High-Quality Multimodal Volume Rendering for Preoperative Planning of Neurosurgical Interventions

Johanna Beyer, Markus Hadwiger, Stefan Wolfsberger, Katja Bühler

Abstract— Surgical approaches tailored to an individual patient's anatomy and pathology have become standard in neurosurgery. Precise preoperative planning of these procedures, however, is necessary to achieve an optimal therapeutic effect. Therefore, multiple radiological imaging modalities are used prior to surgery to delineate the patient's anatomy, neurological function, and metabolic processes. Developing a three-dimensional perception of the surgical approach, however, is traditionally still done by mentally fusing multiple modalities. Concurrent 3D visualization of these datasets can, therefore, improve the planning process significantly. In this paper we introduce an application for planning of individual neurosurgical approaches with high-quality interactive multimodal volume rendering. The application consists of three main modules which allow to (1) plan the optimal skin incision and opening of the skull tailored to the underlying pathology; (2) visualize superficial brain anatomy, function and metabolism; and (3) plan the patient-specific approach for surgery of deep-seated lesions. The visualization is based on direct multi-volume raycasting on graphics hardware, where multiple volumes from different modalities can be displayed concurrently at interactive frame rates. Graphics memory limitations are avoided by performing raycasting on bricked volumes. For preprocessing tasks such as registration or segmentation, the visualization modules are integrated into a larger framework, thus supporting the entire workflow of preoperative planning.

Index Terms—Multimodal Volume Rendering, Hardware Assisted Raycasting, Surgery Planning.

1 Introduction

Minimally invasive neurosurgical procedures are constantly gaining importance with the aim to minimize surgical trauma, shorten recovery times and reduce postoperative complications. For surgery of deep-seated structures, *neurosurgical keyhole procedures* are becoming standard, where a small opening in the skull is sufficient to gain access to a much larger intracranial region via an endoscope or operating microscope. In contrast, interventions in areas directly below the skull require a larger and individually tailored opening of the cranial bone. For both approaches (i.e., surgery of deep-seated structures and near the brain's surface), orientation is necessary to perform the skin incision and bone cover removal at the optimal location. For deep-seated targets, further orientation is crucial to find the structures of interest while additionally preserving the surrounding tissue.

Preoperative planning enables the surgeon to identify anatomical landmarks and critical structures (e.g., large vessels crossing the path of the operating microscope or critical cranial nerves) and in determining the optimal position of incision prior to surgery. It is during this planning session that the physician decides upon the optimal approach by adapting the general surgical plan to the individual patient's anatomy. The medical doctor uses this knowledge during surgery to determine the current location in the skull and the subsequent optimal course of action. Therefore, the success of a surgery, especially in keyhole approaches, depends largely on accurate preoperative planning.

Up to now, the standard approach of presurgical planning is performed using stacks of raw images obtained from medical scanners such as CT (Computed Tomography) or MRI (Magnetic Resonance Imaging). In the field of neurosurgery, MR scans are the medium of choice for depicting soft tissue such as the brain, whereas CT scans are superior in picturing bony structures. Functional MR (fMR) images depict neural activity, Positron Emission Tomography (PET) shows metabolical activity and Digital Subtraction Angiography (DSA) depicts vessels in high quality. However, a mental combination of all these datasets and a correct 3D understanding by simple slice-by-slice analysis is very difficult, even for the skilled surgeon.

- *Johanna Beyer, Markus Hadwiger and Katja Bühler are with the VRVis Research Center, E-mail: {beyer,msh,buehler}@vrvis.at.*
- *Stefan Wolfsberger is with the Department of Neurosurgery, Medical University Vienna, E-mail: stefan.wolfsberger@meduniwien.ac.at.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

3D visualization alleviates this problem by enhancing the spatial perception of the individual anatomy and, therefore, speeding up the planning process. Considering the neurosurgical background, a preoperative planning application has to meet certain requirements: First of all, it should provide a *high-quality, interactive and flexible 3D visualization* of the volumetric datasets using direct volume rendering (DVR). The main advantage of DVR compared to surface rendering lies in the increased amount of information that can be conveyed in one image, as the entire volumetric dataset is used to create the final rendering. Next, a preoperative planning application should offer *multimodal visualization* of datasets from different imaging modalities such as CT, MRI, fMRI, PET or DSA. *Interactive manipulation* of the visualization such as simulated surgical procedures, endoscopic views or virtual cutting planes should be available and, finally, an *intuitive workflow* is necessary, which is integrated into an application framework and ready for use by surgeons or medical staff.

This paper describes an application for preoperative planning of tailored neurosurgical procedures. Our application consists of a multi-volume rendering framework for the concurrent and fused visualization of multimodal datasets (see Figure 1 for several examples), including three main tasks:

- *Planning of the surgical approach to access the brain*, by simulating the skin incision and removal of the cranial bone without any prior segmentation.
- *Visualization of superficial brain areas*, including information from additional volumes such as DSA, fMR or PET to provide further insight into the data.
- *Visualization of deep-seated structures of the brain for (keyhole) surgery*, by including segmentation information.

All visualization modules are integrated into a framework that is designed to support surgeons in the task of preoperative planning, including a preprocessing stage for registration and optional segmentation of the different datasets.

Rendering performs real-time GPU raycasting with perspective projection, in general using a single raycasting pass for 32-bit floating point computations and blending. We employ efficient empty space skipping, early ray termination, and bricking for memory management of multiple volumes. Raycasting is performed through several volumes at the same time, potentially taking into account multiple volumes at a single sample location.

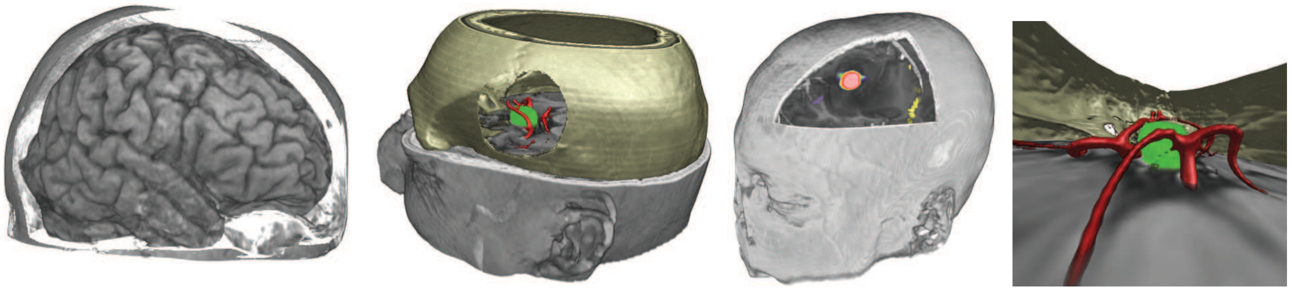

(a) (b) (c) (d)

Fig. 1: (a) Visualization of the brain without prior segmentation using our skull peeling algorithm. (b) Multi-volume rendering of segmented data (green: tumor - MR, red: vessels - MRA, brown: skull - CT). (c) Multi-volume blending (black/white: brain - MR, red: metabolic active part of tumor - PET, yellow: brain areas active during speech - fMR). (d) Perspective volume rendering for simulating keyhole surgery.

The technical contributions of this paper are:

- Unified handling of multi-volume raycasting and bricking with and without segmentation masks. For each sample location, the volume to be sampled is either chosen depending on segmentation information, or multiple volume samples are blended. We circumvent GPU memory constraints by bricking each volume (CT, MR, DSA, PET, fMR), and downloading only active bricks into 3D cache textures (one per modality or unified). Segmentation information is represented as a bricked object ID volume over all modalities, which likewise employs a 3D cache texture.
- *Skull peeling* for selectively removing structures obscuring the brain (skin, bone) without segmentation (Figure 4). In contrast to opacity peeling [18], we consider registered CT and MR data at the same time for more dependable results. The impact of clipping is resolved consistently. Layers can also be peeled away selectively by painting 2D clipping areas (Figure 9).
- The result of skull peeling is view-dependent. In order to employ it for powerful view-independent clipping, we first generate a view-dependent depth map, which is then transformed into volume space and resampled into a static segmentation mask.
- Smooth rendering of segmented object boundaries, taking into account the contributions of multiple volumes. In contrast to our earlier work [9], we do not propose a general solution, but an approach that is customized for the needs of our neurosurgery pipeline that achieves better results in this case. During raycasting, the precise transition between two adjacent materials is reclassified depending on user-specified iso-values and searching the object ID and data volumes along the gradient direction.

2 Related Work

Preoperative planning for neurosurgery has been an active research topic for several years [21, 11, 16], with the main focus often on the integration of imaging, registration, and segmentation into a planning workstation [5], but often falling short of a high-quality visualization of multi-volume datasets. The virtual tumor resection planning of Serra et al. [21] uses volume slicing with basic support for multiple volumes. Even though volume slicing approaches achieve a fast and flexible visualization, they usually do not reach the quality of raycasting methods, especially with respect to close-up perspective views. Other approaches employ iso-surface rendering [16], or the extraction of 3D contours that can subsequently be blended into a mono-volume rendering [11]. This, however, is not optimal for versatile preoperative planning as it does not offer the amount of flexibility needed by surgeons for interactive exploration, e.g., changing the transfer function.

For rendering multimodal data, several methods have been proposed [3, 28, 8, 7]. Their key differences lie in the way how the volumes are combined. Different data intermixing levels (e.g., accumulation level, illumination level, image level) and fusion methods (e.g., one or multiple properties per sample) are used depending on the characteristics of the volumes and the desired results. Manssour et al. [14] use an MRI volume to define the opacity transfer function, while a PET volume determines the color transfer function. Clusters [28] and specialized volume rendering hardware [8, 19] have also been used. Recently, Rößler et al. [19] introduced a slice-based multi-volume rendering method to display a template brain along with patient-specific fMR data, including advanced clipping techniques and render modes. All of the above methods, however, do not address the problem of high-quality rendering of segmented multimodal data. High-quality renderings of the human brain from cranial MR scans usually require segmentation. However, this segmentation process, called skull stripping [1, 22], is not trivial and automatic methods often have problems with noise or require certain MR sequences or scanners. If the brain is rendered without prior segmentation, it is occluded by surrounding tissue of similar intensity values (e.g., skin). Adjusting the transfer function alone, including multi-dimensional transfer functions [12], cannot solve this problem. Methods such as opacity peeling [18] or confocal volume rendering [15] peel away outer, less important regions of a volume to visualize the inner structures. These methods, however, are hard to use in clinical applications because the visual results are very sensitive to several user-defined parameters. For high-quality rendering of segmented data, object boundaries must be determined at the subvoxel level [23, 9, 24], mostly using linear or cubic filtering. Tiede et al. [23] propose a CPU-based method for threshold-segmented objects. They compare the intensity of each sample to the objects in its 2^3 neighborhood to assign the object ID. If the objects have not been segmented via thresholding, trilinear filtering of object masks is used. They also propose to extend their approach to multimodal data. Two-level volume rendering [9] is a flexible rendering method for segmented data with trilinear object boundary filtering and per-object transfer functions and rendering modes. We build on previous research in the area of multimodal volume rendering, and especially GPU-based raycasting. While first approaches were based on slicing [26, 17], GPU raycasting is now a viable and very powerful alternative [13]. The basis for our volume rendering framework is a GPU-based raycaster [20] (requiring Shader Model 3.0) that achieves interactive frame rates also for large volumes. However, we have extended this raycaster considerably in order to support multiple volumes, segmentation masks, flexible per-object as well as view-dependent clipping, and rendering modes tailored for neurosurgical applications.

3 Workflow

For daily use in clinical environments it is crucial for CASP applications (Computer Aided Surgical Planning) to be integrated directly into the clinical workflow. CASP applications should support the surgeon, who usually has a very tight schedule, by offering an intuitive, easy-to-use interface. Therefore, we have integrated our rendering framework as a plugin into the medical workstation and PACS system Impax 6.0 by Agfa Healthcare[1]. The complete workflow of our planning tool can be seen in Figure 2 and contains the following steps: Data acquisition, registration, segmentation, planning of the skin incision and bone removal area, brain surface visualization, and surgery planning for deep-seated structures, tailored to the individual anatomy. The application consists of a preprocessing stage, which was intentionally

[1]http://www.agfa.com

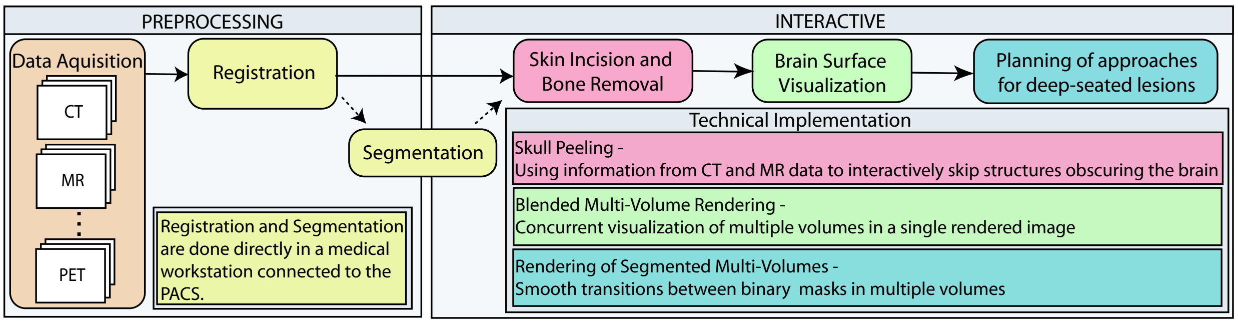

PREPROCESSING
INTERACTIVE
Data Aquisition
CT
MR
PET
Registration
Segmentation
Registration and Segmentation are done directly in a medical workstation connected to the PACS.
Skin Incision and Bone Removal
Brain Surface Visualization
Planning of approaches for deep-seated lesions
Technical Implementation
Skull Peeling - Using information from CT and MR data to interactively skip structures obscuring the brain
Blended Multi-Volume Rendering - Concurrent visualization of multiple volumes in a single rendered image
Rendering of Segmented Multi-Volumes - Smooth transitions between binary masks in multiple volumes

Fig. 2: Detailed workflow of our neurosurgical planning application.

kept as short as possible, and an interactive stage for rendering. After acquisition of the radiological images, the different datasets are registered and segmented directly in the medical workstation before starting the interactive visualization. Segmentation is not mandatory but helps later on by giving additional information during the visualization of deeper structures. Using skull peeling (Section 5.2), the surgeon can take a look at the brain's surface and optionally define the area on the patient's head for the skin incision and ensuing bone removal. The superficial brain can be displayed via direct multi-volume rendering (Section 5.3), showing the brain (MRI) along with other structures such as vessels (DSA), implanted electrodes (CT), or functional brain areas (fMRI, PET). Next, the surgeon may navigate the viewpoint through the keyhole in the cranial bone to access deeper brain structures, using multi-volume rendering of segmented data (Section 5.4). Additionally, binary segmented objects can be displayed in high quality and without staircase artifacts by an on-the-fly subvoxel classification algorithm based on raycasting (Section 5.5). During the entire interactive visualization process, tools for further exploration of the data are available to the surgeon, such as MPR (multiplanar reconstruction) views, clipping geometry, or endoscopic views.

4 Preprocessing Stage

In the registration module, the different multimodal datasets are aligned and resampled. We use an automatic registration algorithm based on mutual information [4] and perform rigid registration (i.e., only translation and rotation). The algorithm usually converges after a few seconds. If the result is not satisfactory, the user can improve the registration manually by interacting with three orthogonal slice views. The segmentation module implements manual segmentation, thresholding and watershed segmentation based on markers [6]. Watershed based on markers is a semi-automatic segmentation algorithm where the user has to draw initial markers for the different objects into the volume data. For each voxel its most probable membership to an object is calculated. The algorithm completes within seconds, however, manual refinement of the initial markers is necessary most of the time. The binary segmented objects are saved in an additional segmentation volume that defines the object membership of each voxel.

5 Visualization Modules

The three visualization modules illustrated on the right-hand side of Figure 2 constitute the main part of our application. Section 5.1 provides a technical introduction to our multi-volume rendering system, which is the basis for all visualization modules. Specifics of the modules are then described in Section 5.2 for *skull peeling* in order to perform view-dependent clipping of "uninteresting" parts of a volume, Section 5.3 for *multi-volume blending* in order to visualize unsegmented data, and Section 5.4 for *segmented multi-volume rendering* for visualizing segmented objects from multiple modalities. Rendering smooth boundaries of segmented volumes is described in Section 5.5. General interaction tools which support the surgeon in the exploration and planning process are explained in Section 5.6.

5.1 Multi-Volume Rendering

A very important issue in multi-volume rendering is how the contributions of different modalities are combined. Our rendering pipeline offers several options that are part of a single consistent framework. Contributions of multiple volumes are combined on a per-sample basis during raycasting. For a given sampling location, either a single volume is sampled and mapped to optical properties via a transfer function, or the contributions of multiple volume samples taken at the same relative location in each modality are blended *after* separate transfer functions have been applied.

A single object ID volume guides these choices for combining multiple volume contributions. Table 1 gives an overview of the most important texture types that are used in our implementation. Each voxel is assigned the ID of the segmented object it belongs to (*tex_objectID*), and each object is assigned the volume it belongs to using a 1D mapping (*tex_volumeID*). For example, "bone voxels" would be assigned to the CT volume, whereas "brain voxels" would be assigned to the MR volume. Each object can use its own transfer function, which is determined using its object ID [9]. The mapping of object ID to volume ID implicitly solves the problem of using different transfer functions for different volumes, as long as each object uses only a single transfer function. Additionally, special *blending object IDs* known by the raycasting shader are used to indicate that for this object a given configuration of multiple volumes should be blended at each sampling location, where each volume uses its own transfer function and blending weight. We currently support only a fixed number of useful configurations, such as blending the MR and the DSA volume used in the visualization module described in Section 5.3. However, adding additional configurations is easy, and our system could easily be extended to full generality using additional look-up textures.

Table 2 illustrates the basic texture look-ups used by multi-volume rendering. The transfer function TF_j in Table 2 is determined by the object ID, which either means $j = o(\mathbf{x})$ for regular object IDs, or a set of predefined j's is used by the shader because $o(\mathbf{x})$ is a blending object ID and thus uses multiple transfer functions. Each object can further have its own set of up to six axial clipping planes, whose positions are obtained in the shader by sampling two 1D look-up textures for minimum and maximum (x,y,z) coordinates, respectively (Table 1). Note that segmentation information is not strictly required. If no object ID volume is available, all parts of our pipeline will implicitly assume that all voxels belong to the same default object. In this case, the only option for combining different volumes is blending them.

Data and Memory Management: Another important challenge in multi-volume rendering is volume data management and coping with memory consumption. We use a bricked volume rendering scheme that subdivides each volume into equally-sized (e.g., 16^3 or 32^3) bricks and maintains 3D brick cache textures. For rendering, active bricks of the volume are held in GPU cache memory, and small 3D

Texture	Dim.+ Type	Function
tex_objectID	3D (I)	map sample position to object ID
tex_volumeID	1D (I)	map object ID to volume ID
tex_volume$_i$	3D (I)	texture cache for volume i
tex_clipmin	1D (RGB)	map object ID to clip planes (min)
tex_clipmax	1D (RGB)	map object ID to clip planes (max)
tex_tf	2D (RGBA)	packed 1D textures TF_j for all j

Table 1: Basic multi-volume rendering textures. Texture type I is single-channel (intensity), and RGB is three-channel, to store three axial clipping plane positions (min or max for x, y, and z, respectively).

obtain ... at $\mathbf{x}$		as
object ID	$o(\mathbf{x})$	$= \text{sample}\big(\text{tex\_objectID}, \mathbf{x}\big)$
volume ID	$i(\mathbf{x})$	$= \text{sample}\big(\text{tex\_volumeID}, o(\mathbf{x})\big)$
volume scalar	$s_i(\mathbf{x})$	$= \text{sample}\big(\text{tex\_volume}_{i(\mathbf{x})}, \mathbf{x}\big)$
transfer function	$\text{TF}_j(\mathbf{x})$	$= \text{sample}\big(\text{tex\_tf}, (s_i(\mathbf{x}), j)\big)$

Table 2: Basic multi-volume rendering quantities and texture look-ups.

layout textures are used for address translation [10] between "virtual" volume space and actual cache texture coordinates. Figure 3 shows an overview of our system. One cache stores segmentation information (an object ID per voxel), with its corresponding layout texture. Each data volume (e.g, CT, PET) has its own layout texture, and either also uses an individual cache texture, or references bricks in a larger unified cache for multiple modalities. Individual caches are less flexible because their size must be chosen at startup, whereas a unified cache allows to change the memory limit dynamically for each modality. However, in this case the GPU must support the texture dimensions required by the larger unified cache. In general, raycasting is performed in a single rendering pass [20] through "virtual" volume space. At each sampling location, one or multiple address translations are performed, and the sample from each modality is obtained from the corresponding cache texture. Although address translation is a relatively fast process, it can optionally be reduced by using a global cache layout for all volumes. This, however, leads to a higher number of active bricks and thus requires bigger caches, since in this case culling cannot be performed independently for each modality.

5.2 Skull Peeling - Surgical Approach to the Brain

This section describes the *skull peeling* algorithm for directly displaying the unoccluded brain from MR data without the need for prior segmentation. Keeping the requirements for clinical applications in mind, the intention is to reliably visualize the brain without the need of tedious preprocessing or complex user interaction. Naturally, a manual segmentation of the brain would achieve the best visual results, but would require a much longer preprocessing time we want to avoid.

Our algorithm is based on the idea of opacity peeling [18], a view-dependent method for peeling away layers in DVR guided by accumulated opacity. Although opacity peeling quickly generates meaningful images, a major problem for medical practice is its dependency on the threshold parameters. Minor changes of these settings can cause major changes in the resulting images (such as a shrinking or expanding brain). Thus, it is an important goal to improve reliability.

Skull Peeling: The skull peeling algorithm simultaneously uses the information of registered CT and MR volumes in order to remove areas along the viewing ray which occlude the brain in the MR volume [2]. While the brain is depicted well in MR scans, CT scans are superior in depicting bony structures with very high intensity values. We exploit this knowledge to decide automatically if a sample lies within a bony area (i.e., the value of the CT dataset is above 1000 Hounsfield units). During raycasting, both the CT and the MR volume are sampled. When the current ray hits a bone for the first time, the accumulated opacity and color values from the MR volume are reset and the ray is advanced until the bony area is exited. At that point, accumulation starts again in order to reveal the brain. This algorithm needs no user input and works well in standard cases where the brain is surrounded by bone (see Figure 1a).

However, when the brain is not surrounded by bone (e.g., after surgery, or when clipping planes are enabled during rendering) this algorithm would fail. The ray would hit a bone for the first time after traversing the brain and everything in front of that hitpoint would be skipped. We therefore added the following extensions:

- The position of the first hitpoint of the skin (i.e., first sample with a density higher than air) is saved. If the ray does not hit a bone within a certain number of steps (defined by threshold T_1) we assume that there is no bone in front of the brain and use the radiance accumulated from the first hitpoint.
- A second extension to the algorithm was made to improve visualization near the skull base. When looking at the brain from below, along the spinal chord, many small bone pieces occlude the view of the brain. We introduce a threshold T_2 which specifies the minimum distance two bony areas must have in order to assume the area in-between to be brain. If this distance is not reached, the area between these two bone areas is skipped and not rendered.

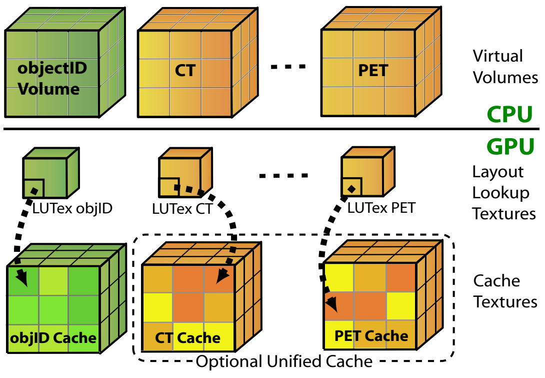


Fig. 3: Bricked memory management for multi-volume rendering. No volume is required to be resident in GPU memory in its entirety. Raycasting is performed in "virtual" volume space, with addresses translated to actual cache texture coordinates using layout look-up textures.

Both thresholds have default values that usually work very well and only need to be adjusted for special cases (e.g., looking at the brain from below). Figure 4 outlines the standard case of skull peeling and a case where threshold T_1 is needed.

Clipped Skull Peeling: Finding the ideal position for the skin incision and subsequent bone removal is important for minimizing the invasiveness of a surgery. For this purpose, we introduce clipped skull peeling to visualize the simulated surgical approach (Figure 11a). The user input consists of the surgeon drawing the areas for the skin incision and subsequent bone removal directly onto the volume-rendered image of the head. After skin removal, the skull is rendered with shaded DVR, as this enhances the 3D perception and helps the surgeon to find anatomical landmark points on the skull, which can be used as orientation aides during surgery. The result of clipped skull peeling is generated in three raycasting passes by using the stencil buffer in order to restrict pixels and thus rays to one of three different cases with one specific rendering mode each: (1) Everything outside the specified clipping areas is rendered using unshaded DVR of the MR volume; (2) Inside the skin incision area the skull is displayed (shaded DVR of the CT data), but accumulation of color and opacity is started only after the threshold for bone has been exceeded; and (3) The bone removal area is skull-peeled. The assignment of these three rendering modes to their corresponding pixels is performed as follows: After clearing the stencil buffer to zero, the polygon that was drawn by the user to simulate the skin incision is rendered, increasing the stencil values of the covered pixels to one. Next, the polygon drawn for bone removal is rendered as well, which increases the stencil values of the

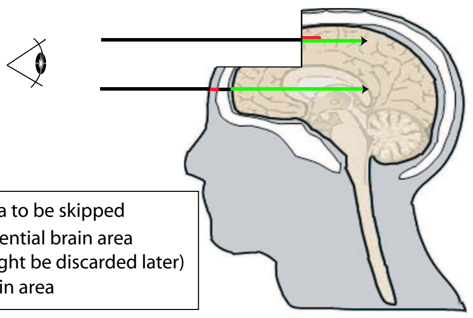


Fig. 4: Skull peeling algorithm. The lower ray displays the standard case where the ray hits the skull prior to the brain. The upper ray depicts the case where the brain is not covered by bone.

corresponding pixels to two. However, for rendering the bone removal polygon the stencil function is set such that stencil values are only modified in areas where the stencil buffer is already one. This ensures that the bone is only removed in areas where the skin has already been removed. Then, the three raycasting passes outlined above are performed, restricting rendering to the corresponding pixels by setting the stencil function accordingly. Note that this algorithm could easily be extended to more general view-dependent clipping methods.

View-Independent Skull Peeling: The skull peeling algorithm is inherently view-dependent. This, however, implies that the peeled volume is static with respect to the image plane instead of the volume, and the part of the volume that is peeled away changes whenever the view is changed. We extend skull peeling to a powerful view-independent approach for volume clipping. During raycasting, a depth image is generated that stores the depths where rays first hit a part of the volume that is not peeled away. In order to generate a segmentation mask that corresponds to the peeled area, each voxel position is transformed according to the view transform into an (x, y) position and corresponding depth z. Comparing the voxel's transformed depth with the depth stored in the depth image at (x, y) determines whether the voxel is included or excluded from the skull peeling segmentation mask, which is equivalent to a voxelized selection volume [25]. This process is very similar to shadow mapping [27]. The generated mask allows to switch back to standard volume rendering with segmented masks, toggling the visibility of the peeled area on demand while it stays constant with respect to the volume, even when the view is changed. Therefore, view-independent skull peeling is the first step for multi-volume brain surface visualization (Section 5.3) as it offers an unobscured view to the brain.

5.3 Multi-Volume Blending – Brain Surface Visualization

Combining multiple modalities in a single rendering can significantly increase the understanding of the actual clinical situation. Surgery at the brain's surface primarily includes tumor resection, epilepsy surgery, and vessel surgery (arteriovenous malformation, AVM). For these cases, after virtually removing the bone cover, the surgeon wants to see the brain surface with additional information such as DSA or functional data (i.e., PET or fMRI). For this task, we employ a visualization approach that combines different modalities without requiring segmented masks. A major motivation for this is that PET data is very diffuse and cannot be segmented well, fMRI data is almost binary, and DSA data can be visualized clearly with a simple ramp transfer function. Also, avoiding the need for segmentation significantly speeds up the preprocessing phase. Therefore, a method that combines multiple volumes based on property fusion is a very good choice for visualizing the brain surface with information from multiple modalities.

Our algorithm only needs two or more registered volumes as input. It is flexible with regard to the number of volumes that can be visualized concurrently, since our bricking scheme (Section 5.1) makes the approach scalable. Each volume has its own individual transfer function. The volume contributions are combined during raycasting by applying the transfer functions of all volumes at each sample location and combining the classified values.

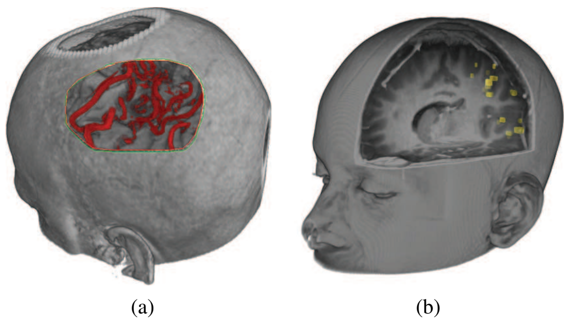


Fig. 5: Multi-volume rendering by blending different modalities. (a) Blending MR and DSA. (b) Blending MR and fMR.

We have implemented different combination techniques ranging from simple linear blending to more specialized combination modes. For blending DSA data with MR or CT, for example, it is sufficient to only display the DSA data whenever its opacity value lies above a certain level. Otherwise, the other volumes are blended and displayed. For visualizing functional data such as PET along with anatomical data, the PET values can be used for color classification while the anatomical data determines the opacity values. Figure 5 shows examples of multi-volume rendering by blending.

5.4 Segmented Multi-Volume Rendering – Deep Lesions

For thorough planning of surgery on deep-seated processes, after skull peeling and brain surface visualization certain structures need to be emphasized, e.g., a tumor or the optical nerve. This is achieved by a prior segmentation of the structures of interest, and individually applying multiple optical properties such as transfer functions and rendering modes. In the following, we assume that an object ID volume is available, which is the combination of multiple binary segmented objects. Our rendering algorithm for segmented volumes is conceptually based on two-level volume rendering [9], which allows to define a separate rendering mode and transfer function for each individual object. However, we use raycasting instead of slicing, which allows for much more flexibility such as the mask smoothing described in Section 5.5. During rendering, the object ID of a sample determines the corresponding rendering mode and transfer function. All 1D transfer functions are stored in a single 2D texture with one transfer function per row. All 2D transfer functions are stored in a single 3D texture. For multi-volume rendering, the user additionally chooses a specific volume for each object ID in order to select the modality that depicts the underlying kind of data best. For example, choosing an MRI as underlying data of a segmented bone is not a good choice, since bone is not depicted well by this modality. Using MRI data for the "brain object," and CT data for the "bone object" surrounding the brain, however, is a very good choice. During rendering, a small 1D look-up texture is used to fetch the corresponding volume ID for each object ID. The shader then simply samples the volume texture corresponding to the volume ID of the sample. Figure 6 depicts examples of multi-volume rendering for segmented data, which also shows that it is possible to specify per-object clipping planes. The clipping plane equations are obtained from two 1D textures, as outlined in Section 5.1, and the shader simply discards fragments that should be clipped. A combination of multi-volume blending with segmented data is also possible, which is determined by special blending object IDs, as described in Section 5.1. In this case, each object can have as many transfer functions as there are volumes, and the result is blended per sample after all transfer functions have been applied.

5.5 Smooth Rendering of Segmented Multi-Volume Data

The main visual problem of rendering binary segmented objects are the staircase artifacts that appear at object boundaries (Figure 7a).

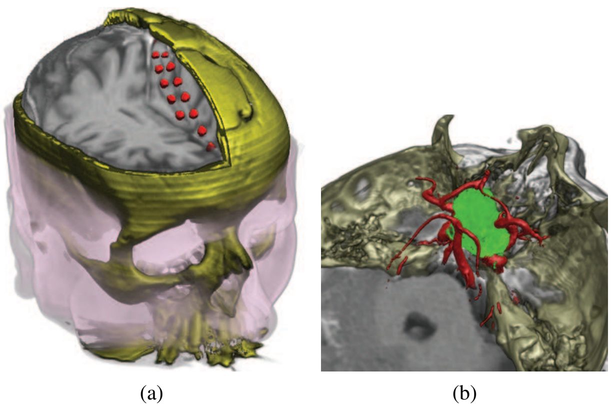


Fig. 6: Multi-volume rendering of segmented data. (a) CT and MR data for visualization of implanted electrodes for epilepsy surgery. (b) CT, MR and MRA data for tumor resection planning.

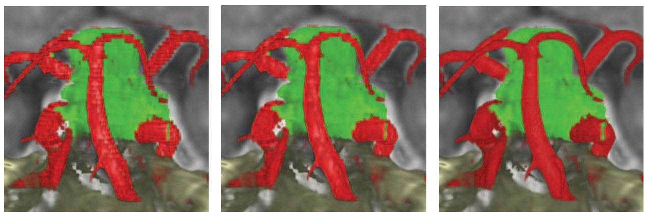

(a) (b) (c)

Fig. 7: (a) Unfiltered object boundaries. (b) Trilinear filtered object boundaries. (c) Our smooth boundary rendering algorithm.

Especially small objects with narrow diameters such as vessels get a ragged appearance with clearly discernable object boundaries of voxel size. For high-quality visualization, the real boundary of the object is needed with subvoxel accuracy. Approaches such as trilinear filtering of object boundaries improve visual appearance (Figure 7b) and work for all kinds of segmented objects [9]. However, trilinear interpolation does not completely remove all artifacts (especially in close-up views), as it takes into account only the binary segmentation information and not the underlying data values. Our approach for smooth rendering of segmented data (Figure 7c) is based on the assumption that the object boundary can be described by an iso-value. Values above the specified iso-value belong to the object whereas values below or equal are outside. This works well for segmented vessels as well as other structures of interest in neurosurgery, such as the bone or implanted electrodes. We take advantage of this existing iso-value boundary to adjust the object ID of each sample on-the-fly during rendering.

The algorithm works as follows: First, for each object that should use improved smooth boundaries, the iso-value corresponding to its boundary must be specified by the user. Then, for each sample during raycasting, we take a number of steps (defined by a user-adjustable parameter) in the positive and negative gradient direction to check if there is another object nearby. The gradient direction is used since new structures are most likely to appear in the direction of the greatest change of intensity. If a sample in the gradient direction belongs to a different object than the original sample, the original sample is treated as *boundary sample*, as below. If the sample is not a boundary sample, standard raycasting for segmented objects is performed as described above. For a boundary sample, the following steps are applied: The intensity of the current boundary sample (I_{cur}) is compared to the predefined boundary iso-value of the current object (Iso_{cur}) as well as to that of the adjacent object found (Iso_{adj}). If the intensity value corresponds to the iso-value of the adjacent object, the current sample is re-classified by changing its object ID (oID_{cur}) to the adjacent object's ID (oID_{adj}). Figure 8 depicts the different steps of the algorithm, and Equation 1 summarizes the reclassification step:

$$oID_{cur} = \begin{cases} oID_{adj} & \text{if } (I_{cur} > Iso_{adj}) \wedge ((Iso_{adj} > Iso_{cur}) \vee (I_{cur} < Iso_{cur})) \\ oID_{cur} & \text{else.} \end{cases} \tag{1}$$

Tiede et al. [23] have presented a similar approach based on threshold-segmented objects and their corresponding min and max threshold values. Their approach, however, only takes into account the eight surrounding voxels of each sample to reassign object memberships, whereas we search a user-defined length along the gradient direction. This gives us the possibility to adapt the boundary to our needs. We

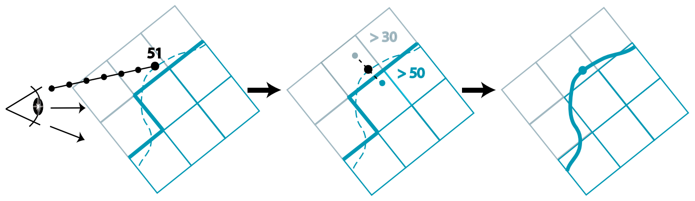


Fig. 8: Smooth object boundary rendering. The density of a sample is compared to the iso-values of neighboring objects in gradient direction. The sample is assigned to the object with the closest iso-value.

can, for example, increase the boundary iso-value of a segmented vessel on-the-fly to show only the interior of the vessel, or lower the iso-value to display the vessel and its vascular hull. The main advantage of our algorithm is that even inexact segmentation masks (e.g., slightly too small or too large masks) can be rendered correctly and with a smooth appearance because the object boundary is adapted to the actual underlying data. When extending this algorithm to multiple volumes one has to be careful to always use the correct volume for iso-value comparison. When comparing the current sample's intensity value to the adjacent object's iso-value, the adjacent object's ID has to be used to fetch the correct volume for getting the intensity at the current sample. Figure 7 shows the visual result of our algorithm compared to standard rendering of segmented masks.

5.6 Interaction Aides

Various features have been implemented to support the surgeon in the task of preoperative planning:

Transfer Function Specification: Colors and opacities are specified over the intensity range in a standard graphical TF editor, in which it is possible to load a set of predefined transfer functions and adapt them manually to the individual dataset.

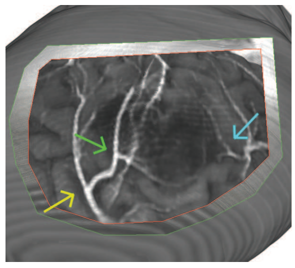
Flythrough Navigation / Microscopic & Endoscopic View: The datasets can be explored by flythrough navigation. To simulate the operating microscope the viewpoint for rendering can be set inside the volume and moved around interactively. An endoscopic lens with an adjustable field of view can be simulated by perspective raycasting.

Slice Views: Next to the 3D visualization window an MPR (multiplanar reconstruction) can be displayed. The MPR consists of three orthogonal slice views (axis aligned) displaying the raw data as well as the segmented objects.

Integration into a PACS: The whole framework is integrated into Agfa's Impax 6.0 medical workstation. Impax 6.0 offers a plugin interface which allows to extend the basic functionality of the workstation to meet the individual demands of the users. Therefore, by integrating our visualization framework, all other features of the medical workstation (e.g., additional segmentation possibilities, data access) can be used in combination with our neurosurgical planning application.

6 Results

We demonstrate the usefulness of our application by presenting two distinct planning cases as they were performed by a neurosurgeon. The cases consist of a tumor resection at the frontal lobe near the brain's surface and a tumor resection near the pituitary gland. The first patient underwent CT, MR, PET and fMRI scans, as the tumor was in the vicinity of the motor language area. First the datasets were registered and resampled to have the same volume dimensions (256^3). After initial exploration of the datasets (e.g. via skull peeling) and positioning of the patient's head as done in real surgery, skin incision and bone removal were performed tailored to the individual anatomy. Next, the datasets (MR, PET, fMRI) were visualized by multi-volume blending where the MR data depicts the anatomy, PET shows the metabolic active parts of the tumor and the fMRI data shows the brain areas involved in language function, which must be kept intact during surgery. Screenshots from different stages of the planning process are depicted in Figure 11. The virtual anatomic structures such as gyri, sulci and blood vessels were found to correlate well with the intraoperative view (Figure 9), thus allowing the surgeon to preoperatively plan the resection borders. Concurrent visualization of fMR data helped in identifying critical "no-touch" areas at the left resection border. PET data revealed a focus of high metabolic activity in the right part of the tumor where consequently a separate specimen was taken and sent to histology during surgery, leading to additional irradiation treatment. The second patient had a deep-seated lesion near the pituitary gland and underwent CT, MR and MR angiography scans (dataset size 512x512x164). In this case, after registration, tumor and vessels were segmented by thresholding. Next, the surgeon used our multi-volume rendering for segmented masks to gain insight into the individual anatomy (i.e., position of critical vessels in relation to the


(a)

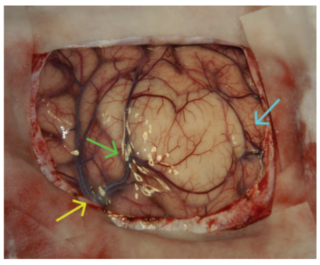

(b)

Fig. 9: Comparison of a skull peeled image (a) with the corresponding intraoperative view (b). Arrows show points of correspondence.

tumor). After he had a clear perception of the location of the tumor and other anatomical landmarks he then used the skull peeling algorithm to plan the optimal position of the surgical approach (Figure 10).

During development we kept a tight feedback loop with the department of neurosurgery at the General Hospital Vienna to iteratively refine the system. The skull peeling algorithm was received very well as it offered a direct view of the brain's surface instantly, without tedious preprocessing. The main drawback is the need of a registered CT dataset which might not always be available. Multi-volume rendering by blending also convinced because of its instant visualization without requiring segmentation and its ease of use. However, the optimal method to blend/combine the different volumes (e.g., linear blending, taking the first volume to define opacity and a second to define color) depends strongly on the type of datasets that are visualized. Therefore, an automatic setting of the combination method depending on the types of datasets could further improve the usability of the entire system. The multi-volume rendering of segmented masks was again perceived as very helpful by the surgeon. A drawback, however, is the amount of parameters that need to be set for each mask individually (assigning a volume to the mask, choosing the rendermode and transfer function, setting the iso-value for smooth object boundaries). Naturally, these parameters offer a very high flexibility for visualizing the datasets, however they also reduce the ease of use of the application. According to the surgeon, the most tedious part of the workflow consists of collecting and registering all the different datasets prior to visualization, especially fMR, PET and DSA data. All things considered, our surgery planning application was very well perceived and is now used almost daily in clinical practice.

Additional User Effort: In the routine clinical setting, preprocessing has been shown to take an average of 10 minutes (Table 3), and initial case setup for visualization up to 3 minutes. After initial exploration, all parameters of a case can be saved in an archived casefile. Case exploration is interactive and re-loading of archived cases takes less than 1 minute. The 3D cases are routinely prepared by young residents and later demonstrated to and discussed with the performing neurosurgeon: For the resident, this has the advantage of teaching and training neuroanatomy of the oncoming surgical approach, for the advanced surgeon time expenditure is thus very small. The cases are chosen by the surgeons either on the basis of anatomical difficulty, individual variations in anatomy, or simply for the convenience of a preoperative 3D visualization.

Preprocessing: (avg 10 min for 3 volumes w/o manual segmentation)	
Collection of image data from interdisciplinary PACS network (radiology, nuclear medicine, neurology)	2-10 min
Image registration	2 min / volume
Threshold segmentation for bone/vessels	1 min / object
Manual segmentation of tumor	$>$ 5 min
Volume rendering initialization	1 min
Initial setup at interactive stage: (avg 2 min for 3 volumes and 2 objects)	
Skull peeling	immediate
Multi-volume blending (TF and blending factor design)	15 sec / volume
Multi-volume rendering of segmented masks (setting of volume, transfer function and render mode)	15 sec / object
Smooth mask rendering (iso-value adjustment)	15 sec / object
Interactive exploration	optional
Loading setup of archived case	$<$ 10 sec

Table 3: User effort for initial case setup (parameter setting).

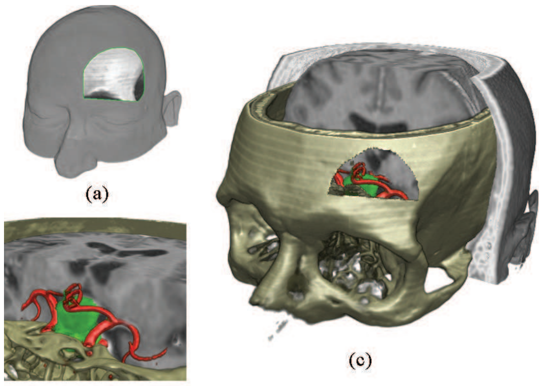


Fig. 10: Planning of a right subfrontal approach for pituitary tumor resection. (a) Skin incision. (b) Operating microscope view. (c) Keyhole approach planning.

Performance: All our visualization algorithms run at interactive frame rates. Naturally, the frame rates vary depending on the transfer functions and rendermodes that are used (e.g., unshaded DVR, shaded DVR). Table 4 gives an overview of the frame rates of both case studies for the different visualization methods. Timings are for a Pentium 4, 3.2 GHz with 3 GB RAM and an ATI Radeon X1800 graphics card.

Multi-volume rendering by either blending or segmented masks achieves the highest frame rates. After activating our algorithm for rendering smooth object boundaries, however, the frame rate drops significantly due to the complex shader for rendering smooth boundaries. This problem can be alleviated by automatically switching back to normal rendering during user interaction (e.g. while rotating the view). The frame rates of the skull peeling algorithm can be explained by the costly branching-statements that have to be performed in the shader to cover all special cases for peeling the skull correctly. On the whole, the framerates were found adequate by the users, since minor viewpoint changes are usually sufficient during preoperative planning.

Memory Usage: The actual memory footprint of our system depends on the cache sizes chosen, and on whether texture dimensions need to be padded to power-of-two dimensions or not. For example, Case Study 2 (Table 4) can use caches of size 512^2x128 each for the CT and MR volumes and of size 512^2x64 for the MRA volume (all 16-bit voxels), and a 512^2x256 cache for the object IDs (8-bit voxels). This yields a memory consumption of 224MB, which would allow this configuration to be rendered even on a 256MB graphics card. In comparison, the original volumes sum up to 287MB, but if padding to power-of-two dimensions is required (e.g., Radeon cards) would actually consume 448MB of GPU memory, which requires at least a 512MB graphics card. However, it is important to note that our caching scheme works better with larger volumes (e.g., 512^2x300 and upward), and especially helps to alleviate power-of-two requirements.

7 Conclusion and Future Work

Our multi-volume rendering system for preoperative planning of neurosurgical interventions was directly inspired by the needs of neuro-

Visualization Method	Case Study 1 CT, MR, fMR, PET (each 256x256x256)		Case Study 2 CT, MR, MRA (each 512x512x164)	
# volumes	2	3	2	3
Skull Peeling	9.5 fps	9 fps	5 fps	4.5 fps
Multi-Volume Blending	23 fps	12 fps	18 fps	12 fps
Segmented Multi-Volumes	35 fps	27 fps	24 fps	20 fps
Smooth Segmented MVs	15 fps	5 fps	6 fps	4 fps

Table 4: Frame rates of the visualization methods (viewport 512x512).

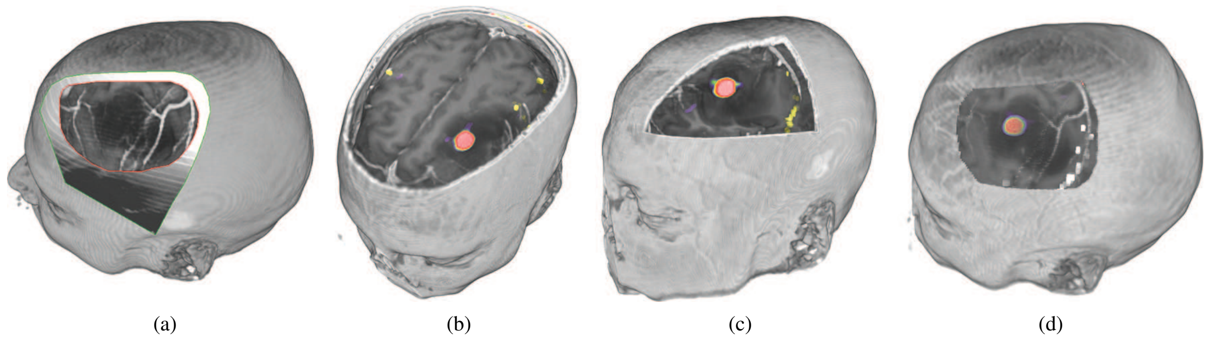


Fig. 11: Workflow for planning a tumor resection near the brain's surface. (a) Planning the surgical approach. (b,c,d) Multi-volume blending for visualization of superficial structures. The visualization includes MR (black/white), PET (red) and fMR (yellow and white) data. The PET transfer function shows an area of high metabolic activity within a low grade glioma. The fMR spots delineate areas activated during speech.

surgeons to visualize multimodal data fast, in high quality, and with as little user interaction as possible. The surgical approach to the brain is simulated by interactively removing surrounding tissue such as skin and bone from MR data by making use of additional information present in a registered CT dataset. Further we developed multi-volume rendering techniques that work either purely on the data or include additional segmentation masks. Rendering of segmented objects was improved by an algorithm for smooth rendering of object boundaries. To encourage use in daily clinical practice, we integrated our multi-volume visualization system into a medical workstation which offers registration, segmentation and interactive data exploration possibilities. In our future work we want to incorporate DTI (Diffusion Tensor Imaging) into our multimodal visualization. Displaying neuronal pathways could further improve the minimal invasiveness and security of neurosurgical interventions. As 3D visualization has become well accepted among neurosurgeons, the next logical step would be to connect our system to a neuronavigation system for tracking of the intraoperative position of the surgeon's instruments. Visualizing the multimodal datasets in parallel to the real surgery could further help the surgeon in identifying structures of interest which are not visible during surgery (e.g. functional areas, optimal skin incision line).

Acknowledgements

This research project was funded by the Austrian KPlus project and by AGFA Healthcare Vienna. The datasets are courtesy of the Neurosurgery Department at the Medical University of Vienna. We would also like to thank Christof Rezk-Salama.

References

[1] M. Atkins, K. Siu, B. Law, J. Orchard, and W. Rosenbaum. Difficulties of T1 Brain MRI Segmentation Techniques. In *Proc. of SPIE Med. Imaging*, volume 4684, pages 1837–1844, 2002.

[2] J. Beyer, M. Hadwiger, S. Wolfsberger, C. Rezk-Salama, and K. Bühler. Segmentierungsfreie Visualisierung des Gehirns für Direktes Volume Rendering. In *Proc. of Bildverarb. für die Medizin*, pages 333–337, 2007.

[3] W. Cai and G. Sakas. Data Intermixing and Multi-Volume Rendering. In *Proc. of Eurographics*, pages 359–368, 1999.

[4] M. Capek, L. Mroz, and R. Wegenkittl. Robust and Fast Medical Registration of 3D-Multi-Modality Data Sets. In *Proc. of Medicon*, pages 515–518, 2001.

[5] S. DiMaio, N. Archip, N. Hata, I. F. Talos, S. K. Warfield, A. Majumdar, N. McDannold, K. Hynynen, P. R. Morrison, W. M. Wells, D. F. Kacher, R. Ellis, A. J. Golby, P. M. Black, F. A. Jolesz, and R. Kikinis. Image-guided Neurosurgery at Brigham and Women's Hospital: The Integration of Imaging, Navigation and Interventional Devices. *IEEE Engineering in Medicine and Biology Magazine*, 25(5):67–73, 2006.

[6] P. Felkel, R. Wegenkittl, and M. Bruckschwaiger. Implementation and Complexity of the Watershed-from-Markers Algorithm Computed as a Minimal Cost Forrest. In *Proc. of Eurographics*, pages 26–35, 2001.

[7] M. Ferré, A. Puig, and D. Tost. A Framework for Fusion Methods and Rendering Techniques of Multimodal Volume Data. *Computer Animation and Virtual Worlds*, 15:63–77, 2004.

[8] A. Ghosh, P. Prabhu, A. E. Kaufman, and K. Mueller. Hardware Assisted Multichannel Volume Rendering. In *Proc. of Computer Graphics International*, pages 2–7, 2003.

[9] M. Hadwiger, C. Berger, and H. Hauser. High-Quality Two-Level Volume Rendering of Segmented Data Sets on Consumer Graphics Hardware. In *Proc. of IEEE Visualization*, pages 301–308, 2003.

[10] M. Hadwiger, C. Sigg, H. Scharsach, K. Buhler, and M. Gross. Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces. In *Proc. of Eurographics*, pages 303–312, 2005.

[11] P. Jannin, O. Fleig, E. Seigneuret, C. Grova, X. Morandi, and J. Scarabin. Multimodal and Multi-Informational Neuro-Navigation. In *Proc. of CARS - Computer Assisted Radiology and Surgery*, pages 167–172, 2000.

[12] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional Transfer Functions for Interactive Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[13] J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proc. of IEEE Visualization*, pages 287–292, 2003.

[14] I. H. Manssour, S. S. Furuie, S. D. Olabarriaga, and C. M. Freitas. Visualizing Inner Structures in Multimodal Volume Data. In *Proc. of SIBGRAPI*, pages 51–58, 2002.

[15] R. Mullick, R. N. Bryan, and J. Butman. Confocal Volume Rendering: Fast Segmentation-Free Visualization of Internal Structures. In *Proc. of SPIE - Int. Symp. on Optical Science and Technology*, 2000.

[16] A. Neubauer, S. Wolfsberger, M. Forster, L. Mroz, R. Wegenkittl, and K. Bühler. STEPS - An Application for Simulation of Transsphenoidal Endonasal Pituitary Surgery. In *Proc. of IEEE Visualization*, pages 513–520, 2004.

[17] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In *Proc. of Graphics Hardware*, pages 109–118, 2000.

[18] C. Rezk-Salama and A. Kolb. Opacity Peeling for Direct Volume Rendering. In *Proc. of Eurographics*, pages 597–606, 2006.

[19] F. Rößler, E. Tejada, T. Fangmeier, T. Ertl, and M. Knauff. GPU-based Multi-Volume Rendering for the Visualization of Functional Brain Images. In *Proc. of SimVis*, pages 305–318, 2006.

[20] H. Scharsach, M. Hadwiger, A. Neubauer, S. Wolfsberger, and K. Bühler. Perspective Isosurface and Direct Volume Rendering for Virtual Endoscopy Applications. In *Proc. of Eurovis '06*, pages 315–323, 2006.

[21] L. Serra, R. A. Kockro, C. G. Guan, N. Hern, E. C. K. Lee, Y. H. Lee, C. Chan, and W. L. Nowinski. Multimodal Volume-based Tumor Neurosurgery Planning in the Virtual Workbench. In *Proc. of MICCAI*, pages 1007–1015, 1998.

[22] T. Song, E. Angelini, B. Mensh, and A. Laine. Comparison Study of Clinical 3D MRI Brain Segmentation Evaluation. In *Proc. of IEEE Engineering in Medicine and Biology Society*, pages 1671–1674, 2004.

[23] U. Tiede, T. Schiemann, and K. H. Höhne. High Quality Rendering of Attributed Volume Data. In *Proc. of IEEE Visualization*, pages 255–262, 1998.

[24] F. Vega Higuera, P. Hastreiter, R. Naraghi, R. Fahlbusch, and G. Greiner. Smooth Volume Rendering of Labeled Medical Data on Consumer Graphics Hardware. In *Proc. of SPIE Med. Imaging*, pages 13–21, 2005.

[25] D. Weiskopf, K. Engel, and T. Ertl. Interactive Clipping Techniques for Texture-Based Volume Visualization and Volume Shading. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):298–312, 2003.

[26] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. In *Proc. of SIGGRAPH*, pages 169–178, 1998.

[27] L. Williams. Casting Curved Shadows on Curved Surfaces. In *Proc. of SIGGRAPH*, pages 270–274, 1978.

[28] B. Wilson, E. B. Lum, and K. Ma. Interactive Multi-Volume Visualization. In *Proc. of Int. Conf. on Comp. Science*, pages 102–110, 2002.