Pacific Graphics 2012
C. Bregler, P. Sander, and M. Wimmer
(Guest Editors)

Volume 31 (2012), Number 7

Adaptive Cross-sections of Anatomical Models

J. Díaz, E. Monclús, I. Navazo and P.Vázquez

MOVING Research Group, Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract

Medical illustrations have been used for a long time for teaching and communicating information for diagnosis or surgery planning. Illustrative visualization systems create methods and tools that adapt traditional illustration techniques to enhance the result of renderings. Clipping the volume is a popular operation in volume rendering for inspecting the inner parts, though it may remove some information of the context that is worth preserving. In this paper we present a new editing technique based on the use of clipping planes, direct structure extrusion, and illustrative methods, which preserves the context by adapting the extruded region to the structures of interest of the volumetric model. We will show that users may interactively modify the clipping plane and edit the structures to highlight, in order to easily create the desired result. Our approach works with segmented volume models and non-segmented ones. In the last case, a local segmentation is performed on-the-fly. We will demonstrate the efficiency and utility of our method.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms.

1. Introduction

The visualization of the internal parts of a volume model is an important topic of research in volume rendering. Inner structures are usually occluded by outer elements and their correct visualization is crucial for a proper understanding of the data. One example is medical data, due to the complexity of the human anatomy. It is important to clearly perceive the 3D relationships of the structures and to be able to analyze certain anatomical structures without occlusion.

Several methods have been proposed to address this issue in the past. Some of them are based on clipping the volume using planes or more complex geometries [WEE03]. The former are included in the majority of volume renderers because they are simple and easy to use. Unfortunately, contextual information, which is useful to better perceive the shape, position and orientation of the clipped structures, is often removed because all structures are equally clipped.

In order to address the loss of contextual information, more sophisticated approaches visualize simultaneously external and internal structures using traditional illustration effects, such as exploded views [BG06] or cutaways [VKG05]. Besides, illustrative visualization techniques also provide the level of abstraction needed to correctly convey the information of complex data sets. However, most of these methods do not allow simple direct structure-aware data editing or the manipulation is indirectly carried out through time-consuming parameter tweaking, which often requires a certain level of expertise to obtain the desired results.

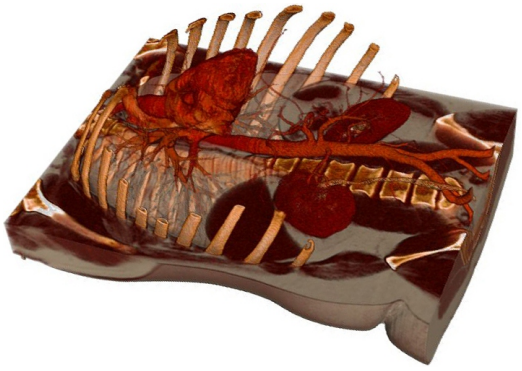


Figure 1: *Adaptative cross-section of the human torso. Extruded structures have been edited from an initial cut with simple dragging operations.*

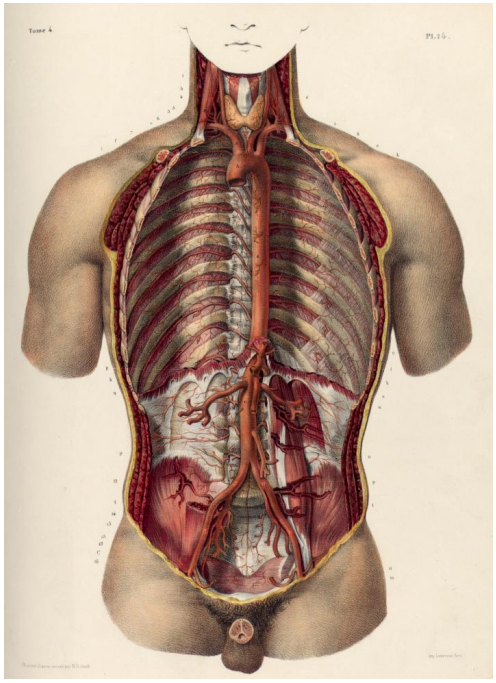


Figure 2: *Drawing of the thorax, abdomen and pelvis in cross-section that reveals the aorta and its branches at a different height from the clipping plane. Taken from* Anatomia Collection: http://link.library.utoronto.ca/anatomia/.

In this paper we present a new manipulation tool that facilitates working directly over a cross-section, and allows the user to extract the interesting structures out from it. In this way, we enrich simple cuts with contextual information (see Figure 1). This process is carried out in a simple, intuitive way, since our method provides tools for the direct manipulation of anatomical structures, where extrusions are generated by automatically adapting the shape of revealed information to the proper anatomical materials. Our inspiration comes from traditional hand drawn illustrations as the one shown in Figure 2, where the aorta and its branches are displayed at a different height from the cross-section.

Our main contributions are:

- An interactive novel approach for the direct extrusion of anatomical structures from cross-sections defined on volume datasets.
- An application architecture that makes the interaction fast and intuitive, and supports on-demand data segmentation if required.

If the input model is not segmented, an automatic segmentation is triggered and performed transparently to the user, whilst keeping interactivity. Additionally, we also provide an extra set of illustrative motifs that users may optionally activate in order to give the illustration a different look. The main advantages of our tool are:

- Intuitive edition that leads to fast generation of illustrations. Target images are obtained in times that go from seconds up to a couple of minutes.
- No need of extra parameter adjusting: Users directly manipulate the desired structures and extrude them off the cross-section.

We have performed an informal user study that demonstrated users were able to generate a set of objective images in little time. Medical experts found the tool suitable for preparing teaching media, and for giving the surgeons a better knowledge of the real 3D shape of their patients' anatomical structures, especially for surgery preparation.

The remainder of the paper is organized as follows: Section 2 reviews the related work. Our approach is introduced in Section 3, Section 4 presents the architecture and implementation of our application. Results and discussion are presented in Section 5. Finally, in Section 6 conclusions and future work are proposed.

2. Related Work

Exploring the inner parts of volumetric data sets is a common operation that has been addressed under different interaction motifs: *cut-away views*, *focus+context visualization*, *importance-driven* volume rendering and *deformation-based* techniques.

Cutaway views is a common paradigm that visually removes a region of the volume from the rendering. This can be achieved by simply defining a cutting plane, or more complex cutting geometries as in Weiskopf *et al.* [WEE03]. The anatomical atlas of Höhne *et al.* [HBR*92] allows the user to place axis-aligned cutting planes for visualizing internal structures. Konrad-Verse *et al.* [KVPL04] use a deformable cutting plane to perform virtual resections while McInerney *et al.* [MB06] propose an editable set of cutting planes to create custom cross-sectional views. Clipping approaches are the most popular techniques because they are simple and easy to use. However, their main drawback is the loss of contextual information of the clipped structures.

Some interactive approaches for generating cutaways of polygonal models have been proposed [DWE03, BF08, MC10]. Li *et al.*'s method [LRA*07] allows the user to modify the appearance of the elements in the clipped region, by means of a rigging system that defines how the cutaway affects each structure. Our approach also provides this feature, replacing the rigging system by a simpler manipulation tool that directly works on the volume data (instead of a polygonal model) and therefore makes our method suitable for even inexperienced users and raw data.

Focus+context methods add cues to preserve the context surrounding the features of interest. These visualizations

may be implemented by modifying the transfer function (TF) of the structures placed between the user and the region or feature of interest like in Bruckner *et al.* [BGKG06]. In order to do this, it is necessary to develop tools that allow the users to easily define important regions, importance information [BHW*07], and the focus of attention [VFSG06]. The definition of the region of interest is a crucial and often complex stage. Furthermore, most of the previous approaches work with pre-segmented data sets and users select by hand the important structures. In [BG06], a volume painting tool [BG05] allows the user to define the element in focus. Chen *et al.* [CSCS08] provide an interactive segmentation tool that makes their technique well-suited for raw data sets, and Owada *et al.* [ONI05] allow the user to select regions of interest by tracing strokes on the screen.

Importance-Driven Volume Rendering is based on the suppression of less important parts of the volume (e.g. using transparency [VKG05, KSW06]). The general concept is to transfer importance to visibility so a visibility priority is assigned to each object. If objects with high priorities occlude each other, the object with maximum priority is rendered. These approaches suffer from similar problems as the maximum intensity projection ones where the depth relations are difficult to recognize. Rautek *et al.* [RBG08] propose a framework for the semantic definition of importance based on a set of illustration techniques. Most of these methods provide the user with an indirect control of the screen-space location of the focus by global and/or data-dependent parameters only. We concentrate on providing maximum flexibility on the decision of the structures to manipulate and direct control on the degree of their edition.

More sophisticated methods are based on interactive manipulation of volume models to create feature-based cutaway visual effects inspired by surgical metaphors [BV09, CSC10, CSC06, MRH08, MTB03, CSCS08, ISC07]. Their main differences concern the deformation technique used to achieve the cutaway and the way the user defines it. Correa *et al.* [CSC06] and Mensmann *et al.* [MRH08] use a template based deformation technique for fast exploration and automatic model peeling to reveal interesting features. Chen *et al.* [CSCS08] propose a set of interactive manipulation tools for drilling, lasering, peeling, cutting and pasting different layers of volume data sets. Birkeland *et al.* [BV09] generate an automatic peel-away visualization based on a feature of interest.

Exploded views is another technique used by traditional illustrators. In this case, occluding structures are decomposed and displaced away to show the inner parts. While [LACS08] and [TKS10] propose different approaches to generate exploded views of polygonal models, Bruckner and Gröller [BG06] present an interactive and editable method for volumetric datasets. In [BRV*10], Bruckner *et al.* present a framework for composing 3D rendered layers and obtaining illustrative visualizations.

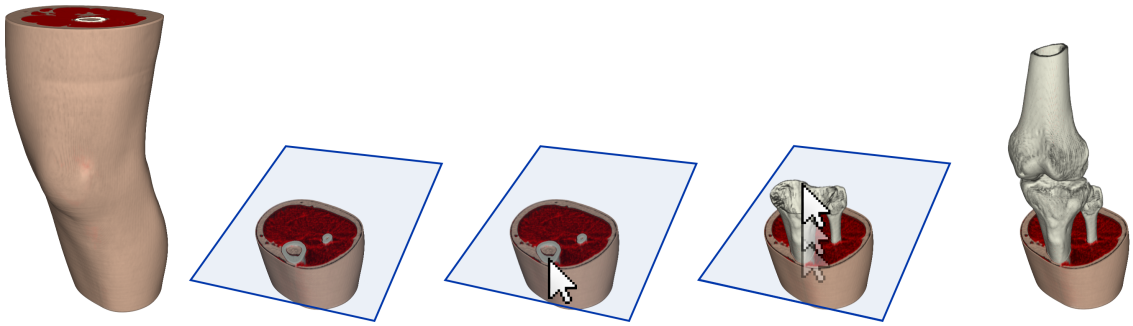
In a recent work, Birkeland *et al.* [BBBV12] enhance the amount of information revealed in the neighbourhood of a clipping plane by using an elastic membrane to clip the volume. This membrane adapts itself to the anatomical structure by defining a potential field that guides the membrane clipping. Their results are similar to ours, but require more intense user involvement. To control the potential field, the user must include anchor points by hand, and results may differ from what the user expected. Our approach is based on direct manipulation of the structures of interest, so the user decides which information must be included and to which extent, and receives immediate feedback of the result.

3. Illustrative Editable Cuts

In this section we introduce the features of our approach and the interaction method. Our main objective is to provide the user with a tool that allows him or her to generate cuts on volumetric models through the manipulation of a clipping plane, and then manipulate individually each of the visible structures appearing on the resulting cross-section of the model. In this way, the user may interactively specify how much portion of each visible structure needs to be extruded off the cutting plane. Hence, the final image shows a richer context and helps the users to get a better understanding of the shape of the structures and their relative position.

The application starts with the initial volumetric rendering of a dataset with a certain Transfer Function (see Figure 3*a*). Then, the user starts the interaction by defining a clipping plane that cuts the volume. As usual, the clipping plane may be moved and oriented. The volume that lies in the positive half-space of the plane is visually removed (see Figure 3*b*). Subsequent manipulations will then occur in this subspace. Once the clipping plane has been set, the user may start the editing stage. Editing is carried out with a number of clicks and mouse dragging, whose use is familiar and predictable, producing an easy to learn process. Each structure editing follows this pattern: first, the user clicks on certain pixel p_c (see Figure 3*c*). As a consequence, the first non-transparent structure (S) that projects onto the clicked pixel is selected. Then, mouse dragging extrudes S at a distance proportional to the mouse movement (see Figure 3*d*). Once the edition ends, the user may also add (if necessary) other illustrative effects tailored to giving the image its appropriate finish (i. e. view-dependent contours, the *Lit Sphere* motif, and so on).

Our rendering algorithm is a GPU-based ray casting. In order to appropriately adapt the extrusion shape to the structure, we must identify the conditions that samples s must satisfy to be considered as part of the Extrusion (E). Let $V \subset \mathbb{R}^3$ be a volumetric dataset, let v_c be the voxel that has been implicitly selected by the user (the voxel visible in pixel p_c) and let Π be the defined clipping plane. Function $pos()$ returns the 3D position of a sample. Then, s will be part of the extrusion, if and only if:



(a) Initial model (b) Clipping plane (c) Structure selection (d) Structure expanding (e) Final image

Figure 3: *Cut editing process. (a) shows the initial model. (b) shows the clipping plane defined by the user. Then, the structure to extrude is selected in (c) and extended in (d) up to the desired final position, shown in (e).*

Prop. 1 $0 < \Pi(pos(s)) \leq dist,$

and

Prop. 2 $Mat(v_c) = Mat(v_s),$

where v_s is the voxel that binds s, $dist$ depends on $\| \vec{m} \|$, $\vec{m}$ being the mouse movement vector on the screen, and $Mat()$ a function that returns the identifier (id) of the material (structure) of the voxel. Note that, for segmented volumes, $Mat()$ only requires querying a 3D structure that stores the identifiers of the structure per voxel.

For non-segmented volumes, we require on-the-fly segmentation. This can be achieved by starting a region growing algorithm with a single seed and detecting a single region with an expansion criterion determined by the homogeneity of materials. Although this process can be implemented in different ways, we simply check that the density value of the candidate voxel lies inside the range of densities previously determined for each material or structure. Formally, this implies a third condition required to consider s a sample that belongs to the extrusion E:

Prop. 3 There exists a path σ of face-connected voxels v_i between v_s and v_c | $\{ \forall v_i, Mat(v_i) = Mat(v_c) \wedge 0 < \Pi(pos(v_i)) \leq dist \}$.

It is important to notice that the connectivity constrain imposed by **Prop. 3** is not necessarily satisfied for initially segmented volumes because they may use unique identifiers for the same material (i. e. *bone* might encompass all metacarps in the hand). In this case, non-connected structures would be extruded together if they have the same *id*. Obviously, this can be overcome by ensuring connectivity between voxels on the positive side of the cross-section although this will be more costly.

In the following sections, we give some details and discuss our implementation. Moreover, we also introduce the interaction method and some extensions we made, notably the inclusion of a second plane that allows more complex illustrations.

4. Application Architecture

As discussed in the previous section, the selection and extrusion of structures is carried out once the clipping plane has been set. In this Section we will present the architecture of the application by exemplifying a concrete interaction: a simple extrusion that is performed defining a clipping plane, selecting and extruding one single structure, and giving the final rendering a certain finish. The data and interaction flows are depicted in Figure 4.

4.1. Process Overview

Without loss of generality, we first describe the different processes assuming the input is a segmented model. Therefore, when interaction starts, the volume (V), the transfer function (TF), and the segmentation structure (VS) −a 3D texture of *ids* that maps each voxel to its corresponding anatomical structure−, are already loaded on the GPU.

Our application has two major blocks: Interaction and Rendering. The interaction control is managed by the CPU, while the rendering is implemented as a multi-pass algorithm on the GPU. Although there are several possibilities for implementing each step, we have designed our algorithm in order to guarantee transparency and smooth interaction through the whole editing process. With this objective, we use two data structures that contain auxiliary information: the *Selection Helper* and the *Extrusion Distances*.

The *Selection Helper* (SH) is a data structure whose resolution matches the one of the viewport. It stores both the 3D coordinates of the selectable pixels and the identifiers of the structures at these points, making the selection very efficient. This structure is implemented as a couple of 2D textures that reside in the GPU.

After the plane definition, we enter the **selection** stage. SH is then initialized with the information of the cross-section using ray casting and finding the first non transparent sample according to the TF. We call this process *Initial Selec-*

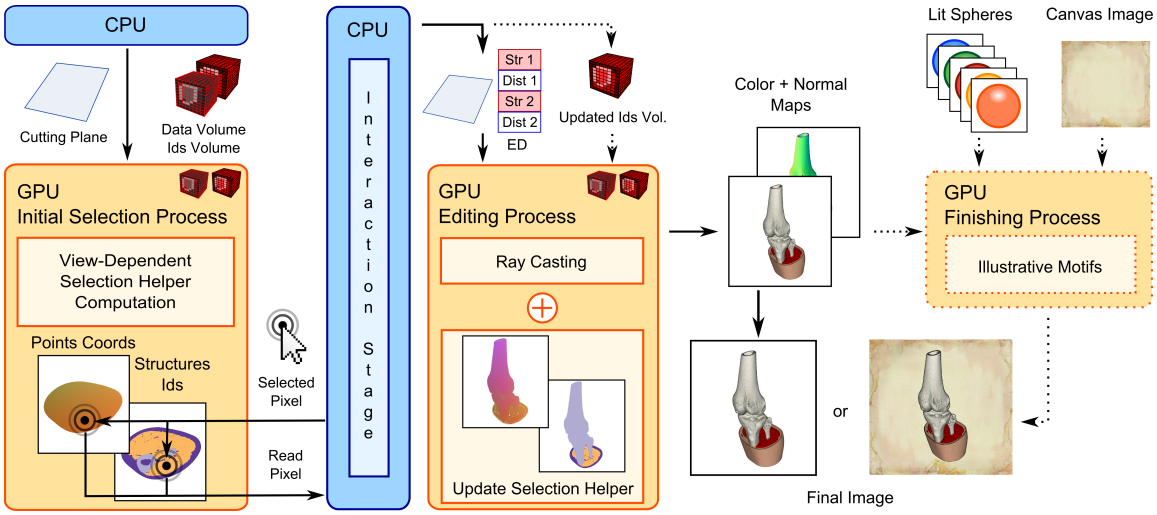


Figure 4: *Basic steps performed by a simple structure extrusion task. Once the clipping plane has been set, users can proceed and select a pixel on it (left) where the structure to extrude is projected. The CPU identifies the structure to be extruded by reading the information from the Selection Texture. Next, the mouse dragging operation modifies the set of visible samples that belong to the selected structure (center). When the edition has been finished, the user may give it the desired finish by adding some illustration motifs (right).*

tion (see Figure 4). Hence, the SH data structure is view dependent. Once the user selects a structure (clicks with the mouse), the **interaction** process reads the clicked pixel and gets the information of the selected structure from SH. Then, the user can define the extrusion using a simple *drag-and-drop* controlled by a CPU thread.

The *Extrusion Distances* (ED) is a 256-element data structure that stores, for each selected structure, the distance it has been extruded to from the clipping plane. For non-extruded structures this value is zero. ED is stored as a 1D texture that is incrementally updated with the user movements. This is uploaded to the GPU and used later on (see Figure 4). The **editing** process performs a GPU-based ray casting that implements a classical Phong shading for the samples lying on the negative part of the clipping plane. For the samples on the positive part of the plane, we must check if they belong to an extruded structure. This can be simply detected by querying the VS texture and checking then whether the identifier has been extruded. Samples not belonging to an extruded structure are rendered transparent. For extruded elements we must determine if the sample is closer to the plane than its corresponding extrusion distance. In this way, the extrusion process renders samples that fulfill properties 1 and 2 defined in Section 3. When the extrusion ends, users can proceed to another structure. Modified structures can also be edited again if desired.

Once the user has ended the manipulation, some **post-processing** effects can be added in order to incorporate illustrative motifs that can be used to emphasize certain features. The editing process generates a color image and a normal map. If the post-processing step is activated, it reads these texture maps and provides a set of effects that include silhouette rendering, the *lit sphere* and background texturing. On the other hand, the post-processing process can also be applied during edition although this seems of little use.

4.2. On-demand Segmentation

We considered useful the possibility of working with non-segmented volumes while keeping a similar user interaction metaphor and results. In this context, we decided to incorporate an interactive segmentation process based on region growing ([RK82]) that is triggered when the input is a non-segmented model. Its propagation approach fulfills Properties 1, 2 and 3 in Section 3 which support the direct extrusion of anatomical structures from cross-sections.

For the process to start, a seed is set in the selected voxel. In this case, SH starts with null values for the *ids* and the corresponding 3D positions of the visible samples. The structure to segment is determined by querying the 3D position in SH, and set as the one whose density value corresponds to the voxel that contains the sample. In our implementation, the range of densities for a certain structure depends on the Transfer Function. Then, the initial region grows to hold all face-connected voxels. Voxels are added to the segmented region iff their density is in the range of the selected structure and has not been previously segmented. The process iterates until there is no change in two successive steps. The relevant feature of our technique is that the process is carried out on

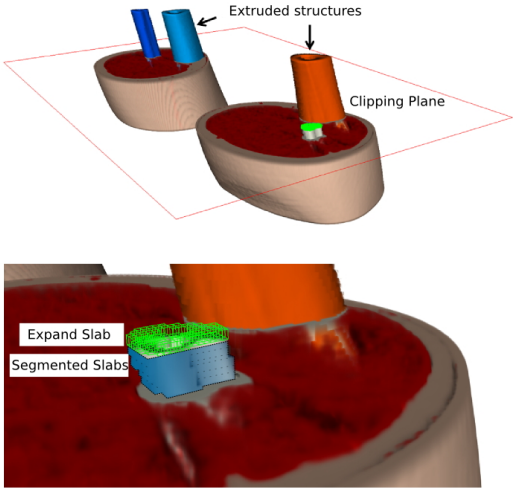


Figure 5: *The incremental segmentation process. For this model, when the user selects the bone, an automatic process starts that segments a slab of the model using region growing starting from the clicked point. If the interaction continues to pull away the structure, the expand region will be the next to analyze if the user continues extruding the same structure.*

demand in an incremental way by dividing the positive half space region of the clipping plane in slabs (see Figure 5). Since we do not know in advance how much distance the user will move from the clipping plane, we only segment a slab that is generated by taking the clipping plane and moving it a certain offset. The offset value we use is two times the voxel diagonal, as discussed in Section 5.3. The information we need to store for the next step of the segmentation is the active front of the current segmentation. This process continuously updates the segmentation texture (VS) which is uploaded to the GPU each time it is modified. As a result, the rendering algorithm is the same for both types of models and therefore, it concurrently supports a combination of segmented and non-segmented regions.

Complex structures may require to impose additional constraints for an accurate seed propagation (such as gradient computations). In this cases, when the user drags fast, the interaction may suffer and might become non-interactive. On the other hand, for structures with fuzzy boundaries or similar density values, a previous accurate full segmentation is desirable [CMB*12].

From the users' point of view, nothing special has to be done for non-segmented models, since the segmentation is triggered automatically if required.

4.3. Interaction Design

The interaction with the application must support several tasks:

- Plane definition: initial process that generates the cross-section of the volume. The plane can be rotated and translated interactively.
- Structure selection: a click on the clipping plane (or on part of the extruded structure, when it has been edited), directly selects the structure to edit.
- Structure extrusion: mouse dragging allows the user to adjust the portion of the selected structure that will be shown.
- Second plane: a second clipping plane is supported, from which the user may also choose to extrude some structures.

Interaction design had two main objectives: flexibility and simplicity. In order to ensure simplicity, all processes have visual feedback: when editing the model, the bounding box of the volume is drawn in wireframe and the cutting planes are also shown. Positive and negative parts of the planes are illustrated by giving a pale transparent look to the negative parts. When a structure is selected, its supporting plane is indicated by doubling its line size. This has two goals: Firstly, it provides a visual cue that indicates that some element has been selected. Secondly, when using two planes, it facilitates disambiguating the extrusion direction of the structure. If the extrusions from two different planes overlap, this is indicated by rendering the intersection with a different color. When selecting a point in the intersection, both planes can be potentially selected, and the user decides the proper one by toggling between them using the space bar. Finally, at any moment, the supporting planes can be changed, and the extruded structures change with them, thanks to the fact that the ED stores information that is independent from the plane positions. Figure 6 shows the different cases our shader must deal with. As the user study will show, the system requires little training.

4.4. Finish

Besides the flexible geometry editing, users can also easily tune the final look of the generated images. In order to do so, we add illustration motifs such as:

- View-dependent contours.
- Structure texturing using the *lit sphere*.
- Background texture.

These effects are applied taking as a basis the color image and the normal map (see Figure 4) by a shader that performs the corresponding steps.

Professional illustrators often emphasize the contours of the objects in order to improve the perception of shape. We have added **view-dependent contours** as an optional effect, computed by using a Sobel kernel [HKRs*06].

The **lit sphere** [SMGG01] is a technique designed to capture custom artistic shading models from sampled artwork. The main idea is to use a reference sphere that shows the

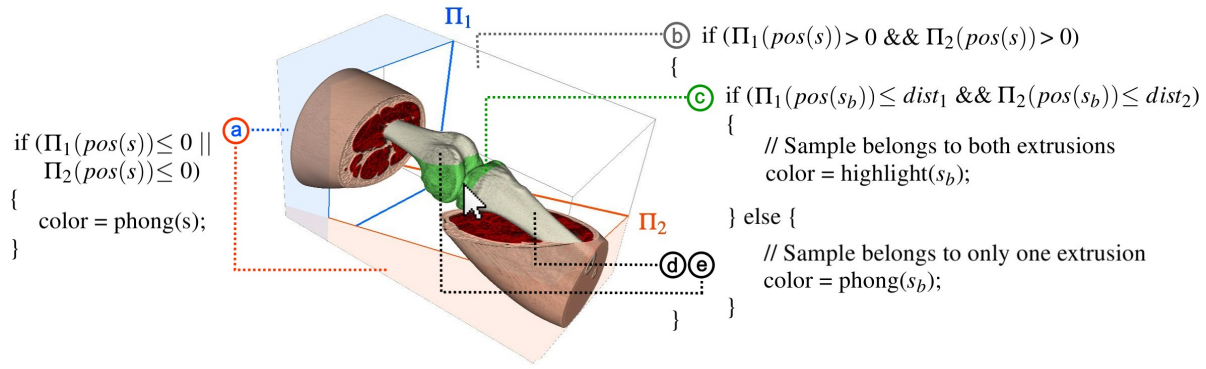


Figure 6: *Structure extrusion from two different planes (Π_1 and Π_2) and the basic algorithm to provide a helpful visual feedback while manipulating the volume. The editing region (b) contains all the samples (s) of the volume within the intersection of the positive half-spaces of the planes. Depending on the extrusion distances of the bone ($dist_1$ and $dist_2$) from both of the planes, the samples (s_b) of the extruded structure belong to a certain region (c, d, or e) and are coloured accordingly. Extrusions intersection (c) is highlighted in green. $\Pi_i(pos(s))$ is a function to determine the signed distance from a sample (s) to a plane (Π_i).*

shading of a certain material. Then, the color of the sphere is blended with the computed color of the ray casting process. As a result, for non-background pixels, the final color composition is

$$rgb = (0.5 * RC_{color} + 0.5 * LS_{color}) * contour_{intensity}$$

where RC_{color} is the color generated by the ray casting process, LS_{color} is the color of the lit sphere texture, and $contour_{intensity}$ is the weight of the edge as detected by the Sobel filter.

Solid backgrounds often produce an artificial look that can be avoided if we want to simulate a professional illustration. We provide the possibility of using a **background texture** (canvas). Our tool has a set of predefined textures that simulate paper. The result of combining the previous effects is shown in Figure 7.

4.5. Implementation Details

The commitment to guarantee ease of use and interactivity has lead us to take some implementation decisions. First, as shown in Figure 3, SH is written in two different steps. This is due to the fact that the first process (initial selection) uses a simple shader that does not account for extruded surfaces. The editing process may update SH at a negligible cost and therefore it is ready for the next selection process. This also avoids an extra rendering pass when selecting another structure.

We have also implemented a basic GPU version for the segmentation process. It segments a volume slice at a time by rendering a quad and checking for each voxel if it belongs to the material of the segmented point and if it is connected with the selected point. The limitations of the OpenGL pipeline impose require a ping-pong of the 3D segmentation texture as well as wastes quite a lot of resources, since the whole slice is treated each step (restricting the region to analyze would require an extra auxiliary structure to be updated each step). As a result, this approach is less efficient than the CPU algorithm in most cases, since it only processes potentially selectable voxels. Another possibility would be to use the newer Fermi GPUs, that would permit CUDA computation, and making the result available for the OpenGL pipeline. We leave this for future work optimizations.

We chose an incremental strategy because we want to prevent the user to wait for the whole segmentation to finish: slabs are segmented at once, trying to segment the data in advance of users' moves. If the user had to wait a long time, this would probably break his or her flow of thought and interaction would become uncomfortable. However, the segmentation process may slow down frame rates if the region to expand is very big. In those cases, the user may notice a slowing in the interaction refresh, as compared to a previously segmented region. In a typical on-the-fly segmentation the frame rate may drop to 4-8 fps. Nevertheless, these frame rates are high enough for the user to perceive continuous response from the system.

5. Results and Discussion

5.1. Results

As we have discussed in Section 4.3, and as the accompanying video shows, the interaction is intuitive, and the frame rates we obtain guarantee a soft and continuous interaction with the structures.

We have tested our tool with different models. The manip-

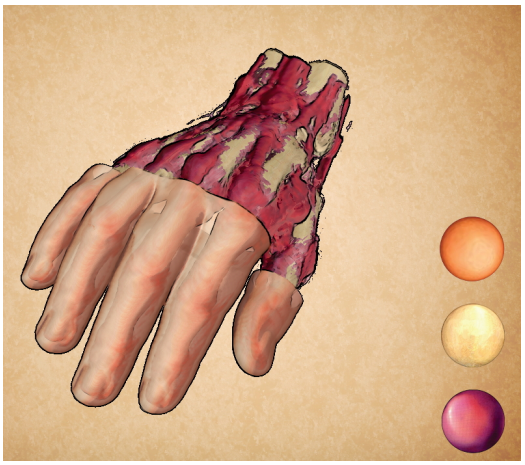


Figure 7: *Final rendering: Texturing over the volume ray casting using the lit sphere, contours and a background texture. On the right we may see the three different lit spheres [SMGG01] used to texture the skin, the bones, and the muscles, respectively. These were taken from [BG07].*

Model	Resol	DVR	Illustrative Cuts			
			Edition		Final	
			1Π	2Π	1Π	2Π
Body	$512^2 \times 512$	16.3	14.2	12.2	14.1	12.2
Head	$512^2 \times 460$	33.2	23.7	21.9	23.5	21.9
Foot	$512^2 \times 256$	41.6	30.2	28.5	30.6	29.1
Hand	$512^2 \times 170$	48.2	33.6	31.2	32.9	31.0
Legs	$512^2 \times 256$	45.0	31.6	28.6	31.3	29.1
Tooth	$256^2 \times 162$	100	69.4	69.4	69.2	66.1
Engine	$256^2 \times 256$	93.6	61.4	61.4	61.4	60.9

Table 1: *Performance of our system with different segmented models in fps. Third column shows a simple DVR rendering using GPU RC with clipping plane* 1Π*. Fourth column shows the average frame rates obtained in several minutes of edition (i. e. extruding structures) using one (*1Π*) and two (*2Π*) planes, respectively. The final column shows average frame rates combining the extruded elements with the finish effects (extrusions, lit sphere, background texture, and contours).*

ulation was carried out in a 800×600 window. The GPU is an NVidia GTX 280 equipped with 1GB of RAM memory. We show two scenarios with segmented models in Table 1: In the first one (*Edition*) the user continuously extrudes the structures with one or two clipping planes (1Π and 2Π in column 4, respectively). The rightmost column shows the frame rates obtained for an edited volume with contours, background, and lit sphere. We can see that we obtain real times for all of the models.

5.2. User Study

As we have seen in Section 2, there is only a small number of methods that allow the direct manipulation of the structures in volume models. These systems do not implement structure-aware volume modification, apart from the elastic membrane clipping [BBBV12], which works in a similar way to us, but requires partial indirect manipulation that may be a time-consuming task. Since we work directly on the extruded structures, comparing with such techniques would not be fair. Thus, we decided to validate the usability of our approach through an informal user study among a set of 12 people, who were either computer scientists or computer science students.

The user study comprised two stages. First, the users got familiar with the system with the aid of a four minutes tutoring video and some minutes of practising time. The whole process took less than 10 minutes to finish for all the users, although the training time was not limited. After that, a set of 4 target images was shown randomly to each user. The goal was illustrated with three different views (an image of each one is shown in Figure 8) and the users had to obtain the same result. Two of the targets required the use of one plane and the other two required two planes, but did not give any clue to the users. The results show that the hand images were obtained in an average time of 41 seconds, while the foot images were obtained in around 56 seconds for the example with one plane, and 114 seconds for the example that required 2 planes. We also appreciate a tendency to reduce the times along the experiment, although a larger sample size should be analyzed in order to raise conclusions. In addition to the user study, we also showed our application to medical experts from different fields of internal medicine. They consider that our approach may be successfully used to create images for teaching media. One of the doctors also said that it might be useful for surgery preparation by using the real datasets of the patients.

5.3. Discussion

The system performs in real time for all the models we tried. Frame rate is slightly slower for non-segmented models but only when the user manipulations effectively trigger the segmentation process, otherwise frame rate does not suffer. However, this may be hardly noticeable, since the time required for the segmentation of one slab, in most cases we tested, is in the range of 0.008 to 0.012 seconds. Usually, this does not affect the interactivity of the edition. For brisk moves (that might involve a large region to segment), the frame rate decays, but we still get continuous interaction. In order to test a worst-case scenario, we simulated the selection of air in an empty region: For a large slab (covering 4 slices of 512×512), the region growing detected 958K voxels and is only 0.19 seconds. Normal extrusions usually generate 12K to 13K voxels, thus, the overhead is perfectly affordable. On the other hand, we may adapt the size of the

Figure 8: *Examples of the test images given to the users. The ones on the left can be obtained with a single plane, while the ones on the right require extruding from two planes.*

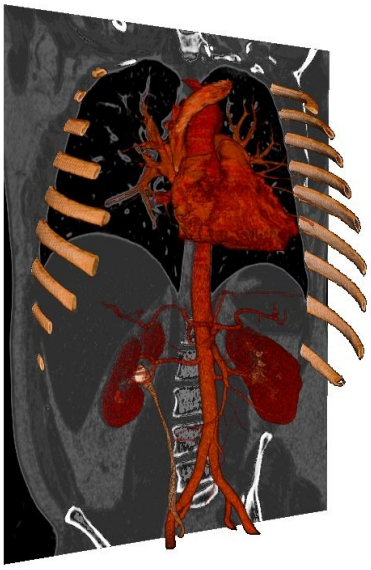


Figure 9: *Enhancing contextual information of a CT image. In this case, the ribs, the heart, the kidneys and the aorta have been extruded from a coronal slice of a human torso.*

slab to sudden moves if required, but in most cases a distance of 2 voxel diagonals from the clipping plane is a good compromise and achieves good frame rates. Moreover, we could also perform the whole segmentation at once, informing the user meanwhile of the underlying process. After that, the interaction would be realtime again. However, we did not find the necessity to swap to this mode during our experiments.

For non-segmented models, the accuracy of the extrusion may be limited by the simple region growing algorithm which only considers density information. The interaction system we designed allows us to incorporate some more sophisticated techniques (e.g multiple transfer functions). However, the correct identification of structures with similar density or fuzzy boundaries, more complex algorithms would be desirable [CMB*12, HGM09, TTFN06]. This algorithms are usually neither fully automatic nor interactive, and are not easy to adapt to our editing proposal. So, they should be applied as a pre-process. Nevertheless, thanks to the data structures design, our system is also able to seamlessly work with partially segmented volumes.

With no need of extra modifications, our approach is also a powerful tool for adding contextual information while inspecting medical images, as is shown in Figure 9.

6. Conclusions and Future Work

We have implemented an interactive tool that allows the easy and intuitive creation of illustrations for the use of medical doctors in scientific documents or as a means of information communication such as in pre-operative studies or teaching. Instead of manipulating several clipping planes or complex bounding geometries, our approach directly extrudes structures from a cross-section. Furthermore, it provides fine manipulation of the image result with simple drag-and-drop movements. A user study demonstrated that the learning process is small (up to 10 minutes). With this little training time, all the users were able to reproduce a set of selected target images in times that ranged from several seconds up to a couple of minutes. The whole interaction process is real time. Even with non-segmented data, we obtain interactive framerates. It is also quite flexible, since it provides the user with the possibility to add illustrative effects. In future, we want to extend our manipulation tool in order to manage other effects different from planar cuts, add some extra illustrative motifs and improve the segmentation algorithm while preserving interactive frame rates.

7. Acknowledgements

The authors want to thank S. Bruckner for providing some lit sphere maps, the Osirix website for providing some medical models and the anonymous reviewers for their valuable comments. This work has been supported by the project TIN2010-20590-C01-01 of the Spanish Government. J. Díaz is also supported by the grant FPI BES-2008-002404.

References

[BBBV12] BIRKELAND A., BRUCKNER S., BRAMBILLA A., VIOLA I.: Illustrative membrane clipping. *Computer Graphics*

Forum 31, 3 (2012), 905–914. 3, 8

[BF08] BURNS M., FINKELSTEIN A.: Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Trans. Graph. 27* (2008), 154:1–154:7. 2

[BG05] BRUCKNER S., GRÖLLER M.: Volumeshop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization 2005* (2005), pp. 671–678. 3

[BG06] BRUCKNER S., GRÖLLER M.: Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1077–1084. 1, 3

[BG07] BRUCKNER S., GRÖLLER M.: Style transfer functions for illustrative volume rendering. *Computer Graphics Forum 26*, 3 (2007), 715–724. 8

[BGKG06] BRUCKNER S., GRIMM S., KANITSAR A., GRÖLLER M.: Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1559–1569. 3

[BHW*07] BURNS M., HAIDACHER M., WEIN W., VIOLA I., GRÖLLER E.: Feature emphasis and contextual cutaways for multimodal medical visualization. In *EG/IEEE VGTC Symposium on Visualization 2007* (2007), pp. 275–282. 3

[BRV*10] BRUCKNER S., RAUTEK P., VIOLA I., ROBERTS M., COSTA M., GRÖLLER M.: Hybrid visibility compositing and masking for illustrative rendering. *Computers and Graphics 4*, 34 (2010), 361–369. 3

[BV09] BIRKELAND A., VIOLA I.: View-dependent peel-away visualization for volumetric data. In *Proc. of Spring Conference on Computer Graphics* (2009), pp. 121–128. 3

[CMB*12] CHICA A., MONCLÚS E., BRUNET P., NAVAZO I., VINACUA A.: Example-guided segmentation. *Graphical Models* (2012), –. doi:10.1016/j.gmod.2012.03.002. 6, 9

[CSC06] CORREA C., SILVER D., CHEN M.: Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1069 –1076. 3

[CSC10] CORREA C., SILVER D., CHEN M.: Technical section: Constrained illustrative volume deformation. *Computers & Graphics 34* (2010), 370–377. 3

[CSCS08] CHEN H., SAMAVATI F., COSTA SOUSA M.: Gpu-based point radiation for interactive volume sculpting and segmentation. *Vis. Comput. 24* (2008), 689–698. 3

[DWE03] DIEPSTRATEN J., WEISKOPF D., ERTL T.: Interactive cutaway illustrations. In *Computer Graphics Forum* (2003), pp. 523–532. 2

[HBR*92] HÖHNE K., BOMANS M., RIEMER M., SCHUBERT R., TIEDE U., LIERSE W.: A volume-based anatomical atlas. *IEEE Comput. Graph. Appl. 12* (1992), 73–78. 2

[HGM09] HU Y.-C., GROSSBERG M., MAGERAS G.: *Survey of Recent Volumetric Medical Image Segmentation Techniques*. 2009. 9

[HKRs*06] HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D., ENGEL K.: *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006. 6

[ISC07] ISLAM S., SILVER D., CHEN M.: Volume splitting and its applications. *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 193–203. 3

[KSW06] KRUGER J., SCHNEIDER J., WESTERMANN R.: Clearview: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 941–948. 3

[KVPL04] KONRAD-VERSE O., PREIM B., LITTMANN A.: Virtual resection with a deformable cutting plane. In *Proceedings of Simulation und Visualisierung* (2004), pp. 203–214. 2

[LACS08] LI W., AGRAWALA M., CURLESS B., SALESIN D.: Automated generation of interactive 3d exploded view diagrams. *ACM Trans. Graph. 27* (2008), 101:1–101:7. 3

[LRA*07] LI W., RITTER L., AGRAWALA M., CURLESS B., SALESIN D.: Interactive cutaway illustrations of complex 3d models. *ACM Trans. Graph. 26* (2007). 2

[MB06] MCINERNEY T., BROUGHTON S.: Hingeslicer: interactive exploration of volume images using extended 3d slice plane widgets. In *Proceedings of Graphics Interface 2006* (2006), pp. 171–178. 2

[MC10] MCINERNEY T., CRAWFORD P.: Ribbonview: interactive context-preserving cutaways of anatomical surface meshes. In *Proceedings of the 6th international conference on Advances in visual computing - Volume Part II* (2010), pp. 533–544. 2

[MRH08] MENSMANN J., ROPINSKI T., HINRICHS K.: Interactive cutting operations for generating anatomical illustrations from volumetric data sets. *Journal of WSCG – 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 16*, 1-3 (2008), 89–96. 3

[MTB03] MCGUFFIN M., TANCAU L., BALAKRISHNAN R.: Using deformations for browsing volumetric data. In *Proceedings of IEEE Visualization* (2003), pp. 53–60. 3

[ONI05] OWADA S., NIELSEN F., IGARASHI T.: Volume catcher. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (2005), pp. 111–116. 3

[RBG08] RAUTEK P., BRUCKNER S., GRÖLLER M.: Interaction-dependent semantics for illustrative volume rendering. *Computer Graphics Forum 27*, 3 (2008), 847–854. 3

[RK82] ROSENFELD A., KAK A.: *Digital Picture Processing*, 2nd ed. Academic Press, Inc., Orlando, USA, 1982. 5

[SMGG01] SLOAN P. J., MARTIN W., GOOCH A., GOOCH B.: The lit sphere: a model for capturing npr shading from art. In *No description on Graphics interface 2001* (2001), pp. 143–150. 6, 8

[TKS10] TATZGERN M., KALKOFEN D., SCHMALSTIEG D.: Compact explosion diagrams. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (2010), pp. 17–26. 3

[TTFN06] TAKAHASHI S., TAKESHIMA Y., FUJISHIRO I., NIELSON G.: Emphasizing isosurface embeddings in direct volume rendering. In *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Bonneau G.-P., Ertl T., Nielson G., (Eds.), Mathematics and Visualization. Springer Berlin Heidelberg, 2006, pp. 185–206. 9

[VFSG06] VIOLA I., FEIXAS M., SBERT M., GRÖLLER E.: Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 933–940. 3

[VKG05] VIOLA I., KANISTAR A., GRÖLLER M.: Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics 11*, 4 (2005), 408–418. 1, 3

[WEE03] WEISKOPF D., ENGEL K., ERTL T.: Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Transactions on Visualization and Computer Graphics 9* (2003), 298–312. 1, 2