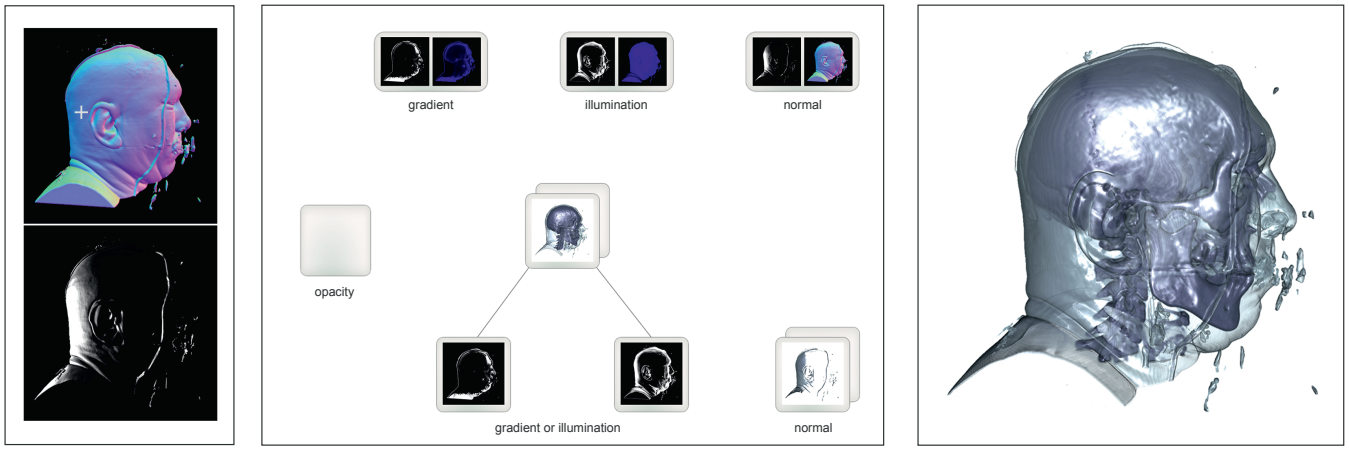
Semantics by Analogy for Illustrative Volume Visualization

Moritz Gerl, Peter Rautek, Tobias Isenberg, Eduard Gröller

Semantics by Analogy for Illustrative Volume Visualization

Moritz Gerl[a,*], Peter Rautek[c], Tobias Isenberg[a,b], Eduard Gröller[c]

[a] *Institute of Mathematics and Computing Science, University of Groningen, The Netherlands*
[b] *DIGITEO in collaboration with LIMSI–CNRS and AVIZ–INRIA, Orsay, France*
[c] *Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria*



Abstract

We present an interactive graphical approach for the explicit specification of semantics for volume visualization. This explicit and graphical specification of semantics for volumetric features allows us to visually assign meaning to both input and output parameters of the visualization mapping. This is in contrast to the implicit way of specifying semantics using transfer functions. In particular, we demonstrate how to realize a dynamic specification of semantics which allows to flexibly explore a wide range of mappings. Our approach is based on three concepts. First, we use semantic shader augmentation to automatically add rule-based rendering functionality to static visualization mappings in a shader program, while preserving the visual abstraction that the initial shader encodes. With this technique we extend recent developments that define a mapping between data attributes and visual attributes with rules, which are evaluated using fuzzy logic. Second, we let users define the semantics by analogy through brushing on renderings of the data attributes of interest. Third, the rules are specified graphically in an interface that provides visual clues for potential modifications. Together, the presented methods offer a high degree of freedom in the specification and exploration of rule-based mappings and avoid the limitations of a linguistic rule formulation.

Keywords: Volume rendering, illustrative visualization, semantic visualization mapping, shader augmentation

1. Introduction

The specification of meaningful mappings from multiple volumetric data attributes to visual attributes is a challenging problem in direct volume rendering. A common approach is to design multi-dimensional transfer functions, which are flexible but also complex and demanding to use. The design of multi-dimensional transfer functions requires expert knowledge and is often provided in suboptimal interfaces. To address this issue, alternative ways of defining the visualization mapping have been investigated. For instance, Rautek et al. [1, 2] present a semantics-driven visualization framework that enables users to specify a mapping from multiple data features to visual attributes as textually formulated rules. This method allows them to explicitly define semantics for attributes of interest. Their approach to parameter specification for volume rendering bears great potential for creating illustrative visualizations. For example, the data attribute *high density* can be mapped to the visual attribute *cartoonish shading* by formulating an according rule. Formulating the visualization rules textually, however, is rather rigid and provides few possibilities for exploring different mappings. Textual rule specification is limited by the ability of the user to mentally envision and assess the effects of potential rules beforehand, and with that devise the appropriate rules. Thus, exploration is rather limited, tedious, and time-consuming. Furthermore, using a function editor to define data semantics (e. g., *high density*) is a rather indirect way of parameter specification. This contrasts the direct specification offered by the semantic-layers concept.

*Corresponding author. Tel +31 50 3634602, Fax +31 50 3633800. SVCG, University of Groningen, P.O. Box 407, 9700 AK Groningen, The Netherlands.
Email address: gerl@cs.rug.nl (Moritz Gerl)
URL: http://www.cs.rug.nl/~gerl (Moritz Gerl)

To address these issues, we present a semantics-driven visualization technique that combines the advantages of a graphical specification of mappings with the illustration capabilities provided by a rule-based approach. The presented technique incorporates a new way of dynamically specifying the input and output of the visualization mapping, allows to define the mapping by analogy, and makes use of a graphical user interface for specifying visualization rules. This provides a more direct control of semantics and allows us to overcome the restrictions of a linguistic rule specification. Our goal is to provide a flexible and general tool for the specification and dynamic exploration of meaningful visualization mappings. We employ the following methods to achieve this goal:

Semantic shader augmentation: We present a technique for automatically augmenting an arbitrary shader program with semantics-driven rendering functionality. This technique replaces the visualization mapping present in an input shader with a rule-based rendering method, while preserving the visual abstraction the input shader originally generates. This semantic shader augmentation enables a dynamic definition of the input and output of the visualization mapping and extends the range of possible mappings. It permits users to rapidly create semantics-driven visualizations based on arbitrary variables in an arbitrary input shader program.

Semantics by analogy: We let users define the visualization mapping by analogy. Specifically, users define contributions of properties to the mapping through brushing on visualizations depicting properties of interest. The final result images are then visually analogous to the brushed data ranges. This brushing on visual data representations permits a direct, flexible, and interactive definition of a semantic mapping. Furthermore, it allows users to intuitively define semantic mappings of two- and three-dimensional data properties, which was not feasible in previous approaches. The brushing interaction improves the usability and directness of specifying semantics-driven mappings.

Graphical rule specification: We present a graphical interface for specifying visualization rules. With this interface, rules can be specified and modified by interacting with dedicated widgets. These widgets provide visual feedback on the semantic entities they represent, which allows to visually assess the effect of rules, or modifications to rules. The interface, therefore, makes use of people's visual information-processing capabilities in the rule specification process and opens up new possibilities for the exploration of semantics-driven mappings. It, thus, reduces the restrictions of a textual rule formulation.

Together, the described concepts form a technique for the interactive exploration of semantics-based visualizations (Fig. 1). It provides a novel way of parameter specification in direct volume rendering. Our technique is particularly well-suited for illustration purposes and can be used as a tool for domain experts to quickly create illustrations from volume data. For instance, in the visualization of medical volume data it allows users to generate case-specific illustrations with respect to the diagnostic purpose or the treatment to be illustrated. Furthermore, our interactive graphical approach to parameter specification is a step towards an illustrative volume rendering system which is suited for being used by scientific or medical illustrators.

We first outline related work in Section 2. Next, in Section 3 we describe our framework for semantics by analogy. In Fig. 3.3 we show and discuss exemplary results that are generated with our technique. In Section 5 we report on an evaluation of our technique. We then discuss its limitations in Section 6. Finally, we conclude the paper and describe possibilities for future work in Section 7.

2. Related Work

The work presented in this paper relates to research in rule-based visualization, selective application of visualization styles, multi-variate data visualization, graphical user interfaces for specifying visualization parameters, as well as to volume rendering using dynamic shader generation.

Early work on automated generation of illustrations based on rules was done by Seligmann and Feiner [3]. They present a system for the generation of intent-based illustrations using design rules. The text-to-scene method introduced by Coyne and Sproat [4] follows a similar approach. It enables the translation of simple semantics to images. Instead of a textual rule specification, we propose a graphical one in this paper. Along these lines, Svakhine et al. [5] use illustration motifs to gear visualizations towards the intended audience. Another semantics-based graphical interface for the specification of a multi-dimensional mapping is presented by Rezk-Salama et al. [6]. Our method differs from these approaches by introducing dynamically generated semantics into the mapping process and by enabling users to specify a mapping based on direct data representations.

The methods that we introduce in this paper build, in particular, upon techniques presented by Rautek et al. [1, 2]. They introduce semantic layers and interaction-dependent semantics which allow to interactively create illustrative visualizations based on textual rules. This is realized with the help of fuzzy logic. Attributes of interest are interpreted as fuzzy sets, whose membership functions describe the attributes' contributions to the mapping. Fuzzy-logic arithmetics is applied to evaluate the illustration rules. In this work, we propose methods that extend the flexibility and exploration capabilities offered by Rautek et al.'s system using a graphical and analogy-based approach.

The application of different visualization techniques for selected subsets of a volume was proposed by Hauser et al. [7] with two-level volume rendering. Other work in this direction [8, 9] further examines the selective application of styles and rendering attributes. Instead of an a-priori selection of volume subsets to map to different visualization techniques, we present a method for interactively exploring the selective application of visual attributes based on semantics.

In this work, we discuss a technique for specifying a mapping from a multi-variate data space to visual attributes. This relates to the notion of multi-dimensional transfer functions introduced by Kniss et al. [10]. We deviate from this concept by basing the mapping on explicitly defined semantics and by providing tools to define the behavior of these semantics interactively. Other approaches also rely on explicitly defined semantics. Both McCormick et al. [11] and Stockinger et al. [12] use mathematical expressions to formulate the visualization mapping. An-

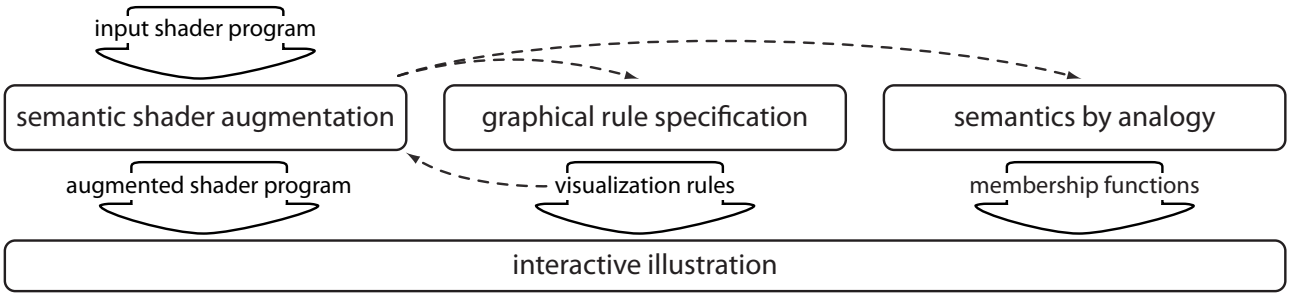


Figure 1: General schematic overview of our framework for semantics by analogy. Three components influence the resulting interactive illustration. The boxes in the center row show the components' main processes. The thick arrows leaving the processes represent the entities which influence the result. The dashed lines indicate the interrelations between the three components.

other rule-based system is presented by Sato et al. [13] who use rules to classify tissue structures in multi-modal datasets. We believe that rule-based visualization bears great potential, but identify formal and textual rule formulations as rather rigid and non-exploratory. For this reason, we combine rule-based rendering with the benefits of graphical user interfaces for defining multi-dimensional mappings. Tzeng et al. [14] present such an interface that permits the user to specify the input to a multi-dimensional data classifier via brushing. Another interface for exploring mappings of multi-dimensional data is presented by Zachow et al. [15]. They allow to brush values of interest in multiple linked abstract data representations to visually explore nasal airflow. All these systems have in common that the mapping is specified in an abstract data space. In contrast to this, our method allows a mapping specification with brushing on a more direct data representation, through visualizing attributes of interest. Apart from this, we also employ design galleries for parameter specification, as introduced by Marks et al. [16].

The technique presented here makes use of automatic shader generation methods. In this context, Rössler et al. [17] propose a technique for dynamically generating shader code for multi-volume raycasting from an abstract render graph. Similar to this approach, we exploit the flexibility and abstraction from GPU programming offered by dynamic shader generation. We differ from Rössler et al. [17] in realizing a rule-based approach and in the user interface for specifying visualization parameters.

3. A Framework for Semantics by Analogy

The overall concept of our framework for semantics by analogy is depicted schematically in Fig. 1. It consists of three components that influence the resulting interactive illustration. The input to our system is an initial shader program selected by the user. The *input shader program* is a combination of GLSL and special tags that surround variables of potential interest. The input shader is parsed and *automatically* augmented with rule-based rendering functionality by a pre-compiler (Section 3.1). This results in an *augmented shader program* as output that encodes the intended semantic mapping. For specifying semantic mappings on an abstract level, we provide a graphical user interface. In this interface, the way the data values are interpreted (i. e., the fuzzy membership function) is defined through analogy (Section 3.2). The mapping to a visual attribute is determined by rules which are graphically specified (Section 3.3).

A key component of our framework is the process of semantic shader augmentation, which is further illustrated in Fig. 2. The input shader in this example is a raycasting program that depicts a volume in a sparse rendering style. The code segment at the top stores the volume z-coordinate in the 'fCoordZ' variable. The segment at the bottom writes the level of sparseness to the 'fSparseness' variable. The level of sparseness is controlled by a single 'fParam' parameter. The sparseness parameter influences the saturation of the color, the usage of a diffuse and specular shading term, and the use of contours. The two variables are marked as a semantic property and as a visual attribute using dedicated tags. The semantic shader augmentation replaces the static mapping of sparseness with a rule-based mapping. The application of the sparse rendering style is then controlled by two visualization rules, which depend on the volume z-coordinate. To achieve this, the process of semantic shader augmentation injects new functionality into the *input shader program*. This results in an *augmented shader program* which is capable of rendering visualizations of semantic properties, and of evaluating semantic visualizations rules. The rules used in the shader augmentation, as well as the membership functions, can be interactively explored and specified in a graphical user interface (Fig. 3).

The described semantic shader augmentation allows to quickly derive a case-specific visualization system from an initial shader program in a collaborative session of a visualization expert and a domain expert or an illustrator, who is then provided with a graphical user interface for exploring and specifying semantic mappings on an abstract level. Together, these processes add more direct control to rule-based rendering and let users explore the mapping space flexibly and dynamically.

3.1. Semantic Shader Augmentation

Shader programs usually encode a certain visual abstraction [18, 19]. For example, a shader can perform illumination computations, can render contours, or perform illustrative volume rendering techniques. All of these techniques typically specify a static visualization mapping. In volume rendering, e. g., this mapping is modeled as a transfer function. By *automatically* augmenting the input shader, we replace the static visualization

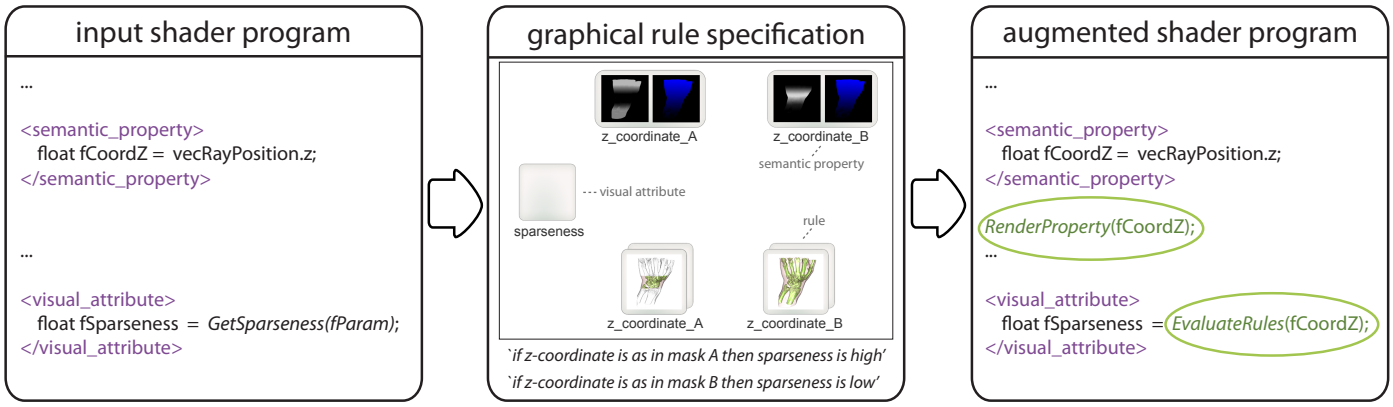


Figure 2: Overview of the concept of semantic shader augmentation. Arbitrary variables in an input shader program are defined as input and output to a semantic visualization mapping by adding dedicated tags to the shader code. The program is then automatically augmented with functionality which allows to specify contributions of input data-properties by analogy, as well as with functionality to realize a mapping of output visual-attributes that satisfies visualization rules.

mapping with a dynamic rule-based mapping, while the general visual abstraction such as 'illumination,' 'contour depiction,' or any illustrative volume-rendering technique is maintained.

We can work with arbitrary visual abstractions in our approach, but we continue to use the example of a GPU raycaster that generates images of different levels of sparseness. This program takes a volume texture as input, aggregates color samples along a ray, and outputs a resulting color value that depends on the pre-defined sparseness parameter. The color samples are computed by reading a scalar value from the volume, executing a transfer function look-up to map the scalar value to a color, and performing the sparseness computation. This means that input data-values (i. e., scalar values) are mapped to output visual-attributes (i. e., sparseness) in a static mapping. For automatically adding to this initial shader rule-based rendering functionality that is more flexible, we use variables in the shader code as input and output parameters of a dynamic mapping. This results in the possibility to apply the sparse rendering method in a selective and flexible way.

To realize this process we perform two *automated* steps: (1) we render images of the data properties so that users can define semantics by analogy and (2) we add functionality to evaluate fuzzy visualization rules. These steps relate to two distinct types of semantic entities. Step 1 refers to semantic data properties which form the input, i. e., the domain to our visualization mapping. For instance, these are properties such as the volume z-coordinate or the normal in the raycasting shader. The other type of semantic entities in Step 2 are visual attributes which represent the output, i. e., the co-domain of our mapping. Examples of these attributes are color, opacity, or parameters of sparseness or stylized shading.

The declaration of properties that form the input and output of the rule-based mapping is realized using dedicated tags in the shader code. An example is shown below:

```
<SemanticProperty name="z_coordinate"
                  variable="fCoordZ"
                  type="float">
   float fCoordZ = vecRayPosition.z;
</SemanticProperty>
```

The tag contains information about the semantic entity: a specifier, the name, and the type of the associated variable. Similar tags can be added for the visual attributes that are encoded in a shader program. By adding such tags to a number of variables of interest, users can declare which variables in the input shader are used as either semantic data properties or as visual attributes. This consequently enables users to dynamically explore the usage of different input and output parameters in the visualization rules specified later. In addition, by using a shader file that has been tagged previously, users can also work with pre-defined input and output features.

An application scenario of this shader augmentation is the rapid development of a case-specific visualization system in a collaborative session of a visualization expert and a domain expert or an illustrator. One case in this scenario is that the domain expert or illustrator has an a-priori understanding which data values or data derivatives are of interest for the desired visualization. In this case our method allows the visualization expert to quickly derive a suitable visualization system from the input shader, simply by adding tags to the respective variables in the shader code. Another case in this scenario is that the domain expert or illustrator has no a-priori understanding which data entities are of interest for the intended visualization. In this case our method provides the possibility to collaboratively identify semantic variables that can be used to achieve the desired results. Identifying and tagging suitable semantic variables can still be a rather time-consuming process, but it has to be performed only once. As soon as suitable variables are tagged by the visualization expert, the domain expert is provided with an abstract control for using these variables in rule-based mappings. This abstract control of the shader program is given by the graphical user interface described in Sections 3.2 and 3.3. Using this interface does not require any programming skills, while it provides the domain expert with a flexible control over GPU-based semantics-driven visualizations.

To augment the input shader automatically we create a copy of the shader file in main memory, extend it with the required functionality, and use the resulting code as a dynamically loaded shader program. This also means that the input-

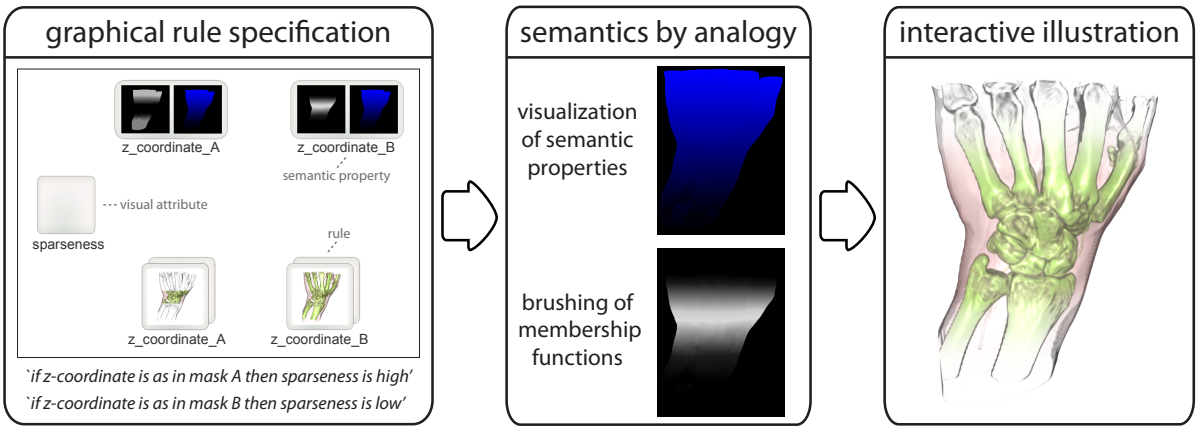


Figure 3: Conceptual overview of our technique for a graphical interaction with semantics-driven visualizations. Graphically specified visualization rules (left) make use of data semantics which are defined by analogy, i. e., by defining membership functions for semantic data properties via brushing on renderings of these properties (center). This technique allows a domain expert to create interactive illustrations (right) on an abstract level, while the underlying entities are automatically derived from an input shader program as illustrated in Fig. 2.

shader code can be edited during runtime. The code generation starts with parsing the source file to detect the tags that define variables as semantic entities. The required functionality is then inserted at the locations indicated by these tags. This dynamic augmentation is one of our key novelties to the original semantic-layers approach [1, 2].There, the rule-based rendering functionality is required to be hard-coded in a volume rendering program, and is thus present a-priori. This restricts the possibilities of quickly exploring different semantic mappings for different visualization techniques, because each of the techniques has to be made usable for the rule-based rendering individually.

The novelty of our semantic shader augmentation is not the dynamic code generation as such. It is the way we employ the well-known tool of dynamic shader generation to realize a dynamic specification of the mapping of arbitrary data values to arbitrary visual attributes based on illustration rules. Furthermore, our method for dynamic shader generation differs from the concepts of the *UberShader* [20] and the *SuperShader* [21]. Both of these concepts use pre-coded fragments of shader code that are combined to a new shader program on the fly. We, in contrast, do not use pre-coded program fragments. The code fragments which we inject into the shader are entirely generated on the fly. This means that our method can be used to augment any existing shader program with a minimal implementation overhead. The only code which has to be added to the input shader manually are the described tags, the remainder of the code is automatically generated. This approach drastically reduces the necessary implementation time for adding the required functionality to a given input shader, and also improves the code's readability.

In order to realize Step 1 of the augmentation, we automatically insert functionality into the shader that generates visualizations of semantic data properties and renders the result to a texture (e. g., Fig. 4 left). Here, we use one color channel of the output texture per dimension of the rendered data property: for one-dimensional variables we use the blue color channel, for three-dimensional variables we use all three channels. To generate the image we perform raycasting, composite the values that are assigned to the semantic variables during the execution of the input shader, and map the composited value to a visible color range. For the compositing, users can select one of three methods: averaging, maximum, or slicing. The semantic-property image in Fig. 4 to the left, for instance, displays the maximum density. To map the composited value to a visible data range, we multiply it with a user-adjustable factor. Although more advanced mappings could be used, we found the described simple ones to be sufficient for our purpose. The average and maximum compositing represent an extension to the semantic-layers approach. They map semantic properties from object-space to image-space. An application of these *image-space semantics* is demonstrated in Fig. 3.3.

In Step 2 of the augmentation, we automatically enhance the shader with the capability to evaluate a semantic rule base. For this purpose we implement the fuzzy-logic arithmetics as described by Rautek et al. [1, 2]. They use two types of membership functions, related to data properties and to visual attributes. Pre-defined functions are employed for both types. The membership functions of the visual attributes define the mapping from membership-function values of a data property to values of a parametrized visual attribute. In our approach, we work with a set of functions that are defined through a simple function editor for the visual attributes. For the semantic data properties, on the other hand, we use dynamic membership functions which are specified by analogy as described next.

3.2. Semantics by Analogy

Users of our system can define data semantics by brushing values of interest in the visualizations of data properties created by the semantic shader augmentation. Data semantics in this context refers to the way data values are mapped to visual attributes by means of membership functions of fuzzy sets. During brushing, these membership functions are adjusted according to the selected color values. This concept allows the user to see and directly assign a meaning to the data ranges of interest. The user can, therefore, visually draw conclusions about which data ranges are meaningful for the visualization he or she intends

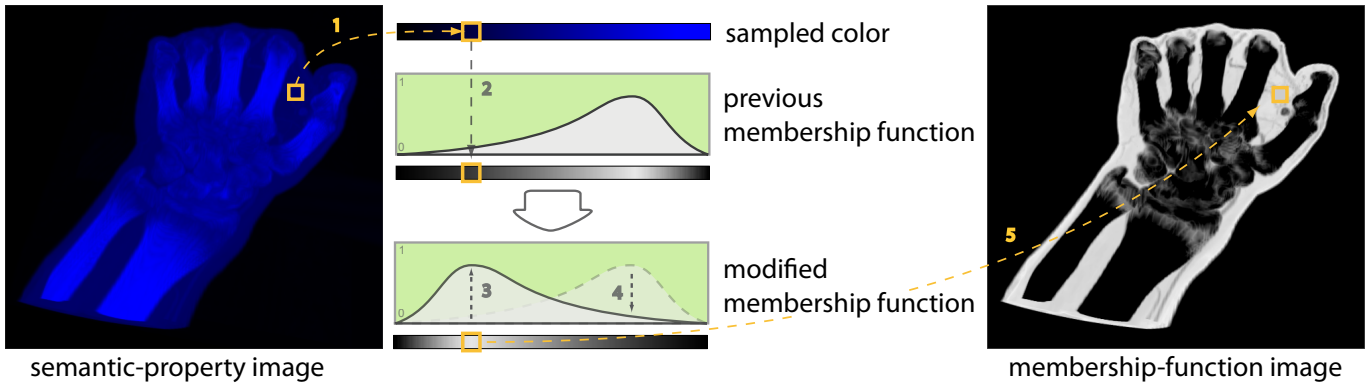


Figure 4: Brushing on a semantic-property image (left: maximum density image) to specify a membership function (center), and generating a membership-function image or mask (right). To brush into a membership function we sample the color at the cursor location (1) and map this color to the domain of the membership function (2). The previous membership function is incremented at the corresponding location (3), a normalization causes the modified membership function to decrease in other data ranges (4). To generate the membership-function image from the semantic-property image, we sample the membership-function value at the location corresponding to a pixel's color and write this value to the mask (5).

to achieve. In this way an appropriate mapping of these data ranges to visual attributes can be defined explicitly. For example, the user might be interested in areas of low density as in the example in Fig. 4. These are visible and can be directly selected in the maximum density image. This is what we denote as the specification of *semantics by analogy: the resulting visualization is visually analogous to the marked data ranges*. The specification by analogy gives a more direct control of the semantics than the use of membership functions specified with a function editor. It is, thus, the second key novelty that we propose as an extension to the semantic-layers technique [1, 2].

Further, in the original approach by Rautek et al. [1, 2] only 1D properties are used due to the complexity involved in specifying 2D or 3D membership functions with a function editor. In contrast to this, our approach of defining membership functions by brushing on a color image natively supports the definition of 2D and 3D membership functions. This facilitates the use of data derivatives such as the normal or the curvature direction in semantics-driven visualizations. Fig. 10 shows an example.

The membership functions are implemented as single-channel textures. The dimensionality of the textures is determined by the number of dimensions of the represented data properties. We here discuss the 1D case as illustrated in Fig. 4: a one-dimensional data attribute that is associated to a one-dimensional membership-function texture. The definition of 2D and 3D membership functions is implemented analogously by simply adding additional color channels to the process. The domain of a membership function represents the scalar values of a semantic property, which in our case are given by the color values in the semantic-property image. The co-domain of the mapping are the corresponding membership-function values in the range of $[0, 1]$. In our realization of the brushing mechanism we provide visual feedback about the current membership function through a gray value mask (Fig. 4, right). This membership-function image is generated by mapping the color values of the semantic-property image to the corresponding membership-function values (Step 5 in Fig. 4).

For brushing into the membership function, we treat the function as a histogram of the color values that are marked by the user. When the user brushes, we sample the color value at the cursor location in the source image (Step 1 in Fig. 4), map the sampled color value to the domain of the membership function (Step 2 in Fig. 4), and increment the membership-function value at this location (Step 3 in Fig. 4, modification through adding). We employ an adaptive brushing behavior that lets the membership function shift to the recently marked data range (Step 4 in Fig. 4). This adaptive selection permits users to rapidly explore different data ranges without having to deselect formerly marked ranges to switch the focus to the recently selected data range. For realizing the adaptive brushing, we apply a normalization of the membership function so that the area below the function equals to one. This ensures that the current membership function represents a normalized probability distribution of brushed color values. This means that the membership function adapts to the frequency of the brushed color values. As an alternative, we also provide a method to subtract from the membership function, which is implemented analogously to adding. Finally, we allow users to modify the membership function in a non-adaptive way. In this case, we skip the normalization step, so that brushing adds to or subtracts from the membership function at only one color range without affecting other ranges.

The described brushing mechanism is related to the concept of dual-domain interaction introduced by Kniss et al. [10]. This concept defines interactions which link the spatial domain of the resulting volume rendering with the domain of the transfer function. The brushing interaction presented here, in contrast, takes place in an intermediate domain.

3.3. Graphical Rule Specification

The semantic-property images enable users to dynamically specify a property's contribution to the visualization mapping by analogy. We also want to facilitate the direct and flexible construction of illustration rules that make use of these properties. We realize this control with a rule specification via a graphical user interface (Fig. 5, 8).

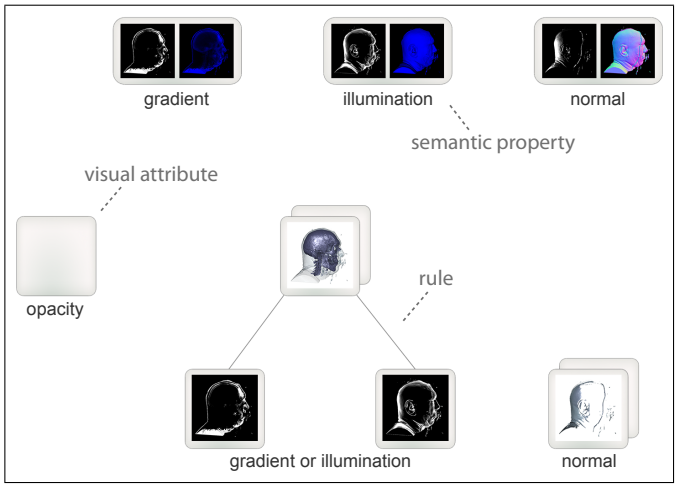


Figure 5: The interface for graphical rule specification. The widgets in the top row represent semantic data properties. They are used for switching between data properties, for creating rules, and for extending rules. Rules are depicted as expression trees in the bottom part. The nodes of the expression trees are interactive elements for combining and extending rules. The widget to the left represents a visual attribute. It serves for using this visual attribute in an illustration rule. Fig. 8 shows an interaction sequence.

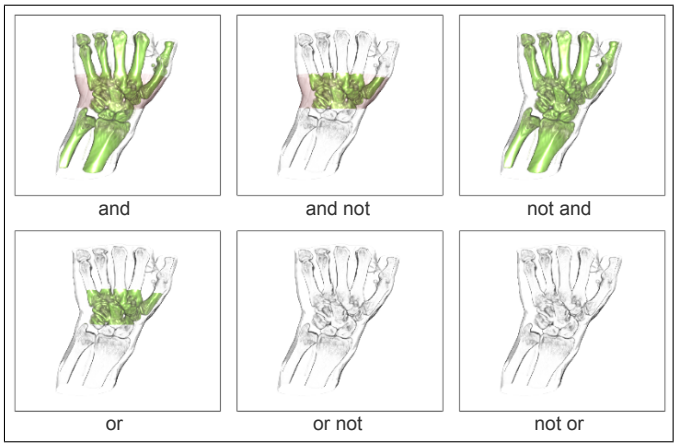


Figure 6: Design gallery with images of possible logical operations for extending the rule of the example shown in Fig. 9(f) with density as third property.

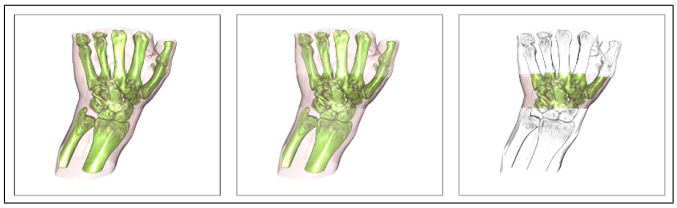


Figure 7: Design gallery with images of possible visual attribute membership functions for the example shown in Fig. 9(f).

The interface for rule specification shown in Fig. 5 contains three types of widgets which show preview images of the semantic entities they represent. These semantic entities are either a data property (top), a node of an expression tree (bottom), or a visual attribute (left). Fig. 8 shows an interaction sequence for creating a rule. The widgets located in the upper part of the interface represent the input data properties (Fig. 8(a)). Such a semantic-property widget contains both the semantic-property image as well as the membership-function image. By dragging a connection between two such widgets the user can graphically specify a rule based on the two respective properties (Fig. 8(b)). To allow users to select a logical operator that combines the two properties, we use a design gallery [16]. We render an array of result images for rules with different combinations of the three operators *AND*, *OR*, and *NOT*. The design gallery in the example in Fig. 6 shows result images for six different logical combinations of the expression *'z-coordinate is peripheral'* with the expression *'density is low'* (e. g., *'z-coordinate is peripheral AND density is low,' 'z-coordinate is peripheral OR density is low,' 'z-coordinate is peripheral OR density is NOT low,'* ...). The visual attribute used in the combined rule is *'sparseness is high.'* Using the *AND* operator in this example results in a sparse rendering of soft tissue in peripheral z-coordinate regions. Using the *OR* operator depicts all soft tissue as well as all peripheral z-coordinate regions with the sparse rendering method. By clicking on the desired result image, users can specify which operator they want to include in the rule (Fig. 8(c)). Once a rule is specified in the described way, a visual representation of the rule is created as an expression tree, located at the bottom part of the interface (Fig. 8(d)).

The nodes of this graphical expression tree are the second type of widgets we employ. Every leaf-node widget of the tree represents one semantic property, and displays the corresponding membership-function image. Every interior-node widget represents a rule or sub-rule and displays the result image generated by evaluating this rule. The expression-tree widgets are interactive elements for extending and combining rules. Drawing a connection from a semantic-property widget to an expression-tree widget allows users to extend the rule represented by the expression tree. The semantic property is included as an additional operand in the rule. By drawing a connection between two root nodes of different expression trees, the user combines the two affected rules to form a single rule.

The third type of widgets we use in our interface represents the visual attributes that can be included in rules. These widgets are located at the left part of the interface. They are used to assign a visual attribute to a given illustration rule by dragging a connection to the root node of an expression tree (Fig. 8(d)). A visual attribute can be associated with a set of pre-defined visual attribute membership functions. Each function describes a different mapping from an aggregated membership-function value to a value of the parametrized visual style. To select one of these visual attribute membership functions for a rule, we also use a design gallery (Fig. 7). We present the user with a set of result images generated by the current rule using different visual attribute membership functions. The user chooses the desired function by clicking on the corresponding preview image (Fig. 8(e)). Once a visual attribute is assigned to a rule in this way, we render an instance of the visual attribute widget and attach it to the root node of the rule's expression tree (Fig. 8(f)). The graphical rule specification is our third major extension to the original semantic-layers approach [1, 2].

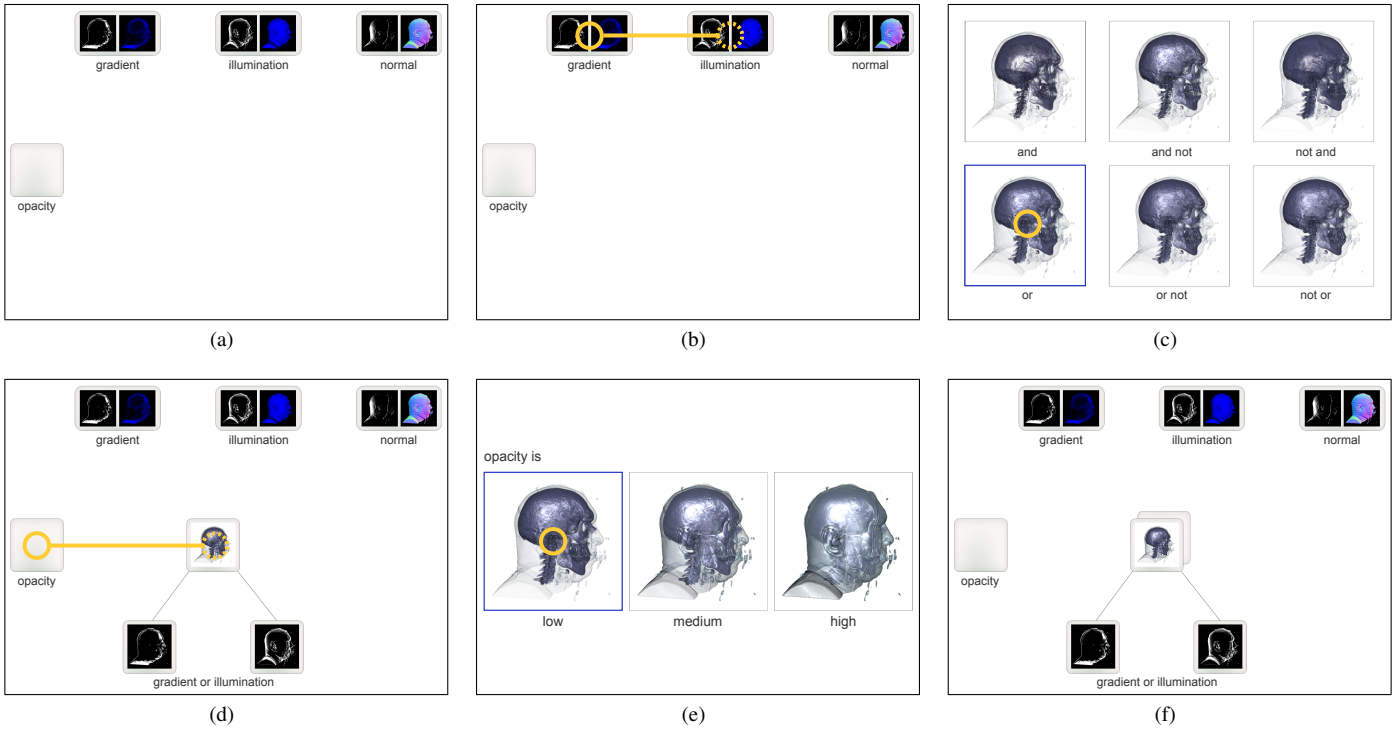


Figure 8: An interaction sequence for the graphical specification of an illustration rule. (a) The widgets located in the upper part of the interface represent the semantic properties. (b) A rule based on two properties is defined by dragging a connection between two such semantic-property widgets. (c) The interface switches to a design gallery. It depicts the results of combining the two chosen properties with different logical operators. A logical operator for the rule is selected by clicking on a preview image. (d) The newly created rule appears as an expression tree at the bottom part of the interface. Then, a visual attribute is assigned to the rule by dragging a connection from the visual-attribute widget on the left hand side to the root node of the expression tree. (e) A design gallery appears which allows the user to select a visual attribute membership function. (f) After the selection, the interface shows the complete visual representation of the specified rule.

4. Results and Discussion

In this section we show explanatory examples of the variety of visualizations that can be achieved with our approach. Fig. 9 demonstrates a simple case of semantic shader enhancement and the selective application of a rendering technique by analogy. In this example, we intend to highlight the central region of the dataset. As an input shader we use a volume renderer that produces images of different levels of sparseness. Fig. 9(a) shows a rendering for a high sparseness parameter, while Fig. 9(b) depicts a visualization for a low sparseness parameter. This shader is used as input to our system to demonstrate the easy enhancement of existing shaders. In the shader file we tag the data attribute *'z-coordinate'* as semantic property. This is the z-component in 3D texture coordinates as they are typically used in raycasting-based volume renderers. Furthermore, we tag the visual attribute *'sparseness'* that is specifically used in this shader. Interactively adding the two simple rules *'if z-coordinate is as in mask A then sparseness is high'* and *'if z-coordinate is as in mask B then sparseness is low'* results in an *automatic* augmentation of the shader. Now we define the regions that we intend to highlight by analogy, i. e., by brushing on the z-coordinate image shown in Fig. 9(c). For rendering peripheral z-coordinate regions with high sparseness, we brush the semantics according to the first rule to mask A in Fig. 9(d). This results in the image shown in Fig. 9(f). In order to render the central area with low sparseness, we add the data semantics for mask B in Fig. 9(e) by brushing on the central area of the z-coordinate. This results in the image shown in Fig. 9(g). This example demonstrates that the enhancement of the shader is simple yet flexible. For example, the smooth interpolation between regions of high sparseness and low sparseness in Fig. 9(g) is achieved with only two rules. In addition, defining semantics by analogy allows to directly apply the sparse rendering to intended regions.

In the example of Fig. 10 we show a simple case of deriving a focus-and-context renderer from an existing raycasting shader. We demonstrate how our approach supports the goal to selectively depict different structures in a dataset (here: bone and soft tissue) with different visual attributes. The selective application of these visual attributes shall be controllable based on semantic properties derived from the data (here: directional information). In order to achieve this goal, we take a simple shader program as input that renders the image with two different transfer functions (one for contextual rendering and one for focus rendering). We then use the two rules *'if average xz-gradient is as in mask A then rendering is contextual'* and *'if average xz-gradient is as in mask B then rendering is focused'*. Note that the keyword *'average'* is a further extension to the semantic-layers approach. It refers to a generalization of semantic properties from object-space to also include image-space properties. In object-space, the fuzzy logic evaluation is done on a per-sample basis while

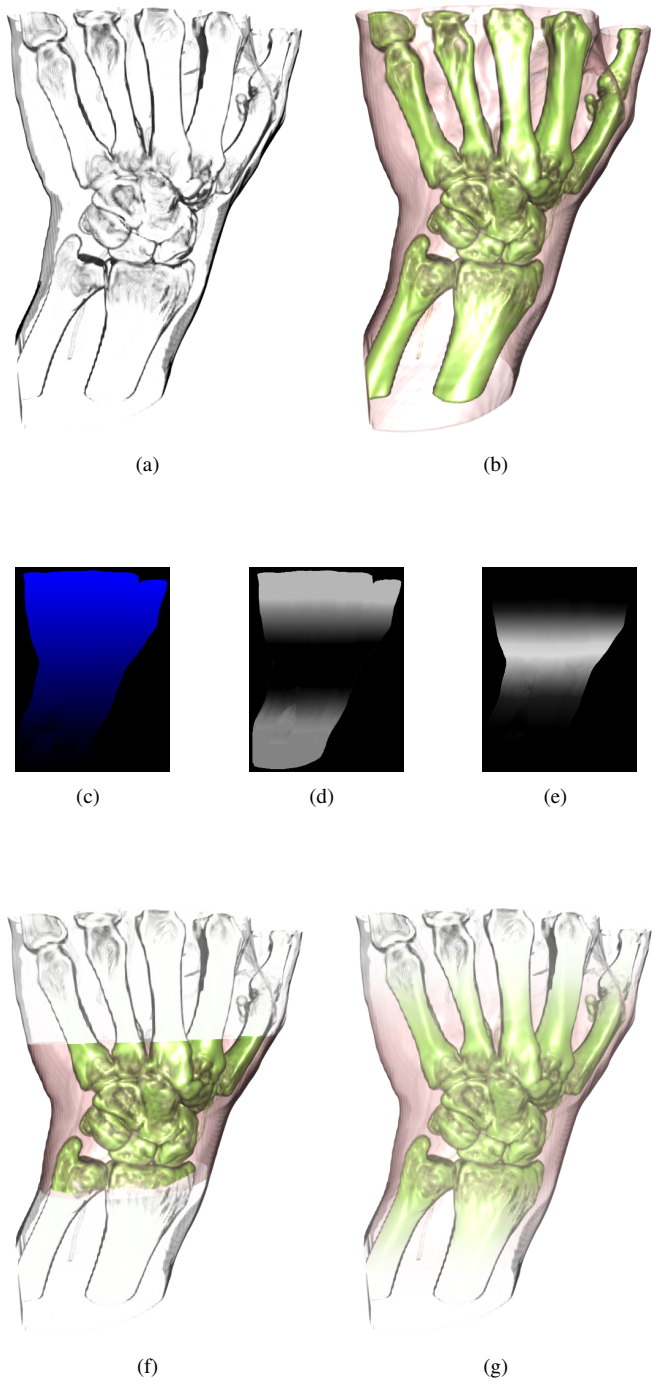


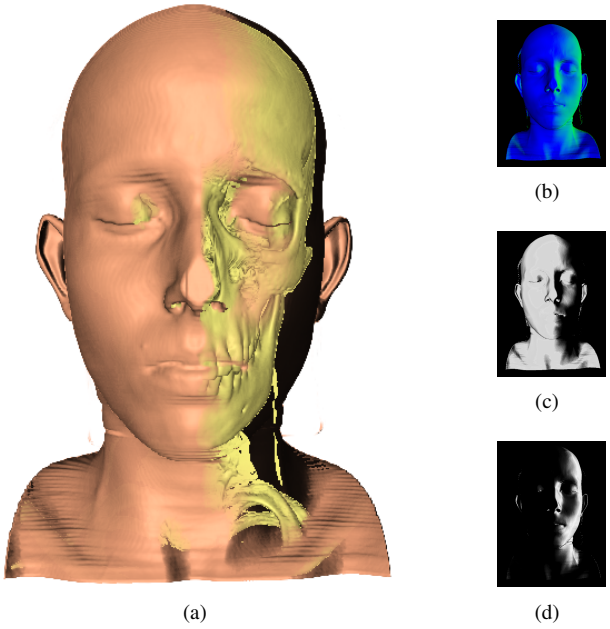
Figure 9: Augmenting an input shader that can produce images of different levels of sparseness. The top row shows renderings (a) for a high and (b) for a low sparseness parameter. In the augmented shader, we control the sparseness parameter with two masks that are defined on the volume z-coordinate. Brushing on the (c) z-coordinate image yields (d) mask A and (e) mask B. We achieve a smooth interpolation of sparseness by adding the rules (f) *'if z-coordinate is as in mask A then sparseness is high'* and (g) *'if z-coordinate is as in mask B then sparseness is low.'*

in image-space it is done on a per-ray basis. To demonstrate this generalization, we here use the image-space property *'average xz-gradient'*. Fig. 10(b) depicts the rendering of this semantic property. The *'average xz-gradient'* is the average of the gradient taken along the ray which is transformed to image-space. We use the x- and z-coordinates of this property to interpolate between the two rendering modes, i. e., context and focus visualization. This allows us to selectively apply the two different rendering modes based on the surface orientation. The two masks created to control the interpolation between the two modes are shown in Fig. 10(c) (mask A) and Fig. 10(d) (mask B). We use our method to apply the focus rendering on surfaces oriented towards the right as seen from the viewer.

The example in Fig. 10 shows a natural extension of the data semantics to a two-dimensional data property. In the original semantic-layers technique it is necessary to use one-dimensional data semantics. This is due to the complexity involved in the specification of multi-dimensional membership functions. With our approach, we can easily use the brushing on images of 2D and 3D properties to interactively and directly specify 2D and 3D membership functions.

In Fig. 11 we demonstrate the progressive adjustment of a visualization using the semantics-by-analogy user interface. In this example we use the *'diffuse illumination'* term of a regular raycasting shader as a semantic property. Our goal in the augmentation is to map the diffuse illumination term to the opacity of each sample. This mapping allows us to make regions that are highly illuminated more transparent. The two rules *'if diffuse illumination is low then opacity is high'* and *'if diffuse illumination is high then opacity is low'* are used to achieve this effect. This visualization mapping is similar to the context-preserving volume rendering technique by Bruckner et al. [22]. This demonstrates the ability of our approach to dynamically specify other (usually hard-coded) visualization techniques by the means of visualization rules. Fig. 11 shows the results (Fig. 11(a, c, e)) and the different masks that were brushed to generate these images (Fig. 11(b, d, f)). The brushing on the data semantics *'diffuse illumination is low'* as done in this example allows users to interact with the illustrative visualization and to adapt it to their needs and/or preferences.

The example in Fig. 12 further demonstrates the use of a semantics-driven opacity modulation. The goal of this visualization is to generate views on a skull-surrounded brain in an MR dataset of a human head. Raycasting of such a dataset is subject to occlusion of the brain by surrounding tissue. In order to cope with this occlusion, we apply our method to a standard raycaster to create see-through views on the brain. We achieve this by using the *'distance along the ray'* as a semantic property and the rule *'if distance along the ray is high then opacity is high.'* This setup allows a flexible application of opacity depending on the penetration depth. Similar to a clipping plane, it removes both the occluding skull and the brain tissue up to a specified penetration depth. This does not preserve the occluding brain tissue as a skull-removal scheme would do. In contrast to the static geometry of a clipping plane, however, this setup implements a dynamic semantics-driven clipping volume which can be interactively defined. The left column in Fig. 12 depicts the results of applying this rule from different viewpoints (Fig. 12(a, d, g)). The right column shows the corresponding semantic-property images (Fig. 12(b, e, h)) and masks (Fig. 12(c, f, i)) that are used to generate the visualizations.



(a) (b) (c) (d)

Figure 10: Controlling contextual and focus rendering modes via the property 'average xz-gradient.' (a) The result is generated from the two rules *'if average xz-gradient is as in mask A then rendering is contextual'* and *'if average xz-gradient is as in mask B then rendering is focused.'* The right column depicts (b) the *'average xz-gradient'* image as well as (c) mask A and (d) mask B.

All results were generated in interactive sessions. The performance of our system depends on the set of rules, the input shader, the membership functions, and parameters such as screen resolution and sampling distance. For all results shown in this paper and in the accompanying video we used a sampling distance of 0.5 voxels and achieved interactive frame rates on a dual core 3.2 GHz PC with a GeForce GTX 480 and 12 GB RAM. We used a 600×600 pixel viewport for the result image and for the rule-specification interface. The images shown in Fig. 9 were rendered at approximately 20 fps. The dataset in this example has a resolution of $122 \times 62 \times 128$ voxels. The example in Fig. 10 was rendered at approximately 15 fps, with a dataset resolution of $256 \times 256 \times 166$ voxels. For the results depicted in Fig. 11, we achieved a framerate of approximately 17 fps, rendering a volume of $128 \times 256 \times 256$ voxels. The framerate for rendering the images in Fig. 12 was approximately 8 fps, for a volume of $512 \times 512 \times 320$ voxels.

5. Evaluation

We evaluated the proposed concepts by gathering user feedback. We conducted two separate user evaluations with two different target audiences. The first evaluation targeted medical domain experts, while the second evaluation examined the usefulness of our system for medical illustrators. Both evaluations were qualitative assessments of the benefits and drawbacks of our method. We designed the evaluations as participatory observational studies combined with contextual interviews. We decided to use a qualitative evaluation methodology because we consider a quantitative evaluation as inappropriate for validating our system. This decision is based on the discussion about the role of evaluation in visualization and human-computer-interaction (HCI) research. In this discussion, Carpendale [23] advocates a more thoughtful application of a greater variety of research methodologies for evaluating information visualizations. She provides a survey of different evaluation methodologies and observes that quantitative methods can be prone to fault and to questionable validity for scenarios such as ours. Amongst other aspects, this is due to the fact that a quantitative experiment requires the rigorous control of many different factors, which are in fact difficult to control as a whole in situations like ours where complex and temporally long interactions are the essential elements of a system. Carpendale argues that a qualitative inquiry is more appropriate in such cases because it allows the researchers to consider the interplay among factors that influence visualizations, their development, and their use, and also to ground the studies in a more realistic setting. Along the same lines, Greenberg and Buxton [24] criticize the dogma of quantitative usability evaluation in HCI research. They stress the importance of choosing the appropriate evaluation methodology for a given research problem. They explain how the insistence on quantitative evaluation as a research methodology has fostered practices of weak science in HCI. Furthermore, Greenberg and Buxton [24] also emphasize that premature usability evaluation can eliminate promising ideas in an early design stage. They recommend to learn from the ways design worthiness is validated in other disciplines, i. e., from the design critique as used in industrial design. We follow their recommendation by selecting a qualitative evaluation of our system involving design critique principles. One may argue that multi-dimensional, in-depth, and long-term case studies [25] are best for evaluating creativity support tools such as ours. Although such an approach would be an appropriate method to validate our system quantitatively, the time required for such an evaluation approach, in particular on the side of the target users, makes it unfeasible at this point.

5.1. Feedback from Medical Experts

The participants in the first evaluation were a neuro-ophthalmologist in a first session and a group of eight radiologists in a second session. Each session comprised an initial demonstration and explanation of the system, a guided experimentation by the participants, and a concluding semi-structured interview. The feedback of the medical experts was positive in general. They quickly understood the interface and were able to use it after a short instruction. For the neuro-ophthalmologist we prepared an example with opacity modulation. This example was very similar to the one shown in Fig. 12, but we made use of different semantic properties. As semantic properties we used the scalar values, the distances from the object-space origin, and the volume texture coordinates. The neuro-ophthalmologist particularly liked the flexibility of our system. He used the opacity modulation to generate cut-away views, and found this procedure more flexible than the usage of cutting planes. He stated that he could imagine to employ a system like ours for intervention-procedure planning and for teaching. The radiologists in the second session particularly liked the abstraction from visualization internals offered by our system. They also

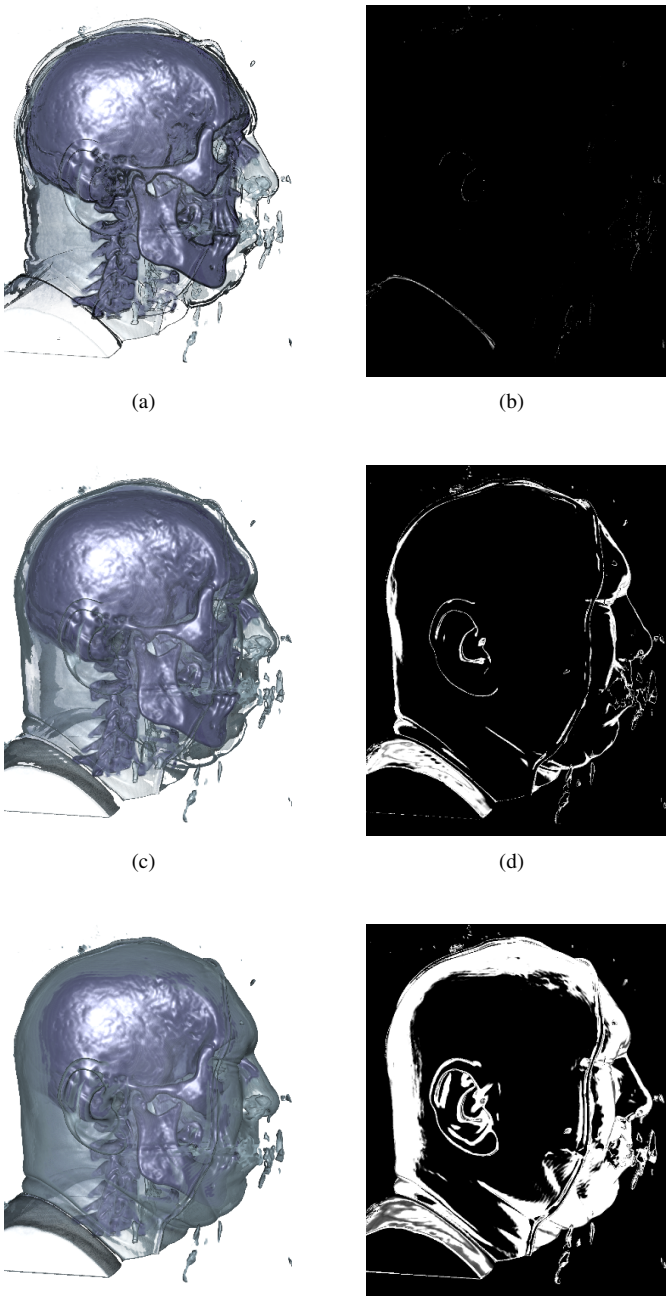


Figure 11: Using the semantics-by-analogy interface for an opacity modulation based on illumination intensity. We use the two rules *'if diffuse illumination is low then opacity is high'* and *'if diffuse illumination is high then opacity is low.'* The left column (a, c, e) shows different results that are generated with using the different masks in the right column (b, d, f) for the data semantics *'diffuse illumination is low.'*

commented positively on the ease of use of the graphical rule specification. They suggested various extensions, for example to include a graphical library of pre-defined semantic properties and visual attributes from which the user can choose. The discussion with the domain experts also revealed the benefit of the collaborative setting we described in Section 3. Semantic properties such as the *'xz-gradient'* or the *'diffuse illumination'* are non-trivial to formulate for most domain experts. But once such properties were visualized in our system and explained to the domain experts, they could easily understand and use them. On the other hand, it might be difficult for a visualization expert to figure out which data properties are useful for the domain experts. In our evaluation, it became obvious that a collaborative effort can help to find such properties.

5.2. Feedback from Medical Illustrators

The participants in the second evaluation were a graduate art student and two professional medical illustrators. The evaluation was conducted in three separate sessions. The art student is in a Master's program for interactive media and environments, and holds a Master's degree in graphics design as well as in fine arts. For simplification, we include her in the group of medical illustrators. The professional medical illustrators are both trained in fine arts as well as in medical illustration. One of them has several years of professional experience as a medical illustrator. The other has several years of professional experience as a photo retoucher and has been transitioning to medical illustration in the past seven years. None of the participants had noteworthy experience with volume data, but all had worked with renderings of polygonal 3D models. Thus, we started the sessions of this evaluation with a brief introduction to volume rendering. Apart from that, we applied the same methodology as in the evaluation with medical domain experts. We found out that some of the proposed concepts might indeed be of use for the target audience of medical illustrators. The participants in this evaluation could understand and use the interface quickly. Again, the feedback was positive in general. At the same time, the evaluation revealed some limitations of our system, which provide inspiration for future work. Interestingly, the feedback was highly congruent between the three participants.

The art student particularly liked the fact that when using our system, she could transfer knowledge and skills from graphics editing software. An example are our membership-function images which resemble the usage of masks in Adobe Photoshop. She stated that the visual feedback provided by the semantic-property images makes it easier to decide on the application of rendering styles. She also liked the feature of working in the intermediate domain of the semantic-property images, while the entire result is updated simultaneously. She stated that she clearly favors our graphical rule definition over a textual one.

The first medical illustrator gave clear confirmation that for her needs and preferences, the graphical rule specification is by far superior to a textual rule formulation. She liked our graphical approach to parameter specification, and stated that it would better meet the demands of "visually oriented" persons than other more programming-oriented interfaces. She stated that she would use our proposed concepts in her work as a medical illustrator, given that they were integrated in a comprehensive system for illustrative volume rendering. Apart from that, she was very interested in direct volume rendering in general. She was particularly fascinated by the accuracy it provides. She emphasized that accuracy is crucial for medical illustration. She commonly uses a 3D atlas of polygonal models as a master to create illustrations. She stated that these atlases—in contrast

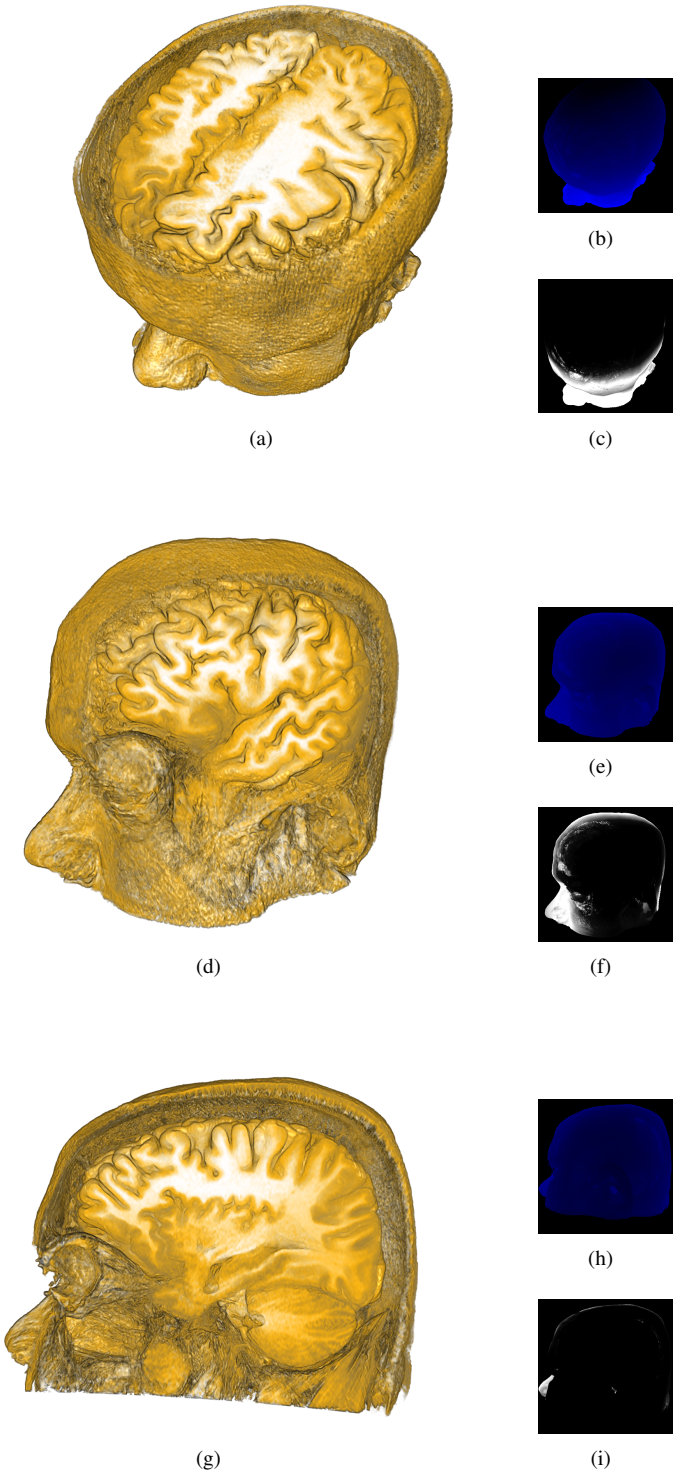


Figure 12: Generating views on a skull-surrounded brain in an MR head dataset. A semantics-driven opacity modulation generates see-through views on the brain. We use the rule *'if distance along the ray is high then opacity is high.'* This allows to apply opacity depending on the ray penetration depth, resulting in an interactive semantics-driven clipping volume. Similar to a clipping plane, this clipping volume removes all brain and skull tissue up to a certain ray position. It does not preserve the occluding brain tissue. The left column depicts results from different viewpoints (a, d, g). The right column shows the corresponding semantic-property images (b, e, h) and masks (c, f, i).

to direct volume visualization—would lack a certain degree of precision because they involve the human factor of the artist creating the 3D models.

The second medical illustrator also revealed this keen interest in volume rendering as a reference for manually creating illustrations. Regarding our system, she also shared many of the views of the other medical illustrator. She confirmed that she clearly favors a graphical rule definition over a textual rule formulation. She liked our use of masks, which allowed her to transfer skills from Adobe Photoshop. She also stated that a system like ours could be of use for her work as a medical illustrator. She also commented very positively on the notion of using brushing as an interaction to modify the rendering, although she was not convinced by the benefit of brushing on the semantic-property images instead of the result image.

All medical illustrators gave rise to the question if it would be preferable to brush on the result image instead of the semantic-property images. They stated that they are used to work on either the image itself, or on a representation which is visually completely unrelated to the image, such as dialog windows. The art student liked the idea of working in an intermediate domain, but the two medical illustrators considered this more critically. They asked for the motivation of this design choice and said that they would be more familiar with brushing directly on the result image. They were not convinced that the visual feedback provided by the semantic-property images is crucial for obtaining the desired results. One of the medical illustrators even felt confused by the additional viewport needed for displaying and brushing on the semantic-property image. For her, it led to a confusion about which interaction has to be performed on which viewport. But both illustrators were convinced that with some training, they would get familiar with brushing in the intermediate domain. However, it would need a long-term evaluation to gain more insight about this learning process.

It is interesting that, in contrast to the medical illustrators, the domain experts did not raise the issue of brushing directly on the result image. They perfectly accepted the brushing on visualizations of data properties. In our opinion, this reflects that the group of medical domain experts is used to approaching the creation of an image in a data-oriented way. They are familiar with thinking of an image as the mapping of data. The illustrators, on the other hand, appeared to approach the creation of an image in a less data-oriented way—maybe because illustration based on real data has not been possible to a large degree until recently. Illustrators seemed to think more in terms of modifying the image directly, instead of modifying data or a mapping which underly the image. However, we assume that, with training, the illustrators would be able to transfer their creative skills to the world of data-oriented image creation.

6. Limitations

The one limitation identified in both of the two user evaluations was that the brushing mechanism can occasionally generate unexpected results. The brushing is influenced by a smoothing of the membership function, which we apply for broadening the

range of marked data values. When the corresponding smoothing parameter is set to a high value, one can easily mark large data ranges. This turned out to have the negative side-effect of unintentionally selecting a too wide range when users intended to select a narrow data range. On the other hand, the creation of smooth volumetric masks is not possible without sufficient smoothing of the membership functions. Without smoothing, only one specific data value can be selected at a time, which leads to the creation of speckled masks. Our brushing mechanism, thus, involves a trade-off between precision and execution time. With execution time, we refer to the time needed for the interaction of defining a desired mask, not the involved computation time. This trade-off implies the drawback that our brushing interaction is prone to working in an either too precise and slow or a too imprecise and fast way.

Another limitation of our method is the robustness of creating the semantic-property images, which form the basis for brushing. The final result is highly dependent on the semantic-property images. These images, in turn, are challenging to create for the general case. Especially for volumetric properties that are not related to iso-surfaces, it is hard to generate renderings which are well understandable and usable as a basis for brushing. In our examples we use either 2D or 3D properties that are directly related to iso-surfaces. In addition to this issue, the image-space selection of 2D and 3D data values can be problematic. Here, the maximum compositing can not be applied, and component-wise averaging results in introducing vectors which are not present in the dataset. With proper smoothing of the membership function, averages of 2D and 3D variables can still be used. We exemplified this in Fig. 10 with using the *'xz-gradient.'* But this smoothing limits the possibility of performing exact selections.

Working with image-space properties such as in Fig. 10 can also result in unexpected behavior. The domain experts here assumed to work with an object-space property and expected the mask to 'stick' to the dataset instead of following image-space directions. However, this unexpected behavior is due to the fact that the participants worked with the system only for a short time. We assume that such problems would be resolved quickly once that users become more acquainted with the system.

Another drawback of our technique is also related to the image-space selection of volumetric data values. The projection of scalar values to image-space requires compositing. When used in combination with a per-sample evaluation of the visualization rules in object-space, the composited scalar value can unintentionally differ from the per-sample scalar value. This then results in a discrepancy between the membership-function image and the final visualization. This discrepancy can be seen in the examples in Fig. 12, where the visualizations do not entirely match the membership-function images. The problem can only be avoided with a proper selection of semantic properties and rules, as well as with appropriate parameter tuning. This restricts the generality of our technique.

Furthermore, a general problem of our interactive graphical approach is the reproducibility of the results. Because interaction is required to specify the masks, it is hard to exactly reproduce results created earlier. This circumstance might let users perceive our system as being unreliable. However, the addition of the possibility to save membership functions and a comprehensive undo functionality may reduce the problem.

Finally, our technique does not address the problem of disjunct expert domains, although it addresses the challenge of semantics-driven parameter specification. A limitation of our approach is that the semantics we make use of do not originate from the problem domain (e. g., medicine), but rather from the solution domain (computer science). This implies that a domain expert using our system is likely not familiar with the semantics with which we provide him or her. The only aid we give for understanding the semantics are the semantic-property images. For example, a medical expert might have difficulties using the property *'distance along the ray.'* He or she might benefit much more from being able to use application-specific semantics such as *'lesion'* or *'tumor'* to steer visualizations.

7. Conclusions and Future Work

In summary, we propose methods that improve the semantic-layers approach by Rautek et al. [1, 2] in several essential ways. First, we introduce a semantic shader augmentation that increases the flexibility of the semantic-layers approach. It makes it possible to automatically *augment semantic shaders* at runtime. This concept can be employed, e. g., to enable a visualization expert to quickly derive a case-specific visualization system from an initial shader program according to the requirements of a domain expert. Second, we introduce the *semantics-by-analogy* approach. It allows users to brush properties to ease the process of defining and exploring data semantics. Third, we describe a user interface for the quick specification and exploration of fuzzy-logic rules. These interactive tools provide domain experts with a direct control of semantics-driven visualizations which abstracts from programming internals. Finally, we extend the semantic-layers method by introducing *image-space semantics*. These are incorporated in the semantic shader augmentation by using the keywords 'maximum' or 'average' in the specification of visualization rules.

One direction for future research is to address the above described problem of disjunct expert domains. It would be interesting to examine how our approach can be extended in order to permit the use of application-specific semantics.

Furthermore, the concepts presented here can be modified to allow users to brush directly on the result image instead of the semantic-property images. This interaction was also suggested by the medical illustrators in our second user evaluation. Many of the described limitations are arguments in favor of this idea.

Apart from this, our approach makes progress towards a semantic markup of the whole volume rendering pipeline. The explicit specification of semantics in the volume visualization pipeline permits us to expose the system's underlying semantics to the user. This allows the user to directly interact with the semantics of the volume visualization process and, hence, to obtain a more direct control and a better understanding.

Although our approach is very flexible, the initial manual tagging of shader files is still time-consuming. However, this process only has to be done once and each tagged shader can be

re-used for further datasets or visualization problems without further tagging. Moreover, we plan to explore more flexible approaches in the future that allow the browsing of a shader file while automatically getting suggestions for data semantics. We believe that the user experience can be greatly improved with such a browsing extension. Further, we think that our approach can be extended with a more general shader markup language. Currently, we only support the markup of semantic properties and visual attributes. A more general solution would allow us to tag resources (such as volumes and textures) and parameters that are (or shall be) exposed in the user interface. With the extension of the shader markup language, a more general visualization system could be realized. The markup of shader files would be sufficient for the rapid generation of new semantics-driven visualization systems.

8. Acknowledgments

We express our gratitude to the anonymous reviewers for their help on improving this paper. We also thank the participants of our user evaluations for sharing their time and expertise. This work has been partially funded by the ViMaL project supported by the Austrian Science Fund (FWF), grant no. P21695.

References

[1] Rautek, P., Bruckner, S., Gröller, M.E.. Semantic Layers for Illustrative Volume Rendering. IEEE Transactions on Visualization and Computer Graphics 2007;13(6):1336–1343. doi> 10.1109/TVCG.2007.70591

[2] Rautek, P., Bruckner, S., Gröller, M.E.. Interaction-Dependent Semantics for Illustrative Volume Rendering. Computer Graphics Forum 2008;27(3):847–854. doi> 10.1111/j.1467-8659.2008.01216.x

[3] Seligmann, D.D., Feiner, S.K.. Automated Generation of Intent-Based 3D Illustrations. ACM SIGGRAPH Computer Graphics 1991;25(4):123–132. doi> 10.1145/127719.122732

[4] Coyne, B., Sproat, R.. WordsEye: An Automatic Text-to-Scene Conversion System. In: Proc. SIGGRAPH. New York: ACM; 2001, p. 487–496. doi> 10.1145/383259.383316

[5] Svakhine, N., Ebert, D.S., Stredney, D.. Illustration Motifs for Effective Medical Volume Illustration. IEEE Computer Graphics and Applications 2005;25(3):31–39. doi> 10.1109/MCG.2005.60

[6] Rezk-Salama, C., Keller, M., Kohlmann, P.. High-Level User Interfaces for Transfer Function Design with Semantics. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):1021–1028. doi> 10.1109/TVCG.2006.148

[7] Hauser, H., Mroz, L., Bischi, G.I., Gröller, M.E.. Two-Level Volume Rendering. IEEE Transactions on Visualization and Computer Graphics 2001;7(3):242–252. doi> 10.1109/2945.942692

[8] Bruckner, S., Gröller, M.E.. VolumeShop: An Interactive System for Direct Volume Illustration. In: Proc. Visualization. Los Alamitos: IEEE Computer Society; 2005, p. 671–678. doi> 10.1109/VIS.2005.135

[9] Lum, E.B., Ma, K.L.. Lighting Transfer Functions Using Gradient Aligned Sampling. In: Proc. Visualization. Los Alamitos: IEEE Computer Society; 2004, p. 289–296. doi> 10.1109/VISUAL.2004.64

[10] Kniss, J., Kindlmann, G., Hansen, C.. Multidimensional Transfer Functions for Interactive Volume Rendering. IEEE Transactions on Visualization and Computer Graphics 2002;8(3):270–285. doi> 10.1109/TVCG.2002.1021579

[11] McCormick, P., Inman, J., Ahrens, J., Hansen, C., Roth, G.. Scout: A Hardware-Accelerated System for Quantitatively Driven Visualization and Analysis. In: Proc. Visualization. Los Alamitos: IEEE Computer Society; 2004, p. 171–178. doi> 10.1109/VISUAL.2004.95

[12] Stockinger, K., Shalf, J., Wu, K., Bethel, E.W.. Query-Driven Visualization of Large Data Sets. In: Proc. Visualization. Los Alamitos: IEEE Computer Society; 2005, p. 167–174. doi> 10.1109/VIS.2005.84

[13] Sato, Y., Westin, C.F., Bhalerao, A., Nakajima, S., Shiraga, N., Tamura, S., et al. Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering. IEEE Transactions on Visualization and Computer Graphics 2000;6(2):160–180. doi> 10.1109/2945.856997

[14] Tzeng, F.Y., Lum, E.B., Ma, K.L.. An Intelligent System Approach to Higher-Dimensional Classification of Volume Data. IEEE Transactions on Visualization and Computer Graphics 2005;11(3):273–284. doi> 10.1109/TVCG.2005.38

[15] Zachow, S., Muigg, P., Hildebrandt, T., Doleisch, H., Hege, H.C.. Visual Exploration of Nasal Airflow. IEEE Transactions on Visualization and Computer Graphics 2009;15(6):1407–1414. doi> 10.1109/TVCG.2009.198

[16] Marks, J., Andalman, B., Beardsley, P.A., Freeman, W., Gibson, S., Hodgins, J., et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In: Proc. SIGGRAPH. New York: ACM; 1997, p. 389–400. doi> 10.1145/258734.258887

[17] Rössler, F., Botchen, R.P., Ertl, T.. Dynamic Shader Generation for GPU-Based Multi-Volume Ray Casting. IEEE Computer Graphics and Applications 2008;28(5):66–77. doi> 10.1109/MCG.2008.96

[18] Rautek, P., Bruckner, S., Gröller, M.E., Viola, I.. Illustrative Visualization: New Technology or Useless Tautology? ACM SIGGRAPH Computer Graphics 2008;42(3):4:1–4:8. doi> 10.1145/1408626.1408633

[19] Rautek, P., Bruckner, S., Gröller, M.E., Viola, I.. Visual Abstractions and Interaction Metaphors for Knowledge Assisted Volume Visualization. In: IEEE VisWeek Workshop on Knowledge-Assisted Visualization. 2008, extended abstract.

[20] Hargreaves, S.. Generating Shaders from HLSL Fragments. In: Engel, W., editor. ShaderX3: Advanced rendering with DirectX and OpenGL; chap. 7.3. 2005, p. 555–568.

[21] McGuire, M.. The SuperShader. In: Engel, W., editor. Shader X4: Advanced Rendering Techniques; chap. 8.1. 2006, p. 485–489.

[22] Bruckner, S., Grimm, S., Kanitsar, A., Gröller, M.E.. Illustrative Context-Preserving Exploration of Volume Data. IEEE Transactions on Visualization and Computer Graphics 2006;12(6):1559–1569. doi> 10.1109/TVCG.2006.96

[23] Carpendale, S.. Evaluating Information Visualizations. In: Information Visualization: Human-Centered Issues and Perspectives; vol. 4950/2008 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg; 2008, p. 19–45. doi> 10.1007/978-3-540-70956-5_2

[24] Greenberg, S., Buxton, B.. Usability Evaluation Considered Harmful (Some of the Time). In: Proc. SIGCHI. New York: ACM; 2008, p. 111–120. doi> 10.1145/1357054.1357074

[25] Shneiderman, B., Plaisant, C.. Strategies for Evaluating Information Visualization Tools: Multi-Dimensional In-Depth Long-Term Case Studies. In: Proc. BELIV 2006. New York: ACM; 2006,. doi> 10.1145/1168149.1168158