Eurographics/ IEEE-VGTC Symposium on Visualization 2008
A. Vilanova, A. Telea, G. Scheuermann, and T. Möller
(Guest Editors)

Volume 27 (2008), Number 3

Interaction-Dependent Semantics for Illustrative Volume Rendering

Peter Rautek, Stefan Bruckner, and M. Eduard Gröller [1]

[1] Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

Abstract

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 that preserves the intention of the user. The semantic layers concept establishes this mapping with a linguistic formulation of rules that directly map data features to rendering styles. With semantic layers fuzzy logic is used to evaluate the user defined illustration rules in a preprocessing step.

In this paper we introduce interaction-dependent rules that are evaluated for each frame and are therefore computationally more expensive. Enabling interaction-dependent rules, however, allows the use of a new class of semantics, resulting in more expressive interactive illustrations. We show that the evaluation of the fuzzy logic can be done on the graphics hardware enabling the efficient use of interaction-dependent semantics. Further we introduce the flat rendering mode and discuss how different rendering parameters are influenced by the rule base. Our approach provides high quality illustrative volume renderings at interactive frame rates, guided by the specification of illustration rules.

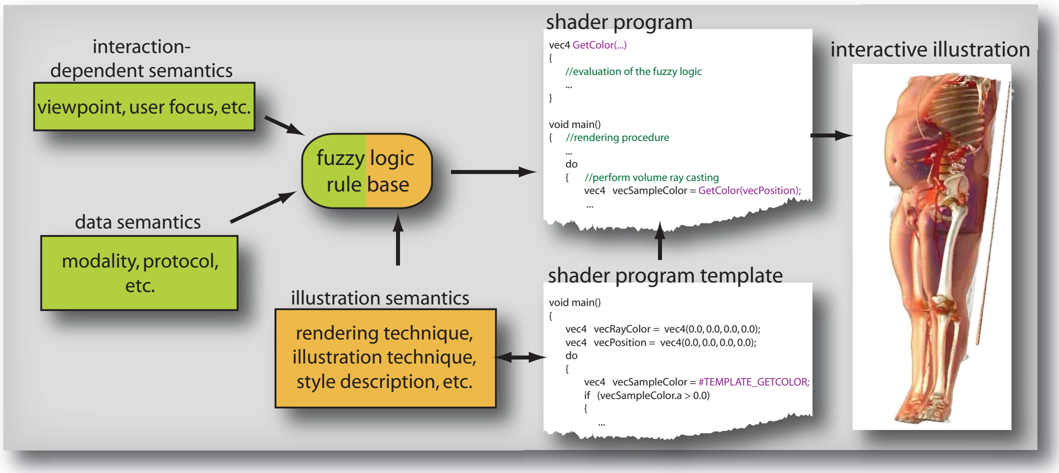
1. Introduction

Medical doctors 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 intent and therefore constrained by the diagnosis, the treatment, or the possible surgical intervention they should convey.

In this paper we introduce the concept of interaction-dependent semantics for illustrative rendering of volume data. In Figure 1 an outline of the rendering system is shown. The central component in our system is a fuzzy logic rule base. The rules are guided by the different types of semantics. Data semantics depend on the available data. CT-scans, for example, provide information on tissue densities. Different ranges of densities correspond to semantically meaningful entities (like air, soft tissue, bone, metal, etc.). Interaction-dependent 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, and the region of user focus (e.g., position of the mouse cursor). These interaction-dependent parameters are used in the fuzzy rules to completely alter the illustration interactively. The interaction-dependent rules allow a specification of the behavior of the interactive illustration.

As shown in Figure 1 different types of semantics are used for the fuzzy logic rules. The *if* part of rules states constraints using data semantics and interaction-dependent semantics. The *then* part of rules describes the consequences for the illustration using illustration semantics. Illustration semantics originate from the area of traditional illustration. Illustrators use specific terms 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



```
vec4 GetColor(...)
{
    //evaluation of the fuzzy logic
    ...
}

void main()
{   //rendering procedure
    ...
    do
    {    //perform volume ray casting
         vec4  vecSampleColor = GetColor(vecPosition);
         ...
```

```
void main()
{
    vec4  vecRayColor =  vec4(0.0, 0.0, 0.0, 0.0);
    vec4  vecPosition =  vec4(0.0, 0.0, 0.0, 0.0);
    do
    {
         vec4  vecSampleColor = #TEMPLATE_GETCOLOR;
         if  (vecSampleColor.a > 0.0)
         {
             ...
```

Figure 1: *Outline of the presented semantics driven rendering framework: The different types of semantics guide the generation of the interactive illustration.*

styles, and the way spatial relations are emphasized or to the contrary ignored to depict occluded structures. As shown in Figure 1 a shader program template is chosen according to the intended illustration technique and the rule base is translated into shader code to complete the shader program template. This approach allows us to implement a wide variety of illustration techniques that are directly controlled by the fuzzy logic rules.

In earlier work [RBG07] the use of data and illustration semantics was introduced for illustrative volume rendering. In this paper we extend this approach with the following main contributions:

Interaction-dependent semantics: We introduce interaction-dependent semantics that are evaluated each frame. Adjustable slicing planes, the position of the mouse cursor, and view-dependent parameters are manipulated by the user. Semantics like *distance to the mouse cursor*, *distance to the slicing plane*, *distance to the image plane* can be used in the antecedent of rules to alter the behavior of the 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. The GPU based implementation allows interactive evaluation of the fuzzy logic enabling interaction-dependent semantics.

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 generalize the semantic layers concept and show the possibility to influence arbitrary rendering parameters with fuzzy logic rules. We demonstrate the capability to influence illustration techniques with the flat rendering mode that resembles the above described illustration technique.

The remainder of the paper is structured as follows: In Section 2 we briefly review the related work. In Section 3 we give a short overview of the implemented system. In Section 4 we explain the evaluation of rendering attributes using fuzzy logic and give details on the GPU based implementation. The general rendering framework and the flat rendering mode are described in Section 5.

2. Related Work

Our approach is a general rendering concept that translates illustration rules into images. 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. Similar to Rezk-Salama et al. [RSKK06], we present a high-level user interface for the specification of a mapping from volume attributes to a visual style by semantically meaningful parameters.

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. Grabli et al. [GTDS04] present a system for programmable line drawing styles.

Our system is able to describe a mapping from a multi-dimensional attribute domain to visual appearance. A related approach that is based on multi-dimensional transfer functions was shown by Kniss et al. [KKH02]. 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. Tzeng et al. [TLM05] show a user interface to specify input for a neural network that classifies volume data in higher dimensions.

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 hot spots. Our system allows similar results with the introduction of interaction-dependent semantics.

3. Semantics Driven Rendering System

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 ..."*, *"if diffusion is low then. . . "*, or *"if curvature is high then. . . "*. Interaction-dependent semantics are represented in the rule base by rules like *"if distance to slicing plane is low then. . . "*, or *"if user focus is close then. . . "*. The rules can further use 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. In fuzzy logic the antecedent of a rule is not simply true or false but can have any transitional value in between. The consequent describes the consequences for illustrative styles if the antecedent of a rule is not false. The consequent in our system describes the resulting rendering attributes, like *". . . then bone-style is transparent"* or *". . . then contours are thick"*.

In our rendering framework the styles and rendering techniques 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.

The interaction-dependent semantics that are used in the antecedents potentially change each frame and make it necessary to evaluate the rules per frame. It is a challenging task to implement a volume rendering system that evaluates all fuzzy rules per sample. 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.

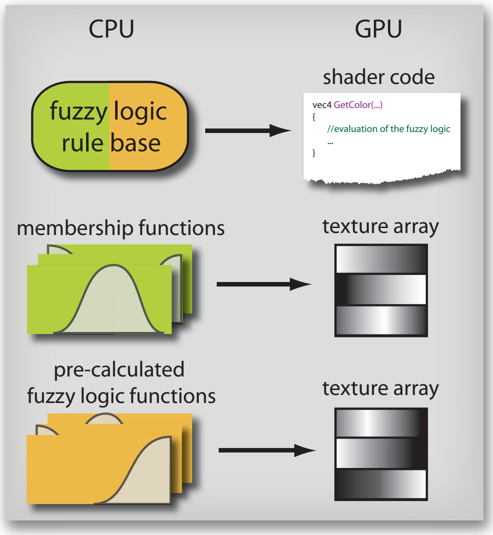

```
vec4 GetColor(...)
{
    //evaluation of the fuzzy logic
    ...
}
```

Figure 2: *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.*

Figure 2 shows the components used on the CPU and the corresponding components on the GPU. 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.

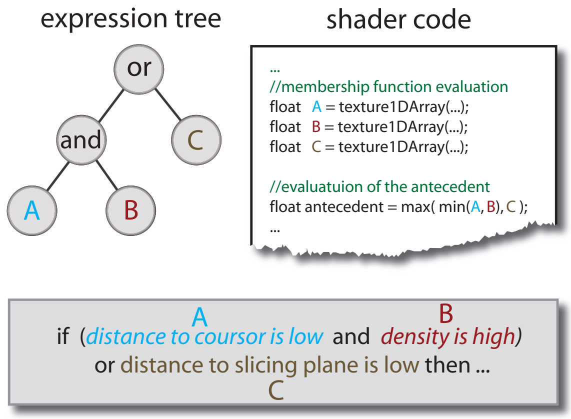
The shader program is adapted automatically for each change of the rule base. The update of the shader program is the most costly operation in our system. However it is only done when rules change and takes less than one second on a modern desktop PC. Changes in membership functions result in an interactive update of the corresponding texture arrays.

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 semantics driven volume rendering. In the following we describe the fuzzy logic used in our framework. A more elaborate discussion on fuzzy logic in general can be found in the literature [YZ92, TU97].

4.1. Evaluation of Antecedents

The evaluation of the antecedents is done using fuzzy logic operations. The antecedent of each rule consists of a logical combination of semantic values of attributes (e.g., *distance to cursor* has semantic values like *low*, *middle*, etc.) The membership functions are evaluated for each attribute that occurs in the antecedent of the rule and combined with the fuzzy logic operations *and* (resulting in the minimum of the operands), and *or* (resulting in the maximum of the operands). Further the unary *not* operation can be used and is evaluated as one minus the operand.


expression tree
or
and
C
A
B
shader code

```
...
//membership function evaluation
float   A = texture1DArray(...);
float   B = texture1DArray(...);
float   C = texture1DArray(...);

//evaluatuion of the antecedent
float antecedent = max( min(A, B), C );
...
```

A
B
if (distance to coursor is low and density is high)
or distance to slicing plane is low then ...
C

Figure 3: *Shader code generation for the evaluation of antecedents. An expression tree and the corresponding shader code are generated from a simple rule.*

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 expression tree with texture lookups in the 1D texture array and expand the expression tree to generate valid shader code. Figure 3 shows an example of a simple rule, the constructed expression tree, and the translation into shader code.

4.2. Implication, Aggregation and Defuzzification

The evaluated antecedent has an implication on the consequent of a rule. Consequents consist of a list of semantic values for styles that are affected by the antecedent. Semantic values for styles are also represented by membership functions. Let the value of an antecedent be $a \in [0,1]$ and the membership function of a semantic value for a given style be $m(x)$ then the implication on this membership function is given by:

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

This results in a truncation of the membership function at the height of a.

Aggregation is the process of building the sum of all membership functions after implication. Aggregation results in one function for each style.

Defuzzyfication is done to deriving a crisp value for each style. We used the centroid method for defuzzyfication. The centroid of the aggregated function is the value that is used as rendering parameter for each style.

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 (i.e., the defuzzyfication) of a potentially arbitrary shaped function 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 earlier work [RBG07] is essential for the implementation of the fuzzy logic on the GPU. We briefly review this derivation: 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:

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

We can rewrite Equation 5 as follows:

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

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.

5. Rendering

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 4 two examples for styles can be seen. Note that for simplicity in the example 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.

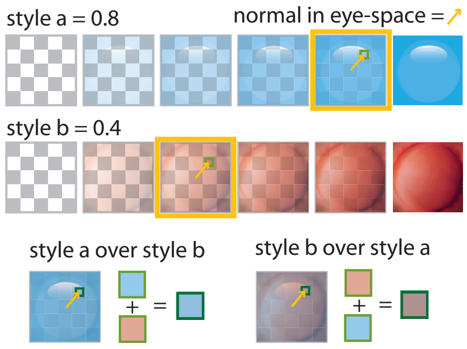


Figure 4: *Example for the compositing of two styles is shown. The exemplary sample has a value of* 0.8 *for style a and a value of* 0.4 *for style b. The corresponding spheres are outlined in yellow.*

The defuzzyfication gives a floating point value for each style. Figure 4 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 4 indicate an exemplary normal in eye-space. The normal in eye-space is used to index the image of the spheres. In Figure 4 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 compositing scheme used in image manipulation programs (such as Adobe Photoshop or GIMP). In Figure 4 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 4). 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 4.

5.2. Flat Rendering Mode Template

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 the artistic volume rendering template to implement this technique. Each style is assigned a priority. 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]. However, our method can be applied gradually and selectively driven by data and interaction-dependent semantics.

Figure 5 depicts the rendering process using the flat rendering mode. The dashed yellow line shows a viewing ray.

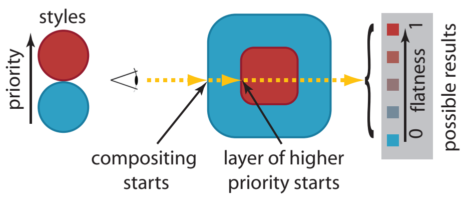


Figure 5: *The flat rendering mode favors samples of higher priority during ray casting. The yellow line depicts a viewing ray. Along the ray common compositing is done until a region of higher priority is reached. The composited color is deemphasized according to the flatness parameter.*

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 according 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.

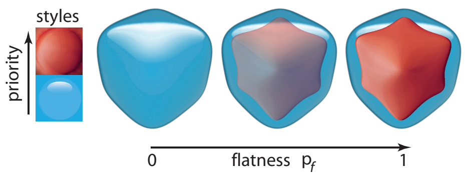


Figure 6: *Example renderings using different values of the flatness parameter. The inner cube has higher priority and is therefore shown for a flatness parameter greater than zero.*

Figure 6 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 6 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 6 uses a flatness parameter of 0.5 and the right most rendering a flatness parameter of 1.0.

In all three examples of Figure 6 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. 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.

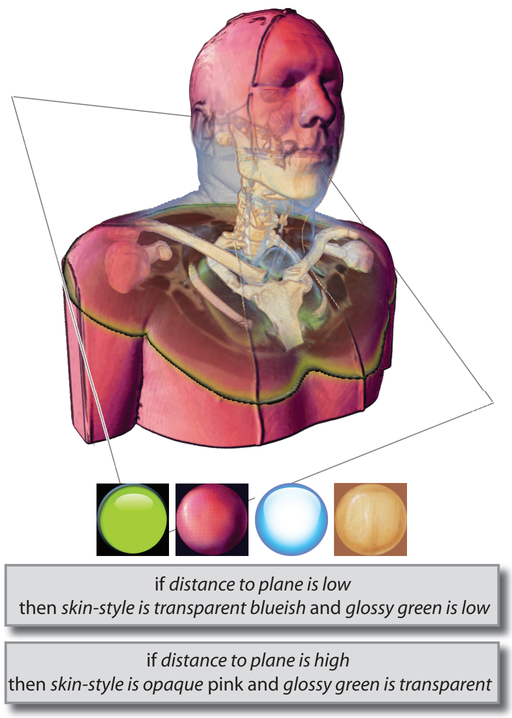
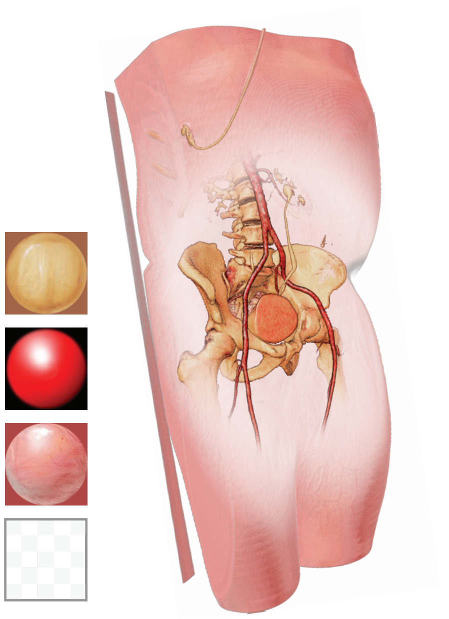

if *distance to plane is low*
then *skin-style is transparent blueish* and *glossy green is low*
if *distance to plane is high*
then *skin-style is opaque* pink and *glossy green is transparent*

Figure 7: *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.*

6. Results

The evaluation of the fuzzy logic on the CPU takes a few seconds, making it impractical to render interaction-dependent illustrations. The presented GPU based implementation enables the use of interaction-dependent rules. Interaction-dependent semantics are capable to produce renderings that put emphasis on a specific focus region and deal with all kinds of view-dependent semantics. Examples for interaction-dependent semantics include *the distance to the image plane* (that allows techniques like depth cueing, depth of field, etc.), *the viewing angle* (e.g. the volume is rendered in a *blue-print* style if it is viewed from the top and in a more tangible style when viewed from the side), etc. Time-dependent semantics are also a subclass of interaction-dependent semantics that can be used to alter the rendering of specific regions over time. We show a few examples of interactive illustrations that can be achieved with our system and demonstrate the possibilities of interaction-dependent semantics and the view-dependent evaluation of the fuzzy rules. Rules that incorporate interaction-dependent semantics define the behavior of the illustration. These rules are shown in the respective Figures. All results that are presented in this paper were achieved in interactive sessions with a GeForce 8800 GTX graphics card. No pre-segmentation of the data was used for the shown examples.



if *penetration depth is low* and *distance to focus is low* then *skin-style is transparent white*

if *penetration depth is high* or *distance to focus is high* then *skin-style is pink*

Figure 8: *The mouse cursor defines the user focus. Depending on the user focus the illustrative rendering is altered.*

Renderings at a view port size of 512^2 and a sample distance of 1.0 are achieved at an average of 23fps for Figure 7, of 20fps for Figure 8, and of 6fps for Figure 9 with the respective rules and styles applied.

In Figure 7 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 7. 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.

In Figure 8 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 8. The lower two styles are used for the skin color. The upper two styles are used for the bones and the vessels and are applied according to rules that solely depend on the density. The dataset shown in Figure 8 has a size of $256^2 \times 415$.

In Figure 9 renderings of a CT-scan of a human leg are depicted. 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 in Figure 9. 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 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 9 has a size of $147 \times 162 \times 429$.

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. Conclusion

We present a rendering concept for interactive illustrations that is based on fuzzy logic rules evaluated on the GPU. The rules linguistically define a mapping from data attributes and interaction-dependent 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 are presented that are examples for the use of the interaction-dependent semantics.

8. Acknowledgements

The work presented in this publication is carried out as part of the ex**vis**ation project supported by the Austrian Science Fund (FWF) grant no. P18322.

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 26*, 3 (2007), 715–724.

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

[GTDS04] Grabli S., Turquin E., Durand F., Sillion F.: Programmable style for *npr* line drawing. In *Rendering Techniques, Eurographics Symp. on Rendering* (2004), pp. 33–44.

[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.

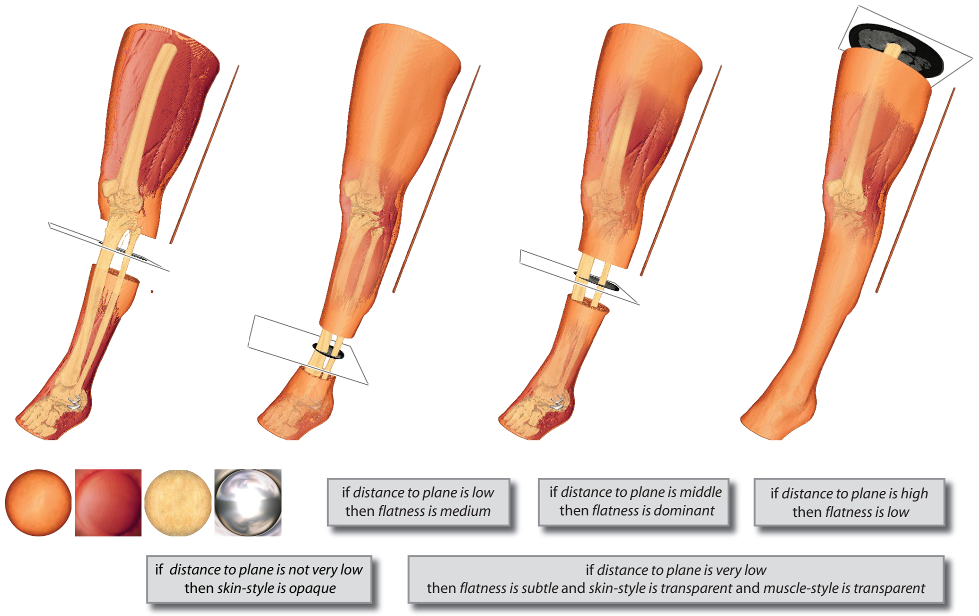

if *distance to plane* is low
then *flatness* is medium
if *distance to plane* is middle
then *flatness* is dominant
if *distance to plane* is high
then *flatness* is low
if *distance to plane* is not very low
then *skin-style* is opaque
if *distance to plane* is very low
then *flatness* is subtle and *skin-style* is transparent and *muscle-style* is transparent

Figure 9: *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.*

[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), 1336–1343.

[RSKK06] REZK-SALAMA C., KELLER M., KOHLMANN P.: High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1021–1028.

[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.

[TLM05] 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 11*, 3 (2005), 273–284.

[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.: Multi-variate, 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 C.S.* Springer, 1992.