Context-Aware Volume Navigation

Stefan Diepenbrock* Timo Ropinski† Klaus Hinrichs‡

Visualization and Computer Graphics Research Group (VisCG), University of Münster

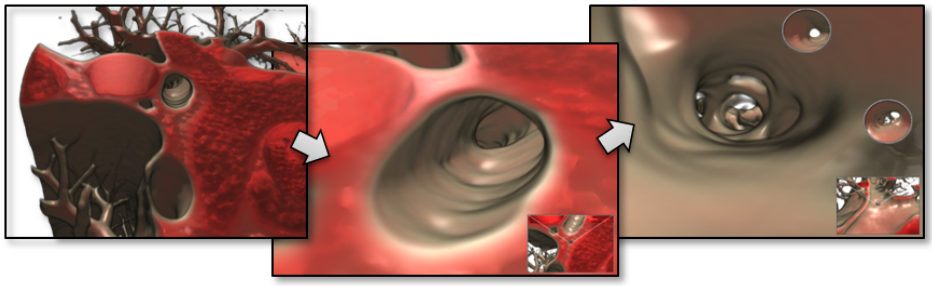

Figure 1: Three subsequent screenshots as made during the usage of our image-based volume navigation metaphor. Without changing the navigation mode, the user is able to inspect the data set in a behavior similar to a trackball and can also fly through internal structures when a collision detection is present. To support the spatial-awareness, appropriate thumbnails are displayed.

ABSTRACT

The trackball metaphor is exploited in many applications where volumetric data needs to be explored. Although it provides an intuitive way to inspect the overall structure of objects of interest, an in-detail inspection can be tedious - or when cavities occur even impossible. Therefore we propose a context-aware navigation technique for the exploration of volumetric data. While navigation techniques for polygonal data require information about the rendered geometry, this strategy is not sufficient in the area of volume rendering. Since rendering parameters, e.g., the transfer function, have a strong influence on the visualized structures, they also affect the features to be explored. To compensate for this effect we propose a novel image-based navigation approach for volumetric data. While being intuitive to use, the proposed technique allows the user to perform complex navigation tasks, in particular to get an overview as well as to perform an in-detail inspection without any navigation mode switches. The technique can be easily integrated into ray-casting based volume renderers, needs no extra data structures and is independent of the data set as well as the rendering parameters. We will discuss the underlying concepts, explain how to enable the navigation at interactive frame rates using OpenCL, and evaluate its usability as well as its performance.

Keywords: Navigation, Volume rendering.

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques;

1 INTRODUCTION

In recent years several algorithms have been proposed which accelerate volume rendering and thus allow interactive frame rates. As a consequence, the user cannot only change rendering parameters interactively, but is also able to navigate within volumetric data. Although navigation is essential in order to get a deeper understanding of the visualized data sets and the need for navigation metaphors has been expressed [8], only little research has been dedicated to support this process within volume visualizations. In fact, in most general-purpose volume visualization systems the trackball metaphor is exploited [25], since it is easy to use and allows predictable navigation [1]. However, the trackball metaphor has several drawbacks. First, navigation is based on the assumption that the shape of the object to be explored is (approximately) spherical, which for instance makes it hard to explore longitudinal structures. Second, in-detail inspections are difficult, since the trackball is fixed to a given center. To deal with this shortcoming, it is often possible to reposition the trackball center, which, however, is tedious and in many cases counterintuitive. Third, the trackball metaphor is not location-aware and does not take visibility into account. This is especially problematic when diving into the volume, where it is hard to orient oneself, in particular in the absence of collision detection. Therefore specialized navigation techniques have been developed for application cases where these drawbacks are limiting factors, e.g., when navigating in virtual colonoscopy [11]. In this paper, we propose a general purpose volume navigation metaphor, which can be integrated into existing volume visualization systems. Besides being intuitive to use, we believe that our navigation metaphor brings forward visualization-based research, since it does not have to be adapted for specific tasks and thus is also applicable to emerging application scenarios.

A navigation task can be understood as a combination of explorative navigation and directed navigation [8]. While explorative navigation is a rather undirected task, where the user interactively inspects the data to gather knowledge, directed navigation supports the user when intentionally visiting structures of interest. In most application scenarios, both navigation types are desirable, and they should therefore be integrated seamlessly. An often recurring pattern, where a seamless integration is required, could be as follows. The user first inspects the whole data set in an explorative manner in order to get an overview and identify potential structures of interest, to which s/he could navigate subsequently. This scenario is also in line with the often cited *overview, zoom, filter out, details-on-demand* concept, which has been introduced by Shneiderman to describe a general visual analysis process [24]. By keeping these

*email: diepenbrock@uni-muenster.de
†email: ropinski@math.uni-muenster.de
‡email: khh@math.uni-muenster.de

considerations in mind and avoiding the mentioned drawbacks of the trackball metaphor, we have developed an intuitive navigation metaphor for volumetric data, which supports explorative as well as directed navigation and is context-aware, thus allowing us to avoid manual navigation mode switches and to provide contextual information to the user. Several challenges occur when developing such a context-aware navigation metaphor for volumetric data. Unlike in scenes consisting of polygonal data, structures are not clearly defined in volumetric data. The set of visible features can be changed easily by adjusting rendering parameters such as the transfer function, clipping planes or when switching between different rendering modes. Therefore existing navigation algorithms, which are based on the assumption that scenes consist of a set of separable features, e.g., McCrae et al. [18], cannot be applied when navigating volumetric data. As a consequence, precomputed data structures would be needed for each possible set of rendering parameters which affect the visualized features. Obviously, this is not a feasible option because of the vast number of possibilities for rendering a data set. An alternative would be to recompute the data structures for each frame by considering the current set of rendering parameters. However, this would require a thorough data analysis and thus result in a significant performance drop, especially for large volume data sets. Therefore many of the specialized volume navigation techniques discussed in Section 2 assume a preset of rendering parameters and analyze the data set to be explored during a preprocessing step. Hence navigating arbitrary data sets with arbitrary rendering parameters is not supported.

In this paper we propose a novel image-based navigation metaphor which meets all of the requirements discussed above. By exploiting the processing power of current GPUs, we are able to analyze the environment surrounding the camera in real-time and thus to extract knowledge to support context-aware navigation, which does not require any manual navigation mode switches. To achieve this goal, we make the following contributions:

- An interactive image analysis technique based on spherical volume ray-casting, which allows our navigation metaphor to adapt to any visible structure and virtually arbitrary rendering techniques.
- A seamless integration of strafing, panning, rotating and flying, which allows convenient proximal as well as distal object inspection.
- Context-aware overlays which help the user to orient by providing an overview of the surrounding region.

2 Related Work

Since our navigation technique is of interactive nature, we do not cover automatic techniques in this section, and refer to the camera survey by Christie and Olivier [8].

Volumetric navigation. Most navigation techniques for volumetric data are specialized for navigation in tubular structures to support various virtual endoscopy applications: Virtual angioscopy [9], virtual colonoscopy [11], virtual sinus endoscopy [16] and virtual bronchoscopy [2]. Most of these techniques attach the virtual endoscope (i.e., the camera) to a precomputed or manually determined centerline [21]. Hence these techniques cannot provide an ad-hoc general-purpose navigation for arbitrary volumetric data. Nevertheless, these applications and algorithms have to be considered when designing a flexible navigation system. Especially relevant for our approach is the active virtual angioscopy navigation technique proposed by Haigron et al. [9]. The authors avoid precomputation by using techniques from the field of mobile robot navigation to automatically steer the camera through vascular structures. They analyze the depth buffer of the current view and direct the camera towards the location of the maximal depth value. In contrast to our technique the authors constrain their analysis to the current field of view, which is sufficient for path planning in tubular structures but cannot provide enough information for a more general navigation metaphor. As a consequence, for instance no collision detection can be performed when moving sideways. Serlie et al. [23] describe a virtual colonoscopy application that uses a cube map to provide a full 360-degree view and speed up rendering. To inspect volumetric data various approaches for optimal viewpoint selection have been proposed. Kohlmann et al. [15] describe a technique called LiveSync to link 2D slices with volume renderings. In contrast to our approach, LiveSync relies on 2D slice representations and does not support individual camera flights. Additionally, several more automatic and thus less relevant view point determination approaches exist [27, 4, 26, 28]. These techniques can be used as starting point for interaction exploration. Viola et al. [29] show how to determine expressive views on predefined features, which can be selected by the user during runtime.

Polygonal navigation. Besides these camera control techniques specifically developed for volumetric data, several relevant techniques for navigating polygonal data exist. Zeleznik and Forsberg [34] describe a technique called UniCam, which allows the user to perform navigation by using a gesture alphabet. While this is a promising concept, the authors report that users initially required several hours of training to get used to the technique. Our technique is inspired by the HoverCam technique proposed by Khan et al. [13]. To support proximal object inspection, a mouse drag is translated into a translation of the camera position, followed by an adjustment of the focus to the point closest to the camera. However, there are several differences between our technique and the HoverCam. First, HoverCam requires an indexing structure called sphere-tree which has to be extracted from the mesh the user wants to inspect. Even the improved HoverCam metaphor [18], which exploits rasterization in order to find the closest point to the camera still requires the scene geometry to be accessible, in order to avoid sudden switching of the camera focus between the objects of the scene. In volume rendering this information is not accessible, since the visualized structures change frequently, based on the chosen transfer function or set clipping planes. Second, the improved HoverCam metaphor requires the use of Proxy Objects of different LoDs in order to speed up rendering, and it utilizes the surface normal at the cursor position. The latter may not be provided by all volume renderers or may not even be clearly defined for arbitrary volumetric data. Third, the HoverCam metaphor does not allow adaptation of the up vector, since the application programmer has to define its behavior for each object. This would be obviously not feasible in volume rendering, where a data set contains a multitude of different objects. Another technique developed for polygonal data is the user designed navigation assistance as proposed by Burtnyk et al. [6, 7]. However, since it depends on predefined camera positions and/or paths, it is not suitable for the initial exploration of a volumetric data set, which we consider an important subtask of scientific visualization. Additionally, the predefined camera positions and paths may convey only little information after the transfer function has been changed, which is a common operation when exploring volume data. In general, 3D camera control as realized in computer games has only limited potential in the area of volume rendering. This is due to the assumption made by these techniques that the camera is linked to a character which is controlled by gravity and has a certain size. Hence we do not further consider these techniques and refer to the survey by Christie and Olivier [8].

3 Design Considerations

To realize an intuitive volume navigation metaphor, several design decisions have to be made. Since seven degrees of freedom come into play, camera control in general is a highly complex task. To allow full flexibility, the user must be able to change the three Cartesian coordinates describing the camera's position,

the three Euler angles describing its orientation, and its focal distance [8]. Obviously, setting all these parameters manually would result in a cognitive overload. In contrast, since in many visualization scenarios exploration plays an important role, navigation constraints are only acceptable up to a certain degree. In particular, a fully automatic navigation system designed for a specific approach would not be able to adapt to the user's intentions. This is a very important prerequisite, since the user might change her intentions during a visual analysis process frequently [3]. Therefore we have combined the strengths of an interactive approach with those of a reactive approach leading to a semi-automatic navigation metaphor. Especially, when a navigation mode change is required this semi-automatic proceeding is beneficial, since such a change should not be performed by using modifier keys or a graphical user interface [20]. Accordingly, we have limited ourselves to use only a standard wheel-mouse, which also eases porting the technique to touch displays. To further improve the usability, we have integrated well-established concepts from existing navigation metaphors where applicable. We have included the three main metaphors described by Ware and Osborne [31]: Camera-in-hand, world-in-hand, and flying vehicle. While with the camera-in-hand metaphor the camera is manipulated as if held in the user's hand, the input mapping is inverted when using the world-in-hand metaphor, and thus the camera rotates around a location fixed in the world. When using the flying vehicle metaphor, the camera can be steered as when sitting in an airplane. By combining these metaphors in a seamless manner, we are able to support proximal object inspection and moving through cavities simultaneously, which we have identified as central tasks in scientific and medical volume visualization.

At this point we would like to emphasize the importance of the flexibility achieved by the semi-automatic nature of our metaphor, which makes it a general-purpose navigation metaphor. This flexibility does not only allow to support distal and proximal object inspection in a seamless manner, but it also supports the navigation of arbitrary data sets, which can be visualized with arbitrary rendering techniques. This decoupling of the data and the renderer is achieved by performing all computations in image space, and it allows to apply the presented metaphor also within yet unknown application scenarios. The benefit of this property becomes clear, when considering other visualization systems. For instance, the virtual endoscopy system proposed by Krüger et al. [16] is a good example. To incorporate collision detection within their GPU-based rendering technique, an extra copy of the data set and knowledge about the renderer and its parameters are required on the CPU. Thus an additional development effort became necessary for navigation purposes only. By having a general purpose navigation metaphor, this interdependency during the development of visualization and navigation algorithms can be dissolved.

The example of the visualization system presented by Krüger et al. [16], highlights also another requirement for navigation algorithms: The need for collision detection and avoidance. This is in line with the work by Wojciechowski, who also emphasizes the importance of collision detection when dealing with navigation metaphors [32]. We believe that collision detection is of even greater importance in the area of volume visualization, since the volumetric nature of the data to be visualized affords the user to dive into it. Allowing this behavior without providing collision avoidance results in a high degree of occlusion, and thus the reduced visibility quickly leads to disorientation. Integrating an occlusion avoidance algorithm improves the situation, but might still lead to orientation problems, for instance when traveling through complex vessel trees. Hence a general-purpose volume navigation metaphor should also provide some contextual information when diving into the data and thus support location-awareness.

The image-based volume navigation metaphor proposed in this paper meets the following design criteria, which we have developed by having typical visualization scenarios in mind. It should:

- support semi-automatic camera control to allow flexible and adaptive navigation (e.g., when the user's intention, the data or the rendering style change),
- be intuitive and easy to learn, i.e., avoid navigation mode switches and integrate well-known concepts,
- allow distant and proximal object inspection, which are essential for most scientific visualization applications,
- integrate collision avoidance algorithms, and
- support location awareness.

Since most of these design criteria require some knowledge about the current visualization, we refer to the presented navigation metaphor as context-aware. To simplify the required knowledge extraction process, we assume that the navigation intended by the user depends on the visualized structures. For instance, if the user excludes structures from the rendering, e.g., by adapting the transfer function, s/he likely does not want to inspect these structures. To comply with this assumption, we have to extract knowledge regarding the visualized structures. This can be done by considering the opacity of each voxel or sampling point in the same way the renderer does it when rendering the data. However, a multitude of parameters can affect the opacity at each sampling point: Transfer functions, segmentation data, clipping, other modalities, derived data. Additionally, transfer functions have to be differentiated further since multiple types exist (e.g., 1D intensity, 2D intensity-gradient, 2D LH [22], 3D [14]) and different segments may also have different transfer functions assigned to them. Incorporating all these parameters within the navigation algorithm would not be a trivial task, and if possible at all it would likely lead to code duplications, which would need to be updated to incorporate additional data sets or formats, new transfer function types, additional rendering parameters, etc. Even if the correct classification could be calculated in the navigation system, the calculated information needs to be stored (possibly at sub-voxel accuracy), processed and regenerated upon changes of rendering parameters. We therefore believe that choosing an image-based approach is not only an option to increase performance but imperative when implementing a flexible volume navigation technique.

4 Navigation Algorithm

In order to satisfy the design considerations introduced above, we have realized the image-based navigation metaphor, which follows the workflow depicted in Figure 2. To be able to achieve interactive frame rates, the workflow has been designed to work with a GPU-based volume raycaster [17], although with slight modifications arbitrary renderers could be used. Based on the data set and the current camera parameters, we generate two pairs of color-coded entry- and exit-points: a conventional one, which is used for the actual rendering, and a spherical one, which is used for the image-based analysis. After the spherical entry- and exit-points have been generated, the subsequent renderer passes its output to the image analyzer. This image analyzer receives incoming mouse events and extracts the required information from the spherical rendering. Based on this information, the current camera is modified, and when necessary, image overlays are generated in order to support spatial awareness. These images are then overlayed over the standard rendering which is displayed on the screen.

By extracting knowledge about the regions surrounding the current camera position, we are able to flexibly react to mouse input to support the desired navigation tasks. In order to get a representation of these regions, we render a 360° fisheye view from the current camera position. All calculations necessary for the navigation are performed based on this image and the corresponding depth values of the first hit points. By exploiting a spherical mapping, we

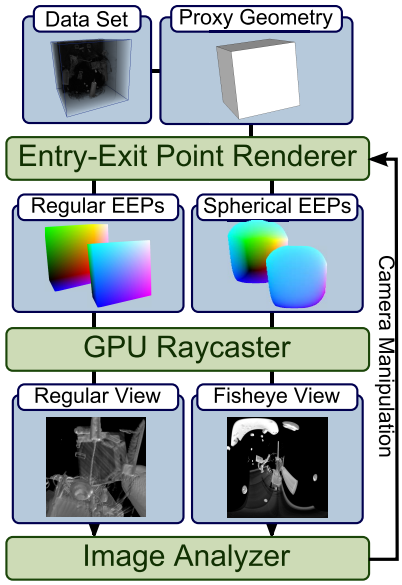

Figure 2: The workflow of our navigation system, integrated into a volume rendering system. The same raycaster is executed twice, one rendering is displayed on the canvas while the other is analyzed to map the users input to camera movement.

are able to get the information for all directions in only one rendering pass instead of six passes required for an alternative cube map approach. This is achieved by mapping each texel position of the entry- and exit-textures to $\theta \in [-\pi, \pi]$ and $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, which are translated to cartesian coordinates. The resulting positions, which are shown as color coded images in Figure 3 (left), are used to set up the rays for which an intersection test with the proxy geometry is performed. To simplify this process, we assume a convex proxy geometry, which should be no real limitation for most renderers and applications. We have to consider two different cases, based on the fact if the camera lies inside or outside the volume. In the first case when the camera is located inside the volume, the entry points are simply given by the camera position, and the exit points are the intersection points with the proxy geometry (see Figure 3 (right)). In the second case when the camera is located outside the volume, the entry and exit points are set to the first and second intersection points, respectively, while the volume is visible.

This proceeding has the advantage that we can use a standard volume raycaster in order to process these entry- and exit-points. Hence, out technique is independent of the volume ray-caster as long as it uses Krüger and Westermann style entry- and exit-points and writes first hit point depth values. Our approach is also independent of the rendering parameters and the data sets. For the actual

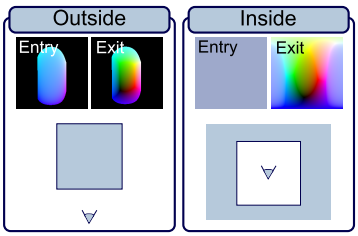

Figure 3: Spherical entry-exit points outside and inside the proxy geometry. The entry points for raycasting from inside the proxy geometry are constant because the rays start at the camera position.

navigation we analyze the output, i.e., the color and depth image, of a raycasting performed by processing these entry- and exit-points. The depth image will be used to modulate flight speed and perform collision detection while the color image is used to generate image overlays providing contextual information.

4.1 Image-Based Volume Navigation

Since one of our design goals was to exploit well-established navigation concepts, we have initially used the generated image data to implement the original HoverCam algorithm [13]. However, preliminary results have indicated that this approach suffers from the fact that the HoverCam metaphor always focuses onto the closest point in a certain search window. This often moves the focus in a direction not exactly matching the mouse movement, which is a known problem of the HoverCam metaphor that results in decreased productivity [5]. Hence we have developed our own concepts, which mimic the behavior of the trackball metaphor, when rotations are desired.

When inspecting an object the camera either rotates around the object (rotating, world-in-hand), moves along the object (strafing) or it rotates around the current position (panning, camera-in-hand) (see Figure 4). While a common solution to enable the user to perform all these three camera movements is the use of modifier keys, our technique instead solves this problem by being context-aware.

We would like to illustrate our approach by explaining two simple cases where the mapping between mouse input and camera reaction is straightforward: The first case is that of a camera positioned in front of a sphere, with the look-at vector pointing directly towards the sphere's center. In this case, the expected camera movement resulting from a mouse drag should be a rotation in the corresponding direction around the center of the sphere (see Figure 5 (a)). In the second considered case a camera is located at the center of a hollow sphere. When performing the same mouse interaction, the user would expect a rotation around the camera position (see Figure 5 (b)). Thus, in the first case the user perceives the camera rotating around the sphere, i.e., the world-in-hand metaphor, whereas in the second case s/he perceives the sphere rotating around the camera or the camera panning inside the sphere, i.e., the camera-in-hand metaphor.

To distinguish between these two basic cases, it is essential to have knowledge about the scene. Since the goal was to realize an image-based technique, the spherical depth image is our only information regarding the scene. Thus, we need to analyze it based on the current mouse input. We compare the depth value at the current screen center and at the current screen center plus the mouse offset. If the depth value in the center is smaller than the other one, we assume that the user intends a rotation, and otherwise a panning camera movement (recall Figure 5).

While panning requires no additional knowledge, for the rotating camera movement a center of rotation is required. Since we do not

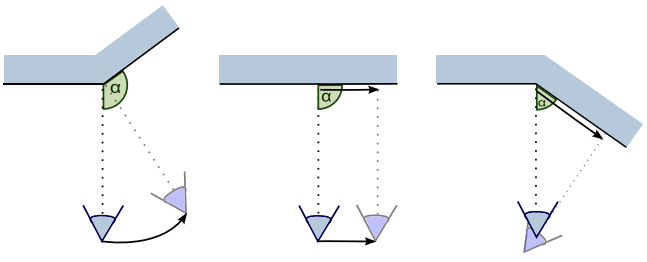

Figure 4: We distinguish between three basic camera movements when dragging the mouse: Rotating, Strafing and Panning. By analyzing the depth image we dynamically decide which movement the user wants to perform.

have such high level information in the depth image, we define the first hit point along the view vector as the center of rotation. While this proceeding does not explicitly integrate strafing, this behavior is inherently given. When for instance applying the presented approach in front of a plain wall, strafing behavior is achieved. In this case constant mouse drags along one direction would be mapped to rapid switches between small rotations and panning. The overall movement is a shaky strafing along the wall (see Figure 6 (a)). The fact that the combination of rotating and panning (roughly) results in strafing can be integrated into the algorithm to smooth the camera movement: Depending on the depth value difference, we interpolate rotating (movement of the camera position) with panning (movement of the focus), resulting in movement of camera position as well as focus, i.e., strafing. To correctly interpolate these two alternatives independent of the distance to the object, we calculate the surface angle α from the difference of depth values in relation to the depth. Based on this angle α the new camera position and focus are determined, α is depicted by the yellow arcs in Figure 4. If α is large ($> 100°$) the camera is rotated around the focus point. If α, on the other hand, is small ($< 80°$) the camera rotates around its current position, which results in a panning operation. For intermediate values both positions are interpolated, resulting in a translation along the surface, i.e., a strafing operation. However, this approximation of strafing will not maintain the initial distance between the camera and the object, as illustrated in Figure 6 (b). This can be fixed easily by setting the distance to the initial value.

In order to account for noise and to be able to move across smaller gaps, instead of just reading the depth values at two distinct locations, we analyze all depth values lying in-between these locations. By applying a Gaussian filter during this process, we are able to achieve smooth camera transitions. Further smoothing is obtained by the fact that the spherical depth image used for analysis has a limited resolution. This allows to integrate a LoD navigation approach, where object clusters are inherently detected. This works, since due to the fixed image resolution, the camera distance directly influences the clustering. Objects further away are projected on fewer pixels and thus small distances between them may dissolve. Thus it becomes possible to rotate around the whole cluster, when further away, while rotating around the individual objects when nearby.

Because rotations may result in an altered up vector, our technique in contrast to the HoverCam metaphor [18] allows to adapt the up vector interactively. This can be done by dragging with the mouse at the corners of the screen in order to initiate an appropriate rotation.

Since the used collision detection is based on well-known potential field approaches, we do not describe it within this Section, and provide a brief overview in Subsection 4.3.

4.2 Supporting Location Awareness

To support the user during the navigation, especially when diving into the volume, we augment the rendering by adding appropriate

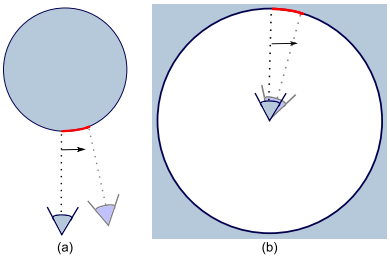

Figure 5: Expected camera movement for two simple cases: Rotating (a) and Panning (b).

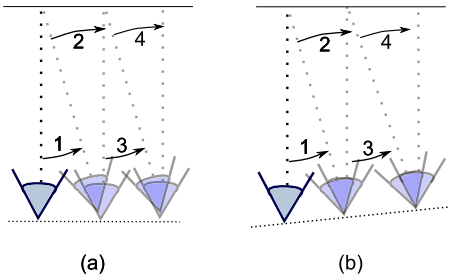

Figure 6: Strafing as combination of rotating and panning, with (a) and without (b) distance adjustment.

overlays.

4.2.1 Contextual Preview Images

To avoid disorientation, we add small preview windows that show parts of the surroundings currently not in the field of view (see Figure 7). These previews are generated by reprojecting the corresponding parts from the spherical rendering. We select the parts positioned on the left, the right, above, below and behind the camera. The latter case can be used for instance to realize a rear mirror metaphor. To render the previews, a view frustum is constructed, and rays are cast through all pixels into the spherical image surrounding the camera. The ray direction is then converted to spherical coordinates and used to look up the texels in the spherical image. Although one might suspect to see distortions in the top and bottom previews this is not the case as shown in Figure 8. Because this is essentially just an image processing and not a new volume rendering pass, the previews can be generated very efficiently.

4.2.2 Overview Map

While the contextual preview images help to get information regarding the surrounding, they provide only little support for estimating the own location. Thus, we also provide a map overlay, which shows the current position from a bird's-eye perspective when diving into the volume. This metaphor is often used in map applications to support spatial awareness. The map is rendered in an additional pass, where we can use the existing volume raycaster again but with a clipped proxy geometry and a different perspective. To define the parts of the volume visible in this overview map, we exploit the plane defined by the camera position and the up vector to clip the proxy geometry and render it from the bird's-eye perspective, i.e., located above and behind the main camera. An example of this overview map is shown in Figure 7. As it can be seen, we have also added the position and the field of view of the camera to support the orientation.

4.2.3 Context-aware Overlays

Both the preview images and the overlay map can be helpful to the user in certain situations but may only consume valuable screen

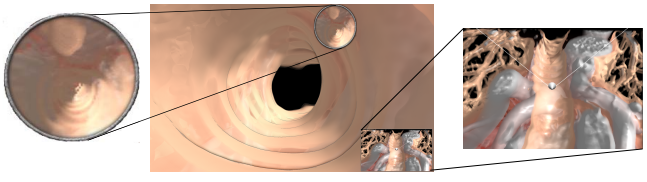

Figure 7: Based on saliency and the overall occlusion our technique overlays a rear preview and an overview map, respectively.

space in other situations. We therefore propose to make these overlays context-aware and use the information stored in the spherical image to dynamically decide which navigation overlays to display.

Saliency-based previews. Besides taking up screen space the content of the previews is not always necessarily helpful to the user - they might not even contain parts of the data set. The first heuristic we implemented just determined to what percentage each preview is filled. Although this correctly disabled most previews when exploring a data set from the outside, all five previews were active when flying for instance through a vessel. Most of the previews were not improving the exploration but simply showing a vessel wall of uniform color. Therefore we employ the notion of visual saliency for color images [12] and depth images [19] to decide the relevance of each preview at the current position in the data set. This heuristic of course assumes that the user has configured the renderer in a way to discriminate features of interest. The more heterogeneous a preview is, the more interesting it is assumed to be. In a similar way, the depth image is considered interesting if the values are non-uniform. Hence previews appear for instance when the user flies by a branch when inside a vessel. If the overall saliency of a preview image exceeds the specified importance threshold, the preview is considered helpful to the user and therefore worth the screen space. Because the rear preview is particularly useful and can be used to reverse the camera direction with a double click, it has a lowered threshold and is therefore visible more often.

Occlusion-based overview. While it makes sense to overlay the map when exploring cavities, it is not very helpful when inspecting an object from the outside. We have therefore implemented a heuristic to decide when the camera is inside a cavity and only then display the map. We consider all preview images except the one representing the front view. If more than 20% of these preview images are occupied with data lying within a closeness threshold, the camera is considered to be within a cavity and thus the overview map is displayed. In our tests this heuristic always led to the desired effect.

4.3 Integrated Camera Control

To unleash the full potential of our context-aware navigation technique, we have integrated it with other camera control metaphors. To combine our approach with a convenient traveling metaphor, we have implemented the flying vehicle metaphor. Thus, by pressing and holding the left mouse button for a short amount of time without moving it the user can start flying through the data set. During this flying, the direction is continuously modified by the offset between the mouse cursor and the center of the screen. The traveling speed is calculated based on the distance to the closest point in a region around the center of the screen. This proceeding is sufficient, since a user evaluation conducted by Ware and Fleet [30] has shown that averaging the depth values had no advantage over just using the closest depth value, when modulating flight velocity based on depth values. To prevent occlusions, we use a force field technique. In particular, we have implemented the force field approach proposed by Xiao and Hubbold [33] to prevent collisions while still allowing smooth flying. To allow mostly unrestricted movement, in our realization the camera is not constantly moved towards the center of the cavity, as it is done when applying navigation in virtual colonoscopy [11].

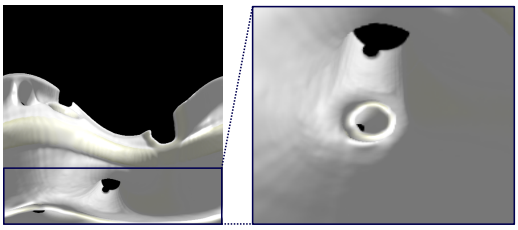

Figure 8: The previews are generated from the spherical image (left) without performing additional raycasting passes. The bottom preview (right) is reconstructed from the lower part of the spherical image without distortions.

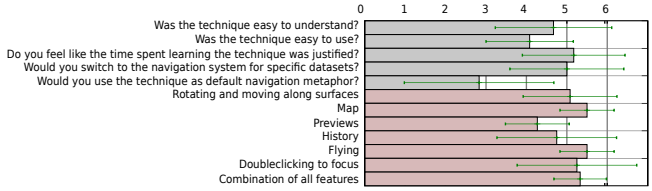

Figure 9: The results of the questionnaire of the conducted user study, which we have evaluated on a seven point Likert scale. Questions 6-12 (red) are about specific aspects of our technique and the participants were asked if they percieved these as useful or good solutions to the problem.

To support zooming, the user can move the camera forward and backward by using the mousewheel. The speed is again determined by the closest point, but no force field is used to provide a real zoom instead of an additional method to fly, which is consistent with the standard trackball. When moving backwards the closest point on the backside is used to check for collisions.

By double clicking the camera rotates to focus the point at the cursor position, which can be used to switch between separate objects. It should be pointed out that while it is of course possible to integrate this operation into a normal trackball navigation it would not be very helpful to the user. In contrast to our technique, which constantly updates the center of rotation, the trackball focus would stay fixed at the clicked point on the surface of the object, and thus the object would return around this point.

Finally, we have added a history function, which allows the user to fly back on the previous navigation path. Instead of adding an option to fly backwards or automatically push the camera out of an object, as proposed by McCrae et al. [18], we allow the user to rewind to previous positions by pressing the right mouse button. We believe that this solution, which has some similarity to forward and backward movement on a path generated for virtual endoscopy applications [10], is more robust and less confusing for the user.

5 Evaluation

5.1 Usability

Our usability evaluation is based on the results of a user study we performed. Additionally, we present our findings from an interview with a medical expert.

5.1.1 User Study

With a lot of mainstream applications like Google Earth using the trackball metaphor we assume that the vast majority of users, especially those that might be interested in navigating volumetric data, are familiar with this navigation approach. We therefore designed our study to investigate how users perceive the implemented navigation system and if they would be interested in using it - as an alternative to the trackball or even more ambitious as the default navigation technique. Another important aspect of our study was to find out how difficult it is for the average user to learn our technique and whether the user thought the time spent to learn the technique was well spent.

11 male and 2 female subjects participated in the study. Most subjects were students or members of the departments of computer science or medicine. All subjects were first-time users of our navigation technique. The total time per subject including instructions,

training, experiment, breaks and debriefing took approximately five to ten minutes. The study itself consisted of a timed navigation task in which we compared our technique with the trackball (two-axis valuator) and a learning task in which the users had to navigate (roughly) as shown in a tutorial video recorded using our technique. The second task was designed to include all aspects of our navigation system. A questionnaire based on a seven point Likert scale has been used to evaluate the users experience.

The users were split up into two groups: The trackball group first performed the navigation task, was then instructed in our technique and performed the learning task. The other group was instructed how to use our technique, then performed the task to learn it, and performed the navigation task using our technique. Since the trackball is not designed to navigate through cavities, we compared only the ability to navigate around objects from the outside in the timed navigation task. The users had to read three letters placed on a sphere in a synthetic data set consisting of a set of objects. All users started at the same position, had to zoom in on the sphere and then navigate around it to read the letters.

The learning task was performed using the Baby data set (a CT scan of a head). The users were shown the following navigation subtasks in a video and had to replicate them:

1. Navigate around the head.
2. Use the transfer function to make skin and brain transparent and navigate inside the skull.
3. Use the rewind functionality to get back to the starting position.
4. Focus the tube, inspect it from the outside (rotate around it) and fly through it.
5. Use the rear preview to focus on the data set after leaving the tube at the neck.

Figure 9 shows the results of the questionnaire. For the users it was generally easy to learn the technique in a few minutes, and they perceived the navigation metaphor as helpful. Many even fully agreed that they would like to use the metaphor for special data sets. However, the rather ambitious question, if the users would in general switch to the metaphor received only little positive resonance. We believe that this partially results from the fact that the users had some previous experience with the trackball metaphor. As seen in Figure 9, all additional navigation techniques were perceived very well. Besides these results, we found that users with the most experience using the trackball had greater difficulties learning the new technique. While the users were in general faster, when using our metaphor, this fact has to be evaluated carefully. However, since we only performed one test and just have evaluated 13 users, this may not be statistically significant.

5.1.2 Evaluation with Domain Experts

To get an expert opinion we interviewed a medical doctor and two medical PhD students, all specialized in cardiovascular medicine. During the interviews, we have demonstrated our technique by loading several different data sets and handed over control to the medical experts. The participating medical experts had less experience with the trackball metaphor and 3D graphics in general than the lay users participating in our study, because they almost exclusively work with slice viewers. Thus, they had more problems performing 3D navigation tasks, also when using the standard trackball metaphor.

However, when using our technique, they were in general very positive and have identified features being of potential interest when inspecting 3D medical data. Especially the possibility of moving along surfaces and traveling through the data set without experiencing collisions was appreciated. When testing the navigation metaphor with their research data, which consists of complex vessel trees as shown in Figure 1, the navigation technique worked as they have expected for flying and rotations. While they also liked the option to strafe along a surface, the subjects expressed the wish to be also able to look ahead in the direction of movement instead of just towards the surface. Furthermore, they stated that medical doctors seldom need to inspect structures from the outside, and therefore the demand for trackball-like behavior was rather limited. As exceptions to this rule the inspection of bones, especially the pelvis with rather large and flat bone structures and the shoulder joint were mentioned.

5.2 Performance

The proposed system has been implemented in C++, OpenGL, GLSL and OpenCL. To minimize downloads from the GPU and speed up computation we exploited OpenCLs ability to interoperate with OpenGL and perform calculations on the spherical image directly on the GPU. Thus, we were able to interweave the visualization and navigation technique in a manner that everything is done in-situ. This allows interactive frame rates, which would not have been possible with CPU realizations. In our implementation, we have parallelized the force field approach, and only a few floats need to be read back from the GPU. Finding the closest point in a direction and reading a line from the depth image is accelerated in a similar fashion. Reading only some scalar results or parts of a texture back from the GPU would have been more complicated using OpenGL exclusively.

The most expensive operations are obviously the additional rendering passes for the spherical view and the map. The spherical view is rendered at a relatively low resolution (we found 256×256 to be sufficient for all tested datasets) while the size of the map is a (configurable) fraction of the main canvas size. For a typical canvas resolution of 1024×1024 the rendering time for one frame increased by 5-10 % for each of these renderings. We found the impact of the actual calculations for the navigation to be negligible. Therefore, if an existing rendering system is able to deliver interactive performance this would very likely not be changed by the integration of our technique.

5.3 Limitations

Although we developed our technique as general as possible we are aware that it will probably not be ideal for all data sets. Large semi-transparent regions (e.g., in data sets resulting from numerical simulations) cannot be explored from the inside because the spherical image provides no usable depth values in this case. If these regions contain less transparent inner parts along which the user wants to navigate, the transfer function could automatically be adapted to make almost transparent voxels entirely transparent. This transfer function would then be used to render the spherical image. Datasets where the objects of interest cannot be separated from the background visually (e.g., due to a low signal to noise ratio) can also be problematic. New rendering techniques that generate clearer images for these datasets can however be integrated easily and thus improve the navigation.

A general problem with semi-automatic navigation methods is the possible conflict with the user's intention. This can be overcome by adding possilities (e.g., modifier keys) to force the navigation technique into a specific mode. Keeping in mind touchscreens as possible application we did not integrate these.

The continous switching between different navigation metaphors can produce occasional jittering. We have tested different approaches to eliminate this jittering. Using a hysteresis to limit switching between the different interaction metaphors resulted in increased jittering. For instance, when rotating the camera around a spherical object, the object would be moved out of focus which would then be corrected by rotating the camera. This results in a unsteady movement of the sphere on screen, which is perceived as irritating. Approximating the center of the object in focus from the

depth image and rotating around it resulted in a smoother, more indirect movement around simpler objects (spheres, cubes), but turned out to be too unreliable for complex objects. If the centerline or medial surface for an object of interest is static and available it should obviously be used for navigation.

6 Conclusions

In this paper we have introduced a general-purpose volume navigation metaphor. By exploiting image analysis of a spherical projection of the camera's environment, we are able to simulate different concepts known from well-established navigation metaphors without requiring navigation mode switches. We have shown how to support rotating, strafing and panning of the camera by exploiting an image-based analysis of its environment. Thus, we enable proximal as well as distal object inspection, which is an important combination in many scientific visualization applications. Furthermore, we have shown how to exploit the knowledge about the camera's context in order to generate preview images as well as overview maps. The latter is of particular importance in the area of volume visualization, since the volumetric nature of the data affords to dive into it, which easily leads to disorientation. Thus, our technique is able to support location-awareness. Since all required analysis tasks are performed based on the rendered image, the presented metaphor can be considered as a general-purpose metaphor, such that it can be used with most visualization techniques and data sets. To our knowledge the presented approach is the first general-purpose volume navigation metaphor. To evaluate the quality of the presented navigation approach, we have performed a usability evaluation and have conducted interviews with medical experts. Since we obtained positive feedback from these tests, we are confident that the presented navigation metaphor can fill the gap between general purpose volume visualization systems using classic navigation metaphors and systems developed for specific application cases that exploit specialized navigation techniques. In the future it would be worth to investigate how to address the limitations discussed in Section 5.3.

Acknowledgements

This work was partly supported by grants from the Deutsche Forschungsgemeinschaft (DFG), SFB 656 MoBil Münster, Germany (project Z1). The presented concepts have been integrated into the Voreen volume rendering engine (www.voreen.org).

References

[1] R. Bade, F. Ritter, and B. Preim. Usability comparison of mouse-based interaction techniques for predictable 3d rotation. In *Smart Graphics*, pages 138–150, 2005.

[2] D. Bartz, D. Mayer, J. Fischer, S. Ley, A. d. Rio, S. Thust, C. P. Heussel, H.-U. Kauczor, and W. Strasser. Hybrid segmentation and exploration of the human lungs. In *IEEE Visualization*, pages 177–184, 2003.

[3] S. Beckhaus, F. Ritter, and T. Strothotte. Cubicalpath - dynamic potential fields for guided exploration in virtual environments. In *Pacific Graphics*, page 387, 2000.

[4] U. D. Bordoloi and H.-W. Shen. View selection for volume rendering. *IEEE Visualization*, 0:62, 2005.

[5] E. G. Britton, J. S. Lipscomb, and M. E. Pique. Making nested rotations convenient for the user. *SIGGRAPH*, 12(3):222–227, 1978.

[6] N. Burtnyk, A. Khan, G. Fitzmaurice, R. Balakrishnan, and G. Kurtenbach. StyleCam: interactive stylized 3D navigation using integrated spatial & temporal controls. In *ACM UIST*, pages 101–110, 2002.

[7] N. Burtnyk, A. Khan, G. Fitzmaurice, and G. Kurtenbach. Showmotion: camera motion based 3D design review. In *ACM I3D*, pages 167–174, 2006.

[8] M. Christie, P. Olivier, and J. Normand. Camera control in computer graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218, 2008.

[9] P. Haigron, M. Bellemare, O. Acosta, C. Göksu, C. Kulik, K. Rioual, and A. Lucas. Depth-map-based scene analysis for active navigation in virtual angioscopy. *IEEE Transactions on Medical Imaging*, 23(11):1380, 2004.

[10] T. He, L. Hong, D. Chen, and Z. Liang. Reliable path for virtual endoscopy: ensuring complete examination of human organs. *IEEE TVCG*, pages 333–342, 2001.

[11] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He. Virtual voyage: interactive navigation in the human colon. In *SIGGRAPH*, pages 27–34, 1997.

[12] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE TPAMI*, 20(11):1254–1259, 1998.

[13] A. Khan, B. Komalo, J. Stam, G. Fitzmaurice, and G. Kurtenbach. HoverCam: interactive 3D navigation for proximal object inspection. In *ACM I3D*, page 80, 2005.

[14] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE TVCG*, pages 270–285, 2002.

[15] P. Kohlmann, S. Bruckner, and A. Kanitsar. LiveSync: Deformed viewing spheres for knowledge-based navigation. *IEEE TVCG*, 13(6):1544–1551, 2007.

[16] A. Krueger, C. Kubisch, B. Preim, and G. Strauss. Sinus endoscopy - application of advanced gpu volume rendering for virtual endoscopy. *IEEE TVCG*, 14:1491–1498, 2008.

[17] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. *IEEE Visualization*, pages 287–292, 2003.

[18] J. McCrae, I. Mordatch, M. Glueck, and A. Khan. Multiscale 3D navigation. In *ACM I3D*, pages 7–14, 2009.

[19] N. Ouerhani and H. Hgli. Computing visual attention from scene depth. *Pattern Recognition*, 1:1375, 2000.

[20] J. Raskin. *The humane interface: new directions for designing interactive systems*. ACM Press/Addison-Wesley, 2000.

[21] T. Schlathoelter, C. Lorenz, I. Carlsen, S. Renisch, and T. Deschamps. Simultaneous segmentation and tree reconstruction of the airways for virtual bronchoscopy. In *SPIE*, volume 4684, pages 103–113, 2002.

[22] P. Sereda, A. Bartroli, I. Serlie, and F. Gerritsen. Visualization of boundaries in volumetric data sets using LH histograms. *IEEE TVCG*, 12(2):208–218, 2006.

[23] I. Serlie, F. Vos, R. Van Gelder, J. Stoker, R. Truyen, F. Gerritsen, Y. Nio, and F. Post. Improved visualization in virtual colonoscopy using image-based rendering. In *IEEE/EG Eurovis*, page 137. Springer Verlag Wien, 2001.

[24] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, page 336, 1996.

[25] K. Shoemake. Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface*, pages 151–156, 1992.

[26] S. Takahashi, I. Fujishiro, Y. Takeshima, and T. Nishita. A feature-driven approach to locating optimal viewpoints for volume visualization. *IEEE Visualization*, pages 495–502, 2005.

[27] P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *Vision Modeling and Visualization*, pages 273–280, 2001.

[28] P. Vázquez, E. Monclús, and I. Navazo. Representative views and paths for volume models. In *Smart Graphics*, pages 106–117, 2008.

[29] I. Viola, M. Feixas, M. Sbert, and M. E. Groller. Importance-driven focus of attention. *IEEE TVCG*, 12(5):933–940, 2006.

[30] C. Ware and D. Fleet. Context sensitive flying interface. In *ACM I3D*, page 127, 1997.

[31] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. In *ACM I3D*, pages 175–183, 1990.

[32] A. Wojciechowski. Potential field based camera collisions detection in a static 3d environment. *MG&V*, 15(3):665–672, 2006.

[33] D. Xiao and R. Hubbold. Navigation guided by artificial force fields. In *ACM CHI*, page 186, 1998.

[34] R. Zeleznik and A. Forsberg. UniCam2D gestural camera controls for 3D environments. In *ACM I3D*, page 173, 1999.