Smart Surrogate Widgets for Direct Volume Manipulation

Sergej Stoppel *
University of Bergen

Stefan Bruckner†
University of Bergen

Figure 1: Illustrative visualization techniques such as cutaways or ghosting views are used to emphasize important concealed structures. Typically such structures are carefully segmented prior to the visualization. Our approach allows for the simple creation of illustrative visualizations without prior processing of the data. The leftmost image shows the visible human data set manipulated with our technique to reveal the skeleton and inner organs. The next image shows an illustrative visualization of the human cochlea. The second image from the right shows an illustrative cutaway of a beetle. The rightmost image shows a cutaway illustration of a high voltage power outlet.

Abstract

Interaction is an essential aspect in volume visualization, yet common manipulation tools such as bounding boxes or clipping plane widgets provide rather crude tools as they neglect the complex structure of the underlying data. In this paper, we introduce a novel volume interaction approach based on smart widgets that are automatically placed directly into the data in a visibility-driven manner. By adapting to what the user actually sees, they act as proxies that allow for goal-oriented modifications while still providing an intuitive set of simple operations that is easy to control. In particular, our method is well-suited for direct manipulation scenarios such as touch screens, where traditional user interface elements commonly exhibit limited utility. To evaluate out approach we conducted a qualitative user study with nine participants with various backgrounds.

Index Terms: [Human-centered computing]: Interaction design process and methods—User interface design

1 Introduction

Volume visualization is a well established method for the exploration, investigation and representation of 3D scalar fields, with various applications in medicine, engineering, physics, architecture and biology. In the recent years volume visualization techniques have been increasingly targeted at non-expert scenarios, for instance in the form of museum installations, and novel interfaces such as touch-based tabletop displays are becoming more common [34]. However, interacting with volume visualizations still remains a complex task requiring training and expertise, and new developments in terms of intuitive interaction design are needed.

*e-mail: sergejsto@gmail.com
†e-mail:stefan.bruckner@uib.no

Interaction interfaces for volume visualizations primarily come in two forms: as spatially-detached widgets allowing for complex and detailed manipulations, e.g., for the specification of transfer functions, and as widgets with direct spatial semantics, that tend to be very general (e.g., bounding boxes for the specification of clipping operations). Since direct manipulation was proposed by Shneiderman [24] in 1982, it had a major influence on interaction design. While direct interaction has its weakness in managing abstract or attribute-rich objects, the directness particularly benefits the novice user in terms of intuitiveness and seamless interaction results.

For these reasons, we present a direct volume manipulation approach targeted at non-expert users inspired by the notion of surrogate objects as discussed by Kwon et al. [14]. We propose visibility-driven smart surrogate widgets that are automatically constructed by fitting a model onto the visible structures of the current view. We allow for several interactions on the surrogate objects which are directly translated to operations on the current visualization. More specifically, we follow four main principles of Shneiderman [24]:

- Continuous representation of the object of interest
- Physical actions instead of complex syntax
- Rapid, incremental, reversible operations whose impact on the object is immediately visible
- Layered or spiral approach to learning that permits usage with minimal knowledge

Our contributions in this paper are the following:

- A novel approach for visibility driven surrogate widget generation
- Incremental, flexible and invertible construction of widgets, integrated directly into the volume visualization

- Seamless translation of simple gestures into volume manipulations
- Intuitive widget design that allows for complex illustrative manipulation through simple, incremental interactions without prior volume segmentation

2 Related Work

Research inspired by art and illustration is a well established subfield of visualization and computer graphics. Illustrative visualization, for example, aims to reproduce and extend traditional illustration techniques in computer-based visualization methods. Much research has been devoted to developing methods for visualizing occluded structures in their spatial context [28]. Hauser et al. [12] introduced two-level volume rendering, an approach that allows for the flexible combination of different rendering techniques for subsets of the data. Bruckner et al., for instance, introduced several illustrative techniques for volume visualizations such as cutaways [4] and exploded views [6]. Viola et al. [29] proposed importance-driven feature enhancement in volume visualization in order to steer the application of abstraction methods. Based on these ideas, Viola et al. [27] presented a system that determines the most expressive view for a user-selected feature. Li et al. [16] proposed a novel cutaway design based on the occluder geometry, where previous approaches focused only on the occluded object. Lawonn et al. [15] presented an approach for the occlusion-free visualization of blood flow and wall thickness in vascular data. Elmqvist et al. [7] discussed a taxonomy of the design space for occlusion management techniques. Birkeland et al. [3] presented a view-dependent technique for the generation of peel-away views. These approaches focus on the visualization of occluded structures. While our approach also can be used to generate cutaways or ghosted views, our primary aim is to support the specification and manipulation of specific features of interest by the user instead of requiring a pre-processing step such as volume segmentation to identify relevant structures.

The benefits of direct manipulation in contrast to command languages were already discussed by Shneiderman [24] in 1983. For more than three decades, the direct manipulation paradigm has influenced interface design, encouraging designers to minimize indirection and to transform domain objects directly into intuitive widget interfaces. A common design pattern for data exploration is the concept of magic lenses. Magic lenses reveal secondary structures through dynamic lens widgets. For a detailed survey on interactive lenses for visualization we refer to Tomisnki et al. [26]. The design of our smart surrogate widgets incorporates the concept of magic lenses by providing focus regions for which alternate optical properties can be specified.

Kniss et al. [13] described a set of direct manipulation widgets for intuitive transfer function manipulation, focusing on probing operations to ease the establishment of correspondences between data attributes and visible features. Guo et al. [11] developed a sketch-based approach for volume rendering manipulation, allowing for quick and easy adjustment of properties such as such as color, transparency, contrast, or brightness. Gerl et al. [10] proposed an interactive approach for the explicit specification of semantics in volume visualization, to visually assign meaning to both input and output parameters of the visualization mapping.

Weiskopf et al. [31] introduced techniques for depth-based clipping in texture-based volume rendering, allowing for the efficient use of complex clipping geometries. Our work instead focuses on the easy creation and modification of clipping geometries, exploiting the information already present in the current visualization state. For this purpose, as proposed in the work of Kwon et al. [14], we use the concept of surrogate interactions that enable the user to interact through easily-understandable proxies instead of more complex domain objects. Yu et al. [36] presented FI3D, a direct-touch data exploration technique for 3D visualization. While they primarily targeted global operations such as rotation, zooming, and translation, their approach also incorporated simple cutting and selection tools.

Instead of considering the underlying data in its entirety, our approach is deliberately based on visibility and hence our widgets are constructed according to the current visualization state, i.e., the structures that are currently visible and identifiable, allowing users to manipulate the objects that they see. This is in contrast to data-driven approaches, that attempt to extract additional semantics irrespective of the current state. Gagvani et al. [9], for instance, presented a technique to animate volumes using a volumetric skeleton tree. McGuffin et al. [17] explored an alternate strategy for volumetric cutting and peeling tools. By using predefined semantic information, the user can apply deformations to selected layers using pop-up menus and integrated 3D widgets. Birkeland et al. [2] developed an illustrative clipping technique with automated feature preservation using an elastic membrane that adjusts itself to salient structures. Le Muzic et al. [18] presented Visibility Equalizer, visualization approach for mesoscopic biological models that applies cutting planes only to certain parts of the data.

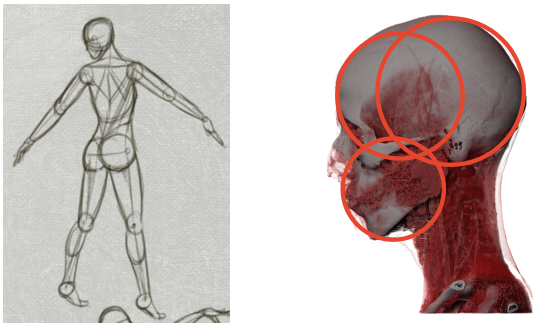
While data-driven methods can be very useful for global manipulations, e.g., in the context of transfer function design [30], more localized operations greatly benefit from the incorporation of visibility and other information. Selecting three-dimensional shapes through a two-dimensional interface can be challenging, and structure-aware or context-aware interaction techniques can be employed to overcome some of the inherent ambiguities. These techniques try to deduce the user intent based on the underlying data and the current state of the provided visualization. Owada et al. [19] solved the challenge of three-dimensional selection by asking the user to draw the perceived outlines of the displayed volume and deducing well-suited segmentations from the sketch. This approach was later extended by preprocessing the 2D domain before applying a 2D-to-3D stroke elevation algorithm [20]. Ropinski et al. [22] proposed an approach for stroke-based transfer function design, where the transfer function is computed to emphasize user-specified features of interest selected by simple sketches.

The work of Wiebel et al. [33] proposed methods to select 3D positions for 2D picking operations based on visibility information. They later extended this approach to map 2D strokes to the most visible features in a 3D volume [32]. Based on this idea, Stoppel et al. [25] extended the method to select visible surface patches in direct volume rendering. Yu et al. [35] presented a family of gesture-driven Context-Aware Selection Techniques (CAST) for the interaction and analysis of large 3D particle clouds. In addition to using contextual information to infer the likely meaning of user interactions, our approach aims to incorporate this information already in the construction of interaction widgets, offering additional guidance to novice users.

3 Smart Surrogate Widgets

A central aspect of our approach is to provide users with easily understandable yet sufficiently powerful and flexible interaction widgets based on the content of the visualization as seen on screen. When confronted with visualizations of volumetric data, novice users frequently seek for ways to ”open up” visible structures in order to look inside, but common interaction widgets such as bounding boxes, due to their uniform structure, can make it difficult to specify a desired operation. In particular, the initial placement of such tools can be challenging, as 3D manipulation is often difficult for novice users. For such users, ideally, a system should ”know” the spatial structure of visible features and should provide them with an ”expected” set of operations.

While it is of course difficult to capture such subjective notions, we can look at existing cutaway and breakaway illustrations for inspiration. Such illustrations frequently use simple and symmetric shapes to partially remove occluding structures, allowing the viewer



a b

Figure 2: (a) Illustrators often approximate complex shapes with a set of circles and the connections between those (Illustration by Paulina Kawenka). (b) A set of only three spheres already gives a good approximation of the shape of a human head.

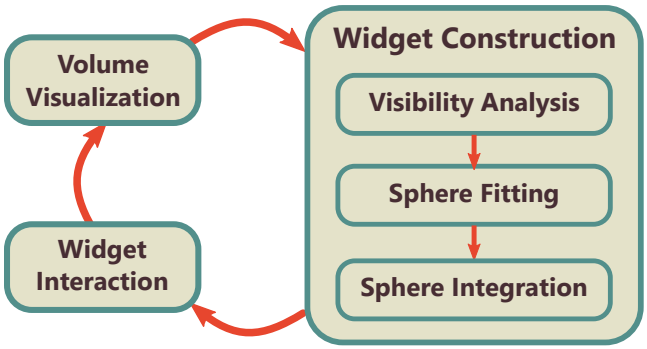


Figure 3: Overview of our approach for smart surrogate widgets and its integration into the volume visualization pipeline. To fit the surrogate widget into the rendered image, we analyze the visibility and compute a set of prominent points for the given visualization state. In the next step, we fit a sphere into the computed point set. The new sphere is combined with the existing widget. The updated widget can be used for further volume manipulation.

to mentally reconstruct the entire object. Furthermore, when sketching an object, artists and illustrators often approximate complex structures with a set of simple shapes such as spheres, as shown in Figure 2(a). Motivated by this, we propose surrogate widgets as constructions of primitive shapes such as spheres and tapered cylinders. In fact, medical and biological scans can often be approximated with a relatively small number of spheres as illustrated in Figure 2(b).

A general overview of our approach, which we detail in the remainder of this section, is shown in Figure 3. First, we discuss the automatic construction of our widgets in Section 3.1 and then proceed to discuss the available interaction mechanisms in Section 3.2.

3.1 Widget Construction

The main idea for the construction of our widgets is the aim of representing the most prominent visible structures using a set of simple primitive shapes that can be easily interpreted and manipulated. Topologically, our widgets are undirected acyclic graphs, where the nodes correspond to spatially anchored spheres with varying radii and the edges correspond to tapered cylinders that connect them. The widgets are constructed incrementally whenever triggered by the user. When requested, our system initiates the construction of a new widget, initially consisting of a single sphere. If more complex operations are desired, additional widget components can be added, which will be connected to the existing spheres via a tapered

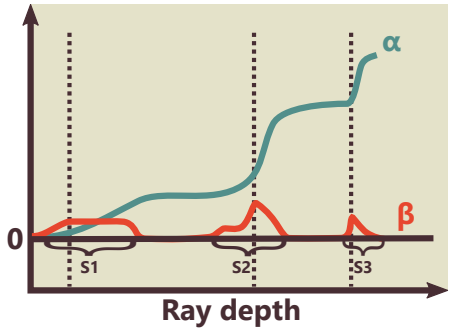


Figure 4: Borders of structures S_k are detected by local maxima of β. This defines a point cloud for the whole scene which is used for widget fitting.

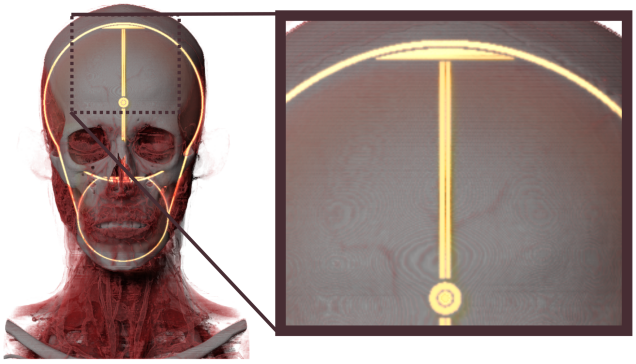


Figure 5: Smart surrogate widget rendered into the visualized volume. The widget is emphasized through a glow in this image.

cylinders based on spatial proximity. Likewise, the user can remove spheres when they are no longer needed or discard the entire widget.

The creation and extension of the widgets requires a reliable method for fitting a sphere into the rendered data. To achieve this, we employ a random sample consensus (RANSAC) model [8]. RANSAC is an iterative learning algorithm that estimates parameters of a mathematical model by randomly sampling the observed data designed to operate on point cloud data. The algorithm works by iteratively identifying outliers in the data and estimating the model with data not containing the outliers. As such RANSAC is tolerant to noise in the data and is commonly applied to similar scenarios such as object recognition.

As mentioned, the RANSAC algorithm requires a point cloud of object surface points, for estimating the model. Therefore, we have to compute surface points for the visible structures in the rendered image. Inspired by the point picking approach proposed by Wiebel et al. [33], we estimate the surface points of visible structures as the most prominent points, i.e. points with the highest opacity gain inside each structure. As illustrated in Figure 4, we evaluate the accumulated opacity α for each point p along the ray. If the point is already inside a widget, its opacity is considered to be zero. From the function α, we compute its derivative function β. Note that the derivative β is positive throughout the ray profile, being zero at plateaus of α. We can then divide β into strictly positive segments, where each segment represents a visible structure along the ray. Following Wiebel et al., we identify the surface points of the visible structures as the most prominent points $p_{k,max}$ of each structure along the ray (indicated by dotted lines in Figure 4). For each structure S_k, the most prominent point $p_{k,max}$ is then defined

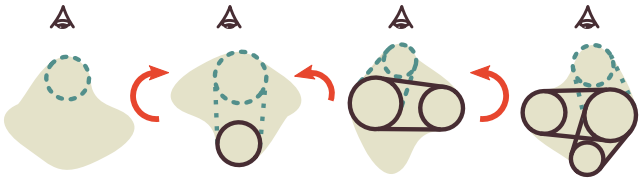
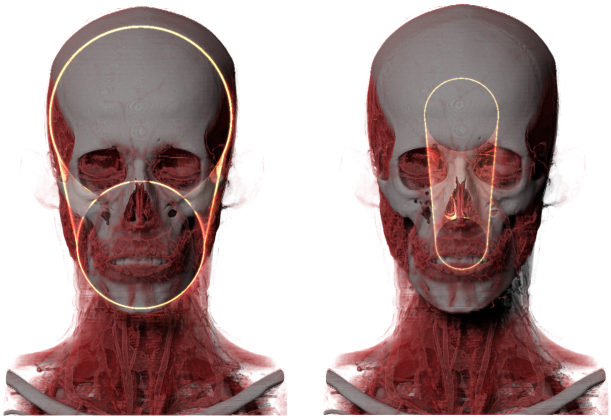


Figure 6: When a new sphere is added to the widget, it is connected to the closest existing sphere. As such the widgets can branch, as shown in the rightmost image.



a b

Figure 7: The widget size can be changed by pulling on the outer frame. The selected widget part is emphasized with a glow.

as:

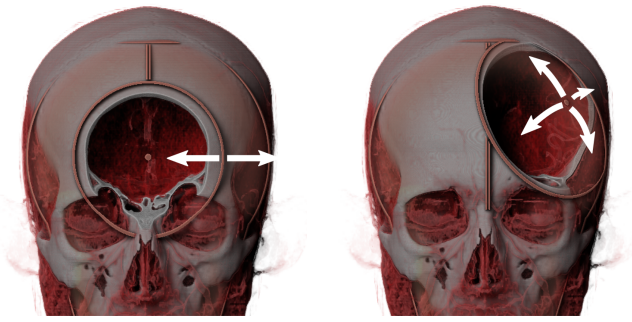
$$\beta(p_{k,max}) \geq \beta(p) \qquad \forall\, p \in S_k \qquad (1)$$

$$d(p_{k,max}) \leq d(p) \qquad \forall\, p \text{ s.t. } \beta(p) = \beta(p_0 k,max). \qquad (2)$$

where $d(p)$ is the distance of point p to the camera. Thus, we define the surface point of each structure as the nearest point with the highest opacity gain inside each structure. Because the RANSAC algorithm considers all points to be equal it would not distinguish between very transparent or very opaque structures when fitting the model. To address this we simply save the surface points of the most prominent structures along each ray by computing the visibility contribution of each structure. This approach provides us with a coherent estimate for the visually most prominent point for each pixel, resulting in a point cloud covering the whole rendered volume that can be used to find the best fitting sphere into the visible data.

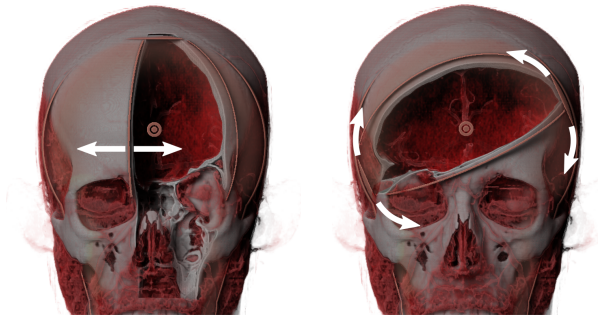
After finding the best matching sphere based on the visibility information, it is either integrated into the existing widget or is declared to be the initial widget, if no widget exists so far. To integrate the new sphere into an existing widget, a tapered cylinder is constructed between the new sphere and the nearest sphere in the existing widget. If a new sphere is added to a widget with more than three spheres, it may be connected with a sphere with more than one existing connection. In this case the widget will branch out, as illustrated in Figure 6.

The radius of fitted spheres is limited to an interval of $[s_{\min}, s_{\max}]d$, where d is the smallest extent of the volume's bounding box. The experimentally determined default values in our implementation are $s_{min} = 0.03$ and $s_{max} = 0.4$ (preventing the matching of overly large or very small objects), which were used for all results in the paper and the video, but the user is able to adjust these values freely.



a b

Figure 8: (a) The iris size can be changed by pulling on the circular iris opening. (b) The iris can be rotated by moving a handle representing the center of the iris opening.



a b

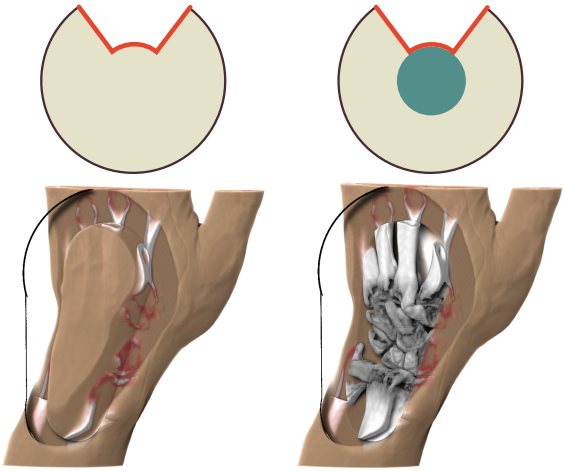
Figure 9: The wedge can be adjusted through two independent sphere arcs (a) and can be rotated around the view vector by pulling on the poles of the wedge (b).

3.2 Interaction Design

Our approach is designed to provide instant and context-dependent means for performing localized manipulations of volume data, including operations such as cutting or highlighting. By adjusting viewing parameters like rotation and zoom, users indicate what they are interested in, and our approach for widget construction automatically provides manipulation facilities based on the visible structures. We deliberately selected a set of simple and easy-to-interpret operations, which still provide considerable flexibility.

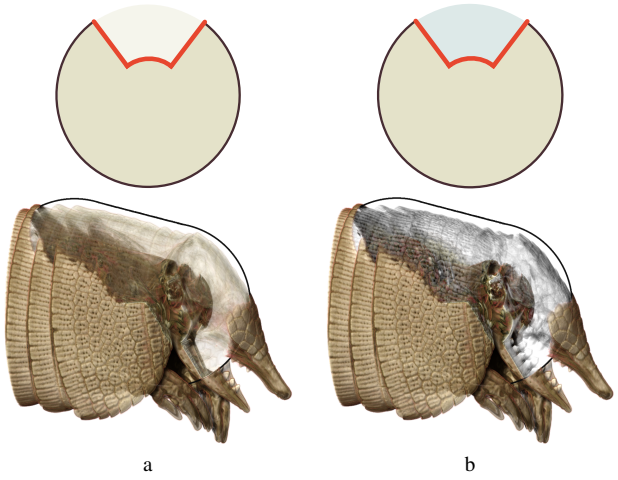
Graphical editor software such as Photoshop typically divide manipulation tools into two components. The first component allows direct interaction via surrogate objects that are embedded into the image, while the second component represents control panels or dialogs that do not have a direct spatial interpretation, such as opacity manipulation or contrast adjustment. The surrogate widgets in graphical editor software typically consist of a frame around the focus object with handles for drag interactions, such as rotation or deformation. Additional properties are controlled via user interface elements such as sliders or text fields which are typically arranged around the image. Our smart surrogate widgets are designed to conform to this common model and consist of a spatial representation shown as part of the rendered image (space widget) and a floating panel displayed in the upper part of the viewport (detached widget).

The region covered by the space widget is subdivided into the focus region and the background. Initially, the focus region is empty, i.e., the widget does not affect the visualization. By clicking on a sphere of the widget, it will be activated and can be resized by



a b

Figure 10: The upper row shows a simplified cross section of the widget. The widget is opened with the wedge tool and the focus area is rendered transparent. The size of an area inside the smart widgets that is unaffected by the cone and the wedge manipulations can be changed with a simple slider. The computed volume can either be shown in with the same transfer function as the remaining data (a) or with a second transfer function (b).



a b

Figure 11: The upper row shows a simplified cross section of the widget. The upper part of the armadillo is covered with a smart surrogate widget consisting of two spheres. Through a combination of cones and wedges it is now possible to look into the body by reducing the opacity inside the widget using (a) the initial transfer function or (b) the alternate transfer function.

dragging its contours, as demonstrated in Figure 7. The focus region can be manipulated by two simple tools, an iris tool and a wedge tool. Changes of the focus region are propagated along all cylinders attached to it, and the final focus region is formed by the union of these subregions. The user can rotate the iris by moving a small circle on a sphere (see Figure 8). To change the size of the iris, users can click and drag a ring around the position circle. The focus is defined by the region enclosed in the cone defined by the iris ring and the sphere center as apex, as shown in Figure 8. The wedge tool can be controlled by moving two independent arcs, and the focus region is set to the volume between the two arcs, as shown in Figure 9(a). The two arcs can be rotated around the view vector by pulling on their poles (see Figure 9(b)).

The main purpose of the detached widget is to control the appearance of the focus region and to provide access to additional global operations. First, we provide a slider to limit the extent of the focus region, essentially generating an inner boundary. Modifying this slider inflates a volume from the center of the widget that is not part of the focus region. By switching between two radio buttons, the inner volume can either be displayed with the same transfer function as the rest of the data (see Figure 10(a)) or with the alternate transfer function (see Figure 10(b)). Furthermore, we provide a slider that can be used to control an opacity multiplier for the focus region. A multiplier with the value 0 renders the focus completely transparent, resulting in a cutting operation, while a value of 1 results in an unmodified rendering with the selected transfer function. In addition to modulating the opacity, the user can also switch to display the focus region with an alternate transfer function. This is illustrated in Figure 11, where a widget consisting of two spheres covers a CT scan of an armadillo. By using a combination of the iris and wedge tools, the body is opened and the opacity inside the focus region is reduced. In Figure 11(a), the focus area is rendered with the initial transfer function, whereas in Figure 11(b) the focus area is rendered using the alternate transfer function. In both cases a smaller volume inside the smart surrogate widget is unaffected by the opacity changes, which allows for better orientation and comparison.

To ensure the third main principle of Shneiderman (rapid, incremental and reversible interaction with immediately visible impact), the widget contains two buttons for adding a sphere and removing the last sphere. This simple mechanism enables the incremental construction of complex widgets through rotation of the volume and fitting the new widget parts to previously covered areas. Of course our system also allows for the manual repositioning of widget parts. When initiated, the user can move the selected sphere on a plane perpendicular to the view direction. However, for all examples shown in this paper, we used the initial automatic placement, as described in Section 3.1.

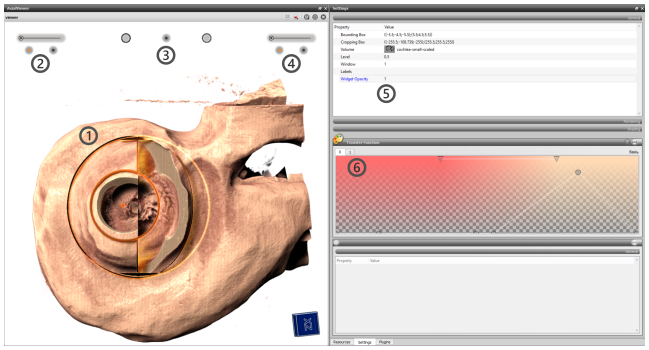


Figure 12: The standard setup of the smart surrogate interface. The numbered interface components are explained in detail in section 3.3.

3.3 User Interface

As we aim for direct interaction our user interface is mostly directly integrated into the rendered volume. A standard setup of out smart surrogate interface can be seen in Figure 12. The center component of the user interface is the smart surrogate widget that is directly integrated into the volume (1) in Figure 12. The detached widget has three components shown in the upper region of the viewport. Component (2) consists of a slider for controlling the size of the inner volume that is unaffected by the focus region and two radio buttons to change between the main and secondary transfer function for the inner volume. Component (3) consists of two buttons for

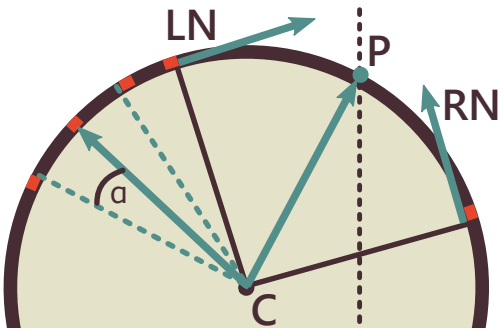


Figure 13: To compute the positions of the interaction tools on the sphere, we construct a vector *CP* and compute the dot product with normalized vectors *LN* and *RN* to estimate if the point *P* lies in the focus region defined by the user interface tool.

adding a new sphere to the widget and removing the last added sphere, as well as a radio button that can be activated to move the widget spheres manually. Component (4) is a slider for the opacity of the focus region and two radio buttons to switch the rendering of the focus region in the primary or secondary transfer function. Further parts of the user interface of our system are a control panel for advanced and expert settings that are hidden by default (5) as well as a transfer function editor (6).

4 Implementation

Our approach for the generation of smart surrogate widgets was implemented in C++ and OpenGL as an extension to an existing volume visualization framework [5]. The system provides several GPU-based rendering algorithms and the widget interface was implemented as part of a plugin. When a widget is created or extended, the visibility information is written to a separate buffer during volume rendering. The widget construction is then performed on the CPU in an asynchronous background thread, and the workflow is not interrupted. When the construction is finished, the new widget is displayed to the user. For sphere fitting, we use the RANSAC implementation of the Point Cloud Library [23].

As widgets can be geometrically complex, intersection testing is performed in two steps. First, for every frame we generate a layered depth image (LDI) from the spheres and tapered cylinders composing the widget. The LDI stores the intersection points with the individual widget elements as well as the corresponding element IDs. The intersection points are evaluated analytically, alleviating the need for tessellating the geometry.

This LDI is then used during rendering to test whether a sample position falls within the region covered by the widget. A uniform buffer containing the state of each widget element (its wedge and iris attributes, opacity multiplier, and transfer function setting) is passed to the corresponding shader. When a position is determined to lie within a particular widget element, the corresponding state attributes are retrieved using the element ID and used to test whether the point is part of the focus region as determined by the wedge or iris properties. For this, we construct a vector from the intersection point *P* to the sphere center or the closest point on the cylinder axis *C* (see Figure 13). The dot products between the normalized vector *CP* and the wedge normals *LN* and *RN* give us enough information to identify the tool positions. Computing the dot product of *CP* and the iris axis allows us to determine if a point is covered by the iris or not. In the illustration in Figure 13, the wedge and iris tool positions are highlighted in orange.

The generated LDI is also used on the CPU when an interaction is initiated by the user. To modify the widget state according to an interaction event, we perform a lookup at the corresponding image position to check if the point lies within a sphere, a cylinder, or both. We construct the vector *CP* where *C* is either the center of the sphere or the closest point on the cone axis. Similar to the procedure shown in Figure 13, we compute a set of dot products to determine the relationship of the point to the interaction tools and modify their positions according to the specific event.

This approach is simple and flexible, and can be easily integrated into existing volume rendering applications. For the results in this paper, we used a renderer based on the algorithm described by Patel et al. [21], but our method supports any standard volume rayasting or slicing approach. In terms of performance, the overhead introduced by our approach is comparably small. Whenever a widget is added or extended, the visibility needs to be evaluated and sphere fitting needs to be performed. The performance of this is dominated by the CPU-based RANSAC fitting. On average, the entire process takes around 300 to 400 milliseconds for the presented data sets. This could potentially be reduced by using a GPU-based implementation for the fitting, but due to the fact that this is performed asynchronously in the background, allowing the user to continue interacting without interruption, the delay is hardly noticeable and was not mentioned by any of the users during our evaluation. Furthermore, for each frame the generation of the LDI and the additional tests during rendering incur a slight overhead, as shown in Table 1. As can be seen, there is a reduction in the frame rate, but the overall impact of our approach on the performance is not substantial. As already mentioned, we use a volume rendering algorithm that includes advanced illumination including dynamic soft shadows, which is why the frame rates in Table 1 are generally lower than with a simpler standard ray caster.

Data set	Dimensions	Regular	Widgets
Beetle	$832 \times 832 \times 494$	11.87 fps	10.67 fps
Cochlea	$527 \times 434 \times 531$	10.14 fps	9.11 fps
Pawpawsaurus	$958 \times 646 \times 1088$	4.38 fps	4.17 fps
Outlet	$425 \times 551 \times 895$	9.33 fps	8.23 fps
Supernova	$432 \times 432 \times 432$	4.87 fps	4.32 fps

Table 1: Comparison of normal volume rendering (regular) to our approach with additional handling of smart widgets (widgets) as measured on an NVidia GeForce GTX 780 GPU with a viewport size of 967×967 pixels.

5 Results

In the following, we illustrate the benefits of our approach for five different scenarios and demonstrate how smart surrogate widgets can be used to easily manipulate different types of volumetric data sets. All the examples discussed here, as all the results presented in this paper, only use automatically determined widget placement without manual adjustment.

5.1 Stag Beetle

Medical and biological illustrations often use cutaway views, where the obscuring tissue such as skin is not drawn to reveal the structures underneath. In the this example, we use a CT scan of a stag beetle. The beetle is rotated to be viewed from above. Our smart surrogate widget covers the thorax and the abdomen. With a combination of the iris tool on the thorax and a wedge on the abdomen, we can easily obtain a cutaway illustration with emphasis on the inner structures of the beetle (see Figure 14). Note that the sphere centers are positioned below the beetle, such that the sphere surface closely matches the exoskeleton, as shown in Figure 15. The inner volume of the widget was inflated to cover the whole widget and was displayed in a transfer function that shows the internal structures of the beetle.

Figure 14: Example of a cutaway illustration generated using a combination of the wedge and iris tool to reveal the inner structures of the beetle.

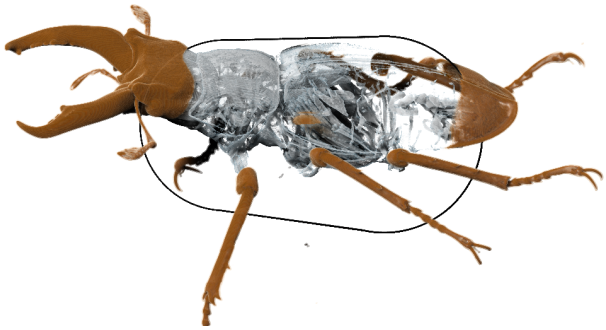
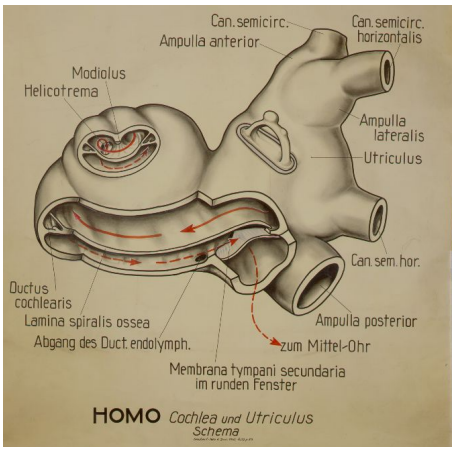
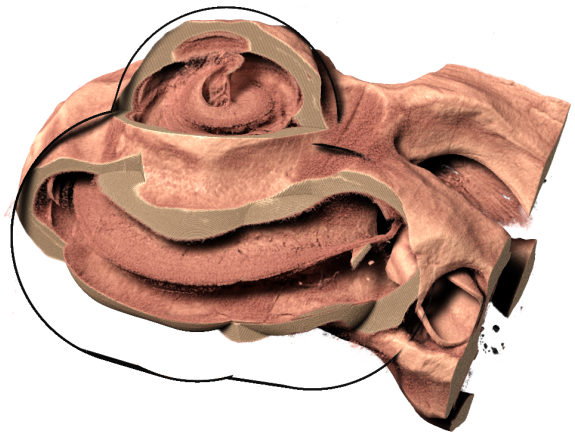


Figure 15: Our algorithm tries to position the spheres close to the selected boundary. When flat surfaces are present this results in spheres that are bigger than the covered structure.



a



b

Figure 16: (a) Medical illustration of the cochlea from Humboldt-University Berlin. (b) An illustrative visualization obtained with our technique by a first-time user during the user study.

5.2 Cochlea

The human cochlea is the auditory portion of the inner ear. It is a spirally-shaped cavity divided into three fluid-filled parts. The complex structure of the cochlea makes it an interesting subject for medical illustrations. Because of its non-regular shape, illustrators often depict the cochlea using a number of non-planar cutaways as shown in Figure 16(a). During our evaluation, we asked the participants to recreate a similar illustration with our technique on a CT scan of the cochlea. One result of this process generated by a non-expert participant of the study can be seen in Figure 16(b). To generate this visualization, the participant created a branched widget with four spheres in total. The cochlea was opened with the wedge tool and a volume that is unaffected by the tool inside the widget was inflated. To achieve this or comparable results the participants needed 7:12 minutes on average ranging from 3:37 minutes to 13:42 minutes. Although the data set does not show strong symmetry, seven out of nine participants stated that they found the that the task of creating this illustration was easy to complete. We discuss further details of our evaluation in Section 6.

5.3 Pawpawsaurus

Paleontologists spend the majority of their time in the field excavating and evaluating fossils. To evaluate new findings and to estimate possible kinship between the new finding and known species, they commonly compare the found bone fragments with already classified findings in a locally stored database. This can be a challenging task as subtle details can be very important for the classification. Structures such as the nasal cavities or the cavities of the inner ear and brain give indications about the relationships between species and help to reconstruct the anatomical features. The state-of-the-art procedure to solve this task is to use cutting planes, but it can be difficult to align them appropriately. The skull of the pawpawsaurus [1] is a relatively well-preserved specimen that was scanned with a high-resolution CT scanner in 2014. Multiple cavities and structures of interest in this data set are obstructed by the jaw, as shown in Figure 17(a). To reveal the nasal cavities, we open an automatically-placed sphere with the wedge tool as demonstrated in Figure 17(b) and (c). Next, a second sphere is placed automatically further back at the location of the brain cavities. We use the iris tool to remove obstructing bones, as shown in Figure 17(d) and (e). The final result, which depicts the nasal cavities, brain cavity, and inner skull fragments, can be seen in Figure 17(f). Experts in paleontology who evaluated our tool particularly valued the possibility to locally change the opacity in the wedge and iris tools, as this helped to quickly switch between detailed inspection and evaluation of the structural relationships between the fragments.

5.4 Outlet

Technical illustrations often use section views, which exploit symmetries in order to depict inner structures. Without additional infor-

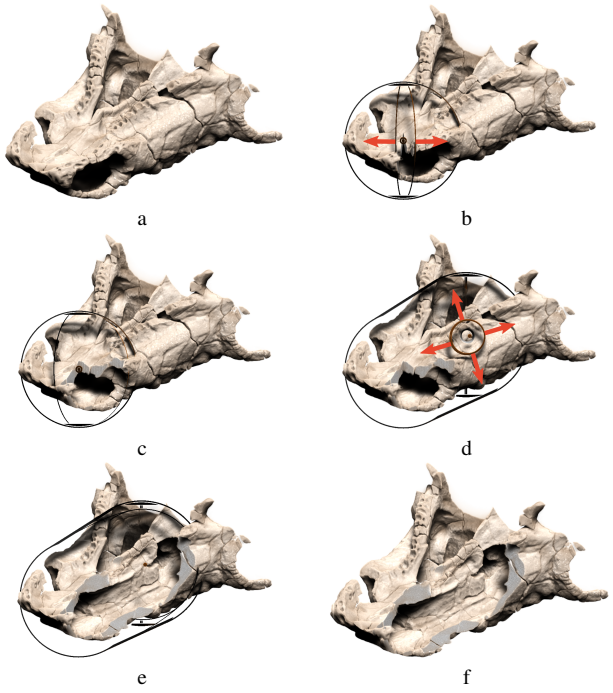


Figure 17: (a) The skull of a pawpawsaurus seen from underneath, cavities of interest are not visible. (b) The first sphere of the widget is created, and we open the wedge tool. (c) The wedge tool is opened to reveal the nasal cavity. (d) A second sphere is added to the widget and iris tool is opened. (e) The iris tool is opened to reveal the brain cavity. (f) The final illustration showing the nasal and brain cavities.

mation such as segmentation, this is difficult to achieve in volume visualization. Our widgets allow us to quickly achieve similar results without any preprocessing. We illustrate this on a CT scan of a high voltage power outlet. Using standard volume rendering, it is hard to show the inner structures and their positions inside the volume. The transfer function can be set to show a clear outer geometry hiding the inner structures (see Figure 18(a)), to show the inner structures (see Figure 18(b)), or it can be set to show the outer geometry semi-transparently (see Figure 18(c)). In either case the spatial relationships between the outer and inner parts are not entirely clear. Using our smart surrogate widgets, which are automatically are placed on the top and bottom of the volume, we can open the outlet with the wedge tool and the iris tool, as shown in Figure 19(a). The volume covered by the wedge and iris tool can be shown in a transfer function emphasizing the inner parts as depicted in Figure 19(b). Furthermore, we can inflate an inner volume inside the widget to be displayed in the same transfer function as the wedge as shown in Figure 19(c). This visualization depicts the structure and position of the inner parts clearly, and the shape of the outer parts can still be inferred.

5.5 Supernova

In this example, we use an entropy field from single time step of a supernova simulation made available by Dr. John Blondin at the North Carolina State University through the US Department of Energy's SciDAC Institute for Ultrascale Visualization. The data set consists of multiple interleaving layers with varying entropy levels. A volume rendering of the entropy can be seen in Figure 20(a). Physicists are interested in the structural development of the different entropy levels over time. To study the different entropy levels, a sophisticated transfer function has to be constructed that clearly

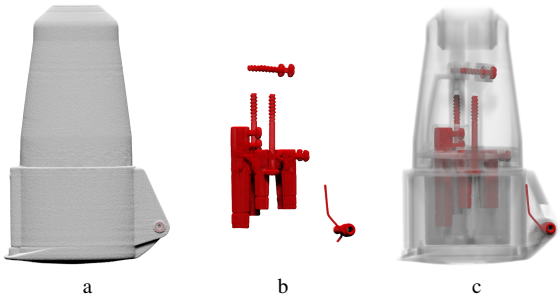


Figure 18: Using standard volume rendering methods the transfer function can be set to (a) show the outer geometry, to (b) show the inner structures, or to (c) show both using transparency.

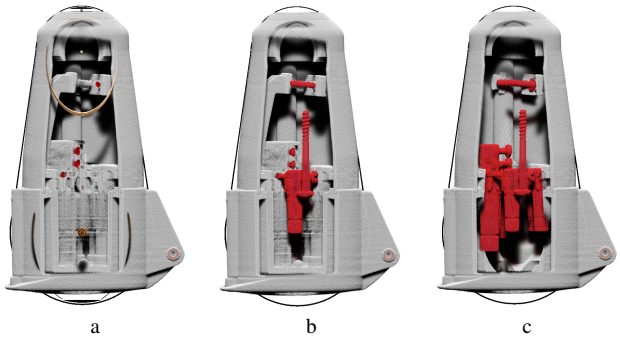


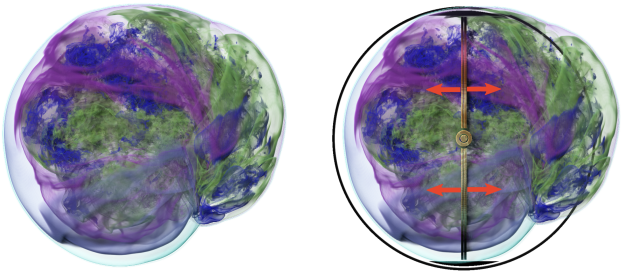
Figure 19: (a) The smart surrogate widget covers the outlet with two spheres first fitting to the rounder top part and the lower end. To open the volume, we used the wedge tool for the lower sphere and the iris tool in the upper sphere. Both spheres are active in this image. (b) The volume covered by the iris and the wedge is shown with a different transfer function to depict the structure in front of the cut. (c) An inner volume is inflated and shown with the same transfer function as used for the wedge and iris tool.

separates them. However, due to the high degree of occlusion, local manipulations can help to improve the clarity of such visualizations.

The automatically-placed widget represents a good match for the overall structure of the dataset, as shown in Figure 20(b). Using the iris tool, we can open the data and display the inner volume with a copy of the original transfer function that is set to zero opacity except for one entropy level (see Figure 21(a)). We can also inflate an inner volume in the widget to be displayed in the same transfer function as the volume covered by the iris tool, to see the structure of the single entropy level more clearly (see Figure 21(b)). Changing the size of the inflated volume isolates the single entropy level, giving the user insight into where this entropy level is located in the data as well as its relationships to other entropy levels.

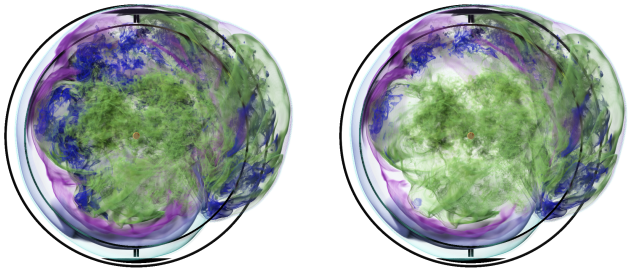
6 User Feedback

To gain feedback on the utility and usability of our approach, we conducted a qualitative study with 9 participants with ages ranging from 26 to 52 years and various backgrounds, including paleontology, scientific illustration, biology, and museum curation. All participants received a quick introduction to the basic concept of smart surrogate widgets and our prototype. They could then freely experiment with the tool and ask further questions if needed. Afterwards, they performed a task of recreating an illustrative visualization of the cochlea (see Figure 16) and evaluated the automatic fitting of the widget against manual fitting. We then performed a semi-structured



a b

Figure 20: (a) Direct volume visualization of a supernova simulation entropy field. (b) Automatically placed smart surrogate widget.



a b

Figure 21: (a) The smart surrogate widget is opened to remove outer layers of the data. (b) The inner volume of the widget is displayed with only a single entropy level.

interview with questions about the usability of our tool and general impressions of the approach.

All participants with at least some experience in 3D software were able were able to recreate the cochlea illustration to their satisfaction, with completion times ranging from 3:37 to 13:42 minutes (7:12 minutes on average). However, two participants, who had no prior experience with 3D software at all, were overwhelmed by the combination of possibilities of the individual widgets tools, and commented that they would need to spend more time with the system in order to achieve the desired result. All participants stated they would consider to use the tool in their work or for educational purposes.

The second task was to evaluate the automatic widget fitting against manual fitting with position manipulation through sliders. The participants were asked to rate to the quality of the fitting on a scale from 1 (”bad fitting”) to 5 (”according to my intentions”). Seven out of nine participants rated the automatic fitting with 5 and two rated it as 4. Two participants explicitly stated they were surprised how well the placement of widgets matched their intentions. Interestingly, only five out of nine participants rated the manual fitting with 5. The remaining four participants rated it as 4, 3, 2 and 1, and stated that they perceived the manual fitting as a very difficult task. To complete the manual fitting, the participants required from 2:03 to 3:17 minutes with an average of 2:13 minutes.

After the participants finished the tasks, they were asked to give qualitative feedback on the concept and our prototype. When asked if the participants would consider using smart surrogate widgets for their work one participant answered: ”Yes. This a very useful for identifying hidden or rather tricky morphological features. It is a good additional tool to the cutting plane. Cutting planes can be useful for larger scale structures. This tool would be more useful for intricate structures such as the skull or smaller bone fragments”. A professional illustrator stated: ”Yes, I would use it at two stages of illustration: First - for research, so that I understand the shape, anatomy and functions of the object. Second - as a template for the illustration, so that I can get the right 3D shapes, angles and textures”. Two participants that are specialized in multimedia exhibitions for museums stated that the smart surrogate widgets would have ”high applicability” as an educational tool for museums ”in a simplified version for children. For instance only with the wedge functionality”.

When asked how intuitive the participants perceived the interaction with the widgets, six rated the interaction as very intuitive, stating ”The interaction is very intuitive. The widget behaves as I intend it to and has a logical structure” or ”The interaction is very intuitive. It is nice that you can start illustrating without thinking of where to put the sphere”. One participant rated the intuitiveness as medium, stating ”Activating is the widget is not intuitive, perhaps a set of buttons would be better. A button each for the action you will perform: wedge, iris, move, and scale. However, when actually using the tool, it is easier when you can toggle on and off the widgets in the interface”. Two participants rated the interaction as ”not very intuitive” and ”not intuitive”, suggesting that a short animation of the functionality when hovering over the widget parts would improve the intuitiveness. With respect to the functional completeness of the smart surrogate widgets, most participants agreed that the functionality is sufficient. One experienced participant stated that he would like to be able to change the transfer function for each sphere separately. Another participant wished for an undo or reset button for the widget tools, so that he could easily close the whole volume again.

One negative comment noted by several participants was that they felt that the interaction handles were slightly too small, making it somewhat difficult to hit them when interacting quickly. Furthermore, many participants commented that they also consider the interface to be well-suited for touch-based interaction. While the feedback was gathered using conventional mouse interaction, our system actually supports touch interfaces as well. While this was not part of the task description, we noticed that some participants used the smart surrogate widgets as an exploration tool similar to a magic lens. With only one sphere, the widget was opened with the wedge tool and moved around in the volume to study the inner structure.

Overall, the feedback was very encouraging and we believe that our approach can provide a simple yet powerful tool for interacting with volumetric data. We are currently exploring the possibility of creating an interactive museum installation based on our prototype, and hope to use this opportunity to gather further data from a much larger user group.

7 Discussion

In our experiments, we found out that smart surrogate widgets provide an intuitive and fast method for the creation of expressive visualizations as well as data exploration. We deliberately chose a minimalist design, avoiding the introduction of additional clutter or occlusion, and only provide a limited set of interactions. Nonetheless, we found that the simple tools presented in this paper can be efficiently combined to quickly perform complex operations that would be difficult to achieve using conventional approaches. While our widgets are composed of two simple primitive shapes (spheres and tapered cylinders), they adapt well to a variety of spatial structures. In fact, initially we attempted to incorporate a larger set of basic widget elements, such as ellipsoids or rectangular prisms. However the resulting interface was significantly more complex without a noticeable increase in functionality.

Our approach can be easily integrated into existing visualization pipelines, and only introduces a small overhead. While the sphere

fitting computation can take several hundred milliseconds, it is an infrequent background operation that does not negatively affect overall usability. As widget construction is based on visibility, our approach is not restricted to any particular type of data or transfer function model. Moreover, even though in this paper we only considered scalar data sets, we also see great potential for using our widgets in the context of multimodal data. For instance, the focus region of a widget could be used to display a second modality instead of using an alternate transfer function.

8 Conclusion

In this paper we presented the new interaction concept of smart surrogate widgets for direct volume manipulation. Inspired by the notion of interaction through surrogate objects, we designed a visibility-driven approach for incrementally building a complex volume-covering widget consisting of only two primitive parts. We discussed our design on several examples and showed how it facilitates quick and easy volume manipulation. To evaluate our approach we conducted a qualitative user study. Our results show that smart surrogate widgets represent a promising and powerful approach suitable for non-experts in various scenarios.

Acknowledgements

The research presented in this paper was supported by the MetaVis project (#250133) funded by the Research Council of Norway.

References

[1] A. Paulina-Carabajal, Y-N. Lee, and L.L. Jacobs. Pawpawsaurus campbelli. `http://digimorph.org/specimens/Pawpawsaurus_campbelli/`. Accessed: 2017-09-21.

[2] Å. Birkeland, S. Bruckner, A. Brambilla, and I. Viola. Illustrative membrane clipping. *Computer Graphics Forum*, 31(3):905–914, 2012.

[3] Å. Birkeland and I. Viola. View-dependent peel-away visualization for volumetric data. In *Proc. Spring Conference on Computer Graphics*, pp. 133–139, 2009.

[4] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569, 2006.

[5] S. Bruckner and M. E. Gröller. VolumeShop: An interactive system for direct volume illustration. In *Proc. Visualization*, pp. 671–678. IEEE, 2005.

[6] S. Bruckner and M. E. Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 2006.

[7] N. Elmqvist and P. Tsigas. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1095–1109, 2008.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[9] N. Gagvani, D. Kenchammana-Hosekote, and D. Silver. Volume animation using the skeleton tree. In *Proc. IEEE Symposium on Volume Visualization*, pp. 47–53, 1998.

[10] M. Gerl, P. Rautek, T. Isenberg, and E. Gröller. Semantics by analogy for illustrative volume visualization. *Computers & Graphics*, 36(3):201 – 213, 2012.

[11] H. Guo, N. Mao, and X. Yuan. WYSIWYG (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2106–2114, 2011.

[12] H. Hauser, L. Mroz, G. I. Bischi, and M. E. Gröller. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252, 2001.

[13] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proc. IEEE VIS*, pp. 255–262, 2001.

[14] B. c. Kwon, W. Javed, N. Elmqvist, and J. S. Yi. Direct manipulation through surrogate objects. In *Proc. ACM CHI*, pp. 627–636, 2011.

[15] K. Lawonn, S. Glaer, A. Vilanova, B. Preim, and T. Isenberg. Occlusion-free blood flow animation with wall thickness visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):728–737, 2016.

[16] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin. Interactive cutaway illustrations of complex 3D models. *ACM Transactions on Graphics*, 26(3):31, 2007.

[17] M. J. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proc. IEEE VIS*, pp. 53–61, 2003.

[18] M. L. Muzic, P. Mindek, J. Sorger, L. Autin, D. Goodsell, and I. Viola. Visibility equalizer: Cutaway visualization of mesoscopic biological models. *Computer Graphics Forum*, 35(3):161–170, 2016.

[19] S. Owada, F. Nielsen, and T. Igarashi. Volume catcher. In *Proc. Symposium on Interactive 3D Graphics and Games*, pp. 111–116, 2005.

[20] S. Owada, F. Nielsen, T. Igarashi, R. Haraguchi, and K. Nakazawa. Projection plane processing for sketch-based volume segmentation. In *Proc. Symposium on Biomedical Imaging*, pp. 117–120, 2008.

[21] D. Patel, V. Šoltészová, J. M. Nordbotten, and S. Bruckner. Instant convolution shadows for volumetric detail mapping. *ACM Transactions on Graphics*, 32(5):154:1–154:18, 2013.

[22] T. Ropinski, J. Prassni, F. Steinicke, and K. Hinrichs. Stroke-based transfer function design. In *Proc. Eurographics/IEEE VGTC Conference on Point-Based Graphics*, pp. 41–48, 2008.

[23] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Proc. IEEE International Conference on Robotics and Automation*, 2011.

[24] B. Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, 1983.

[25] S. Stoppel, H.-C. Hege, and A. Wiebel. Visibility-Driven Depth Determination of Surface Patches in Direct Volume Rendering. In *Proc. EuroVis Short Papers*, pp. 97–101, 2014.

[26] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann. Interactive lenses for visualization: An extended survey. *Computer Graphics Forum*, pp. 173–200, 2016.

[27] I. Viola, M. Feixas, M. Sbert, and M. E. Gröller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):933–940, 2006.

[28] I. Viola and M. E. Gröller. Smart visibility in visualization. In *Proc. Eurographics Workshop on Computational Aesthetics*, pp. 209–216, 2005.

[29] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, 2005.

[30] P. Šereda, A. Vilanova, and F. A. Gerritsen. Automating transfer function design for volume rendering using hierarchical clustering of material boundaries. In *Proc. EuroVis*, pp. 243–250, 2006.

[31] 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.

[32] A. Wiebel, P. Preis, F. M. Vos, and H.-C. Hege. 3D strokes on visible structures in direct volume rendering. In *Proc. EuroVis Short Papers*, pp. 91–95, 2013.

[33] A. Wiebel, F. M. Vos, D. Foerster, and H. C. Hege. WYSIWYP: What you see is what you pick. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2236–2244, 2012.

[34] A. Ynnerman, T. Rydell, A. Persson, A. Ernvik, C. Forsell, P. Ljung, and C. Lundström. Multi-Touch Table System for Medical Visualization. In *Proc. Eurographics*, pp. 1775–1784, 2015.

[35] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg. CAST: Effective and Efficient User Interaction for Context-Aware Selection in 3D Particle Clouds. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):886–895, 2016.

[36] L. Yu, P. Svetachov, P. Isenberg, M. H. Everts, and T. Isenberg. FI3D: Direct-touch interaction for the exploration of 3d scientific visualization spaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1613–1622, 2010.