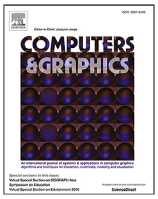
Computers & Graphics 67 (2017) 77–85

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag



Special Section on SCCG 2017

Data-sensitive visual navigation

Peter Mindek[a,*], Gabriel Mistelbauer[b], Eduard Gröller[a,c], Stefan Bruckner[d]

[a] Institute of Computer Graphics and Algorithms, Faculty of Informatics, TU Wien, Austria
[b] Department of Simulation and Graphics, Faculty of Computer Science, University of Magdeburg, Germany
[c] VRVis Research Center, Austria
[d] Department of Informatics, Faculty of Mathematics and Natural Sciences, University of Bergen, Norway

ARTICLE INFO

Article history:
Received 10 April 2017
Revised 19 April 2017
Accepted 23 May 2017
Available online 13 June 2017

Keywords:
Navigation
Exploration
Medical visualization

ABSTRACT

In visualization systems it is often the case that the changes of the input parameters are not proportional to the visual change of the generated output. In this paper, we propose a model for enabling data-sensitive navigation for user-interface elements. This model is applied to normalize the user input according to the visual change, and also to visually communicate this normalization. In this way, the exploration of heterogeneous data using common interaction elements can be performed in an efficient way. We apply our model to the field of medical visualization and present guided navigation tools for traversing vascular structures and for camera rotation around 3D volumes. The presented examples demonstrate that the model scales to user-interface elements where multiple parameters are set simultaneously.

1. Introduction

In interactive visualization systems, parameters are typically set through a wide variety of user interface elements. It is challenging to design an interface, which would allow the users to efficiently explore and analyze complex datasets. It is often necessary to provide two separate sets of control elements, which respectively provide coarse and fine parameter adjustments. This approach clearly does not scale with the increasing complexity of the data, or with the increasing generality of visualization systems.

Specifically, given the non-linear nature of the visualization-mapping functions used to display the data, the changes in the user input do not always proportionally reflect the visual changes in the output image which can make the exploration process counter-intuitive. This is, for instance, often an issue in general-purpose volume rendering applications, where it is not always possible to prepare adequate presets. In such applications, transfer functions have to be adjusted manually, which can be challenging, particularly for users without a strong technical background. Small changes in certain parts of the transfer function may cause significant changes in the visualization, while modifying the rest of the transfer function may not have a strong effect on the output image. In general, it is hard to predict how the parameter changes will influence the output image.

Similar problems exist in colorimetry. Most of the color spaces are perceptually non-uniform, such as the RGB, HLS, or *CIE XYZ* color model. Changes of colors will be perceived differently by the human observer depending on their specific location in the color space. For this reason, color spaces such as *CIELUV* were devised, which are perceptually uniform. Here, a unit color change leads to a unit perceptual change, independent of where in the color space this change occurs. We address a similar issue, but in the area of human–computer interaction with interactive visual systems. Our goal is to make changes in the input parameters more predictable with respect to the perceived changes in the output image and to actively support users in the anticipation of the expected effects of the interaction.

Specifically, we address the problem of unpredictable interaction in medical visualization, where complex 3D volume data are explored on a regular basis. For this purpose, various types of input elements are investigated, realizing different ways of user interaction.

To be able to enhance different kinds of input elements, we introduce a general model of *data-sensitive navigation* consisting of two elements:

- *Data-sensitive manipulation* describes how the behavior of input elements is modified so that changes of the visual output are made proportional to changes in the user input.
- *Data-sensitive guidance* refers to the visual encoding of the information used for normalizing the output changes to support users in steering the interaction towards specific goals.

* Corresponding author.
E-mail address: mindek@cg.tuwien.ac.at (P. Mindek).

http://dx.doi.org/10.1016/j.cag.2017.05.012
0097-8493/© 2017 Elsevier Ltd. All rights reserved.

We demonstrate our model by proposing *TreeSlider*, a novel data-sensitive navigation tool for the investigation of tree structures. TreeSlider can be used to traverse vessel trees in order to identify potential pathologies such as stenoses or occlusions. Radiologists typically have to inspect hundreds to thousands of slices, and important pathologies can be easily missed. By adapting the interaction mechanism to the underlying data and explicitly encoding information about potentially important regions in the data, we show how our approach can assist users in this tedious and time-consuming task. In the second example, we apply our approach to the more general scenario of camera manipulation. Here, we show how viewpoint relevance measures can be employed to provide additional visual guidance in the interaction with 3D visualizations.

2. Related work

While the motivation of our work are specific tasks from the field of medical visualization, our model is generally applicable to various computer graphics areas where interaction is employed as well. Therefore, we subsequently review existing literature from both areas, namely interaction in computer graphics as well as medical visualization.

Non-linear interaction. Lindow et al. [1] present a method for transforming input parameters of visualizations to obtain a linear relationship between the input and the output. Gavrilescu et al. [2] customize common interaction elements by visualizing the amount of change caused by modifying these elements. van Wijk and Nuij [3] describe a metric for efficiently navigating between two regions in large 2D environments. Their non-linear approach simultaneously performs zooming and panning and ensures a constant change of velocity. Blanch et al. [4] present the *semantic pointing* interaction metaphor. Objects within a user interface have two different size properties: one in motor space (importance for interaction) and the other one in visual space (the displayed visual size). Elmqvist and Fekete [5] describe several mapping functions for picking objects in a 3D scene by adjusting the ratio between motor and visual space. In a region without target objects, the cursor moves fast and is enlarged, whereas when approaching an object, the cursor progressively becomes smaller and slower. Chapius et al. [6] present an interaction technique that couples the size of the cursor to its speed while minimizing visual distraction. When approaching an object (the closest one), the area of the cursor decreases until it behaves like a common point-cursor when hitting the object. Elmqvist et al. [7] propose a space deformation technique for exploring large geometric spaces such as maps or networks. They fold the space similar to an accordion in order to reduce the consumed screen space while simultaneously preserving the overall context. Ji and Shen [8] describe an approach that selects the optimal view of a static scene by analyzing the opacity, color and curvature of images from various viewing angles. By employing an optimization to maximize the perceived information, they achieve smooth and constant visual changes of the optimal viewpoint in dynamic scenes of time-varying data. Kohlmann et al. [9] present an interaction metaphor for linking 2D slice views with 3D volume rendering. Depending on a minimal set of input parameters such as patient orientation, local object shape, and viewpoint history, an optimal viewpoint from a user-picked slice position is determined. In their follow-up work [10], they additionally adjust the transfer function within the 3D view in order to reveal the structure of interest without obstruction. Wörner and Ertl propose *SmoothScroll* [11], an interaction element for browsing one-dimensional data, which can be hierarchically aggregated at multiple levels. This allows for browsing of very large datasets. Our method is applicable to controls associated with one-dimensional data, but also to more complex structures such as trees. We demonstrate this by applying our method to *TreeSlider*, a novel interaction element for browsing tree structures.

Furthermore, we propose a general model that unifies the concept of remapping the user interaction impact on the input parameters and the guidance, which visually hints on how the impact is remapped. Willett at al. [12] propose guidelines for displaying such information. We apply our concept on more complex input elements, where existing methods are not usable.

Angiography. The analysis of blood vessels is a crucial and fundamental procedure in medicine for investigating embolisms, calcifications, stenoses, or occlusions within the lung, the brain, or the peripheral arteries, such as the legs. All these examples share a tree of the arterial system as basis for their analysis. Starting with the most basic, but widespread, procedure for analyzing vascular pathologies, all axial or transverse slices of an acquired medical volume dataset have to be viewed and precisely inspected. Since this demands a lot of manual work from physicians, Kanitsar et al. [13] discuss Curved Planar Reformation (CPR), a visualization technique that creates a cut along a blood vessel in order to inspect its interior (or lumen) in an obstruction-free manner. This approach significantly reduces the number of images to investigate. Auzinger et al. [14] propose a technique to view the lumen of blood vessels from unrestricted viewing angles. Portugaller et al. [15] investigate the importance of viewing slice-images together with supporting techniques such as Maximum Intensity Projection (MIP). They conclude that transverse slices still play an essential role in detecting arterial lumen narrowings, although techniques such as CPR reduce the number of images to analyze. Motivated by this fact, we enhance the well-established and accepted role of transverse-slice images in a data-sensitive manner. The layout of vascular structures is important when browsing through the vessels. In order to reflect the spatial arrangement of the investigated blood vessels, Borkin et al. [16] developed a horizontal arrangement for analyzing the endothelial shear stress of heart arteries. Although this layout represents the vascular system well, its spatial orientation would be problematic when investigating arteries, such as the ones of the human lower extremities, since these are mostly vertically oriented. We propose an extended interface element that remedies this problem while utilizing a well-known interaction paradigm, the slider. Inspired by the simplicity of a slider and the necessity to browse through a vessel tree, we combine both aspects into one interface element, called *TreeSlider*. Oeltze and Preim [17] describe an organically looking visualization for vascular structures based on convolution surfaces. The vessel tree is convolved with a Gaussian filter and artifacts at branching points, such as bulging or blending, are handled by reducing the kernel size. Wu et al. [18] propose an adaptive refinement for surface-based vascular visualization to have more polygons in regions with high curvature. A comparison of different surface modeling approaches for visualizing blood vessels is given by Wu et al. [19]. Based on the focus and context principle proposed by Straka et al. [20], we blend the vessel tree over a MIP of the underlying dataset.

3. Data-sensitive navigation

To address various problems arising in the interaction with visualization systems through diverse input elements, we present a model of data-sensitive navigation. This model can be applied as a basis for enhancing interaction in various vessel visualization scenarios. In its generality, the model could be employed in other application areas as well.

In visual computing, interaction consists of a feedback loop between the user input and the visual output. The user sets the

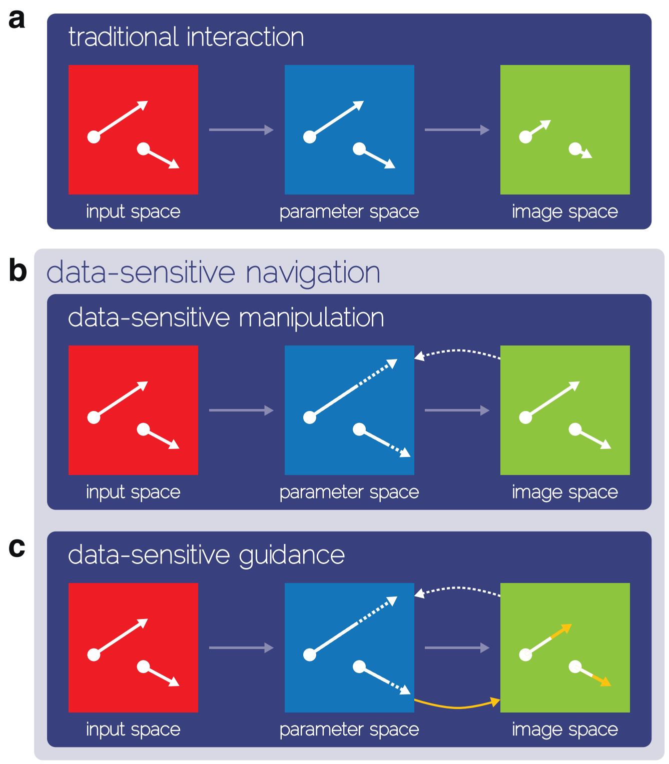


Fig. 1. White arrows represent changes in input space, parameter space, or the output visualization. (a) User input is directly mapped to the input parameters. The changes of the input are not proportional to the changes of the output. (b) Data-sensitive manipulation, where the underlying data or the output are used to dynamically scale the changes in the parameter space. In this way, changes in the output are proportional to the changes in the input space. The mapping is illustrated with the dashed arrows. (c) The mapping information is displayed in the image space to guide users during the data-sensitive navigation (yellow arrows). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

parameters, he observes the generated output image, and refines the parameter settings accordingly until the output is satisfactory. To model this process, we split the data-sensitive navigation into two parts – *manipulation* and *guidance*. Data-sensitive manipulation transforms the user input with respect to the underlying data, so that the changes in the input are made proportional to the changes of the output. Data-sensitive guidance augments the visual output with information derived from the remapping used in the manipulation stage. The guidance informs the user about the effects of potential future changes of the input elements. Depending on the interaction method, the guidance information can overlay either the input elements themselves or the output image. The location of the overlay is chosen so that the user is not distracted during the interaction, yet the guidance information is properly communicated.

Fig. 1 gives an overview of our proposed model. Fig. 1(a) shows the traditional interaction. Changes of the user input are proportional to the changes of the parameter values, which often do not have direct semantic relevance to the user. Changes in the output image, providing the only visual feedback for the user to steer the input, are typically not proportional to the input changes. The worst-case scenario is that certain desired outputs are impossible to achieve with the given input elements.

3.1. Model overview

Our model addresses this problem in two different ways. Fig. 1(b) shows *data-sensitive manipulation*. The goal of this concept is to non-linearly modify the sensitivity of input elements to make the changes in the input proportional to visual changes in the output images. The sensitivity of an input element is the amount of parameter-value change caused by a unit interaction, i.e., how much does the underlying parameter value change if a slider handle moves a unit distance. This non-linear sensitivity modification ensures that the screen-space of the input elements is properly utilized for efficiently controlling the changes in the output. For instance, a slider could be non-linearly rescaled so that larger portions of the slider-track map to the areas of high interest, thus making the sensitivity of the slider lower around these areas.

On the other hand, data-sensitive guidance (Fig. 1(c)) displays gathered information about interesting aspects of the underlying data in a way that helps users to find them more quickly. An example is a slider augmented with non-uniform ticks, indicating non-linear changes in the output with respect to linear slider changes. This form of visualization reduces the amount of trial-and-error exploration by steering users in the specification of parameter settings to obtain desired results. Depending on the type of the input element, this information can be shown in the output image, or it can be used to augment the input element itself to make it easier to use. This is illustrated by yellow arrows, which represent the displayed guidance information.

3.2. Data-sensitive manipulation

To consider various input elements, which are used to control parameters of different types, we use a formalism that describes the problem of non-proportionality between input and output in a general way.

A common characteristic of input elements is that they are employed to specify user input I, which is transformed into parameter settings P. These parameter settings are used to generate an output image O. Typically, the changes in the input I are not proportional to the changes in the output O. This is due to the fact that there is usually a non-linear relationship between P and O, while I is linearly mapped to P ($\Delta I \propto \Delta P \not\propto \Delta O$). To rectify this situation, it is necessary to apply a transformation T which introduces a non-linear relationship between I and P:

$$P = T(I) \mid \Delta I \propto \Delta O \tag{1}$$

During data-sensitive navigation, parameter values are determined by the transformation $T(I)$ instead of the traditional linear mapping of I to P. This makes the user interaction proportional to the visual changes in the output images.

In most cases, the transformation T cannot be defined analytically, as this would require a complex mathematical model of the visualization mapping as well as of the underlying dataset. Such a model is typically not available. Therefore, we assume that it is possible to define a function $d(P)$ which estimates the *importance* of the respective output image O generated from the given parameter settings P. Alternatively, $d(P)$ can be defined as the magnitude of change of O with respect to P. The transformation T is then constructed from d as described below. The function d is obtained by sampling the parameter space and applying an adequate interpolation. Such an approach was employed by Lindow at al. [1].

Fig. 2 illustrates how the transformation T is constructed from the sampled importance function d. First, n samples of the function d are taken by analyzing the corresponding output images. These sampled importance values are then normalized so that their sum is equal to the range of the given input element. The normalized sample values are denoted as $d'(s_i)$. The input domain is split into regions corresponding to individual samples. The size of each region is proportional to the normalized importance $d'(s_i)$ of the respective sample s_i. If all outputs would have equal importance, the input domain would be split into regions of equal size and the

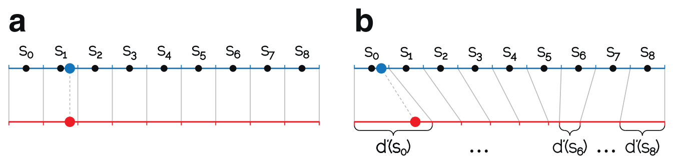

a
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8
b
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8
$d'(s_0)$ … $d'(s_6)$ … $d'(s_8)$

Fig. 2. Transformation T of an input to get a parameter value, indicated by gray arrows. The red dot indicates the input, e.g., a slider position, the blue dot shows the parameter value. The black dots are the sample positions s_i in the parameter space, while $d'(s_i)$ is the normalized importance $d(s_i)$ of the sample s_i. (a) Transformation T if importance values of all samples are uniform (data-sensitive manipulation is disabled). (b) Transformation T if the samples have non-uniform importance values (data-sensitive manipulation is enabled). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

input would be mapped to the parameter values in a traditional, linear way (Fig. 2(a)). If each output would be assigned a different importance, parameter-space regions with higher importance would be assigned larger regions of the input domain, e.g., ranges in a slider (Fig. 2(b)). Essentially, the sensitivity of the input element is modified in such a way that its resolution is increased in those areas, which account for more visual changes in the output image.

3.3. Data-sensitive guidance

As previously explained, data-sensitive manipulation changes the way how the user input maps to the displayed output in traditional interactive environments. This mapping, achieved through the transformation T, allocates more screen-space of the interaction elements to those parts of the parameter space that account for greater visual changes of the output images. However, this might not always be apparent by observing the output image alone. Even though the resolution of the input element is increased in potential regions of high interest, the non-linear mapping between the input and the output through the abstract parameter space might cause confusion for the user. Therefore, in addition to data-sensitive manipulation, we employ the concept of data-sensitive guidance. Data-sensitive guidance is a way of displaying the importance function in image space to steer the user while they are interacting with the system. In this context, the image space is either the output visualization, any linked view, or the portion of the screen where the graphical representation of the input element itself is placed. Hence, it is possible to show the guidance information wherever it is most suitable for the given application and interaction type. This is indicated in Fig. 1(c) as the yellow arrow between the parameter space and the image space.

Since there are many different ways how the importance can be shown, we categorize them into two groups:

Local encoding. The importance value is shown for the current parameter settings or its close neighborhood. This type of encoding is useful if a continuous path within the parameter space needs to be explored by the user. The local encoding of the importance guides the user where to go next from the current parameter settings, or how fast can they proceed. For instance, when traversing blood vessels, the user moves through the vessels continuously. It is unlikely that they will jump from the current position within the vessel to other than neighboring positions. In this case, the local encoding is sufficient to provide the guidance in the traversal process.

Global encoding. The importance is visualized for the whole input domain or the parameter space. This type of encoding is useful if the parameter space is not necessarily explored in a continuous manner. Therefore, the importance is shown for all available choices to help the user decide on how to change the input. An example is selecting adequate viewpoints in the visualization of 3D objects. It is not essential that the viewpoints are continuously explored, but good ones need to be found. In this case, the global encoding of the importance is appropriate.

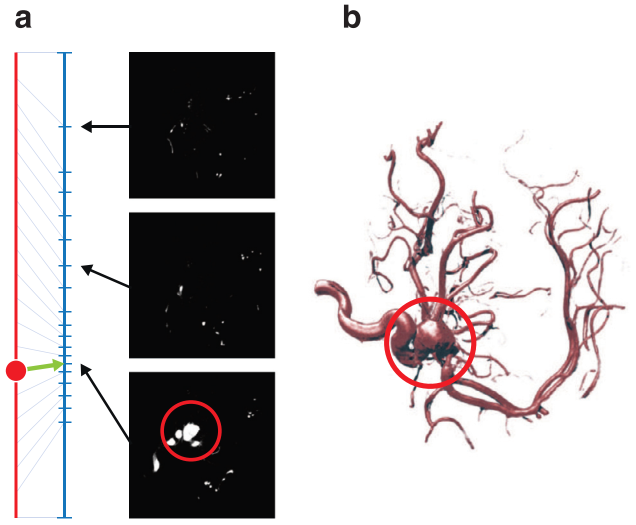


Fig. 3. Data-sensitive navigation applied to a simple slider for volume slicing of a CTA dataset of an aneurysm (marked with the red circle). (a) The red line represents a slider. The visual guidance information shown to enhance the usability of the data-sensitive slider is depicted in blue (global encoding) and green (local encoding) color. (b) A volume rendering of the entire dataset for overview. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Displaying importance instead of (or in addition to) the non-linear mapping of the input to the parameter values helps guiding the user towards areas of potentially high interest within the explored data. In the following sections we present several ways how this type of visualization can be utilized to steer the navigation in interactive visualization systems.

Fig. 3 illustrates data-sensitive navigation on a simple slider (shown in red color) used for traversing a stack of slices of a volume dataset. The sensitivity of the slider is modified using data-sensitive manipulation, so that it provides more resolution around the areas of interest (in this case, an aneurysm). A simple pixel difference between consecutive slices is used as the importance function, while the transformation T is constructed as illustrated in Fig. 2. In this way, the slices, where significant visual changes occur, are considered as areas of high interest. Because of the uneven shape of the aneurysm, this metric assigns high importance to the areas around it.

The importance function used to construct the transformation T from the input to the parameter space is visualized next to the slider in blue color. This visual guidance in the form of ticks indicates how the screen-space of the slider is allocated to individual parts of the slice stack. Areas with higher importance values are indicated by a higher density of the ticks. This is an example of global guidance encoding. It gives the user the possibility to quickly locate areas of potential interest, as well as means for estimating their extent. The green arrow is an example of local guidance encoding. It links the current slider position to the visualized importance function.

4. Applications

The goal of this paper is to describe how data-sensitive navigation can be employed to address interaction issues in medical visualization. Therefore, we apply our concept in two interaction

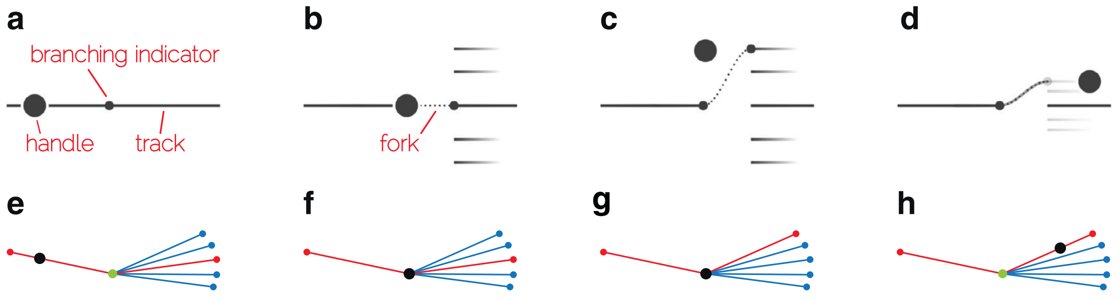


Fig. 4. Different states of a TreeSlider are shown in (a)–(d). (a) TreeSlider and its components, (b) If the handle is moved close to a point where the tree branches, a fork continuously appears. (c) If the handle is within the fork area, it can be moved vertically to choose other branches of the tree for traversal. (d) The traversal continues on the chosen path. In (e)–(h), the traversed tree is shown. The active path is depicted in red color, the green circle represents a branching node, while the black circle represents the current position set by the TreeSlider. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

use-cases typical in medical visualization. The first use case is the examination of vascular structures. It is an interaction task commonly performed by medical personnel. The second use case is a more general one, i.e., the rotation of 3D objects. This task is performed in situations where 3D visualization is applied, such as surgery planning.

4.1. Data-sensitive vessel traversal

The examination of vascular structures is an important topic in medical visualization [13,14,17]. Vascular pathologies such as stenoses, i.e., an abnormal narrowing of blood vessels, or vascular calcifications, i.e., mineral deposition inside a blood vessel, potentially leading to a blockage of the blood flow, are often small in relation to the acquired data. However, their examination is of critical importance. Since vascular structures usually consist of many branches, examining all of them simultaneously or sequentially is a cumbersome task. The vascular structures are modeled by a tree referred to as *vessel tree*.

4.1.1. TreeSlider

To free physicians from the laborious process of inspecting the entire vessel tree by repeatedly choosing a path and traversing it using a simple slider, we introduce a novel interaction element, called *TreeSlider*. TreeSlider allows users to specify a position within a tree structure. In case of blood vessel traversal, a slice of the underlying data perpendicular to the vessel tree at the position specified by the TreeSlider is taken and presented to the user. The perpendicular slices are calculated using *rotation minimizing frames* [21]. In this way, using TreeSlider it is possible to traverse the entire vessel tree without the necessity of traversing the same part of the tree multiple times.

In order to make the TreeSlider an efficient interaction element for traversing tree structures, it is required to:

- be scalable – the image-space dedicated to the TreeSlider has to be as small as possible even when traversing large trees,
- allow users to visit every part of the tree,
- support data-sensitive navigation.

To satisfy these requirements, we utilize the design of a classic slider, which consists of a single line called *track*, and a *handle* which is dragged along the track. The position of the handle represents the slider value, which is mapped to a position on a path between the root node and a specific leaf node of the vessel tree. Since only one path is traversed at a time, the screen-space of the slider does not have to account for all the tree branches at the same time, thus maintaining scalability, as defined in the first requirement. The components of the TreeSlider are shown in Fig. 4(a).

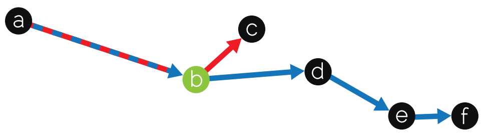


Fig. 5. A directed acyclic graph with two paths (marked blue and red). The relative positions of the fork node *b* on these two paths are not equal. Therefore, if the user changes between the paths, the position of the fork node *b* has to be remapped so that the respective branching indicator does not move during the switching of the paths. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

According to the second requirement, the user has to be able to visit every part of the tree. Therefore, the path mapped to the slider track can be changed through an element called *fork*. Forks are elements displaying all branches of the tree originating in a specific node. Forks are placed on the slider track at positions corresponding to these nodes where the tree branches (nodes with more than one child). Through these nodes, which we refer to as *fork nodes*, the currently traversed path can be changed. By default, the forks are invisible and their positions on the track are illustrated by dots called *branching indicators*.

If the handle is near a branching indicator, the fork appears. It consists of several parallel lines, each representing one alternative path where the user can go from the respective branching node. At this point, the handle can be moved vertically to switch between possible paths. This process is illustrated in Fig. 4(b) and (c). As the user moves the handle further along the track, the fork continuously collapses to bring the handle back to the track (Fig. 4(d)). Now the track maps to a different path within the tree. Fig. 4(e)–(h) shows how the traversed path is changed using the fork.

The problem with switching the paths is that the relative positions of the fork node within different paths may be different. The problem is illustrated in Fig. 5. Suppose the current path is the one marked in blue. If the handle reaches the fork node *b*, it is possible to switch the current path to the one marked in red. However, the position of the fork node *b* is much further down the red path than it is within the blue path.

This would result in discontinuities in the interaction with the TreeSlider. Therefore, when switching the paths through a fork node, its position within the new path is linearly remapped so that it matches the position in the previous path. This is achieved by scaling the portion of the TreeSlider track representing the path before the fork node and the portion after it. In this way, the respective branching indicator does not move during the path switching. As the user moves the handle away from the fork after switching the paths, the scaling factors of both parts of the TreeSlider track are continuously interpolated to the original values. In this way, the remapping is continuously reduced as the user

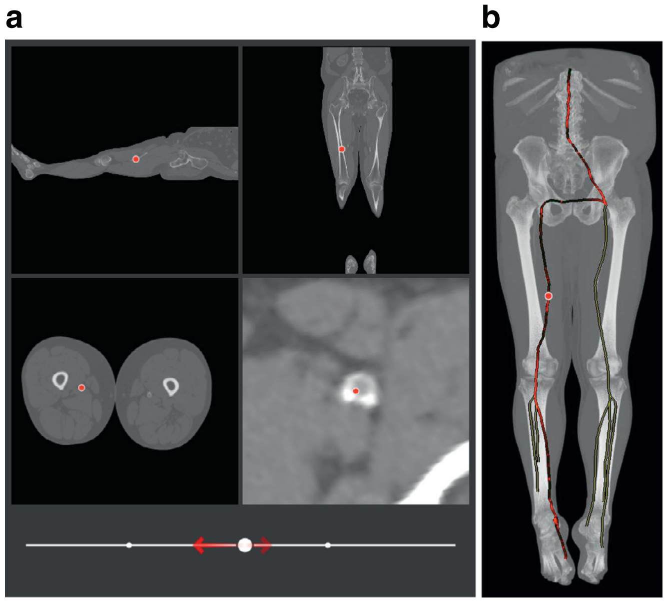


Fig. 6. Traversal of a vessel tree in a CTA scan using the TreeSlider. (a) Axial slices through the selected point (marked with red circles) and a slice perpendicular to the vessel tree at the current position are shown above the TreeSlider. (b) The vessel tree is overlaid on top of a 3D visualization of the dataset as an overview. All views in (a) and (b) are linked together. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is moving the handle. This allows users to seamlessly switch between different paths within the tree.

This mechanism might cause discontinuities in the interaction if two forks nodes are close to each other as it might be impossible to smoothly remove the remapping before the second fork is reached. Therefore, for multiple fork nodes, which are closer than a specified threshold, we only use a single fork. This fork contains all the branches of the concerned fork nodes. This is useful in scenarios such as blood vessel traversal, as vessel trees rarely contain nodes with more than two branches. However, several branching nodes may be close together. Using a single fork for these nodes is therefore an efficient option.

4.1.2. Data-sensitive TreeSlider

In order to make the interaction with the TreeSlider data-sensitive, we split the traversed tree into segments separated by the fork nodes. Since there is exactly one path between each pair of fork nodes, we can treat each segment in the same way as if it were a simple one-dimensional slider.

In order for the data-sensitive navigation to work, we need the function $d(P)$ specifying the importance for every position P on the tree. In our specific case, we use the difference of consecutive slices based on the *mean square* metric. The resolution of the TreeSlider is effectively increased in those areas where strong changes in the slices around the blood vessels occur.

Additionally, we also want to take the structure of the vessel tree into account. The most important structural features of the vessel tree are branchings, i.e., the fork nodes. Therefore, we increase the importance values for all points along the vessel tree which are close to a fork node. In this way, the resolution of the TreeSlider is also increased near branchings of the vessel tree. This demonstrates that complex data models can be used to construct the transformation T to realize data-sensitive navigation.

Fig. 6 shows our interface for vessel traversal using the TreeSlider. In Fig. 6(a), the TreeSlider with the locally encoded guidance (red arrows) is shown. The guidance indicates how sensitive the TreeSlider will be when moving the handle in the respective directions. Since the TreeSlider is used to continuously traverse the tree, locally encoded guidance is adequate.

Above the slider, four slices of the volume data are given. These are three axial slices passing through the slider position, and a slice perpendicular to the path in the vessel tree. The slice views are linked with a 3D overview visualization (Fig. 6(b)), where the entire vessel tree overlays a Maximum Intensity Projection (MIP) of the dataset. In each of these linked views, the slider position is depicted as a red dot. The 3D overview also contains globally encoded guidance information, where high importance along a blood vessel is shown in red, while low importance is indicated in black. In this way, the users immediately see areas of high importance.

4.1.3. TreeSlider evaluation

In order to gain an insight of how people interact with the TreeSlider, we conducted a usability experiment. The experiment was performed with nine participants, all of them Master and Ph.D. students in computer science familiar with medical volume data. Since the participants are not medical professionals, we designed the task to show whether the TreeSlider can be easily understood in a general use scenario, not necessarily linked to the medical domain.

The participants used the interface shown in Fig. 6. They could interact with the TreeSlider, while observing the cursor movement along the vessel tree. After a short explanation of the functionality of the TreeSlider, we asked them to perform the following task: *a blue dot is displayed at a random position on the vessel tree; use the TreeSlider to move the cursor along the vessel tree to the position of the blue dot. After the position of the dot is reached, the dot moves to a new random position and the task is repeated.*

We asked the subjects to always move the cursor to the position of the dot in the most efficient way, i.e., using the shortest possible path. This task tests the ability of the users to navigate through the tree with the TreeSlider in a desired manner.

For each run, we record the starting position, the position of the dot, and the path along which the cursor is moved. We determine the *optimality* of the recorded path, which we define as the percentage of the recorded path that overlaps with the shortest possible path calculated by the Dijkstra algorithm.

In total, the participants performed 232 iterations of the task. The average *optimality* of all recorded paths was 95%. This was mainly due to movements of the cursor around forks. This finding suggests that the TreeSlider can be efficiently used to traverse tree structures.

Additionally, we gathered informal user feedback from the test subjects. In general, they liked the concept of the TreeSlider. Some of them reported that it was enjoyable to interact with it. The subjects did not report any problems with the scaling of the TreeSlider track during branch switching. We attribute this to the fact that the linear remapping of the fork positions only temporally changes the sensitivity of the TreeSlider interaction and the users can compensate for it by a continuous movement of the handle.

Some concerns were raised regarding the mapping between a fork of the TreeSlider and a branching in the traversed tree. From the visual representation of the fork, it might not be immediately clear, which branch maps to which line within the fork and, hence, it has to be determined by trial-and-error. This is not a significant issue when traversing vessel trees, since the forks typically only consist of two to three branches. The problem could be solved by visual mapping, such as using different colors for different branches both in the TreeSlider and the underlying visualization.

However, in the medical scenario this might be unnecessary, since the TreeSlider is designed in a way that the user does not have to look at it during the interaction. This is due to the fact that it uses relative changes of the mouse input instead of absolute positions. As confirmed by a domain expert in radiology, this supports the usual workflow in radiology. It is important that the domain experts can keep their attention focused at the underlying visualization rather than the user interface while examining the data. The locally encoded guidance, which augments the TreeSlider, is mostly useful when users wish to continue the interaction after it was interrupted.

We asked our test subjects whether they were looking at the TreeSlider while interacting with it. Some of them realized that this is not necessary, even though we did not instruct them not to look at the TreeSlider while completing the task. This suggests that it is intuitive to use the TreeSlider without looking at it, making it suitable for the use in the medical domain.

Despite these findings, it is a point for future work to clearly indicate the mapping between the TreeSlider forks and the tree branchings in the visual representation. This will ensure that the TreeSlider can be used outside of the medical domain and to efficiently traverse trees with nodes of higher degrees.

4.2. Data-sensitive 3D object rotation

So far we have shown how data-sensitive navigation can be applied in a 2D interaction scenario. Such scenarios are often used in the diagnostic process.

However, there are applications for the use of 3D visualization in the medical domain as well. These include interdisciplinary communication, knowledge transfer, but also more specific tasks such as surgery planning. In these applications, more complex methods of interaction are usually required in order to generate desired views. Generally, these interaction methods require an intuitive control of several parameters at the same time, e.g., 3D camera settings.

Our model of data-sensitive navigation can be applied to interaction widgets, which control multiple parameters at the same time, and therefore support various tasks utilizing 3D visualization. To demonstrate this, we apply the model to *Arcball* rotation of 3D objects [22].

The data-sensitive navigation applied to the object rotation makes sense if we want to increase the precision of the rotation around interesting viewpoints, while keeping the rotation relatively fast around viewpoints without interesting features. In this particular example, we apply our model to a visualization of a simple CT scan of a human head, where there is a clear distinction between feature-rich and featureless areas (face and neck versus occipital and parietal bones). In this way, the principles of the data-sensitive navigation are demonstrated in a tangible way.

The assumption of applying the data-sensitive navigation is that there is a function $d(v)$ which evaluates the importance of the given viewpoint v. There are numerous viewpoint-evaluation techniques that can be used [23,24]. For our application, we employ a simple importance function, which evaluates the number of edges visible from each viewpoint. This is achieved by summing the pixel intensities of the image rendered from the given viewpoint processed by the Sobel operator. The assumption is that more edges indicate more visible features.

In Arcball rotation, the user input consists of a screen-space 2D input vector, usually provided as mouse input, which maps to a quaternion-represented rotation of the displayed 3D object. In order to apply data-sensitive navigation, we sample the importance function $d(v_c)$ at each possible viewpoint v_c, which can be obtained from all the possible input vectors sampled with a fixed resolution. In this way, we get a 2D importance map z of the sized dependent on the sampling resolution.

In order to apply data-sensitive manipulation, we cannot simply apply the transformation T to the input vector, since this might result in the discontinuities in the transformed input.

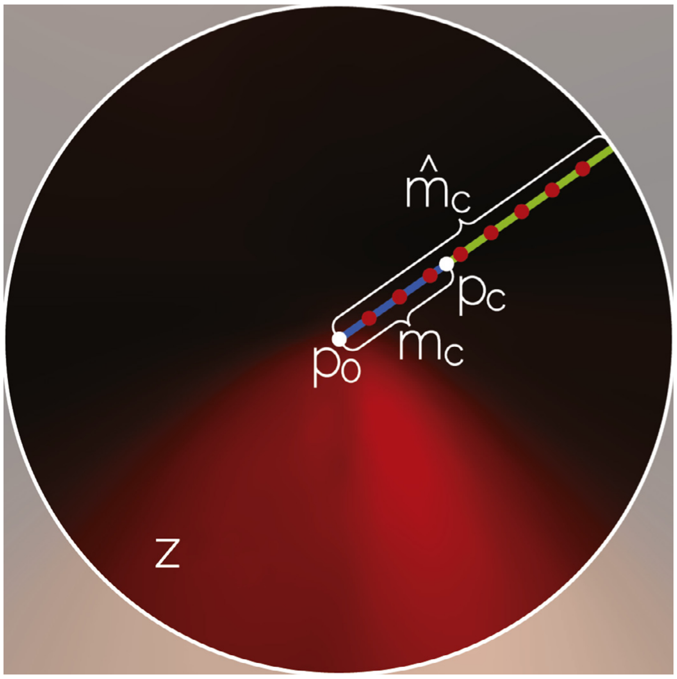


Fig. 7. Mouse input of the Arcball rotation. m_c (blue) is the input vector whose length is limited to the interval [0, 1]. $\hat{m}_c$ is the normalized vector m_c. p_0 is the original mouse position, p_c is the new mouse position. For every mouse movement, we can only consider importance values along the vector $\hat{m}_c$. In this way, the 2D mouse input is in each step reduced to a 1D problem as illustrated in Fig. 2. $\hat{m}_c$ represents the whole slider and p_c represents the current slider position. The red dots show where the importance map z (show in the background – black areas are least important, red areas are most important) is sampled in order to produce the importance function for 1D data-sensitive navigation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Therefore, we reduce the transformation of the Arcball input to a one-dimensional problem.

Fig. 7 shows how the 2D Arcball interaction can be reduced to a 1D remapping problem. $\hat{m}_c$ is the normalization of the input vector m_c. It represents the maximum length of the input vector in this direction. $\hat{m}_c$ can be viewed as a one-dimensional slider as shown in Fig. 2, where p_c is the current slider position. By sampling the map $z(m_c)$, which is transformed so that p_0 is in its middle, the importance along the slider is obtained. Now it is possible to transform the point p_c through data-sensitive manipulation as if it was a handle of a one-dimensional data-sensitive slider. In this way, the speed of the Arcball rotation is adjusted so that it is proportional to the perceived change of the rendered image by taking the underlying data into account. In this way, the rotation is made slower, and thus more precise, over areas of high importance.

In addition to data-sensitive manipulation, we apply both the global and the local encoding for data-sensitive guidance. The globally encoded guidance consists of a sphere placed in the lower right corner of the image, which we refer to as *navigation sphere*. The navigation sphere has the same orientation as the rendered objects. The surface of the navigation sphere encodes the importance for all viewpoints in color (black for least important, red for most important). Additionally, we overlay the importance map z onto the image whenever the user rotates the object. This encoding indicates where the user has to drag the mouse in order to display viewpoints with high importance. We apply a simple image-processing filter on the map z which increases the contrast of the image, helping the user to pick important regions. According to the specific application, this filter could be replaced by a different one,

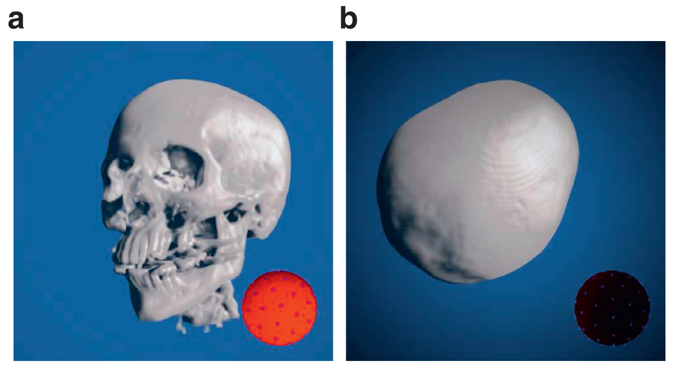


Fig. 8. Visualization of the importance using the navigation sphere and vignetting. (a) A feature-rich area is shown without vignetting. The front-facing part of the navigation sphere is red, indicating high importance. (b) If a featureless area is displayed, vignetting is introduced to indicate the low importance. It is also visible through the black color of the navigation sphere. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

e.g., an iso-contour detection filter to group regions of similar importance together.

For locally-encoded guidance, we employ a vignetting effect, whose strength is modulated by the importance of the current viewpoint. This means that the edges of the image get brighter if more important viewpoints are shown, and thus guide the user's attention. Both vignetting and the navigation sphere are shown in Fig. 8, while the importance map overlay is shown in Fig. 9.

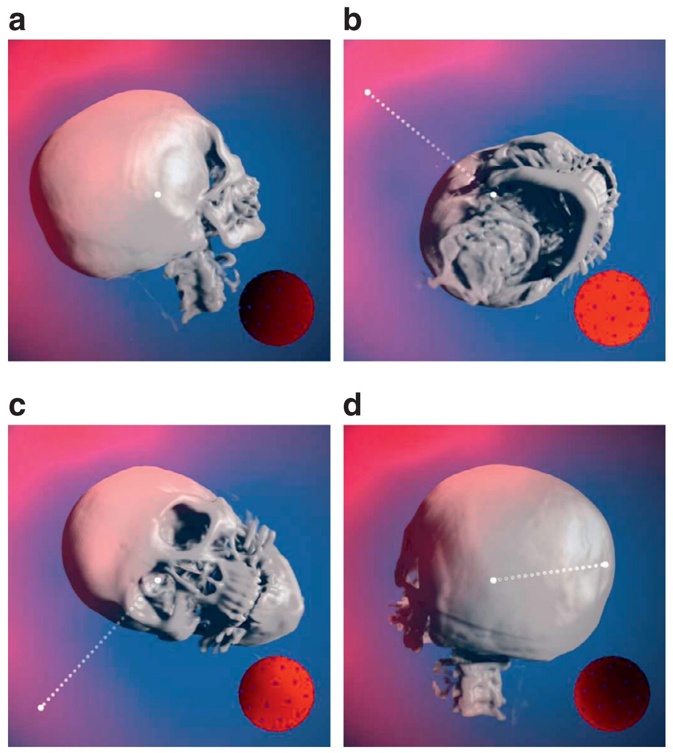


Fig. 9. Visual guidance for 3D object rotation. The white dotted line indicates the input vector for the Arcball rotation. (a) Initial position. The red color indicates where to move the mouse to rotate the object to a more interesting viewpoint. (b) The mouse was moved towards the red area to reveal interesting features around the neck and jaw. (c) The mouse was moved towards another red area to show the face. Since a large portion of the image comprises the featureless forehead, this red area is less pronounced. (d) The mouse moves towards an area of low interest denoted by the lack of red color. Featureless occipital and parietal bones are shown. The low importance is visible from the dark navigation sphere, as well as darker borders caused by the vignetting. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Discussion and limitations

The aim of our work is to enhance input elements used in medical visualization. Since the commonly employed elements are very diverse, we propose a model of data-sensitive navigation, which is independent of the user interface. The model describes how the underlying data can be utilized to make changes in the input proportional to the changes of the output, and how these changes can be visually encoded.

The computational overhead of applying our model is the evaluation of the importance function for the individual data samples. Therefore, it is highly dependent on the dataset size and the complexity of the importance function. In our examples, the overhead was in the order of milliseconds for the vessel exploration, and in the order of seconds for the 3D rotation, for medium-sized datasets (approx. 512^3 voxels). However, the importance function can be precalculated, which ensures that the overhead during the interaction is negligible.

In addition to the concept of data-sensitive navigation, we proposed the TreeSlider. It is a a novel interaction element implementing the data-sensitive navigation model, which we employ for the traversal of vascular structures.

Unlike our 3D rotation example, the TreeSlider extends the way how such widgets are commonly understood and interacted with, since the same visual element represents different tree branches. Therefore, we performed an experiment to help us understand how people interact with the TreeSlider.

The main limitation of the TreeSlider as a general-purpose input element is its inability to handle tree nodes of high degrees. Without making the screen-space of the TreeSlider wider, the too many branches originating in a single fork node might have to be displayed too close together in the fork, preventing users from easily switching between the branches. This limitation could be overcome by using the mouse wheel to switch between branches instead of using the vertical movement. However, this limitation does not apply in scenarios where degrees of the nodes are relatively low, such as in the traversal of large vessels in the peripheral vascular system.

We showed TreeSlider to a radiologist, who found the idea interesting and potentially useful. In the future, we are planning to conduct a thorough evaluation of the TreeSlider applied in radiology.

We realized the TreeSlider as a self-contained, open-source *Qt* widget. The source code is available for download.[1]

6. Conclusions

In this paper, we address the problem of unpredictable user input for medical visualization. To tackle the challenge of an efficient and intuitive interaction, we propose a general model, which can be used to improve navigation in visual-computing applications in general. The model consists of data-sensitive manipulation, a way of making changes in the user input and output proportional, and data-sensitive guidance, which displays the remapping information to steer the user interaction.

To showcase our model, we propose the TreeSlider, a novel interaction element for traversing tree structures. We demonstrate that both data-sensitive manipulation and guidance can be applied to the TreeSlider to efficiently traverse vascular structures. Additionally, we apply data-sensitive navigation to the rotation of 3D objects. This indicates, that our method is scalable to multidimensional input elements as well.

[1] https://cg.tuwien.ac.at/downloads/treeslider.

Although we only used medical-visualization examples, data-sensitive navigation seems applicable to many other areas as well, such as information visualization, visual analytics, and more generally visual computing.

Acknowledgments

The research presented in this paper was supported by the MetaVis project (#250133) funded by the Research Council of Norway, ManyViews, OeAD Scientific & Technological Agreement SK 14/2016, the Vienna Science and Technology Fund (WWTF) through project VRG11-010, and by EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.cag.2017.05.012.

References

[1] Lindow N, Baum D, Hege H-C. Perceptually linear parameter variations. Comput Graph Forum 2012;31(2pt4):535–44.
[2] Gavrilescu M, Malik MM, Gröller ME. Custom interface elements for improved parameter control in volume rendering. In: Proceedings of the Fourteenth International Conference on System Theory and Control; 2010. p. 219–24.
[3] van Wijk JJ, Nuij WAA. Smooth and efficient zooming and panning. In: Proceedings of the Ninth IEEE Conference on Information Visualization; 2003. p. 15–22.
[4] Blanch R, Guiard Y, Beaudouin-Lafon M. Semantic pointing: Improving target acquisition with control-display ratio adaptation. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 2004. p. 519–26.
[5] Elmqvist N, Fekete J-D. Semantic pointing for object picking in complex 3d environments. In: Proceedings of the 2008 Graphics Interface; 2008. p. 243–50.
[6] Chapuis O, Labrune J-B, Pietriga E. Dynaspot: speed-dependent area cursor. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09; 2009. p. 1391–400.
[7] Elmqvist N, Riche Y, Henry-Riche N, Fekete J-D. Mélange: Space folding for visual exploration. IEEE Trans Vis Comput Graph 2010;16(3):468–83.
[8] Ji G, Shen H-W. Dynamic view selection for time-varying volumes. IEEE Trans Vis Comput Graph 2006;12(5):1109–16.
[9] Kohlmann P, Bruckner S, Kanitsar A, Gröller ME. Livesync: deformed viewing spheres for knowledge-based navigation. IEEE Trans Vis Comput Graph 2007;13(6):1544–51.
[10] Kohlmann P, Bruckner S, Kanitsar A, Gröller ME. Livesync++: enhancements of an interaction metaphor. In: Proceedings of the 2008 Graphics Interface; 2008. p. 81–8.
[11] Wörner M, Ertl T. SmoothScroll: A Multi-scale, Multi-layer Slider. Springer Berlin Heidelberg; 2013. p. 142–54. ISBN 978-3-642-32350-8.
[12] Willett W, Heer J, Agrawala M. Scented widgets: improving navigation cues with embedded visualizations. IEEE Trans Vis Comput Graph (Proc InfoVis) 2007;13:1129–36.
[13] Kanitsar A, Fleischmann D, Wegenkittl R, Felkel P, Gröller ME. CPR – curved planar reformation. In: Proceedings of the 2002 IEEE Visualization Conference; 2002. p. 37–44.
[14] Auzinger T, Mistelbauer G, Baclija I, Schernthaner R, Köchl A, Wimmer M, et al. Vessel visualization using curved surface reformation. IEEE Trans Vis Comput Graph 2013;19(12):2858–67.
[15] Portugaller HR, Schoellnast H, Hausegger KA, Tiesenhausen K, Amann W, Berghold A. Multislice spiral CT angiography in peripheral arterial occlusive disease: a valuable tool in detecting significant arterial lumen narrowing? Eur Radiol 2004;14(9):1681–7.
[16] Borkin MA, Gajos KZ, Peters A, Mitsouras D, Melchionna S, Rybicki FJ, et al. Evaluation of artery visualizations for heart disease diagnosis. IEEE Trans Vis Comput Graph 2011;17(12):2479–88.
[17] Oeltze S, Preim B. Visualization of vasculature with convolution surfaces: method, validation and evaluation. IEEE Trans Med Imaging 2005;24(4):540–8.
[18] Wu J, Ma R, Ma X, Jia F, Hu Q. Curvature-dependent surface visualization of vascular structures. In: Proceedings of the 2010 Computerized Medical Imaging and Graphics; 2010. p. 651–8.
[19] Wu J, Hu Q, Ma X. Comparative study of surface modeling methods for vascular structures. In: Proceedings of the 2013 Computerized Medical Imaging and Graphics; 2013. p. 4–14.
[20] Straka M, Köchl A, Cervenansky M, Sramek M, Fleischmann D, Cruz AL, et al. The VesselGlyph: focus & context visualization in CT-angiography. In: Proceedings of the 2004 IEEE Visualization Conference; 2004. p. 385–92.
[21] Wang W, Jüttler B, Zheng D, Liu Y. Computation of rotation minimizing frames. ACM Trans Graph 2008;27(1)2:1–2:18.
[22] Shoemake K. ARCBALL: a user interface for specifying three-dimensional orientation using a mouse. In: Proceedings of the 1992 Conference on Graphics Interface. San Francisco, CA, USA; 1992. p. 151–6.
[23] Takahashi S, Fujishiro I, Takeshima Y, Nishita T. A feature-driven approach to locating optimal viewpoints for volume visualization. In: Proceedings of the 2005 IEEE Visualization Conference; 2005. p. 495–502.
[24] Vázquez P-P, Feixas M, Sbert M, Heidrich W. Viewpoint selection using viewpoint entropy. In: Proceedings of the 2001 Vision Modeling and Visualization Conference; 2001. p. 273–80.