Institut für Computergraphik und Algorithmen
Technische Universität Wien
Karlsplatz 13/186/2
A-1040 Wien
AUSTRIA
Tel: +43 (1) 58801-18601
Fax: +43 (1) 58801-18698

Institute of Computer Graphics and Algorithms
Vienna University of Technology
email:
technical-report@cg.tuwien.ac.at
other services:
http://www.cg.tuwien.ac.at/
ftp://ftp.cg.tuwien.ac.at/

TECHNICAL REPORT

Semantics Driven Illustrative Rendering

Peter Rautek, Stefan Bruckner, and M. Eduard Gröller

TR-186-2-07-10
October 2007

Keywords: Illustrative Visualization, Focus+Context Techniques, Volume Visualization

Semantics Driven Illustrative Rendering

Peter Rautek, Stefan Bruckner, and M. Eduard Gröller

October 5, 2007

Abstract

Illustrative visualization aims to effectively communicate a specific information. In traditional illustration the choice of appropriate styles and rendering techniques is guided by the intention of the artist. For illustrative volume visualizations it is difficult to specify the mapping between the 3D data and the visual representation.

We introduce a layer between data and visual appearance that allows the semantic specification of the visualization mapping. Our framework enables a linguistic formulation of rendering styles and techniques. A fuzzy logic rule base determines the constraints of the resulting interactive illustration. The fuzzy logic is automatically translated into a shader program and is evaluated directly on the GPU. The framework provides high quality illustrations that are guided by the specification of semantic illustration rules. The evaluation of the fuzzy logic on the GPU allows user-guided and view-dependent changes of the resulting illustrations. We show examples of semantics driven illustrative volume rendering at interactive frame rates.

1 Introduction

Illustrations are depictions that aim to intuitively explain topics of interest. The illustration is typically non-photorealistic and guided by its intention. Two illustrations showing the same object but with different intention might look completely different. The intention of an illustration defines the way it is drawn and which features of the underlying object are shown. Using the appropriate styles and depicting the features of interest accurately is the most important task of scientific illustration. Interesting regions are drawn to become salient features in the final image. Regions of less interest are drawn using more subtle styles or are even overdrawn by more important regions. The visual appearance of features in the final rendering is guided by the semantics that describe the intention of the illustration.

Medical doctors for instance use simple illustrations for the purpose of patient briefing. The illustrations describe a specific diagnosis, the future treatment of diseases, or a planned surgical intervention. In the optimal case patients are shown illustrations that are consistent with their anatomy and the special instance of their disease. However, hand drawn (traditional) illustrations are elaborate pieces of art usually done in a very time consuming way. Therefore it is impossible to create high quality hand-drawn illustrations for each patient.

One goal of illustrative medical visualization is to produce patient specific illustrations derived from measured data. CT-scans or MRI-scans provide measurements of the patients anatomy and can be used to automatically generate illustrations. The illustrations are dependent on the intention and therefore constrained by the diagnosis, the treatment, or the possible surgical intervention. *The illustration must be guided by the underlying semantics.* Similar to medical illustrations, semantic driven illustrations are used in various other expert areas as well (e.g., architecture, biology, machine construction, etc.).

In this paper we demonstrate the concept of semantics driven illustrations on volumetric (mainly medical) data. However, the presented concept can be adapted to other areas as well. The semantics that guide the illustration emanate from different sources. We differentiate between high-level semantics (expert semantics), and low-level semantics (data semantics, illustration semantics, and user-driven semantics). Expert semantics as the most obvious kind of semantics originate from the specialization field of the expert. The purpose of a CT-scan, which parts of the body are scanned as well as the diagnosis are examples for known semantics that can be used in the automated generation of illustrations. Data semantics are depend on the available data. CT-scans, for example, provide information on tissue densities. Different ranges of density correspond to semantically meaningful tissues (like air, soft tissue, bone, metal, etc.). Illustration semantics originate from the area of traditional illustrators. Illustrators use specific

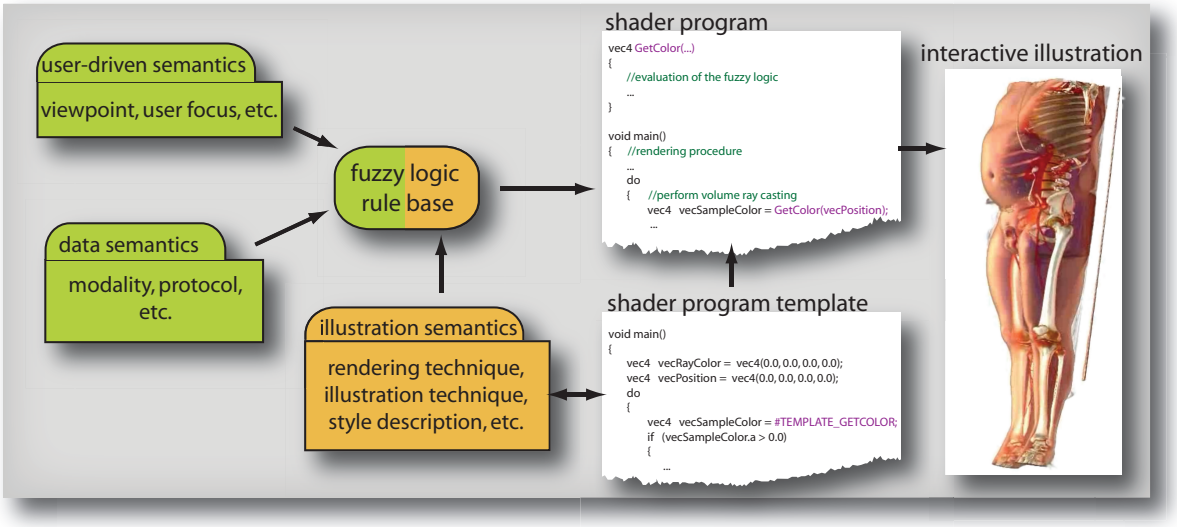

Figure 1: Outline of the presented semantics driven rendering framework: The different kinds of low-level semantics guide the generation of the interactive illustration.

semantics for the description of styles and of rendering techniques. Examples include:

- The way regions of interest are emphasized and the remaining features are drawn to provide context.
- The description of rendering styles.
- The way spatial relations are emphasized or to the contrary ignored to depict occluded structures.

User-driven semantics originate from the interactive illustration itself. Examples are:

- The direction the depicted object is viewed from.
- The distance between features and the image plane.
- The region of user focus (e.g., position of the mouse cursor).

These user-driven parameters taken be used to interactively change the illustration. The user interacts with semantically meaningful parameters that affect the appearance of the depicted object.

In Figure 1 an outline of semantics driven illustrative rendering is shown. The central component in our system is a fuzzy logic rule base. The rules are guided by the different kinds of semantics. The *if* part of the rule states constraints using data semantics and user-driven semantics. The *then* part of the rules describes the consequences for the illustration using illustration semantics. A shader program template is chosen according to the intended illustration technique. The rule base is translated into shader code to complete the shader program template. The result is an interactive semantics driven illustration. The user explores the depicted object interacting with semantically meaningful parameters.

Expert semantics as high-level semantics are not shown in Figure 1. The fuzzy rule base for one specific instance of an illustration application is affected by the low-level semantics. The expert semantics define the specific case itself and therefore completely determine the set of rules that are applicable. In this paper we assume that expert semantics are known for the given use case. We concentrate on the technical issues of illustrative rendering that is driven by low-level semantics. In earlier work [RBG07] we introduced the use of data and illustration semantics for illustrative volume rendering. In this paper we extend the previously presented framework with the following main contributions:

User-driven semantics: We introduce user-driven semantics for the generation of interactive illustrations. Interactive parameters are taken to derive user-driven semantics. For example, adjustable slicing planes, the mouse cursor, and view-dependent parameters are manipulated interactively. Semantics like *distance to the mouse cursor*, *distance to the slicing plane*, *distance to the image plane* can be used to influence the illustration. With user-driven semantics it is possible to change rendering styles and the opacity of features automatically and in a viewpoint-dependent manner, resulting in an interactive illustration.

GPU based evaluation of the fuzzy logic pipeline: In this paper we describe a GPU based implementation of the fuzzy logic component. All steps in the fuzzy logic reasoning are evaluated directly on the GPU. The shader program is automatically generated according to the fuzzy logic rule base and adapted each time the rule base changes.

Flat rendering mode: Illustrators often ignore spatial relations and draw layers with more important features on top of other layers. This results in a flat depiction of the important features on top of more contextual regions. In this paper we present the flat rendering mode that uses the concept of prioritized layers. Layers of high priority overdraw layers of lower priority resembling the above described illustration technique.

Presentation of a general rendering concept: We use automatically generated shader programs. The programs are generated using shader program templates. The templates provide the rendering code. Placeholders are used in the shader program templates and are automatically substituted depending on the given fuzzy logic. The automatic shader program generation is independent of the rendering template. Different shader program templates can be used to generate shader programs for volume rendering as well as for surface rendering.

The remainder of the paper is structured as follows: In Section 2 we briefly review the related work. In Section 3 we give an brief overview of the implemented system. In Section 4 we explain the evaluation of rendering attributes using fuzzy logic. This process is carried out on the GPU. We give details on the implementation of the GPU based fuzzy logic. The evaluated rendering attributes define the visual appearance of the depicted object. Image generation employs those rendering attributes that have been evaluated with fuzzy logic. The process of rendering is described in Section 5.

2 Related Work

The presented approach is a general rendering concept that translates low-level semantics into images. A mapping from data semantics and user-driven semantics to illustration semantics is specified using fuzzy rules. Because of the wide variety of diverse illustration techniques that can be achieved with this framework extensive related work exists. However, the strength of our approach is the linguistic definition of the different illustration techniques within a uniform framework.

Earlier work dealing with the automatic generation of imagery from semantics was done by Seligmann and Feiner [SF91]. In their system they use design rules to achieve intent-based 3D illustrations of geometric objects. The work of Coyne and Sproat [CS01] follows a similar idea. Their *text-to-scene* approach translates simple semantics into images. Svakhine et al. [SES05] use illustration motifs to adjust the illustrations to the intended audience.

Hauser et al. [HMBG01] introduce two-level volume rendering that allows the combination of multiple methods in one final rendering. Based on volume attributes other previous work [BG05, LM04] showed the selective application of specific styles and rendering attributes. For the parameterization of rendering styles we use an approach that is based on the work of Sloan et al. [SMGG01]. They present a technique to render pre-defined artistic styles.

A mapping from multi-dimensional attributes to visual appearance was shown by Kniss et al. [KKH02]. They present a method based on multi-dimensional transfer functions. Further, the quantification of statistical measures of multiple fuzzy segmentation volumes was shown in related work [KUS*05]. The formulation of a mapping from attributes to visual appearance using mathematical expressions was shown in the work of McCormick et al. [MIA*04] as well as Stockinger et al. [SSBW05]. Set operators and numerical operators were used by Woodring and Shen [WS06] to compare multi-variate as well as time-varying data. Sato et al. [SWB*00] use classification rules to identify tissue structures in multi-modal data. Viola et al. [VKG04] present importance-driven volume rendering that is conceptually similar to our flat rendering mode. However, our focus lies on the semantic specification of the importance. Krüger et al. [KSW06] show a technique for the visualization of hotspots. Our system allows similar results with the introduction of user-driven semantics.

3 Semantics Driven Rendering System

The idea of semantics driven rendering makes use of the semantics that accompany the process from acquiring data to drawing an illustration. We use the semantics in a fuzzy rule base. Rules employ data semantics such as *"if density is high then ..."* or *"if diffusion is low then..."*. Other rules employ illustration semantics such as *"if curvature is high then..."* or *"if feature is focus then..."*. User-driven semantics are represented in the rule base by rules like *"if distance to slicing plane is low then..."*. The rules can further use

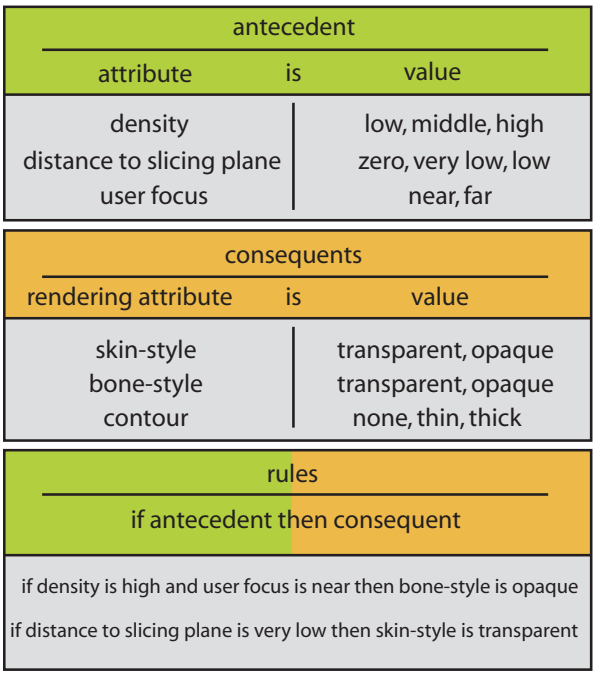

Figure 2: Fuzzy logic rules consist of antecedents and consequents. The antecedents (i.e., the *if* part) state a condition that has implications on the consequent (i.e., the *then* part). The gray boxes show examples for logical components that can be used in the antecedents, examples for consequents and exemplary rules that can be used in the rule base of our system.

any logical combination of the above mentioned semantics such as *"if distance to slicing plane is low and density is high then..."*. The *if*-part of rules is called the antecedent, the *then*-part is called the consequent. The consequent describes the consequences for illustrative styles if the antecedent of a rule is true. In fuzzy logic the antecedent of a rule is not simply true or false but it can also have any transitional value in between. The consequent in our system describes the resulting rendering attributes, like *"...then bone-style is transparent"* or *"...then contours are thick"*. Figure 2 shows the structure of fuzzy rules. Antecedents consist of logical combinations of semantic values for attributes. The consequent consists of a list of semantic values for rendering attributes. The gray boxes in Figure 2 show corresponding examples for antecedents, consequents and fuzzy logic rules.

The rendering styles are parameterized. Each parameter is evaluated separately using all fuzzy logic rules that have consequences for the parameter. The antecedents of the rules are evaluated describing to which degree a rule is *true*. Implication, aggregation and defuzzyfication are the remaining steps that are performed to derive a value in the interval 0..1 for each rendering attribute.

It is a challenging task to implement a volume rendering system that evaluates fuzzy rules per sample to determine rendering attributes interactively. Modern CPUs are not capable of evaluating fuzzy logic rules for a 3D volume several times per second. On the other hand modern GPUs do not offer the flexibility to fully implement a fuzzy logic system. Our implementation makes use of the flexibility of CPUs and the processing capabilities of modern GPUs.

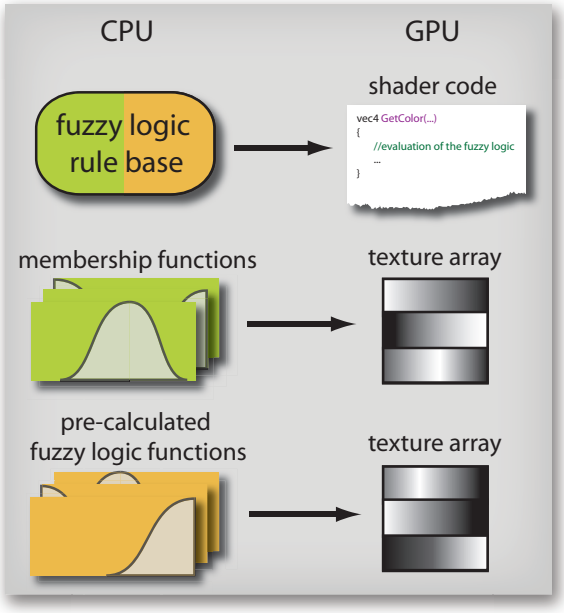

Figure 3: System overview: The fuzzy logic rule base is parsed and translated into shader code on the CPU. Membership functions and pre-calculated fuzzy logic functions are encoded in 1D texture arrays. The GPU makes use of the generated shader program and the texture arrays to perform interactive semantics driven illustrative rendering.

In Figure 3 an overview of the components of the system is shown. The rule base is translated into shader code on the CPU. The shader code is used to generate a shader program for volume rendering performed on the GPU. The membership functions as well as pre-calculated fuzzy logic functions are stored in 1D texture arrays that are used on the GPU to efficiently evaluate these functions.

The realized framework provides a user interface for the design of styles, the construction of membership functions, the definition of the interactive attributes, and for the input of the rules. The shader program has to be

adapted for each change of the rule base. The generation of the shader code and the update of the shader program is the most costly operation in our system. However it has only to be done if the rules change and does not take longer than a second. Changes in membership functions result in an update of the corresponding texture arrays. The update of the texture arrays is faster than the update of the rule base and can be done interactively.

In Section 4 we give details about the generation of shader code, the pre-calculation of fuzzy logic functions, and the fuzzy logic evaluation on the GPU.

4 Fuzzy Logic on the GPU

Our framework parses the fuzzy logic rules and generates appropriate shader code for the fuzzyfication, fuzzy logic operations, aggregation, implication and defuzzyfication. The entire fuzzy logic is carried out on the graphics hardware allowing for interactive semantic driven volume rendering. In the following we describe the fuzzy logic used in our framework. We describe the evaluation of the antecedents, the implication, the aggregation of multiple rules, and finally the defuzzyfication. A more elaborate discussion on fuzzy logic in general can be found in the literature [YZ92, TU97].

4.1 Evaluation of Antecedents

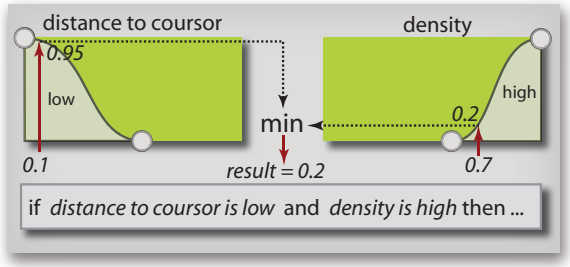

Figure 4: Evaluation of antecedent of an exemplary rule. The membership functions are evaluated and combined with fuzzy logic arithmetics. In this example the distance to cursor is 0.1 and the density is 0.7.

The evaluation of the antecedents is done using fuzzy logic operations. The antecedent contains a logical combination of semantic values of attributes. In Figure 4 two examples of attributes are shown. The *distance to cursor* is a user-driven attribute that encodes the distance to the mouse cursor for each sample. In the example of Figure 4 the membership function of the values *low* is shown for the attribute *distance to cursor*. The attribute *density* has the semantic value *high* in Figure 4. The membership functions are evaluated for each attribute that occurs in the antecedent of the rule. In Figure 4 we assume the value for *distance to cursor* to be 0.1 resulting in a membership of 0.95 for the semantic value *low*. Thus the *distance to cursor is low* with a membership of 0.95 in the example above. The evaluation of the membership function for *density is high* results in a membership of 0.2 for an assumed density of 0.7. The fuzzy logic operation *and* results in the minimum of the operands. In Figure 4 the result of the antecedent of the rule *if distance to cursor is low and density is high then* ... is the minimum of the two operands of the logical *and* and is therefore 0.2.

Other fuzzy logic operations are the *or* and the unary *not* operation. The *or* is evaluated as the maximum of the operands and the unary *not* is one minus the operand.

In our implementation the rules are parsed and translated into shader code. We build a fuzzy logic expression tree containing the operations *and*, *or*, and *not*. The nodes of the expression tree are substituted with the corresponding operations *min*, *max*, and $1.0-\ldots$. The leafs of the tree are the operands of the fuzzy logic expression (i.e., the membership functions). We store the membership functions as 1D textures and combine them into a 1D texture array. We substitute the leaf nodes of the

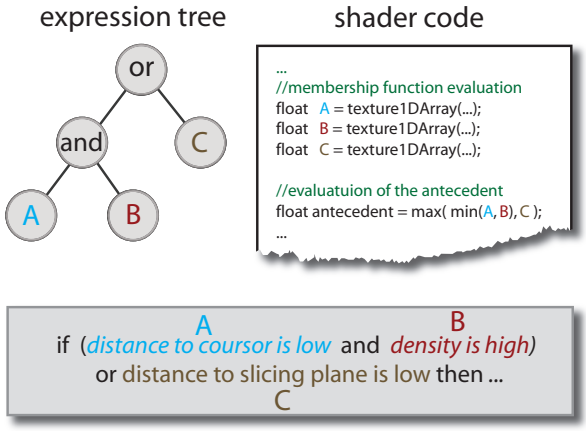

Figure 5: Shader code generation for the evaluation of antecedents. An expression tree and the corresponding shader code are generated from a simple rule.

expression tree with texture lookups in the 1D texture array and expand the expression tree to generate valid shader code. Figure 5 shows an example of a simple rule, the constructed expression tree, and the translation into shader code.

4.2 Implication

The evaluated antecedent has an implication on the consequent of a rule. The membership functions of the styles in the consequents are affected by the antecedent. Let the value of an antecedent be a and the membership function of a style be m(x) then the implication on this membership function is given by:

$$m'(x) = min(m(x), a) \tag{1}$$

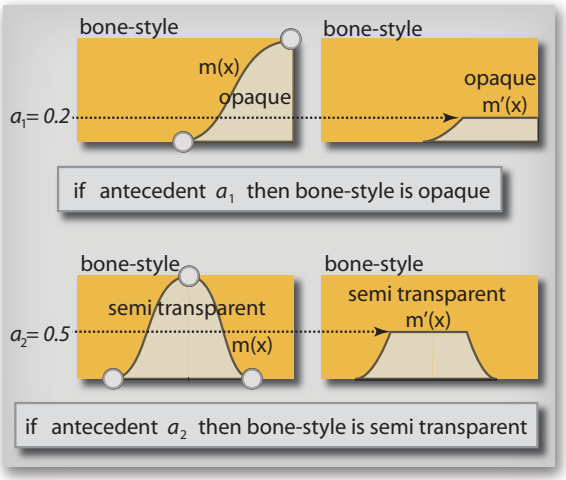

Figure 6: Implications of two different rules affecting the same style. The membership functions *opaque* and *semi transparent* are truncated according to the values of the respective antecedents.

The membership function of a style is truncated at the height of a. In Figure 6 two examples of implications are shown. The upper example shows the implication on the membership function *opaque* of the style *bone-style*. The antecedent value of a_1 truncates the membership function at a height of 0.2. The lower example in Figure 6 shows the effect of implication on a second membership function of the same style. The membership function *semi transparent* is truncated at a height of 0.5 because of the exemplarily chosen value of a_2.

4.3 Aggregation

Aggregation is the process of building the sum of all membership functions that are non-zero after implication and that affect the same style. Aggregation has to be done for each style separately. Each aggregation results in a (usually more complex) function for one style. In Figure 7 the aggregation of two functions for the style *bone-style* can be seen. The two functions are the result

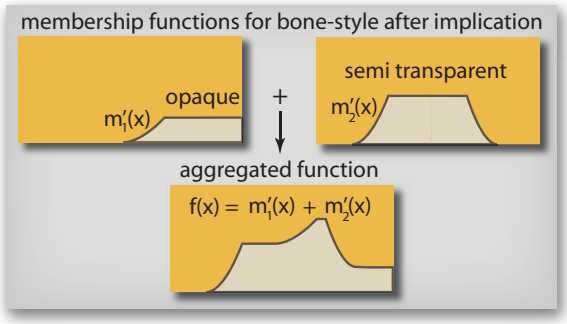

Figure 7: Aggregation of two membership functions of the same style. The aggregated function is the sum of all truncated membership functions affecting the same style.

of the previous implications and are aggregated using the sum.

4.4 Defuzzyfication

The defuzzyfication is the process of deriving a crisp value for each style used in one or more consequents. Defuzzyfication for one style is done by finding the centroid of the aggregated function for this style. The defuzzyfied value is used as rendering attribute and influences the visual appearance of the sample.

4.5 Implication, Aggregation and Defuzzification on the GPU

Obviously implication, aggregation and defuzzyfication are operations that are not straightforward to implement on the GPU. The representation of 1D functions (i.e., the membership functions of semantic values for the styles), the truncation of these functions (i.e., the implication), the sum of the truncated functions (i.e., the aggregation) and the calculation of the centroid of a potentially arbitrary shaped function (i.e., the defuzzyfication) are tasks that are hard to achieve on the GPU. We show that the computationally most expensive tasks can be precomputed and stored in textures.

The derivation of the defuzzyfication as described in our earlier work [RBG07] follows: For defuzzyfication we want to find the centroid of the aggregated function. Let $f(x)$ be the result from the aggregation, then its centroid c_f is given by the equation:

$$c_f = \frac{\int x f(x) dx}{\int f(x) dx} \tag{2}$$

Let the semantic values respectively the membership functions of one style be $m_j(x)$. The membership function for the semantic value affected by rule i after the implication is then given by the equation

$$m_i'(x,a_i) = \min(a_i, m_i(x)) \tag{3}$$

where a_i is the antecedent value of the rule i. The aggregated membership function $f(x)$ is then given by

$$f(x) = \sum_{i \in I} m_i'(x,a_i) \tag{4}$$

where I is the set of indices of rules that affect the given style. The centroid of the aggregated function can then be calculated by substituting Equation 4 in Equation 2:

$$c_f = \frac{\int x \sum_{i \in I} m_i'(x,a_i) dx}{\int \sum_{i \in I} m_i'(x,a_i) dx} \tag{5}$$

We can rewrite Equation 5 as follows:

$$c_f = \frac{\sum_{i \in I} \int x m_i'(x,a_i) dx}{\sum_{i \in I} \int m_i'(x,a_i) dx} \tag{6}$$

In Equation 6 it can be seen, that the integrals (i.e., the summands in the nominator as well as in the denominator) do solely depend on the a_i. This allows us to pre-compute the summands of the nominator as well as of the denominator and store them in a lookup table. During evaluation the a_i are used as index for the lookup tables. For each rendering attribute Equation 6 has to be evaluated, resulting in a total of $2n$ texture lookups for the precomputed nominators and denominators, $2(n-1)$ summations and one division, where n is the number of rules that affect the rendering attribute. The shader code for the implication, aggregation and defuzzyfication resulting from Equation 6 is automatically generated.

5 Rendering

In Section 4 the evaluation of the fuzzy logic on the GPU is described. The result of the fuzzy logic is a floating point value for each rendering attribute that was used in the consequents of the fuzzy rules. The flexibility of our framework is achieved using shader program templates. A shader program template implements a rendering technique. It specifies rendering attributes, that can be used in the consequent of rules. Placeholders are put in the shader program template at the fuzzy logic specific parts of the rendering procedure. The parts containing the fuzzy logic evaluation of the rendering attributes are generated automatically and complete the shader program template.

We describe two different shader program templates implementing volume rendering techniques. In Section 5.1 the artistic volume rendering template is described. Section 5.2 deals with the more advanced flat rendering mode template.

5.1 Artistic Volume Rendering Template

We use a direct volume rendering approach for the visualization of volumetric data. Viewing rays are cast through the volume and sampled at equidistant sample positions. For each sample an opacity transfer function is evaluated determining the visibility of the current sample. Samples with opacity greater than zero are colored. In the artistic volume rendering template the color evaluation is the only part of the shader program template that depends on the fuzzy logic rules. The color evaluation is done using artistic styles. Each style is a rendering attribute that is evaluated according to the fuzzy rules. The rules ensure that styles are applied gradually and selectively to different regions. The styles are described using style transfer functions [BG07]. Style transfer functions allow the parameterization of artistic styles. A style transfer function is given by a set of images of shaded spheres. In Figure 8

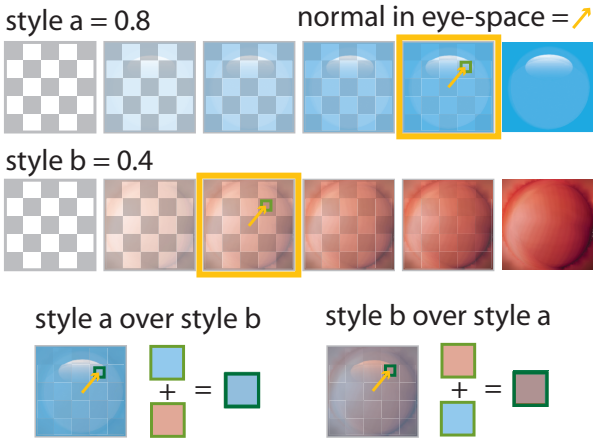

Figure 8: Example for the compositing of two styles is shown. The current exemplary sample has a value of 0.8 for *style a* and a value of 0.4 for *style b*. The corresponding spheres are outlined in yellow. The normal in eye-space for the current sample is shown as yellow arrow. The color for each style is evaluated accordingly (outlined in light green). The compositing is done following the priority of the styles. The final color is outlined in dark green.

two examples for styles can be seen. Note that in the example for simplicity both styles vary from transparent

to opaque but this is not necessarily the case. Another example could be that the line thickness of a style is parameterized.

The defuzzyfication gives a floating point value for each style. Figure 8 shows exemplary results of the defuzzyfication outlined in yellow. In this example the defuzzyfication for *style a* resulted in 0.8 and for *style b* in 0.4. The resulting color for each style depends on the normal of the current sample in eye-space. The yellow arrows in Figure 8 indicate an exemplary normal in eye-space. The normal in eye-space is used to index the image of the spheres. The evaluation of a style is described in more detail in previous work [BG07]. In Figure 8 the resulting colors for the styles are outlined in light green. The final color of the sample is composited from all used styles. The styles are prioritized and composited from the bottom to the top style, following the widely known compositing scheme used in image manipulation programs (such as Adobe Photoshop or GIMP). In Figure 8 two possibilities for the resulting color are shown. The result depends on the priority of the styles. If the priority of *style a* is higher then the priority of *style b* (i.e., style a over style b) then the resulting style is a blueish sphere and the final color of the sample is blue (outlined in dark green in Figure 8). If the priority of *style a* is lower then the priority of *style b* (i.e., style b over style a) then the resulting style is a violet sphere. The final color of the sample is also outlined in dark green in Figure 8.

5.2 Flat Rendering Mode Template

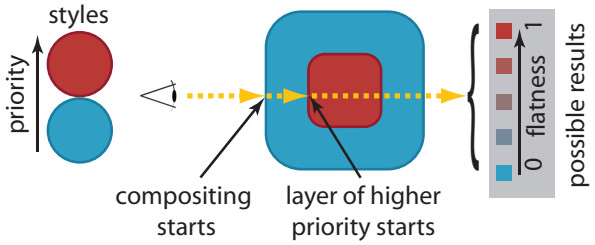

Figure 9: The flat rendering mode favors samples of higher priority during ray casting. The yellow line depicts a viewing ray. Along the ray common compositing is done until a region of higher priority is reached. The composited color is deemphasized according to the flatness parameter.

Spatial relations are often emphasized to aid the viewer of an illustration in correctly interpreting the image. However, in traditional illustration spatial relations can also be completely ignored in order to show hidden structures of higher importance. The flat rendering mode template extends an artistic volume rendering template. It uses the prioritization of styles to achieve the effect of ignoring spatial relations. If the flat rendering mode is used, regions in the volume that use styles of higher priority overdraw regions with lower priority. This is conceptually similar to the work of Viola et al. [VKG04]. Our method can be applied gradually and selectively driven by data and user-driven semantics.

Figure 9 depicts the rendering process using the flat rendering mode. The dashed yellow line shows a viewing ray. The blue and red boxes denote two regions that use different styles according to specific rules. Samples along the viewing ray are evaluated and composited. If the ray reaches a region of higher priority the ray-color is influenced accordingly to the flatness parameter. A flatness parameter of 0 results in common volume rendering. A flatness parameter of 1 always shows the regions with highest priority. At each position the ray enters a region of higher priority the ray-color $c_r(x_i)$ is set to:

$$c_r(x_i) = c_r(x_{i-1})\left(1 - p_f\right) \tag{7}$$

where x_i is the current sample position, x_{i-1} is the position of the last sample and p_f is the flatness parameter.

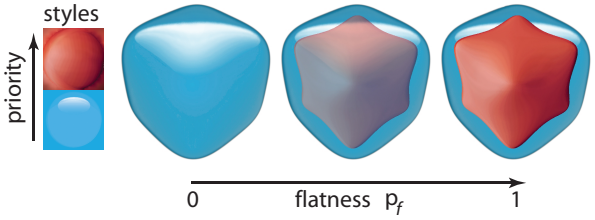

Figure 10: Example renderings using different values of the flatness parameter. The inner cube has higher priority and is therefore shown for a flatness parameter greater than zero.

Figure 10 shows a volume rendering of a cube dataset where densities increase towards the cube center. A simple rule states that the reddish style is high for regions of high density. The left most rendering of Figure 10 shows just the outer surface of the cube. The region with the style of higher priority remains hidden. The rendering in the middle of Figure 10 uses a flatness parameter of 0.5 and the right most rendering a flatness parameter of 1.0.

In all three examples of Figure 10 the flatness parameter is set globally. However, the *flatness* is a semantic parameter that describes the trade-off between showing spatial relationships and showing important regions. Shader program templates can offer illustration semantics for the use in the fuzzy rules. The flat rendering

mode template offers the property *flatness*. The *flatness* as any other rendering attribute offered by shader program templates can be used in the consequents of fuzzy rules and is dynamically evaluated per sample. This results in a local semantics driven application of the *flatness* parameter.

The flat rendering mode cannot make use of early ray termination. This reduces the performance for large datasets. However, the method can be used for the generation of high quality illustrations or offline rendered animations.

6 Results

We show a few examples of interactive illustrations that can be achieved with our system and demonstrate the possibilities of user-driven semantics and the view-dependent evaluation of the fuzzy rules. Rules that use only data semantics define the visual appearance of the depicted object. The behavior of the interactive illustration is defined with rules that incorporate user-driven semantics. All results that are presented in this paper were achieved in interactive sessions using our system without any pre-segmentation of the data. The rules defining the behavior of the interactive illustrations are shown in the respective Figures. Rules that solely depend on the density are not shown in the Figures.

In Figure 11 an illustration of the upper part of the visible human dataset is shown. A slicing plane is used to specify a focus region. The slicing plane additionally shows the histological cut data of the visible human dataset. The spheres used to define the styles are shown at the bottom of Figure 11. The left most style is applied in regions very close to the slicing plane. The second and third styles are used to render the skin. Rules that depend on the distance to the slicing plane are specified to modulate the style used for the skin. The right most style is used for regions of high density (i.e., bones). The dataset has a size of 256^3. Renderings with the same rules and styles applied are achieved at an average of 23fps for a view port size of 512^2 and a sample distance of 1.0.

In Figure 12 an interactive illustration of a human body is shown. The user focus is defined at the position of the mouse. Rules using the distance to the mouse define the appearance of the skin. The skin is shown completely transparent close to the mouse, in unshaded white at a farther distance and in pink for high distances. The spheres used for the styles are shown on the left of Figure 12. The lower two styles are used for the skin color.

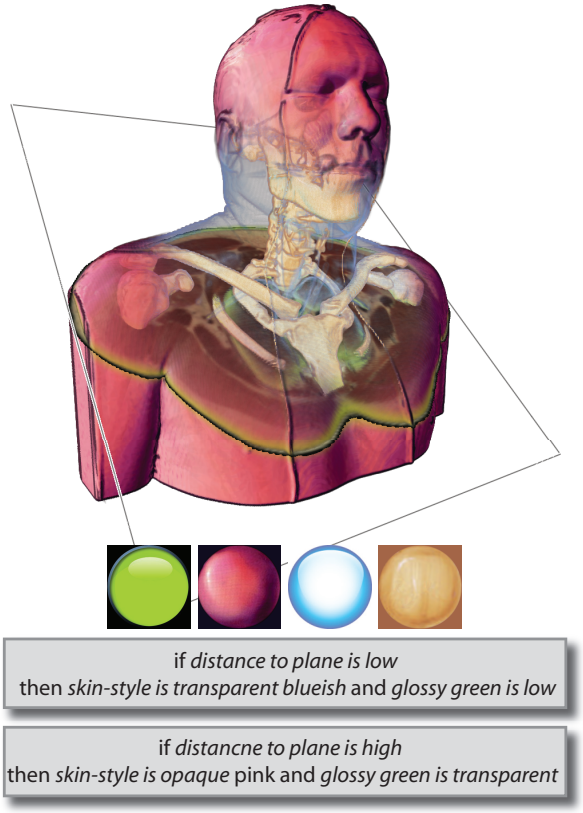

Figure 11: Rendering of the visible human dataset. A slice plane of the histological data is shown. The CT-data is used for the volume rendering providing the context for the slice plane.

The upper two styles are used for the bones and the vessels and are applied according to rules that do solely depend on the density. The dataset shown in Figure 12 has a size of $256^2 \times 415$. Images with the same rules and styles applied are rendered at an average of 20fps for a view port size of 512^2 and a sample distance of 1.0.

In Figure 13 renderings of a CT-scan of a human leg is shown. A slicing plane is used to show the CT-data. Rules dependent on the distance to the slicing plane are specified to influence the transparency of the skin and the soft tissue. Skin and soft tissue close to the slicing plane are made transparent. The spheres used to create the styles are shown on the left of Figure 13. From left to right the spheres are used to color skin regions, soft-tissue regions, bone regions, and the metallic implants of the patient. Further, rules were specified that influence the *flatness* of the illustration in dependence on the distance to the slicing plane. The left most rendering shows a rendering fully applying the flat rendering mode in all regions. The other renderings use the

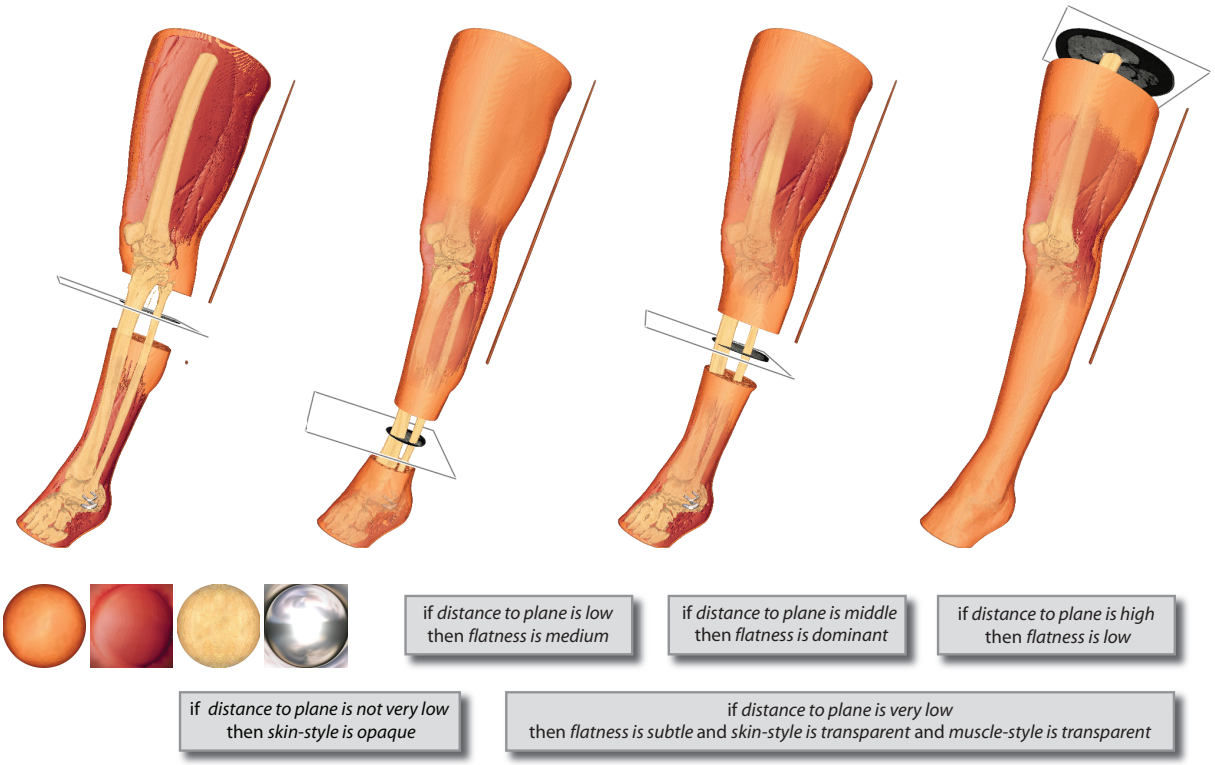

Figure 13: Rendering of a CT-scan of a human leg. The distance to the slicing plane influences the rendering styles and the flatness parameter. The left rendering shows the flat rendering mode fully applied ignoring the spatial relations. The remaining three renderings show the effect of the rules. The flatness parameter is applied gradually and selectively. Spatial cues are provided in regions close to the slicing plane.

flat rendering mode gradually only in regions of middle distance to the slicing plane. This results in illustrations, that preserve the spatial relations close to and far away from the slicing plane, but ignore spatial relations in between. The dataset shown in Figure 13 has a size of $147 \times 162 \times 429$. Images with the same rules and styles applied are rendered at an average of 6fps for a view port size of 512^2 and a sample distance of 1.0.

These simple examples illustrate the power and flexibility of our approach. The system is easily extensible for other interactive illustration scenarios. The interactive behavior of our system can be seen in the accompanying video.

7 Conclusions and Future Work

We presented a rendering concept for interactive illustrations that is driven by low-level semantics. In our system fuzzy logic rules are specified. They linguistically define a mapping from data attributes respectively user-driven semantically meaningful parameters to visual styles and rendering techniques. Our framework handles a great variety of rendering techniques in a uniform way. We showed the interactive semantics driven specification of rendering attributes such as the flatness parameter of the flat rendering mode. Interactive illustrations were presented that are the result of the semantic driven rendering concept.

For the future we identified major research challenges where the current framework might be of significant value. One of these challenges is the automatic illustration for medical diagnosis and treatment. Often patients are examined with expensive equipment. The diagnosis and the further treatment of the patient are communicated in the language of experts. For example, a typical diagnosis using a CT-scan is full of semantics encoded in medical language. We believe that the language of the medical experts can be automatically translated into illustrations. These illustrations are undoubtable of help for the patient to understand her/his situation. The presented framework is a first step in the direction of automated semantic illustrations using domain expert lan-

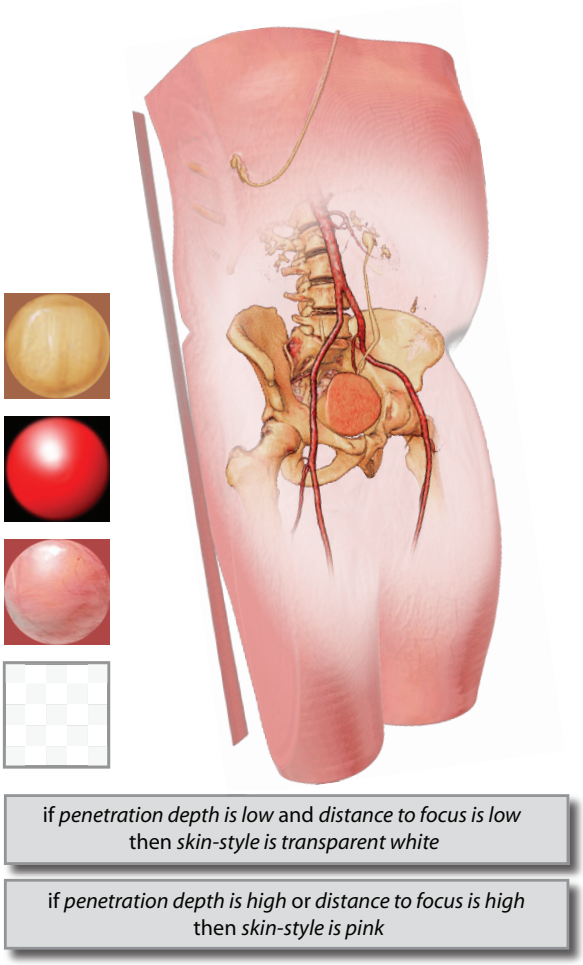

Figure 12: The mouse cursor defines the user focus. Depending on the user focus the illustrative rendering is altered.

guage. We show with our work how low-level semantics can be translated into interactive illustrations.

Acknowledgements

The work presented in this publication is carried out as part of the exvisation project (www.cg.tuwien.ac.at/research/vis/exvisation) supported by the Austrian Science Fund (FWF) grant no. P18322. We also want to acknowledge the Laboratory of Neuro Imaging, UCLA, USA for the monkey atlas dataset and for kindly providing details about the PET data. The engine block dataset is courtesy of General Electric, USA.

References

[BG05] BRUCKNER S., GRÖLLER M. E.: VolumeShop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization 2005* (2005), pp. 671–678.

[BG07] BRUCKNER S., GRÖLLER M. E.: Style transfer functions for illustrative volume rendering. *Computer Graphics Forum (accepted for publication) 26*, 3 (2007).

[CS01] COYNE B., SPROAT R.: Wordseye: an automatic text-to-scene conversion system. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 487–496.

[HMBG01] HAUSER H., MROZ L., BISCHI G.-I., GRÖLLER M. E.: Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 242–252.

[KKH02] KNISS J., KINDLMANN G., HANSEN C.: Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 270–285.

[KSW06] KRÜGER J., SCHNEIDER J., WESTERMANN R.: Clearview: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 941–948.

[KUS*05] KNISS J., UITERT R. V., STEPHENS A., LI G.-S., TASDIZEN T., HANSEN C.: Statistically quantitative volume visualization. In *Proceedings IEEE Visualization 2005* (2005), pp. 287–294.

[LM04] LUM E. B., MA K.-L.: Lighting transfer functions using gradient aligned sampling. In *Proceedings of IEEE Visualization 2004* (2004), pp. 289–296.

[MIA*04] MCCORMICK P., INMAN J., AHRENS J., HANSEN C., ROTH G.: Scout: a hardware-accelerated system for quantitatively driven visualization and analysis. In *Proceedings of IEEE Visualization 2004* (2004), pp. 171–178.

[RBG07] RAUTEK P., BRUCKNER S., GRÖLLER M. E.: Semantic layers for illustrative volume rendering. *IEEE Transactions on Visualization and Computer Graphics* (2007), to appear, www.cg.tuwien.ac.at/research/publications/2007/TR–186–2–07–05.

[SES05] SVAKHINE N., EBERT D. S., STREDNEY D.: Illustration motifs for effective medical volume illustration. *IEEE Computer Graphics and Applications 25*, 3 (2005), 31–39.

[SF91] SELIGMANN D. D., FEINER S. K.: Automated generation of intent-based 3D illustrations. In *Proceedings of ACM Siggraph 1991* (1991), pp. 123–132.

[SMGG01] SLOAN P.-P., MARTIN W., GOOCH A., GOOCH B.: The lit sphere: A model for capturing NPR shading from art. In *Proceedings of Graphics Interface 2001* (2001), pp. 143–150.

[SSBW05] STOCKINGER K., SHALF J., BETHEL W., WU K.: Query-driven visualization of large data sets. In *Proceedings of IEEE Visualization 2005* (2005), pp. 167–174.

[SWB*00] SATO Y., WESTIN C.-F., BHALERAO A., NAKAJIMA S., SHIRAGA N., TAMURA S., KIKINIS R.: Tissue classification based on 3d local intensity structures for volume rendering. *IEEE Transactions on Visualization and Computer Graphics 6*, 2 (2000), 160–180.

[TU97] TSOUKALAS L. H., UHRIG R. E.: *Fuzzy and Neural Approaches in Engineering*. Wiley & Sons, 1997.

[VKG04] VIOLA I., KANITSAR A., GRÖLLER M. E.: Importance-driven volume rendering. In *Proceedings of IEEE Visualization 2004* (2004), pp. 139–145.

[WS06] WOODRING J., SHEN H.-W.: Multivariate, time varying, and comparative visualization with contextual cues. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 909–916.

[YZ92] YAGER R. R., ZADEH L. A. (Eds.): *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, vol. 165 of *International Series in Engineering and Computer Science*. Springer, 1992.