International Journal of Computer Assisted Radiology and Surgery manuscript No.
(will be inserted by the editor)

GPU-based Smart Visibility Techniques for Tumor Surgery Planning

Christoph Kubisch · Christian Tietjen · Bernhard Preim

Received: date / Accepted: date

Abstract *Purpose:* The rating of distances and infiltrations to vital structures is important for the planning of tumor surgery or interventional procedures. To support such an assessment, the target structures should be clearly emphasized in a 3D visualization by ensuring their visibility.
Methods: Smart Visibility techniques such as *Ghosting Views* and *Breakaway Views* are employed. *Ghosting Views* highlight focus structures by fading out occluding structures and are often used in anatomical illustrations. *Breakaway Views* reveal the structure by cutting into surrounding structures. As a result, an intersection surface is created that allows relating the focus structure with its surroundings. In this contribution a specialized GPU-based implementation of these techniques is presented for polygonal models derived from a segmentation of the anatomical structures.
Results: We present different rendering styles of the techniques and apply them to highlight enlarged lymph nodes in the neck, as well as tumors inside the liver. Compared to other methods, we focus on polygonal models and optimizations. Thus, very high frame rates could be achieved on consumer graphics hardware. Furthermore, we employed markers that support the estimation of distances within the scene and possible infiltrations around the focus structures.
Conclusion: The parameters for the techniques are defined automatically to aid the employment in clinical routine. Such an application is also supported by the combination and refinement of established rendering techniques.

Keywords real-time rendering · visualization · smart visibility · interactive exploration · spatial perception

C. Kubisch
Otto-von-Guericke University of Magdeburg, Germany
Institute for Simulation and Graphics
Tel.: +49-391-671-2527
E-mail: kubisch@isg.cs.uni-magdeburg.de

C. Tietjen
Otto-von-Guericke University of Magdeburg, Germany
Institute for Simulation and Graphics
Tel.: +49-391-671-2527
E-mail: tietjen@isg.cs.uni-magdeburg.de

B. Preim
Otto-von-Guericke University of Magdeburg, Germany
Institute for Simulation and Graphics
Tel.: +49-391-671-8512
E-mail: preim@isg.cs.uni-magdeburg.de

1 Introduction

Illustrative 3D visualization have a great potential to support complex surgical planning processes. Various related topics in this context have been addressed, such as viewpoint generation [1], animation [2], visibility and performance, as well as exploration and measurement tools for training systems. When it comes to the actual rendering techniques for emphasizing relevant focus structures within the anatomical dataset, recent work by Baer et al. [3] examined how the evaluation on the efficiency of different techniques can be performed. In this paper, we describe how such techniques can be efficiently implemented and applied to selected surgical planning procedures. If possible primary tumors and metastasis are treated with local therapies, such as surgery, radiation treatment or interventions. The location of tumors, especially the proximity to surrounding risk structures, such as major blood vessels, is important for treatment decisions. The latter deal with the resectability or extent of a surgical procedure as well as with an appropriate access to the tumor. Therefore, it is important to rate distances and infiltrations into adjacent anatomic structures correctly. For this purpose, the visualization of the immediate surroundings of the tumors is crucial. Different 3D visualization techniques exist to depict structures.

Medical image data, such as CT and MRI, is rendered either using direct volume rendering (DVR) or surface-based triangle rasterization. DVR has the benefit of being able to represent the raw scan data directly. However, its quality mostly relies on the scan resolution and classification of the data values to appropriate colors using transfer functions. A segmentation as pre-processing step is necessary to tag the voxels to individual anatomical structures. Turning the segmented results into triangle meshes is more efficient for rendering on the optimized graphics hardware. The major benefit of using meshes, however, is the possibility to use advanced surface reconstruction methods, e.g. for liver segmentation [4] or bones [5]. For surface meshes different methods exist to smoothen the surface and therefore not only reduce visual artifacts, but also increase data fidelity (refer to [6] for details).

Ideally, the structures are rendered fully opaque to assess spatial relationships between objects. However, this may result into *focus structures* being occluded within the 3D scenery. Using transparency may not be helpful to resolve this issue, as depending on the degree of transparency, ambiguities in spatial perception are introduced [7]. Other visualization techniques have been developed that guarantee visibility without depending on transparency. With such techniques, regions occluding the target structure are removed. These techniques are referred to as *Smart Visibility* [8].

By selectively fading out structures or cutting into the medical data, a clearer view on structures can be achieved. This is particularly useful when the focus structures are small and surrounded by larger structures, such as enlarged lymph nodes within the neck (Fig. 1, right). In another application we show that cutting into the liver to reveal a tumor also provides spatial context information, such as vascular structures and which Couinaud liver segments are nearby (Fig. 1, left). As the result images provide correct depth data, measurements can be performed or additional data such as distances to risk structures can be projected.

To achieve a high performance and provide a good rendering quality, we focus on a (hardware-supported) GPU implementation. In the following contribution, several GPU-based implementations are presented for two base techniques in this area. The methods keep otherwise occluded focus structures visible in polygonal models of the patient anatomy. Furthermore, automatic parameterizations are described, so that the techniques can be used without additional overhead for the user. Throughout the paper, we present examples from liver and neck surgery. The techniques have been developed in close collaborations with medical doctors in the field, however they are applicable for a large variety of other interventions as well.

2 Related Work

Feiner and Seligman [9] have presented the key methods and variants of creating *cutaways* and *ghosting* to satisfy visibility constraints in dynamic 3D scenes, which this work builds upon.

For intervention planning in surgery, radiation treatment or interventional radiology, the interactive exploration of large 3D visualizations is essential. Bichlmeier et al. have successfully used 3D scenery in augmented reality [10]. They overlayed the anatomical structures on top of the live image of the patient and faded out the virtual image at its borders in the context of minimal-invasive interventions. This way, the surgeon can virtually look inside the patient and see larger structures from the outside than through the endoscopic view, also known as *virtual X-Ray*. Furthermore, the use of shadows allows enhanced perception of the location of the surgical tools within the anatomical structures of the patient. In the preoperative stage of planning neurosurgical interventions Beyer et al. [11] have successfully used *cutaways* to reveal structures and areas within the brain. This work focuses on providing similar benefits when obstructions within the virtual scenery occur.

With DVR, several *Smart Visibility* techniques can be employed. By applying *ghosting*, the tumor's visibility is guaranteed, as all occluding parts of structures in front of it are faded out. Krüger et al. have achieved this in an efficient GPU implementation and called their technique *Clear View* [12]. Alternatively, the visibility of segmented structures can be assured using *Maximum Importance Projection* [13]. With this technique, structures are cut by cones to visualize the depth of the cut into surrounding structures. We refer to this technique as *breakaway*. Diepstraten et al. [14] defined the cut volume for *breakaways* as view-dependent and as object-dependent for *cutaways*. Rautek et al. have used the focus object's silhouette to create cuts, that give better context information conforming to the target structure's shape [15]. Ray-casting allows discarding the shading result of previous sample or intersection points at any time during the traversal of the ray. Thus, contributions of occluding samples can be discarded or faded out, once a focus structure is hit. It is also possible to stop or start the ray-casting at any custom boundaries. Therefore, in DVR these techniques

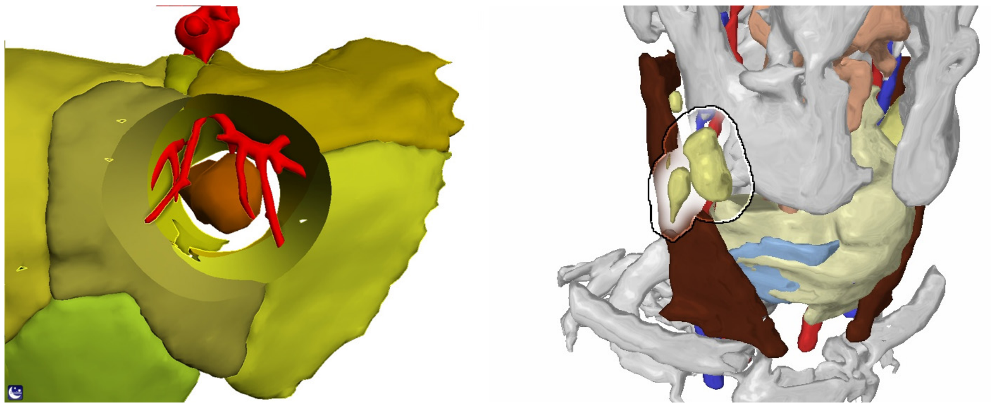


Fig. 1 The *Section View* (left) was applied to a liver dataset to reveal an (artificial) tumor and the vascular structures. Enlarged lymph nodes are highlighted through *Ghost Views* (right). Some small artifacts can be seen on the liver, as the triangle mesh did not have a closed surface.

are easier to achieve through the use of GPU ray-casting [16], than compared to polygonal models rendered with classic rasterization. As polygonal models still play an important role in medical visualizations, we focussed on techniques for rasterization.

In the rasterization pipeline, each object is rendered separately and any form of higher-level scene description is typically lost (compared to a volume texture containing the entire data for DVR). Hence, different approaches are needed when dealing with *Smart Visibility* techniques for polygonal mesh scenes [14,17]. Efficient *ghosting* has been achieved using off-screen rendering by Elmqvist et al. [18]. The same authors have also performed a user study, that showed superior performance in both correctness and efficiency for perceptual tasks. Compared to their technique, our *ghosting* variants make use of the smoothness in anatomic shapes for creating the blending weights, and use latest GPU features to compute the masking primitive dimensions. One of the key issues in generating *breakaways* for polygons is, that triangles can only represent the surface of a structure. When a structure is not cut along the viewing direction, no geometry for the interior exists, therefore structures appear “empty”. Generating new triangle surfaces on the fly may not be practical to do in real-time and with appropriate quality. The systems examined so far solved these issues by using *Constructive Solid Geometry* (CSG) [19]. The rendering of CSG can be realized by making extensive use of the stencil and depth buffer in the hardware rasterization pipeline. In our contribution we refine the key concepts of CSG and could therefore optimize their runtime behavior. The effect is similar to the work of Burns et al. [20], who generate cuts based on the object's shape. Their algorithm is restricted to closed geometry that is free of self-intersection. Our algorithm, however, had to take into account overlapping meshes and especially enclosures, which often arise from segmentation results (e.g tumor within another organ). Furthermore we could simplify the generation of the object dependent cut-shape, because of the smooth surfaces organic structures have. In contrast to these previous techniques, we added effects on the cut surface itself, to aid the perception of distances and intersections, which are important aspects for medical planning systems.

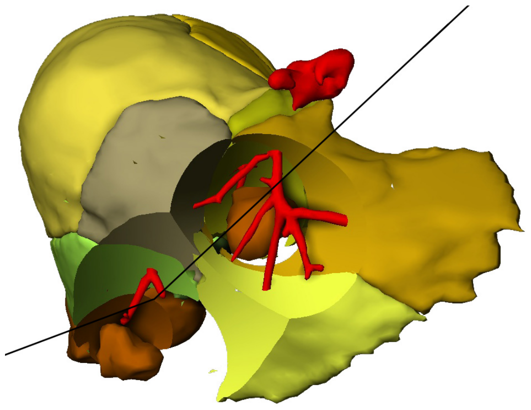


Fig. 2 The depth perception of the objects is enhanced through shadows (lower right). The comparison of the blurred depth buffer with the unmodified scene depth yields smooth shadows from a spot light that is attached to the camera.

Shadows and *Ambient Occlusion* are commonly used to give clues about occlusion among structures. Depending on the proximity of structures, the self-shadowing within the scene results in different *Ambient Occlusion* values. Pre-computation for these effects, however, is less ideal for *Smart Visibility*, as the scene's exposure to light and the object's visibility is changed directly by the user. In rasterization pipelines, the most common way to approximate *Ambient Occlusion* like effects for dynamic scenes is to use the final image's depth buffer, as it is the only efficient way to derive spatial information from the entire scene. Luft et al. have used *unsharp masking of the depth buffer* to create halos and shadows around structures [21] (Fig. 2 shows our implementation of this technique). In the context of highly dynamic scenes, e.g. in computer games, *Screen Space Ambient Occlusion* has been developed [22], in which *Ambient Occlusion* is approximated by sparse sampling of surrounding pixels in the depth buffer and comparing the depth values between them. We chose the *unsharp masking* technique due to ease of implementation and high performance. To create the look of manually created illustrations, we apply *lit sphere* shading by Sloan et al. [23].This approach was also successfully used by Bruckner et al. [24] in volume rendering of medical data.

3 Methods

A common pre-requisite for all presented techniques is the segmentation of all relevant structures, as well as the triangular mesh surface generation. All objects are represented by closed manifold surfaces, otherwise small holes in the surface are unintentionally emphasized, as seen in Fig. 1. The surface normals used for shading are smoothed per vertex, i.e. the vertex normal is the average of all connected triangle normals. In general, the scenery is split into three types of objects by the user [25]:

– **Focus.** Typically, the tumor or other pathologic structures are unaffected by intersections and serve as input for the generation of the cut volume.
– **Context Detail.** Detail objects, such as small vessels, are also not intersected, as their relationship to the tumor might be important and their occlusion contribution might be insignificant.
– **Intersectable.** The majority of the objects are intersected by the cut volume so that focus and detail structures are revealed.

At first, we will present the *ghosting* algorithm. The implementations vary in their complexity and necessary CPU interaction. We will provide different forms of automatic parametrization of the cut volume. These are partially also applicable for the *breakaway*, which is described in Sect. 3.2.

When *Smart Visibility* techniques are applied, it is crucial to detect those surface areas, which lie between the tumor and the viewer. Our approach is similar to texture-based cutout by Diepstraten et al. [14], with the added restriction that the camera is within the cut volume. Since the goal is to highlight focus structures, we assume that the cut volume will always contain the camera and is itself convex (Fig. 3, left). This simplification means that a single depth comparison with the cutter's depth z_c is sufficient to classify a surface fragment's depth z_f as either occluding or non-occluding (depth values range from $0-1$, near to far plane):

$$z_f - z_c \begin{cases} < 0, \Rightarrow z_f \text{lies within cut volume} \\ \geq 0, \Rightarrow \text{outside of volume} \end{cases} \qquad (1)$$

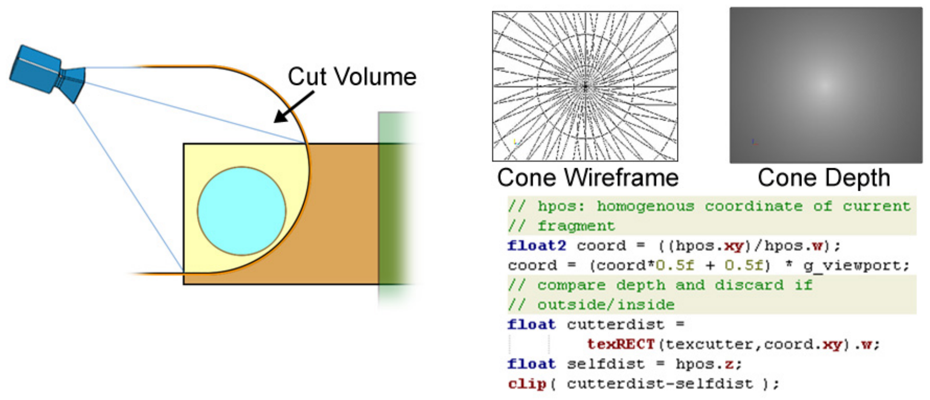
This classification is performed through the use of fragment shaders during rasterization in the programmable graphics pipeline. Using custom criteria, the shaders allow the explicit discard of pixels otherwise contributing to the framebuffer (Fig. 3, right). In the following subsections we will discuss different strategies to define the cut volume, i.e. z_c based on the focus objects. Special care has to be taken in *breakaways* when the surface of the intersection has to be visualized as well (Fig. 4, left).

Masking and Cut Shapes. Two strategies for the shapes that reveal the focus structures, have been explored: object-based and primitive-based. The object-based approach has the benefit of accentuating the object's shape, and allow more control on the impact on the the rest of the scenery. However, the shape might also add too much noise to the scenery, and it might not be clear whether it is part of the effect or not. Using standard primitive shapes, such as cones, boxes and cylinders draws attention due to their regularity compared to the irregular organic shapes. Since tumors typically exhibit blob-like shapes, cylinders and cones seem appropriate containers. As cylinder and cone have only a single curvature on their shell, a straight "virtual ruler" can be aligned to them, to ease distance measuring which will be presented in Sec. 3.2. A rounded cone is less suited for the measuring guide, but removes rapid illumination changes at the tip and interfers less with the rest of the scenery.

3.1 Ghosting

In *ghosting* the visibility of the tumor is provided by fading out the intersection. The masking process can either be binary, i.e. discarding all pixel contributions within the cut volume, or weighted by smoothly blending from occluding structures to those behind. The latter technique eventually is frequently used in artist-created illustrations and is the name-giving element in *ghosting*. When smooth blending is needed, the render setup is as follows:

1. Silhouette Cut: Draw cut volume into *cut buffer*.
   Cylindrical Cut: Determine cylinder properties based on focus objects.



```
// hpos: homogenous coordinate of current
// fragment
float2 coord = ((hpos.xy)/hpos.w);
coord = (coord*0.5f + 0.5f) * g_viewport;
// compare depth and discard if
// outside/inside
float cutterdist =
        texRECT(texcutter,coord.xy).w;
float selfdist = hpos.z;
clip( cutterdist-selfdist );
```

Fig. 3 Since camera is within the convex cut volume (left), the cut volume can be represented by the depth buffer of the cut geometry, i.e. the *cut buffer*. The top images on the right show such a depth buffer for the cone geometry (wire-frame). The cut operation is realized by comparing the cut volume's depth distance with the surface distance for the surface fragment (right).

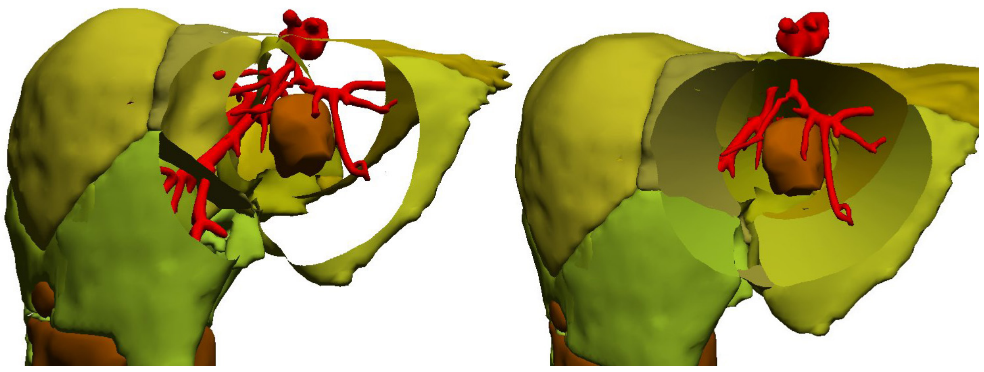


Fig. 4 Simply discarding the surface fragments based on depth (left) leaves too many open areas when dealing with polygonal data. *Breakaways* close these intersection surfaces and therefore lead to a better representation of spatial relationships (right).

2. Draw all intersectable objects without cutting them. Store color output result into a temporary *solid buffer*.
3. Draw all intersectable objects with active cutting.
4. Draw all focus and detail objects.
5. Blend *solid buffer* with the frame buffer based on custom weighting.

The final blend as well as the two intersectable passes are not required when binary masking is used. Next to the resulting speed-up binary masking does not require an appropriate weighting function.

3.1.1 Cylindrical Cut

The cylinder volume is created in the clip space after projection, and is always aligned with the camera plane. Furthermore, the cutting circle edge is highlighted by a colored border. At first, the center point and the radius of the highlighted tumor are calculated. The fragments within the screen space circle will be masked. The depth position of the cylinder also depends on the maximum depth of the tumor, so that structures behind it remain visible.

The fragment shader does not only handle the masking, but also the coloring for the border. Given a cylinder with the center point (x_c, y_c), radius r_c and a maximum depth z_c, as well as the rendered surface fragment with the position (x_f, y_f, z_f), the following comparison is applied:

$$\sqrt{(x_c - x_f)^2 + (y_c - y_f)^2} - r_c \begin{cases} < 0 - \epsilon, \Rightarrow \text{fragment masked using } z_c \\ = 0 \pm \epsilon, \Rightarrow \text{colored border} \\ > 0 + \epsilon, \Rightarrow \text{fragment treated normally} \end{cases} \quad (2)$$

The width of the colored border is controlled by a user parameter ϵ. To determine the cylinder properties, all vertices of the tumor have to be transformed into screen space and are subsequently used to derive the center point in this 2D point set. When the blended *ghosting* is used, the blending weight function is a sim-

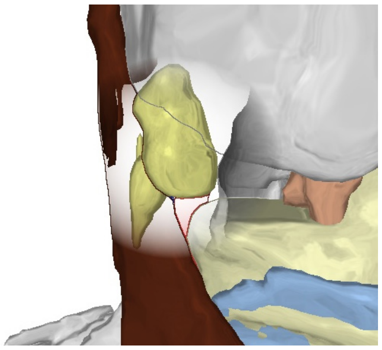


Fig. 5 The silhouette of occluded structures is preserved in this cylindrical *ghosting*. The silhouette was generated through a laplace edge detection filter.

ple power function of the normalized distance to the circle center.

Cylinder Properties. The chosen center and radius are derived from the enclosing circle of the screen space bounding box of all projected vertices and the furthest projected vertex of the focus object is used for z_c. These parameters can be computed efficiently by rasterizing all tumor vertices as *POINTS* into a 1×1 multiple render target, comprised of two floating point textures, *max buffer* and *min buffer*. A special shader renders all vertices at the screen space center, so that all vertices contribute to this render target. Theit screen space positions for the regular camera are written into the two buffers, whilst negating all components for the *min buffer*. As during the process *MAX* blending is active, the lower left and the upper right corners can be reconstructed from reading the two buffers and flipping signs again for the *min buffer* contents. Calculating the bounding circle from the box corners is performed prior the cylinder tests as seen in Eq. 2.

Fading out surface parts or entire objects in front of the focus structures results in a certain information loss. As suggested by Krüger et al. the silhouette of the cut objects shall be made visible within the ghost area [26]. To achieve this, the alpha value of the *solid buffer* encodes to which object the pixel belongs. With 8-bit color buffers, 255 objects can be represented in the scene. An edge detection filter is run on the alpha channel and yields object silhouette lines for otherwise discarded surfaces (see Fig. 5).

3.1.2 Silhouette Cut

When the cut volume is defined by a circle or other regular shape, it may contain a large screen space area that eventually contains no parts of the focus structure at all. As a consequence, many surface parts of ob-

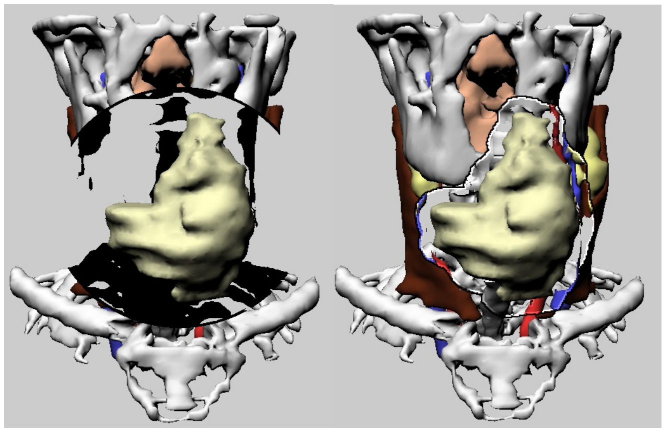


Fig. 6 The *cylinderical ghosting* is used to highlight a tumor within the neck (left). It significantly removes more surrounding structures than a silhouette-based approach (right).

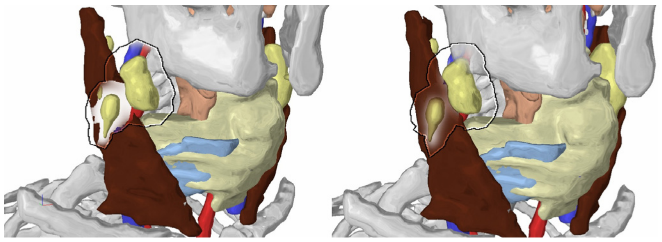


Fig. 7 The back faces for the silhouette depth generation are extruded with a fixed length (left). This length can be decreased when the normals face along the view direction (right). As a result, less surfaces that lie behind the object are affected.

jects around the focus objects are discarded (see Fig. 6, left). Another issue is the combination of multiple focus objects. Either all focus objects contribute to a single circle shape, meaning that even larger areas of wasted space, or multiple shapes need to be tested. For the latter the most efficient approach is to render the shapes into a special screen-sized buffer. Thus, the per-pixel test of each intersection only needs to sample from the property buffer to get all required attributes for the according screen space position. Furthermore, the shape of the focus object might provide additional information, e.g. the shape of the tumor is an important indicator for its malignancy.

This leads to directly using the objects silhouette for the cut volume generation (see Fig. 6, right). The cut volumes of all focus objects are rendered into a special *cut buffer*, which contains the furthest per-pixel distance by using a depth test of *GREATER* when rasterizing the cut volumes. Due to the aforementioned simplifications regarding the cut volumes attributes, it is sufficient to render the backfaces of the focus objects. By applying a vertex shader that extrudes all vertices along the smooth vertex normals, the cut volume is expanded along the object silhouette as well. The extrusion distance is affected by the angle that the normal has towards the viewer. A larger distance is given to normals perpendicular to the viewing direction, so that less of the surrounding structures with similar depth values are affected, as is shown in Fig. 7.

The intersection test (Eq. 1) can be performed efficiently through the use of hardware shadow mapping. To benefit from the GPU's support for fast comparison of depth values with depth buffers, the *cut buffer* is simply implemented as depth render target. For the blended *ghosting* the weight is created through the angle of the objects' frontface normals and the view direction. Overlaps between focus objects are resolved by using *MAX* blending when creating a *weight buffer* used as input for the later weighting. As the anatomic focus objects, such as tumors, lymph nodes and metastases, are similar to blob shapes and are tessellated uniformly, this normal-based approach works well for both weight creation and silhouette extrusion. Finally, a border around the focus structure's ghost region can be created by applying a laplace filter on a buffer that masks all pixels that are affected by the cut volume. For efficiency, this information can be stored into the alpha value of the *weight buffer*.

3.2 Breakaway

The main difference in the *breakaway* compared to the *ghosting* is that the intersection surface between intersectable objects and the cut volume is visualized as well. Similar to the previous technique we will present two different cut volumes that share the same rendering setup. At first, the cut volume is rasterized into the RGBA *cut buffer*. However, this time, not only depth is captured but illumination as well. In the RGB channel of the buffer the classic Phong lighting, which is based on the cut volume's normals, is stored and the alpha value holds the depth information. For more complex shading, the normals could be stored directly. Enough precision for the depth comparison is required in the buffer. In our test scenes, which did not have a high viewing distance, 16-bit floats were sufficient. The algorithm basically works as follows: At first, the structures which intersect with the volume have to be identified. The cut result can be classified into three cases:

1. the structure lies completely outside the volume
2. the structure lies completely within the volume
3. the structure is cut by the volume.

In this identification stage, only for case 3, a visible surface needs to be generated. We chose to use the color of the object, whose front face pixel is closest to the cut volume. The pixels, that passed all tests, are colored based on the illumination information stored in the *cut buffer*.

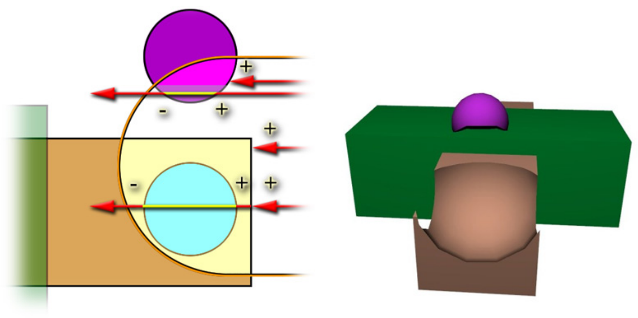


Fig. 8 In the illustration (left) the arrows represent a screen space pixel, ie a ray shot from the camera. Through rasterizing the front and back faces with different stencil operations, the stencil buffer is incremented or decremented accordingly. The stencil bias allows to classify each pixel at the end, whether it is part of a cut surface (positive bias) or not (less or equal null). The right image shows how the cyan ball was entirely removed through this operation, as the object generated only pixels with a bias of null.

To determine the pixels of interest, a combination of stencil and depth test is employed, similar to the setup Coffin and Höllerer [27] have chosen to visualize the border geometry. This setup is required due to the enclosures within the scene. Otherwise a much simpler setup of rendering all fragments outside the volume (backfaces first, then frontfaces) could be used [20]. Only those pixels within the cut volume are tested. Every object in the scene has to be processed in two steps:

- **Mark Phase:** Using the stencil buffer, all pixels in the image, that are part of an intersection, are marked. This is achieved by rendering front and back faces with different stencil operations. For each object the stencil buffer is initialized at the "middle" (e.g. 127 for 8-bit stencil buffer). Thus, incrementing or decrementing may not result into hitting the limit of 0 or 255 early when no wrapping stencil operations are supported. The fragment shader discards all fragments outside of the volume. Front faces increment and back faces decrement the stencil value. Therefore, a positive bias means the object is intersected, because the corresponding back faces were outside the volume and did not close the object. This holds only true if the camera does not intersect with the objects themselves and if no faces are beyond the far plane, i.e. all faces contribute to the stencil buffer and are not clipped. Through the use of the *GL_NV_depth_clamp* extension, this limitation can be circumvented. This procedure is analogous to the setup for *stencil shadow volumes* [28] and illustrated in Fig. 8.

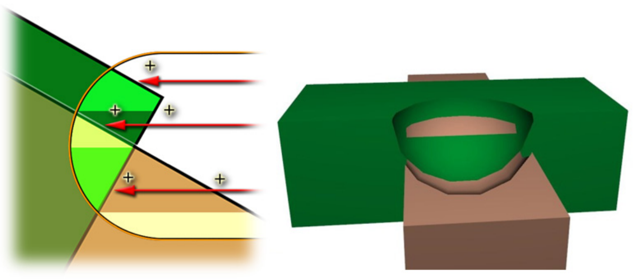


Fig. 9 In the color phase those pixels with a positive bias are colored according to the closest front face to the cut surface. Without a clear volume description, artifact-like visualization may occur when closed surfaces intersect.

– **Color Phase:** The front faces of the object are rendered using the illumination from the *cut buffer*. They use a depth test of *GREATER* and write into the depth buffer.

Between each object's phases the stencil buffer is cleared, but the depth buffer is kept, so that the color contributions of the pixels with the largest z-value are preserved. Given the simplifications of the cut volume, this corresponds to the pixels closest to the cut volume's outer surface, as only the fragments inside the volume are rendered. This works fairly well, even if different objects overlap each other slightly (which frequently happens when segmented meshes are smoothed). However, as the effect is entirely surface-based, no clear definition of each object's volume exists, and therefore ambiguous results will remain (see Fig. 8, left).

After all intersectable objects have been processed, the frame buffer now contains the shaded intersection surfaces. The stencil buffer is used to prevent all the surface pixels from being overwritten. At this point it only contains the intersection of the last object. However, all pixels, where no intersection occurred, still have their depth set to 0, i.e. the near plane. Therefore, a full-screen quad is drawn at the front plane and only those pixels, which are in front of the surface pixels pass the depth test (*LESS*) and mark this intersection in the stencil buffer. In the next step, all intersectable objects are rendered again with a regular depth test only outside the volume, while the stencil test protects the intersection surface pixels. As a result, the color buffer now contains the intersectable objects and their appropriate intersection surfaces.

Before the detail and focus objects are rendered, the depth buffer of the intersection surfaces has to be corrected. The stencil buffer still marks all intersection pixels, so that rendering the cut volume into the depth buffer at appropriate pixels can be achieved. With both

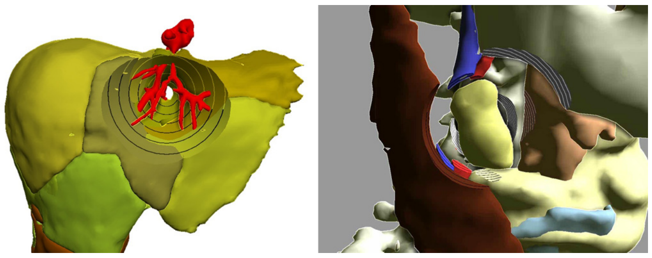


Fig. 10 Distance lines are created on the cone intersection surface. The lines serve as a hint to the penetration depth in a liver segment (left) or thickness of objects near to a highlighted lymphnode (right)

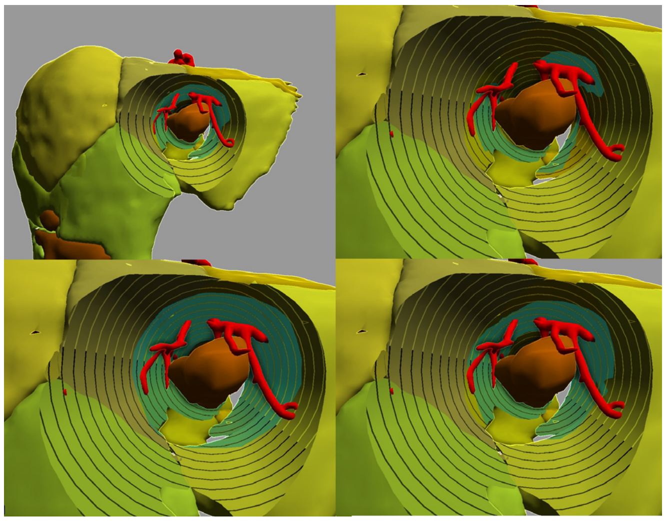


Fig. 11 The intersection of a user specified distance around the tumor with the cutter surface is highlighted by a green surface. The range of interest around the tumor can be changed at runtime as seen in the close-up images. Closer distance lines have been used in the detail images as well.

color and depth buffer now containing the correct information after the intersection, the detail and focus objects can be drawn with the regular depth test.

To further enhance the depth perception, we used the previously mentioned *unsharp masking the depth buffer* technique [21]. The method is simple to implement as it only requires a blurred depth buffer of a given scene as additional input and yields smooth surface shadows (see Fig. 2). The result of this shading operation was combined with an edge detection on the depth buffer. The entire composition pipeline is presented in Fig. 12.

Distance Markers. Measuring distances is a frequent task in pre-operative planning of tumor surgery, in particular to assess resectability within a certain security margin. Therefore, distance lines were added to the intersection surface by modifying the fragment shader

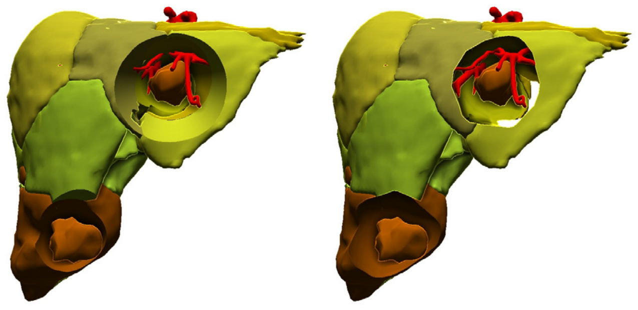


Fig. 13 The rounded cone cut creates smooth intersection surfaces with the liver (left). However, when using a silhouette-based approach, additional clues about the object's shape and distance to the invisible backside are given through the surface shading, which is driven by the back faces (right).

that solely computed lighting before. The lines represent virtual slices that are perpendicular to the direction from the focus object's center to the camera. When a regular shape, such as a cone, is employed for cutting, the effect is equivalent to holding a ruler along the surface. This way the intersection surface is accentuated even more and the lines serve as a guide to estimate thickness of structures nearby (see Fig. 10). The lines are generated by adding an offset and performing a wrapped modulo operation on the fragment's illumination value.

Tietjen et al.[29] have colored the surfaces of surrounding structures depending on the minimal distance to enlarged lymphnodes. A similar effect can be achieved by masking the intersection of the tumor with the cutter's surface. The range of interest can be altered by the user at runtime, as well as the sizes of the distance lines as shown in Fig. 11.

3.2.1 Cone Cut

To generate the cut volume data, a cone or rounded cone can be used (Fig. 13, left). When a rounded tip is chosen, it creates less illumination changes than the sharp cone tip and thus yields smoother intersection surfaces. The cone angle can be defined by the user; the placement, however, is performed automatically based on the bounding sphere of the focus object. The cone mesh has its tip along the local z axis. The cone plate center is placed at the camera position c and oriented to have the z axis point to the bounding sphere center b_c (Fig. 14). The scale factors (s_z and s_{xy}) are derived from the bounding sphere's radius b_r and position, as well as the cone angle α using trigonometry:

$$s_z = (b_r/\sin(\alpha)) \cdot \cos(\alpha) + |(b_c - c)| \tag{3}$$

$$s_{xy} = \tan(\alpha) \cdot s_z \tag{4}$$

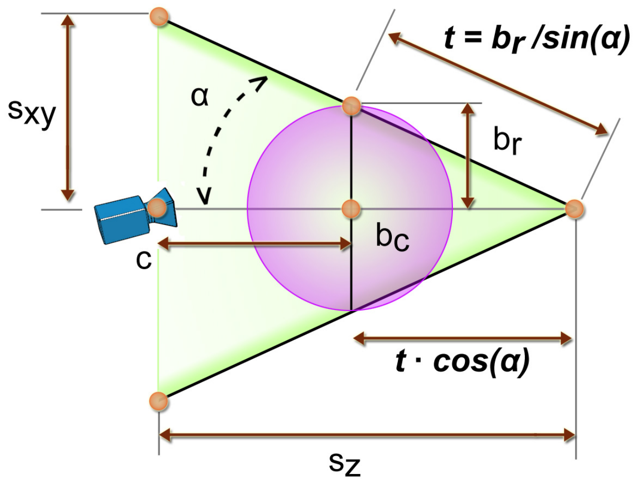


Fig. 14 Through the use of trigonometry, the scale factors (s_z and s_{xy}) for the green cone mesh are computed from the focus object's purple bounding sphere and a user specified cone angle α.

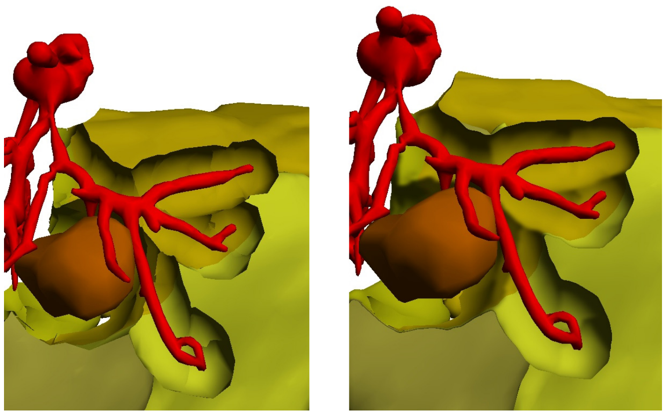


Fig. 15 When using objects with complex shape and topology for the silhouette cut, a lot of sharp illumination changes can occur (left). To decrease these effects, the illumination and depth values of the *cut buffer* are blurred and result into smoother surfaces (right).

This calculation is very lightweight and performed on the CPU. Because of the smooth and angled surface of the cone's mantle the cut is emphasized and easily recognizable within the scenery.

3.2.2 Silhouette Cut

Deformed back faces of the focus objects are also possible cut shapes (Fig. 13, right). In contrast to the method used in Sect. 3.1.2, the vertices are not weighted, but equally translated along their normals. This allows to define cut regions of specific dimensions around the object, which is of frequent interest in preoperational planning. Compared to Burns et al. [20] we create

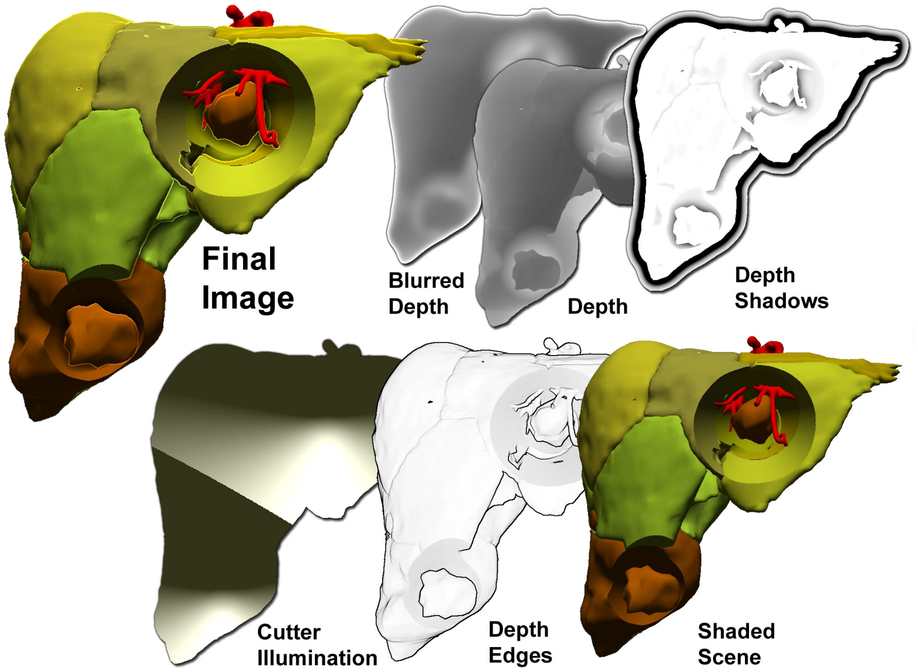


Fig. 12 The final image composition is a result of various effects. The cut surfaces are shaded based on the cutter's surface. Through the use of *unsharp masking the depth buffer*, a shadow effect is employed. Finally, an edge detection is run on the depth buffer and additively blended on top of the image to highlight sharp edges and silhouette lines.

the *cut-buffer* directly by rasterization, which is much faster than computing the distance field. With complex focus shapes, the resulting cut volume can create a lot of noise, as many different shapes intersect each other to create the final distant cut surface. To reduce this effect, the *cut buffer* can be blurred, which results into a smoother cut surface and less harsh illumination changes (see Fig. 15). As a side effect of preferring smooth data, the *cut buffer* can be rendered at lower resolution than the main buffer. That increases the blur performance and does not hinder quality due to bilinear filtering when sampled at full resolution. This approach to generate the silhouette cut is an original optimization for medical data, as other CSG methods have to deal with surfaces where hard edges appear.

4 Results

Both techniques and their variants have been tested on a system composed of an NVIDIA GeForce 9600 GT graphics card and an Intel Core2Duo 2.3 GHz, 2 GB RAM. All rendering was performed at a resolution of 1024 × 768. Implementation was done using the Luxinia 3D engine, which is written in C and programmed through the high-level language *Lua*. The engine uses *OpenGL* and *Cg* for the rendering. Two different datasets were created through the use of MeVisLab and served as test scenes (see Tab. 1). The rounded cone geometry consists of 1000 vertices and 2000 triangles and is drawn for each of the two focus objects.

Table 1 Test Scenes

Name	Vertices	Triangles	Objects
ghosting, neck	50,000	85,000	23
breakaway, liver	35,000	75,000	11

In Tab. 2 all techniques are compared among each other. When blending is enabled, the *ghosting* effect has about half the performance, as the scene has to be drawn twice. The performance impact of deriving the cylinder properties for the tumors (300 vertices) is mostly bound to a certain setup overhead, and less to the amount of actual vertices. Therefore, it will mostly be slower than the silhouette approach. The *breakaway* is the slowest method due to many objects being drawn multiple times. The depth shading requires additional render-to-texture setups and various blur passes, as a consequence the effect is quite costly. The effects are mostly bound by overheads introduced through their setup and due to per-pixel operations, so the object's resolution could be increased without significant performance loss. All methods presented still deliver very high frame rates, even when all features are enabled.

Table 2 Rendering Speed

Method	FpS	FpS Fx
ghosting, none	1800	–
ghosting, cylinder	700	400
ghosting, silhouette	720	370
breakaway, none	1600	1000
breakaway, cone	350	215
breakaway, silhouette	400	240
breakaway, silhouette blurred	270	180

The Fx version means that for *ghosting* blending was enabled and in case of *breakaway*, the depth buffer shading and distance lines were activated.

The results have been presented to ENT surgeons, who perform neck surgery. Especially the possibility to reveal intersection surfaces in a user defined distance to the tumor was found useful to analyze infiltrations. It was also stressed by them, that the various possibilities are suitable for an authoring system for educational purposes. The 3D anatomy of the neck region could be conveyed adequately when learning to understand sonographic image data. Both, regular cut shapes and object-based shapes were found useful for different highlighting tasks. The *breakaway* methods for the liver have been presented to a liver surgeon. Following his input, we experimented with using those liver segments as cut volume, which would be removed during surgery. Additionally we overlayed the entire vessel system with transparency to make it always visible (see Fig. 16). The surgeon had concerns regarding the contour edge highlights, but supported the idea of using the contour shadows. Furthermore, he suggested using the resection plane, a free-form surface that separates the kept from removed liver tissue, as source for the cut volume. A quantitative evaluation could not be performed, as a direct comparison with volume rendering techniques on the same datasets was not possible. Whether the rendering of polygonal meshes or volumes provide a more true to life visualization is outside of the scope of this work.

5 Conclusion

The presented smart visibility techniques can be used in therapy planing. The *ghosting* is applicable to emphasize one or more tumors, as only a small region surrounding the tumor is modified. For larger structures the diameter and depth of the cylindrical cut becomes a problem. The silhouette-based method results into less discarded areas. As the surrounding surfaces are entirely removed, it is harder to estimate the distances to close structures.

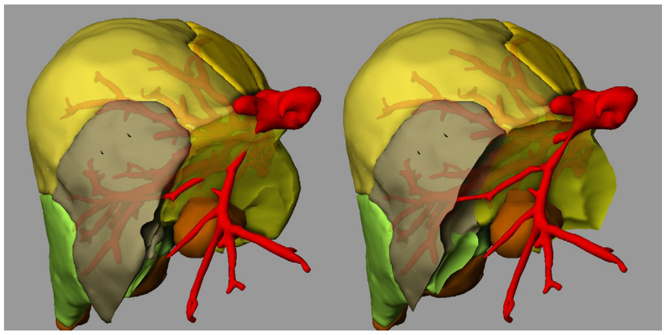


Fig. 16 The *breakaway* is generated from the geometry of two invisible liver segments. These segments would be removed along with the tumor during surgery. Different ranges to the segments were chosen for the cut volume, to visualize the impact on the other segments. As suggested by a liver surgeon, the vessel system is kept visible by overlaying it with transparency.

When the technique is further developed to a *breakaway*, it can overcome this disadvantage. The surrounding tissue of the target structure can be explored better, as the cut surfaces provide additional spatial information. Additional shading effects, similar to those in manually created illustrations, may further enhance the perception.

For a detailed analysis, i.e. surgical neck tumor dissection, an evaluation is planned to determine whether the target group can estimate spatial relationships faster and more accurately. As a first step, Baer et al. have presented an experimental user study of emphasis techniques used in medical visualizations for focus and context illustration [3]. The cylindrical *ghosting* as well as other techniques were used in their framework to support the detection of enlarged lymph nodes in the neck. An evaluation of the presented work should be helpful to determine for which application cases the many possible parameters and variants of the techniques are suited. Furthermore, automatizing parameter choices, such as cone angle and guideline dimensions based on distance to the focus structure, should be investigated.

The techniques were implemented efficiently for polygonal meshes. These meshes are a common result when processing segmented structures. Certain limitations and pre-requisites were chosen to achieve this. The *breakaway* and the silhouette-based variants only work well when all objects consist of closed surfaces. We assume that the camera is within the cut volume, and that the volume is convex, i.e. can be represented solely by image space depth. Currently, no transparent objects can be used, as this would add another level of complexity in combination with *depth peeling* approaches like Everitt et al. [30]. By generating a second depth for the entry positions into the cut volume, it would be possible to create arbitrary shadow maps. These could be used to deliver more accentuated shadows, which help to increase depth perception.

Further advances in hardware capabilities, such as arbitrary read and write buffers and finer control on atomic operations, may, however, soon make it possible to create single-pass implementations of the methods. Another recent possibility is using hardware ray-tracing of polygonal scenes (*Nvidia Optix*). Ray-tracing the scene would make it much easier to generate the cut views and shadows. The benefit of the methods presented here is that they run on a variety of current generation consumer hardware. Once the capability for *hardware shadow mapping* exists, it is even possible to recreate the effects with the fixed-function *OpenGL ES 1.x* pipeline as found in mobile devices.

Acknowledgements. We thank the surgeons C. Logge, G. Strauß and A. Böhm for their time and valuable feedback during the informal evaluation. We thank T. Grosch, S. Glaßer and the reviewers for their input.

References

1. Mhler K, Neugebauer M, Tietjen C, Preim B. Viewpoint Selection for Intervention Planning. In: IEEE/Eurographics Symposium on Visualization (EuroVis); 2007. p. 267–274.
2. Mhler K, Bade R, Preim B. Adaptive script based animations for intervention planning. In: Proc. of Medical Image Computing and Computer-Assisted Intervention (MICCAI); 2006. p. 984–991.
3. Baer A, Adler F, Lenz D, Preim B. Perception-based Evaluation of Emphasis Techniques Used in 3D Medical Visualization. In: Vision, Modeling, and Visualization Workshop. Braunschweig; 2009. p. 295–304.
4. Ling H, Zhou SK, Zheng Y, Georgescu B, Sühling M, Comaniciu D. Hierarchical, learning-based automatic liver segmentation. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition; 2008. p. 1–8.
5. Braude I, Marker J, Museth K, Nissanov J, Breen DE. Contour-based surface reconstruction using MPU implicit models. Graphical Models. 2007;69(2):139–157.
6. Bade R, Konrad O, Preim B. Reducing Artifacts in Surface Meshes Extracted from Binary Volumes. Journal of WSCG. 2007;15(1-3):67–74.
7. Elmqvist N, Tsigas P. A Taxonomy of 3D Occlusion Management for Visualization. IEEE Trans Vis Comput Graph. 2008;14(5):1095–1109.
8. Viola I, Gröller ME. Smart Visibility in Visualization. In: Proc. of EG Workshop on Computational Aesthetics in Graphics, Visualization and Imaging; 2005. p. 209–216.
9. Feiner S, Seligmann DD. Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. The Visual Computer. 1992;8(5&6):292–302.
10. Bichlmeier C, Wimmer F, Heining SM, Navab N. Contextual Anatomic Mimesis Hybrid In-Situ Visualization Method for Improving Multi-Sensory Depth Perception in Medical Augmented Reality. In: ISMAR '07: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality; 2007. p. 1–10.
11. Beyer J, Hadwiger M, Wolfsberger S, Bühler K. High-Quality Multimodal Volume Rendering for Preoperative Planning of Neurosurgical Interventions. IEEE Transactions on Visualization and Computer Graphics. 2007;13(6):1696–1703.
12. Krüger J, Schneider J, Westermann R. ClearView: An Interactive Context Preserving Hotspot Visualization Technique. IEEE Trans Vis Comput Graph. 2006;12(5):941–948.
13. Viola I, Kanitsar A, Gröller ME. Importance-Driven Volume Rendering. In: Proc. of the IEEE Visualization; 2004. p. 139–145.
14. Diepstraten J, Weiskopf D, Ertl T. Interactive Cutaway Illustrations. Computer Graphics Forum. 2003;22(3):523–532.
15. Rautek P, Bruckner S, Gröller ME. Interaction-Dependent Semantics for Illustrative Volume Rendering. Computer Graphics Forum. 2008;27(3):847–854.
16. Krüger J, Westermann R. Acceleration Techniques for GPU-based Volume Rendering. In: Proc. of the IEEE Visualization; 2003. p. 38–45.
17. Li W, Ritter L, Agrawala M, Curless B, Salesin D. Interactive Cutaway Illustrations of Complex 3D Models. In: ACM SIGGRAPH 2007 papers. ACM Press; 2007. p. 31.
18. Elmqvist N, Assarsson U, Tsigas P. Employing Dynamic Transparency for 3D Occlusion Management: Design Issues and Evaluation. In: Proc. of Human-Computer Interaction - INTERACT (1); 2007. p. 532–545.
19. Kirsch F, Döllner J. OpenCSG: A Library for Image-Based CSG Rendering. In: Proc. of the USENIX Annual Technical Conference. USENIX; 2005. p. 129–140.
20. Burns M, Finkelstein A. Adaptive cutaways for comprehensible rendering of polygonal scenes. In: Proc. of ACM SIGGRAPH Asia; 2008. p. 1–7.
21. Luft T, Colditz C, Deussen O. Image Enhancement By Unsharp Masking The Depth Buffer. ACM Transactions on Graphics. 2006;25(3):1206–1213.
22. Mittring M. Finding next gen: CryEngine 2. In: SIGGRAPH '07: ACM SIGGRAPH 2007 courses. New York, NY, USA: ACM; 2007. p. 97–121.
23. Sloan PPJ, Martin W, Gooch A, Gooch B. The lit sphere: a model for capturing NPR shading from art. In: GRIN'01: Graphics interface 2001. Toronto, Ont.; 2001. p. 143–150.
24. Bruckner S, Gröller ME. Style Transfer Functions for Illustrative Volume Rendering. Computer Graphics Forum. 2007 Sep;26(3):715–724.
25. Tietjen C, Isenberg T, Preim B. Combining Silhouettes, Surface, and Volume Rendering for Surgery Education and Planning. In: IEEE/Eurographics Symposium on Visualization (EuroVis); 2005. p. 303–310.
26. Krüger A, Tietjen C, Hintze J, Preim B, Hertel I, Strauß G. Interactive Visualization for Neck Dissection Planning. In: Proc. of the IEEE/Eurographics Symposium on Visualization; 2005. p. 295–302.
27. Coffin C, Höllerer T. Interactive Perspective Cut-away Views for General 3D Scenes. In: Proc. of the IEEE Symposium on 3D User Interfaces; 2006. p. 25–28.
28. McGuire M, Hughes JF, Egan K, MK, Everitt C. Fast, Practical and Robust Shadows. Austin, TX: NVIDIA Corporation; 2003.
29. Tietjen C, Dornheim J, Krüger A, Preim B, Hertel I, Strauss G. Computer Assisted Surgery Planning for Neck Dissections. In: Computer Aided Surgery around the Head – 3rd International Symposium; 2005. .
30. Everitt C. Interactive Order-Independent Transparency. NVIDIA; 2001.