Fast Contextual View Generation in 3D Medical Images Using a 3D Widget User Interface and Super-Ellipsoids

Ken Lagos and Tim McInerney(✉)

Department of Computer Science, Ryerson University,
Toronto, ON M5B 2K3, Canada
tmcinern@ryerson.ca

Abstract. This paper presents a 3D widget user interface (UI), super-ellipsoid shape primitives and a customized volume rendering algorithm that together create an effective system for generating contextual views in 3D medical images. The widget UI supports the fast and precise positioning of a super-ellipsoid "paint blob". The paint blob can be deposited and automatically blended with previously deposited blobs to form an arbitrarily complex-shaped region of interest (ROI) enclosing target image features. The rendering of these "focus" regions can be controlled separately from the surrounding contextual region, allowing medical experts to examine and measure image features relative to the surrounding structures, regardless of the level of occlusion. The system's core algorithms execute in parallel on graphics processing units, resulting in real-time interaction and high-quality visualizations. The focus plus context visualization system is validated via a user study and a series of experiments.

Keywords: Visualization · 3D medical images · User interface

1 Introduction

Direct Volume Rendering (DVR) controlled by Transfer Functions (TFs) is a standard algorithm for generating visualizations in 3D medical images, especially for CT scans. Despite the extensive amount of research in TF design [1], quickly defining a single TF that generates a desired contextual view of a spatially localized focus region or target anatomical structure remains a non-trivial task. This may be due to the fact that TF specification is primarily a mapping of ranges/characteristics of voxel intensity values to color and opacity. Additional mechanisms (e.g. 2D TFs) provide more control to the user over the visualization output but often at the cost of more complex user interfaces that are still primarily indirect in nature. Fast, simple *direct* selection of a spatial sub-volume can complement a TF specification process by supporting separate and more easily defined TFs for this "focus" region and for the context region surrounding it.

However, designing interactive techniques enabling a user to quickly generate and easily modify a contextual view by directly selecting a focus ROI or selecting a specific target structure in a volume rendered image is a challenging human-computer interaction (HCI) task. Such a technique needs to augment TF specification by supporting the selection of objects adjacent to other objects that have been mapped by the TF to similar opacity and color values. Furthermore, desired ROIs may vary from regularly shaped regions, such as rectangular blocks or spheres, to arbitrarily complex, curving shaped objects/features, or to elaborate branching objects such as vasculature. In addition, the TF specification may result in a volume rendering in which a target ROI contains several visually disconnected objects or an ROI that is visually occluded by parts of other structures. Another important requirement of interactive ROI selection techniques to support the fast modification of the ROI as a user is exploring the volume from new viewpoints. Finally, the interaction issues are inherently tied to issues of the user's depth perception of the target ROI as well as the design of appropriate supporting visual cues, including interface signifiers to indicate affordance and help bridge the gulf of execution of the UI during ROI selection.

This paper presents an interactive view generation and ROI selection technique that is based on the manipulation and rendering of mathematically-compact and blend-able convex shape primitives known as super-ellipsoids (Fig. 1). Specifically, a user interface (UI) design that uses a mouse and 3D widgets for manipulating the super-ellipsoids is presented[1]. The main hypothesis is that the combination of a mouse, the 3D widget UI, blend-able super-ellipsoids and a standard TF interface is an efficient and flexible technique for quickly creating contextual views of spatially localized regions. Super-ellipsoids, referred to as "blobs", are defined using implicit functions and can be seamlessly and tightly blended to form volumetric "paint" that envelopes regions and image features in the volume image. Inside a selected ROI defined by the blobs, a special TF may be applied that controls color and opacity separately from the surrounding structures. Note that this work does not compare higher degree of freedom (DOF) input devices to our mouse-widget approach for 3D volume view generation. In our experience, for desktop systems, higher DOF input devices suffer from precision, muscle fatigue, comfort and occlusion problems.

This paper presents work that improves and significantly expands on previous work [2]. In [2] a more restrictive spherical paint blob and blob manipulation UI that utilized a (data iso-) surface painting metaphor (referred to in this paper as a *surface brush*) was employed. In this new work we instead use 3D widget handles to manipulate a super-ellipsoid blob, referred to as the *blob tool* (Fig. 2). The widget handles act as intuitive signifiers that allow users (from any scene viewpoint) to flexibly and accurately position, orient, resize and deposit blobs that envelope a target ROI anywhere in the volume. Another major contribution is the addition of a *blob region-grow tool*. The same widget UI and blob provides an intuitive, convenient mechanism for interactively constraining and steering a GPU region growing segmentation algorithm in real time. These combined capabilities, unified under a single widget UI, support the highly-efficient selection and modification of complex-shaped volumetric regions, individual objects, parts of an object, or objects with complex geometry and topology - such as artery networks - allowing for fast and flexible volume exploration in 3D images ranging from relatively clean CT scans to moderately noisy images such as MR scans.

[1] Several videos of the tools comprising the technique can be viewed at https://www.youtube.com/watch?v=NsgipyU1lfQ&list=PLtEweT4itM8Ef2D9HLQDAKu6G1Ou1Q3Yr&index=1.

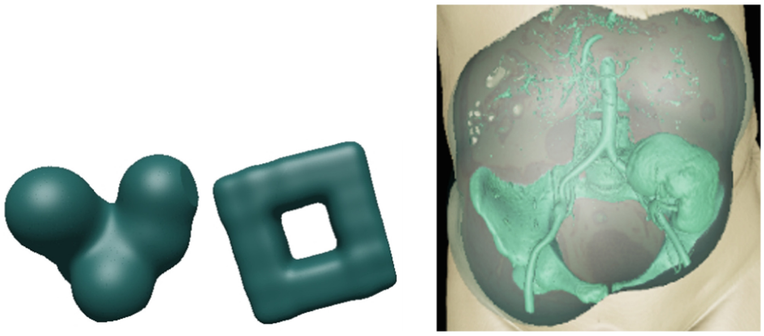
Fig. 1. Left – examples of super-ellipsoid blob blending. Right – blended blobs defining a region of interest. The interior region has separate TF controls from the exterior.

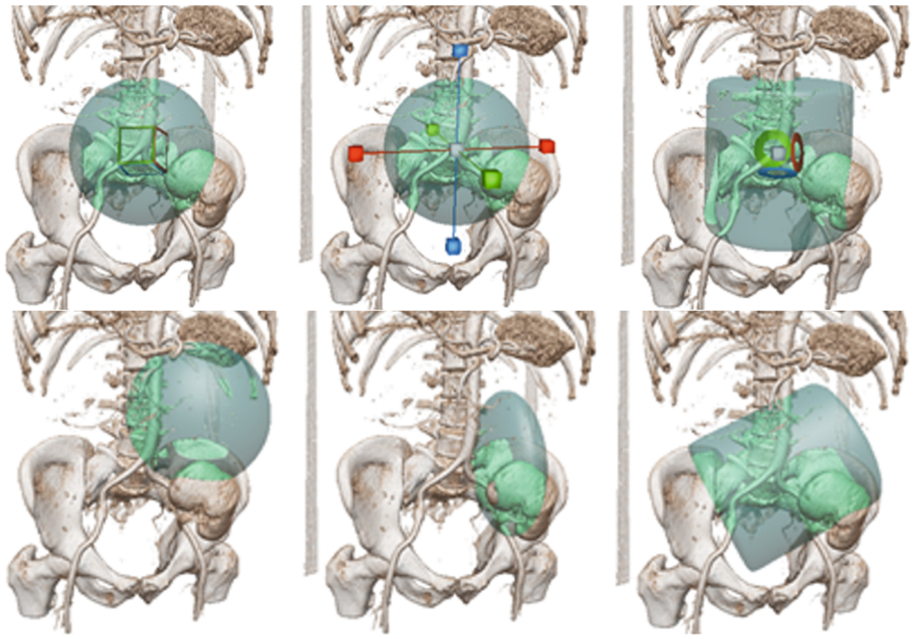
Fig. 2. Super-ellipsoid blob tool and its widget handles. Upper row, left to right: translation handle, resize handle, and rotation handle. Lower: effect of widget actions.

In addition, a new *open-view* capability can be activated for both tools to support real-time examination and measurement of occluded structures with respect to surrounding structures. Finally, the results of a user study comparing the 3D widget-based blob tool UI, the previously developed surface brush style UI and a standard cylinder "screen-space paintbrush" UI for defining a ROI are presented. Several experiments demonstrating the use of our visualization technique for creating different views are also performed in order to further validate the effectiveness of the technique. Note that this paper will primarily focus on the UI and system functionality. Details of the blob and blob blending mathematical formulation, GPU shader algorithms and data structures, as well as tables detailing the user study results can be found in [14].

2 Related Work

A large amount of research has been carried out to create UI's and TF's that enable medical professionals to generate insightful volume renderings of complex volume images. In general, many of the 1D TF's and 2D TF's developed do not utilize global or local spatial information. However, how much spatial information is needed is perhaps task dependent [1]. Focus plus context visualizations attempt to visually combine a user-selected local spatial region of primary interest (the focus) with the surrounding information—or context—into a single display. One of the most common focus plus context techniques is to make occluding objects semi-transparent thereby revealing hidden objects [3]. For un-segmented volume data, the opacity of a sample point along a ray can be controlled by a function of shading intensity, gradient magnitude, distance to the viewer, and previously accumulated opacity [4]. In general, the use of transparency is limited – it does not provide a strong visual cue of the depth of the hidden objects and it can be visually confusing if there are multiple overlapping semi-transparent layers. The use of a distance function as part of the overall TF specification does use spatial information [5]. However, the distance-functions are typically radial in nature and do necessarily provide much local and shaped spatial control. Several researchers have developed techniques to generate focus plus context views where the focus region is more explicitly defined with a convex-shaped 3D "lens" [2, 3, 6–9]. The lens geometry is often a cylinder, sphere or cone or some other compactly defined mathematical function. Similar to some of the work presented in this paper, Bruckner and Gröller [10] uses a 3D volumetric painting approach with a 3D Gaussian lens. Lenses realized with super-ellipsoids have also been used by other researchers, albeit often in more restricted ways than in our work. Luo et al. [8] uses a super-ellipsoid distance function to define their focal probe region. Within this region, a different rendering style may be used than in the surrounding context. Radeva et al. [11] also uses a super-ellipsoid "lens". In our work the lens can also act as a tool tip to deposit multiple super-ellipsoid paint blobs that are smoothly blended to define a complex–shaped paint region. Another issue with a convex lens approach is the target region/object may be occluded (i.e. hidden) by other objects. Luo et al. [8] incorporates a view-dependent cone region to cutaway occluding voxels in front of the focal probe. We use a similar approach but the shape of the auxiliary lens is not restricted to a cone but rather is configurable and is designed to provide depth cues by orienting the cut surfaces of the occluding objects toward the viewer such that the relative position and depth of an object is easier to perceive (Sect. 3.3).

2.1 Selecting a Complex-Shaped ROI as a Focus Region

Interactive selection of *complex-shaped* 3D ROI selection techniques can be categorized in several ways. One categorization is data dependent techniques versus data independent techniques. A data independent technique specifies a ROI spatially (i.e. geometrically). A data dependent technique uses data attributes, such as voxel intensity or voxel gradient magnitude, to label voxels as part of the same structure. Data independent techniques are purely geometric and are primarily based on user interactive spatial specification of a ROI. The advantage of these techniques is that they can be applied to any volume, regardless of noise. However, they place emphasis on the user to navigate to an ROI and define an envelope and their effectiveness hinges upon the UI design. Techniques for interactive geometric specification of a ROI can also be categorized based on the underlying interaction metaphor used. Tracing, painting, sculpting, and deforming are among the most common metaphors used [2].

Data dependent selection (i.e. segmentation) techniques are a heavily researched field and advanced segmentation algorithms are often required to select structures in noisy images. Nevertheless, simpler highly-parallel GPU-based segmentation algorithms, such as voxel region growing [12, 13], can be used with great effect in volume rendered images such as CT and MR scans to quickly define a ROI and generate a contextual view for previewing and examining of a target region. These techniques may be sufficiently accurate to perform a visual analysis and measurement of hidden structures by supporting fast selection and removal of occluding structures. Region growing algorithms are susceptible to noise and are prone to leaking into neighboring structures and therefore benefit from interactive control of the overall growing/shrinking process. Some implementations allow for rough interactive control over the *localized* growing/shrinking of the selected voxels [12]. The volume visualization package LiveVolume [13] supports fast GPU region growing to select complex connected voxel regions. The UI is simple and effective – an initial point is selected, and the user uses the mouse to control the growing/shrinking of this region in 3D by moving the cursor away from, or towards this initial point. Chen et al. [12] support fast GPU region growing for segmentation using a UI where the user sketches curves on the screen to initiate the region growing. Localized region shrinking is supported by sketching additional curves to indicate regions to stop and reverse growth.

3 Implementation

This section provides an overview of the implementation and interface of the system. The super-ellipsoid blobs are defined using implicit functions and the functions are evaluated at each voxel location in a special 3D grid of voxels referred to as the *blob grid*. The blob grid has dimensions matching that of the input volume. The implementation allows users to place a large number of blobs, if desired, in real time. To ensure quick lookups into the grid, the blob grid is stored in GPU memory. A disadvantage of our implementation is that it requires a large amount of GPU memory but results in overall higher performance by avoiding expensive calculations in the volume ray cast shader program. Specifically, this gain in performance is achieved by computing the blob grid values in a separate rendering phase using a special, highly-efficient GPU vertex shader program. As the user deposits a new blob, it is blended with the existing blob grid field values computed from previously deposited blobs - there is no need to re-compute the grid field values for all blobs. Consequently, during the volume rendering phase, at each step along the ray these grid values can then be efficiently accessed from the volume rendering fragment shader program. In addition to the blob grid, two other grids are required to support the system functionality. The dimensions of both of these grids also match the input volume grid dimensions. A *region grow grid* is used to record voxels selected by the region grow tool. The *accumulation grid* is used to combine the voxel selections of the individual blob tool or region grow tool selections. The volume rendering algorithm blends the values from the input volume and the various grids to form the final image.

3.1 Blob Tool

Widget handles are designed to suggest their function (i.e. translate, rotate, scale) and operation as well as provide an additional visual depth cue for the blob (Fig. 2). Handles appear via a key press. When a handle is selected, it is then hidden to allow the user to clearly see the effects of their mouse movements on the tool tip blob. When the user releases the left mouse button, the handles reappear. GUI sliders can be used to change the shape and opacity of the blob. As the blob is manipulated, voxels inside are highlighted to provide visual feedback of the selection region. If a user is satisfied with the selection, they can deposit the blob (i.e. blend it with any previously applied paint blobs) via a key or GUI button. The user can instantly undo applied blobs in the order of last applied to first.

3.2 Blob Region-Grow Tool

As mentioned, a GPU-based region grow algorithm is a fast segmentation technique for selecting a complex-shaped connected structures. However, region grow algorithms are noise sensitive and are prone to “leaking” into neighboring regions with similar voxel intensity ranges to that of the target. Therefore, interactively controlling the region grow process prevents a user from wasting time manually correcting the region grow output by removing the leaked regions. In addition, the ability to interactively update the region growing allows users to generate new views “on the fly”.

In our work, we use the widget UI to interactively constrain and “steer” 3D region growing. This *blob region-grow tool* (Fig. 3) can be applied directly to the volume image to select connected voxels, for example an artery “tree”, or alternatively can be applied to connected voxels within a previous ROI selected by the blob tool - allowing the user to select ROIs even within a moderately noisy volume image.

The constrained region grow is implemented on the GPU following the algorithm in [12]. The user initiates region growing by first using the translation widget handles to place the tool tip blob around an anatomical structure they wish to select (Fig. 3 left). The user then moves the mouse along the surface of this structure and a small green circular region is highlighted, indicating a valid seed voxel is within the bounds of the blob. If the circular region is colored red, this indicates the seed voxel is outside the blob.

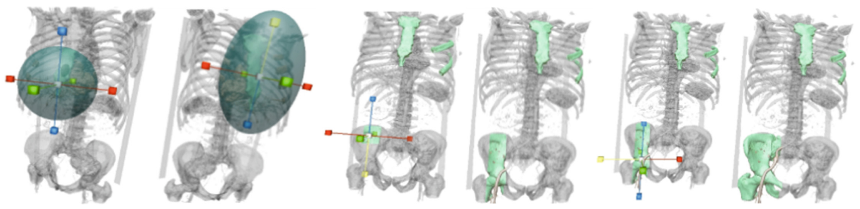

Fig. 3. Examples showing the blob region-grow tool selecting connected structures. 1, 2: the region growing algorithm is spatially-constrained by the super-ellipsoid blob. 3–6: The grown region can be dynamically resized/reshaped in real-time by using the widget handles to adjust the blob size/shape (rectangular blob not rendered). (Color figure online)

A valid seed point is then selected with a mouse click. The region grow process is then activated and all valid connected voxels within the blob boundary are selected, essentially instantaneously. Valid voxels are voxels that are visible in the scene and not already selected. The blob tip can then be interactively resized, reshaped or rotated using the widget handles and the selected region will automatically grow and/or shrink, in real-time, to stay within the blob (Fig. 3, 3–6). This includes one-sided control over resizing using one of the six resize handles. Other target regions can be dynamically added by reinitiating the region grow tool and selecting new seed voxels. The user can also use a key or GUI button to instantly undo any selected ROI. By having the region growing controlled by the same 3D widget UI as the blob tool, the user can precisely select regions of connected voxels that might be otherwise difficult and/or time-consuming to isolate using only a pure geometric blob tool - the tools are complementary.

3.3 Open-View

In situations where the 3D blob tool itself is occluded or the target object is occluded, the user may activate an *Open-View* mode to automatically remove the occluding objects without losing the surrounding context. The open-view capability is implemented with an (invisible) auxiliary "lens" with one lens endpoint fixed to the viewpoint and the other endpoint attached to the blob tool tip (Fig. 4). As the user manipulates the blob tool or rotates the scene, the open-view lens automatically cuts away any occluding visible voxels between the viewpoint and the tool tip blob. The shape of the Open-View lens is defined by a shaft region and an adjustable super-ellipsoid cap (Fig. 5). The cap can be translated, scaled, rotated and tapered using a similar widget interface as the blob tool. GUI sliders are also available to adjust the shape of the cap – which in turn automatically adjusts the shape of the shaft to match. A separate rounded or tapered cap creates an effective perceptual depth cue of the cutaway region by orienting the cut surfaces towards the viewer. Combining the open-view capability with the widget blob tool UI allows the user to quickly create specialized view dependent cutaway regions such that the contextual view is enhanced with effective depth cues provided by the surface of the cutaway region (Fig. 4).

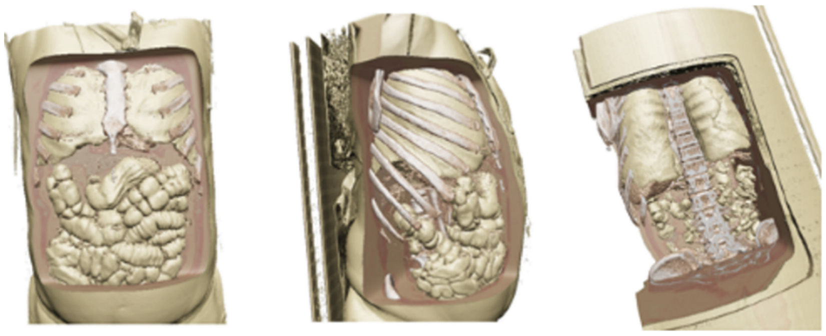

Fig. 4. Using *open-view* to create a cutaway region to reveal a user's earlier blob tool selection of the lungs and intestines. These structures can then be viewed relative to skin and bone.

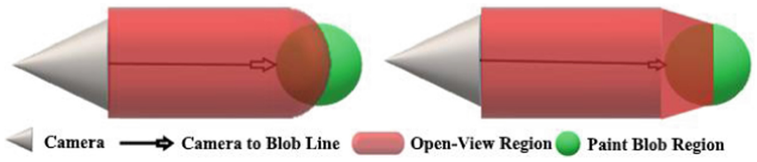

Fig. 5. An open-view lens" representation for spherical and tapered cubical cap shapes. The cap shapes are designed to orient cut surfaces toward the user, regardless of the length of the shaft.

4 Results and Validation

A user study was performed in an effort to quantitatively evaluate the UI efficiency and accuracy, as well as qualitatively evaluate the ease of use, flexibility, and rendering control of the super-ellipsoid based technique and its associated widget UI for selecting and visualizing a wide range of target ROIs and anatomical structures.

4.1 User Study Results

The user study was divided into four parts. Each part required participants to select a previously highlighted target object. Sixteen people participated in the study. The participants were asked to select several regions of interest with four different paint-based selection tools and then fill out a questionnaire. Efficiency, accuracy and user tool preference were recorded (See [14] for details of measurements). Participants consisted of 14 males and 2 females aged between 18–30 years old. Participant's average mouse usage per week was approximately 28 h, with an average of 11 h a week for video games and 2.4 h for 3D modeling software. The user study experiments were performed on a Windows 10 desktop computer with a mouse and keyboard. The computer was equipped with Nvidia 1080 GTX graphic card to ensure the system ran at a smooth 60 fps on a 1920 × 1080 native resolution monitor. The four paint tools used in the study were the blob tool, region-grow blob tool, a surface brush and a screen brush.

The UI of the surface brush and UI of the screen brush are briefly described below. Although the surface brush was implemented in a separate system, the system was updated to use matching super-ellipsoid blobs and matching volume rendering parameters. The surface brush employs an adjacent GUI panel to control brush size, orientation, and shape. The surface brush has two painting modes, a camera view-plane sliding mode and data iso-surface sliding mode. In both cases the blob orientation is automatically defined from the plane/iso-surface orientation. As the user moves the mouse the brush automatically adheres to and slides along the plane/surface. The screen brush is a screen space painting technique similar to 2D painting applications. A circular outline depicts the brush bounds and the brush is resized using a separate GUI slider. Users can paint on a volume rendered image by dragging the mouse along the screen to paint strokes. The brush also paints in depth and selects voxels based on the camera's perspective projection. The depth is set to the far side of the volume. An eraser is also supported. Therefore, to select a target object, users rotate the scene with the mouse to find a screen view such that the target object is visually separated from the surrounding objects. After painting and erasing the object from this viewpoint, the users then repeat this process from new viewpoints until the object has been selected.

The study used two CT scans and was comprised of four different target anatomical structures (Fig. 6). Each structure was specifically selected to tease out the strengths and weaknesses of each UI. In the hands of an experienced user, the selection tools used in the study can be used to quickly isolate a complex-shaped ROI consisting of multiple anatomical structures. However, in this study the participants are naïve users. Consequently, an ROI consisting of single, clearly defined anatomical structures were chosen as the targets, highlighted in yellow and with the remaining visible voxels de-emphasized by coloring them gray (Fig. 6). If the user correctly selects these voxels, they are instantly painted green. Incorrectly selected voxels (i.e. voxels inside the blob boundary but outside the target) were instantly painted red.

Both efficiency and accuracy were measured. The purely geometric tools (blob tool, surface brush, screen brush) were first compared to each other. In terms of efficiency almost all selection tasks were completed in well under two minutes. When comparing the tools to each other the results were mixed, with the screen brush on average the most efficient. However, the blob tool was consistently the most accurate selection technique for all trials. The widget region-grow tool was then introduced and compared to the other tools. As this technique selects connected voxels within a preset intensity range, it is slightly less accurate than the geometric tools. However, it is easily the most efficient technique for selecting a single connected structure, like those in the study. Details of the efficiency and accuracy results can be found in [14].

After the trials for each tool were completed, participants were asked to fill out a questionnaire to evaluate their level of satisfaction with the tool using a 7-point Likert scale. Users were also asked to pick their favorite tool before and after the introduction of the region grow tool. Before the introduction of the region-grow tool participants generally favored the screen brush as it provided a simple and familiar interface for the selection tasks. The blob tool was the second favorite as users with gaming or 3D modeling experience were familiar with the handle-based controls. Participants

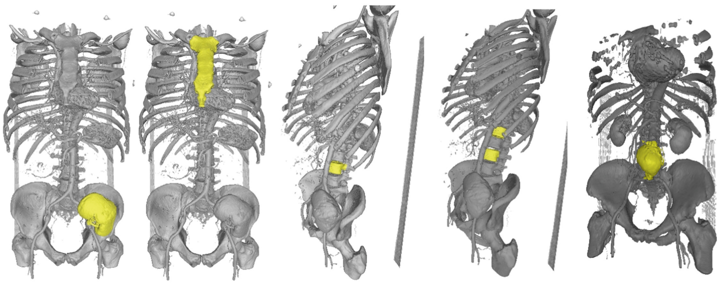

Fig. 6. A series of anatomical structures used in the user study. From left to right: kidney, sternum, single vertebra, double vertebra and aneurysm. (Color figure online)

overwhelmingly favored the blob region grow tool once it was introduced. The intuitive widget interface combined with the region-grow functionality allowed users to quickly and accurately select a single target object. As mentioned, because the participants were naïve users, single-object targets were used in the study for clarity and simplicity. This experimental setup may have introduced some bias towards a tool specifically designed for this task. The least favorite technique was the surface brush, most likely due to having the brush size and shape controls on a separate GUI panel, which several participants found frustrating for some target object selection tasks. This result was not unexpected. Although the surface brush is a more direct manipulation technique for positioning and automatic orientation, the cost of this capability is the difficulty of integrating blob resizing, depth control and blob orientation fine tuning into the interface.

4.2 Experiments

Additional experiments were performed by the authors to demonstrate the capabilities and flexibility of the system and UI for creating contextual, spatially localized views of a wide range of target structures. All experiments were performed in less than 5 min and a brief description of the selection process is provided for each.

Aneurysm. The region of interest for this CT dataset (down-sampled to $256 \times 256 \times 144$) was the aneurysm and connecting arteries (Fig. 7). An aneurysm is an excessive localized enlargement of an artery cause by a weakening of the artery wall. An initial voxel visibility intensity range was set to display bones and arteries via the TF. A region-grow tool was then used to initially select the aneurysm itself. At this point, the aneurysm volume can be measured, if desired. The widget handles were then used to resize the super-ellipsoid blob to instantly grow the region into the connecting arteries and the selected voxels were added to the accumulation grid. The minimum voxel visibility range was then lowered to render surrounding skin, muscles, and bones. A rectangular open-view lens was activated to create a view that allows users to clearly see the aneurysm's position in relation to various surrounding structures. The open-view lens also allows the user to view the aneurysm from any angle and quickly gain additional insight.

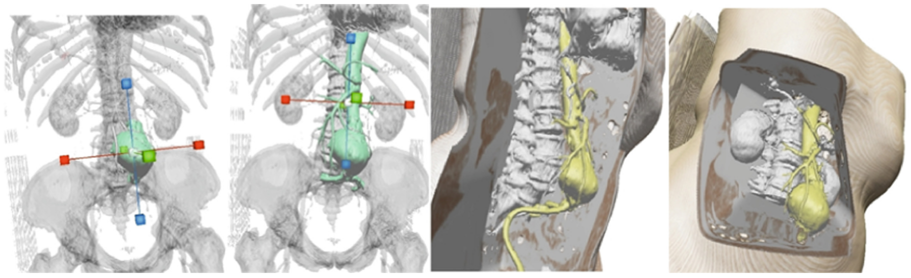

Fig. 7. Aneurysm selection experiment. From left to right: 1. Minimum and maximum visibility set to view aneurysm and surrounding structures. 2. Aneurysm selected with region-grow tool and tool adjusted to grow into connecting arteries. 3, 4. Selected voxels added to the accumulation grid and open-view lens activated to create various contextual views by changing the viewing angle and visibility settings.

Kidney. This experiment was conducted on a 256 × 256 × 296 (down-sampled) CT scan and showcases how a user can create various contextual views of a kidney transplant (Fig. 8). Initially, the hip bone was selected using the blob tool and the selected voxels were added to the accumulation grid. The kidney and connecting arteries and vertebrae were then selected using a region-grow tool and were also added to the accumulation grid. A cubical blob tool was used and interactively adjusted (along with TF visibility settings) to create several contextual views. The kidney transplant position can be analyzed.

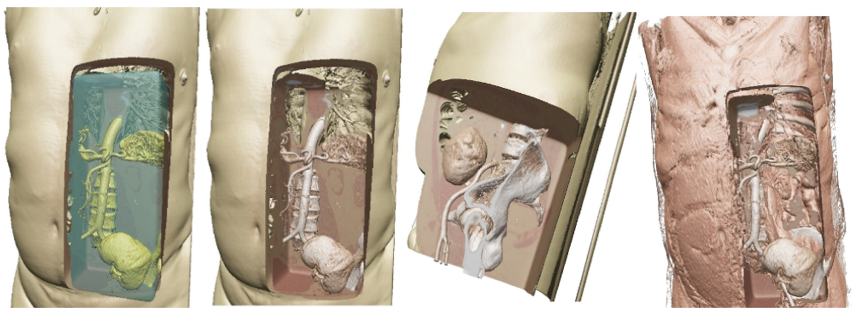

Fig. 8. Kidney transplant experiment. Initially a blob tool was used to select the hip bone. Left to right: 1. Region-grow tool used to select the connected arteries and the kidney. 2–4: Blob tool to create contextual views by changing the viewing angle and visibility settings.

Brain Tumor. This experiment uses a (down-sampled) 288 × 288 × 22 MRI scan and demonstrates how the system can be used with nosier volume images to create effective views (Fig. 9). In this case, the object of interest is a tumor inside the brain. The tumor has a very high intensity value. To select the tumor, the minimum voxel visibility range was set to a high value and the region grow tool was used to select and add the tumor to the accumulation grid. To create cross sections in the brain, a rectangular blob tool was used with its minimum voxel visibility range set to the maximum value. Since voxels within the accumulation grid are not affected by the visibility settings of the blob tool, it allows users to interactively create cross sections by translating, rotating, or resizing the tool. This allows users to quickly ascertain and measure the tumor's position relative to other objects within the dataset.

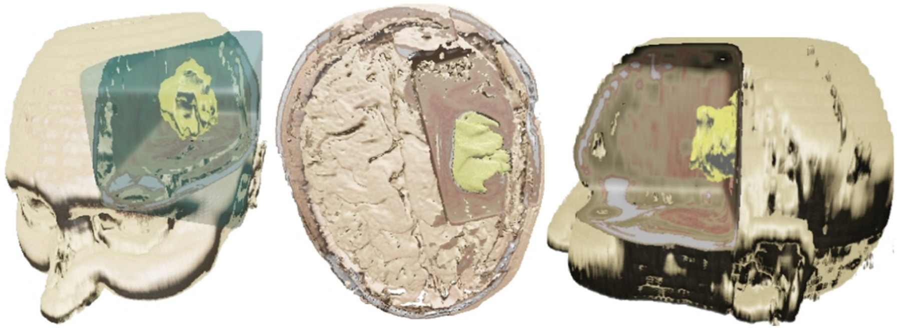

Fig. 9. MRI brain tumor selection experiment. Left: Set initial minimum and maximum visibility settings to view brain tumor without losing any detail. Select tumor with blob region grow tool and add to accumulation grid. Middle, Right: Use rectangular blob tool and set TF such that voxels are invisible. Tumor position and extent can now be measured from various angles.

Pulmonary Stent. This experiment was conducted on a 512 × 512 × 308 CT scan and demonstrates how the system can be used to select a stent and inspect its position within the artery. Starting from the top left of Fig. 10, the voxel visibility range was adjusted to view the lungs, but unfortunately, the stent wasn't clearly visible until the voxel intensity range was reset to a higher value. As the surrounding area was clean, the blob tool was used to select the stent and the selected voxels were added to the accumulation grid. The voxel visibility range was then lowered to view the stent's position around the heart and top of the lungs. To get a clearer view, a blob was placed around the area of the stent and all outside voxels were removed. This allowed the user to zoom in and create an enlarged view of the stent and its surrounding objects for inspection. Voxels of lower intensity were then brought back, and an open view lens was added to show cross-sectional views of the stent within the artery. These two techniques allow the user to inspect the stent's position relative to other objects and to potentially verify correct placement.

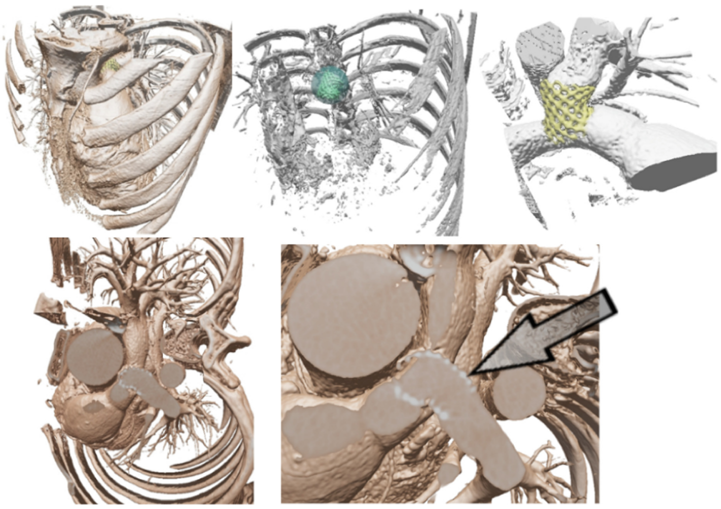

Fig. 10. Pulmonary stent experiment. From top left: 1. Create initial scene of stent and lung region using TF. 2, 3. Increase minimum visible voxel intensity via TF to isolate the stent and use the blob tool to select the stent and add it to accumulation grid. Bottom 4, 5. Add open-view lens to create contextual views. The arrow indicates the stent cross-section.

5 Conclusion

The goal of simply and efficiently exploring and selecting regions of any shape or level of occlusion, in order to examine and focus on specific structures and their spatial relationship to surrounding structures, can be difficult to achieve using only a TF approach. Fast and flexible interactive volume navigation and ROI selection tools can potentially augment TFs to realize this goal. However, the problem is shifted to one of designing an effective tool and UI. This paper presented one design and compared it to two other common selection tools with well-known UIs. Virtual surface paintbrush style UIs are well-known technique for easily selecting and removing outer region "layers" of voxels belonging to a specific tissue type, such as skin [2]. However, due to the nature of the painting style they mimic, they may not be optimal for enveloping (i.e. selecting for highlighting or removing) structures with varying thickness. They also cannot be easily used in noisy images. A 3D widget-based UI, on the other hand, is more "volume-oriented" and amenable to highly accurate envelopment but may require more user interaction to position and manipulate. However, unlike the surface brush it can be applied to noisy datasets. Finally, the standard screen-space paintbrush UI is familiar, highly intuitive and often more efficient for selecting some 3D objects. However, for fast modification of the ROI to support volume exploration and iterative generation of contextual views, the 3D widget UI is more flexible than both the surface and screen brush UIs. In addition, and more importantly, the widget UI provides a convenient and flexible control mechanism for steering a 3D region growing segmentation technique, adding powerful selection capabilities. The mixed results of the user study indicate that it may be beneficial to incorporate the best features of the surface and screen brush tools into the widget UI.

References

1. Ljung, P., Kruger, J.H., Groller, E., Hadwiger, M., Hansen, C.D., Ynnerman, A.: State of the art in transfer functions for direct volume rendering. Comput. Graph. Forum **35**, 669–691 (2016)
2. Faynshteyn, L., McInerney, T.: Context-preserving volumetric data set exploration using a 3D painting metaphor. In: Bebis, G., et al. (eds.) ISVC 2012. LNCS, vol. 7431, pp. 336–347. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33179-4_33
3. Kriger, J., Schneider, J., Westermann, R.: ClearView: an interactive context preserving hotspot visualization technique. IEEE Trans. Vis. Comput. Graph. **12**(5), 941–948 (2006)
4. Bruckner, S., Grimm, S., Kanitsar, A., Gröller, M.E.: Illustrative context-preserving exploration of volume data. IEEE Trans. Vis. Comput. Graph. **12**(6), 1559–1569 (2006)
5. Tappenbeck, A., Preim, B., Dicken, V.: Distance-based transfer function design: specification methods and applications. In: Simulation and Visualization (2006)
6. Zhou, J., Döring, A., Tönnies, K.: Distance based enhancement for focal region based volume rendering. In: Tolxdorff, T., Braun, J., Handels, H., Horsch, A., Meinzer, H.P. (eds.) Bildverarbeitung für die Medizin 2004. Springer, Berlin (2004). https://doi.org/10.1007/978-3-642-18536-6_41
7. Monclus, E., Dıaz, J., Navazo, I., Vazquez, P.P.: The virtual magic lantern: an interaction metaphor for enhanced medical data inspection. In: The 16th ACM Symposium on Virtual Reality Software and Technology, Kyoto, Japan (2009)
8. Luo, Y., Iglesias Guitián, J.A., Gobbetti, E., Marton, F.: Context preserving focal probes for exploration of volumetric medical datasets. In: Magnenat-Thalmann, N. (ed.) 3DPH 2009. LNCS, vol. 5903, pp. 187–198. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10470-1_16
9. Ropinski, T., Steinicke, F., Hinrichs, K.: Tentative results in focus-based medical volume visualization. In: Butz, A., Fisher, B., Krüger, A., Olivier, P. (eds.) SG 2005. LNCS, vol. 3638, pp. 218–221. Springer, Heidelberg (2005). https://doi.org/10.1007/11536482_19
10. Bruckner, S., Gröller, M.: Volumeshop: an interactive system for direct volume illustration. In: 16th IEEE Conference on Visualization (VIS 2005), Baltimore, MD (2005)
11. Radeva, N., Levy, L., Hahn, J.: Generalized temoral focus + context framework for improved medical data exploration. J. Digit. Imaging **27**, 207–219 (2014)
12. Chen, H., Samavati, F., Sousa, M.: GPU-based point radiation for interactive volume sculpting and segmentation. Vis. Comput. **24**(7), 689–698 (2008)
13. LiveVolume. www.livevolume.com. Accessed 01 June 2017
14. Lagos, K.: Fast contextual view generation and region of interest selection in 3D medical images via superellipsoid manipulation, blending and constrained region growing. Master's thesis, Department of Computer Science, Ryerson University, Toronto, ON, Canada (2019)