works quite well in practice, there are several limitations. The used TPS interpolation limits single template shapes to 2.5D shapes. Consequently refinement tasks requiring more complex template shapes have to be broken down into multiple sub-problems. Template shape modeling based on radial basis functions with finite support similar to [26] can be used to overcome this limitation, although requiring to extract iso-surfaces from the volumetric result. In the current implementation self-intersections can occur temporarily. Techniques to guarantee artifact free surface evolution as proposed in [31] for triangular meshes are subject to future work. Beyond numerous other technical enhancements, a more complete clinical study of the overall system, involving data acquisition, segmentation and refinement, as well as the actual surgery planning, is scheduled for the near future.

9. REFERENCES

[1] W. A. Barret and E. N. Mortensen. Interactive live-wire boundary extraction. *Medical Image Analysis*, 1(4):331–341, 1997.

[2] R. Beichel. *Virtual Liver Surgery Planning: Segmentation of CT Data*. PhD thesis, Graz University of Technology, 2005.

[3] R. Beichel, S. Mitchell, E. Sorantin, F. Leberl, A. Goshtasby, and M. Sonka. Shape- and appearance-based segmentation of volumetric medical images. In *Proc. of ICIP 2001*, volume 2, pages 589–592.

[4] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):567–585, 1989.

[5] A. Bornik, R. Beichel, and et al. A hybrid user interface for manipulation of volumetric medical data. In *Proc. of IEEE Symposium on 3D User Interfaces 2006*, pages 29–36.

[6] A. Bornik, R. Beichel, B. Reitinger, G. Gotschuli, E. Sorantin, F. Leberl, and M. Sonka. Computer aided liver surgery planning: An augmented reality approach. In *Proc. of SPIE 2003*, volume 5029.

[7] Y. Boykov and M.-P. Jolly. Interactive organ segmentation using graph cuts. In *Proc. of MICCAI 2000*, pages 276–286.

[8] H. Delingette. General object reconstruction based on simplex meshes. *International Journal of Computer Vision*, 32(2):111–146.

[9] G. Donato and S. Belongie. Approximation methods for thin plate spline mappings and principal warps. In *Proc. of ECCV 2002*, pages 21–31.

[10] T. Emerson, J. Prothero, and S. Weghorst. Medicine and virtual reality: A guide to the the literature (MedVR). Technical Report B-94-1, HITLab, University of Washington, USA, 2001.

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

[12] J. L. Foo. A survey of user interaction and automation in medical image segmentation methods. Technical Report ISU-HCI-2006-2, Iowa State Universtity - Human Computer Interaction, 2006.

[13] H. K. Hahn, B. Preim, D. Selle, and H. O. Peitgen. Visualization and interaction techniques for the exploration of vascular structures. In *Proc. of IEEE Visualization 2001*, pages 395–402.

[14] M. Harders, S. Wildermuth, and G. Székely. New paradigms for interactive 3D volume segmentation. *Visualization and Computer Animation*, 13:85–95, 2002.

[15] T. Heiman, M. Kunert, and H.-P. Meinzer. New methods for leak detection and contour correction in seeded region growing segmentation. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXV, 2004.

[16] G. Hesina, D. Schmalstieg, A. Fuhrmann, and W. Purgathofer. Distributed Open Inventor: a practical approach to distributed 3D graphics. In *Proc. of VRST ’99*,

pages 74–81.

[17] H. Hoppe. Optimization of mesh locality for transparent vertex caching. In *Proc. of SIGGRAPH '99*, pages 269–276.

[18] H. Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.

[19] M. Jackowski, A. Goshtasby, and M. Satter. Interactive tools for image segmentation. In *In. Proc. of SPIE '99*, volume 3661.

[20] Y. Kang, K. Engelke, and W. A. Kalender. Interactive 3D editing tools for image segmentation. *Medical Image Analysis*, 8:35–46, 2004.

[21] P. Kry, D. James, and D. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *Proc. of SIGGRAPH '02*.

[22] H. Lamecker, T. Lange, and M. Seeba. A statistical shape model for the liver. In *Proc. of MICCAI 2002*, pages 422–427.

[23] A. E. Lefohn, J. E. Cates, and R. T. Whitaker. Interactive, GPU-based level sets for 3D segmentation. In *Proc. of MICCAI '03*.

[24] H.-P. Meinzer, P. Schemmer, and et al. Computer-based surgery planning for living liver donation. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXV, pages 291–295, 2004.

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

[26] B. S. Morse, T. S. Yoo, and et al. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proc. of SMI '01*, pages 89–98.

[27] M. Nienhaus and J. Döllner. Blueprints: illustrating architecture and technical parts using hardware-accelerated non-photorealistic rendering. In *Proc. of GI '04*, pages 49–56.

[28] M. Nienhaus and J. Döllner. Sketchy drawings: a hardware-accelerated approach for real-time non-photorealistic rendering. In *Proc. of SIGGRAPH '03*.

[29] B. Reitinger, A. Bornik, and R. Beichel. Efficient volume measurement using voxelization. In *Proc. of SCCG '03*, pages 47–54.

[30] S. Senger. Visualizing and segmenting large volumetric data sets. *IEEE Computer Graphics and Applications*, pages 32–37, 1999.

[31] A. Sharf, T. Lewiner, A. Shamir, L. Kobbelt, and D. Cohen-Or. Competing fronts for coarse-to-fine surface reconstruction. In *Proc. of Eurographics 2006*, volume 25. in print.

[32] L. Soler, H. Delingette, and et al. Fully automatic anatomical, pathological, and functional segmentation from CT scans for hepatic surgery. *Computer Aided Surgery*, 6(3):131–42.

[33] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. Robust quasistatic finite elements and flesh simulation. *ACM Transaction on Graphics*, 2005.

[34] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proc. of SIGGRAPH '87*, pages 205–214.

[35] I. Wolf, M. Hasenteufel, and et al. ROPES: a semiautomated segmentation method for accelerated analysis of three-dimensional echocardiographic data. *IEEE Transactions on Medical Imaging*, 21(9):1091–1104, 2002.