Sonstige wissenschaftliche Arbeiten
zur Habilitationsschrift

Visual Exploration and Analysis of Volumetric Data

ausgeführt in den Jahren 2008-2012
zum Zwecke der Erlangung der *venia docendi* (Lehrbefugnis)
im Habilitationsfach "Praktische Informatik"

eingereicht im April 2012
an der Technischen Universität Wien
Fakultät für Informatik

von

Dipl.-Ing. Dr.techn. Stefan Bruckner
Flesichmanngasse 7/1B, A-1040 Wien
geboren am 2. Juni 1980 in Oberwart
bruckner@cg.tuwien.ac.at

Wien, im April 2012

Contents

Science is a wonderful thing if one does not have to earn one's living at it.

— *Albert Einstein*

Preface

THIS collection of selected publications is a supplement to my habilitation thesis entitled *Visual Analysis and Exploration of Volumetric Data*. It contains additional related scientific contributions published in the period from 2008 to 2012. All included articles appear in their unmodified form as published, only the original page numbers, headers, and footers have been removed for aesthetic purposes. The papers are sorted chronologically in ascending order of publication date.

Vienna, Austria, April 2012 *Stefan Bruckner*

1 Interaction-Dependent Semantics for Illustrative Volume Rendering

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.

The following paper appears in its original format published as:

P. Rautek, S. Bruckner, and M. E. Gröller. Interaction-dependent semantics for illustrative volume rendering. *Computer Graphics Forum*, 27(3):847–854, 2008.

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

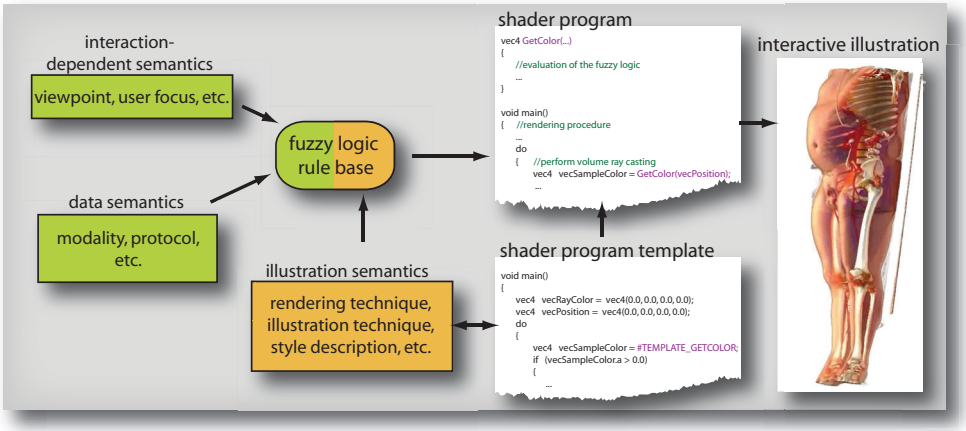


Figure 1: *Outline of the presented semantics driven rendering framework: The different types of semantics guide the generation of the interactive illustration.*

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 multivariate as well as time-varying data. Sato et al. [SWB*00] use classification rules to identify tissue structures in multimodal 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.

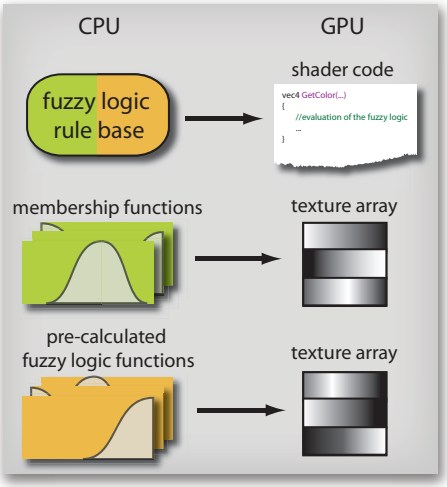


Figure 2: *The fuzzy logic rule base is parsed and translated into shader code on the CPU. Membership functions and pre-calculated fuzzy logic functions are encoded in 1D texture arrays. The GPU makes use of the generated shader program and the texture arrays to perform interactive semantics driven illustrative 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.

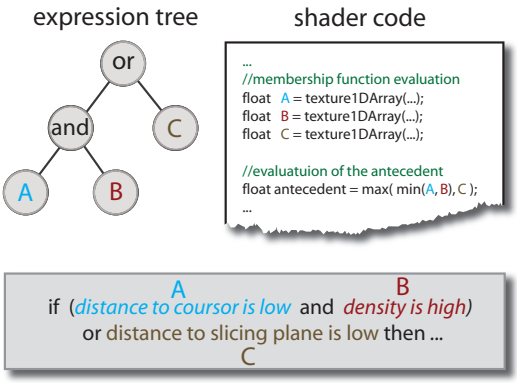


Figure 3: *Shader code generation for the evaluation of antecedents. An expression tree and the corresponding shader code are generated from a simple rule.*

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.

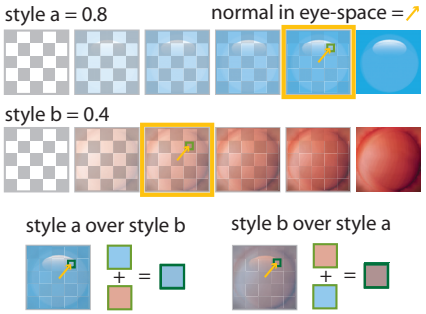


Figure 4: *Example for the compositing of two styles is shown. The exemplary sample has a value of* 0.8 *for style a and a value of* 0.4 *for style b. The corresponding spheres are outlined in yellow.*

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.

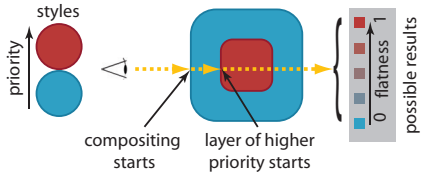


Figure 5: *The flat rendering mode favors samples of higher priority during ray casting. The yellow line depicts a viewing ray. Along the ray common compositing is done until a region of higher priority is reached. The composited color is deemphasized according to the flatness parameter.*

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:

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

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.

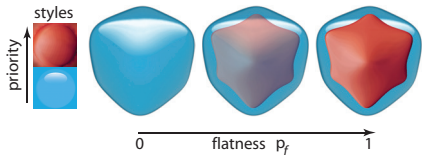


Figure 6: *Example renderings using different values of the flatness parameter. The inner cube has higher priority and is therefore shown for a flatness parameter greater than zero.*

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.

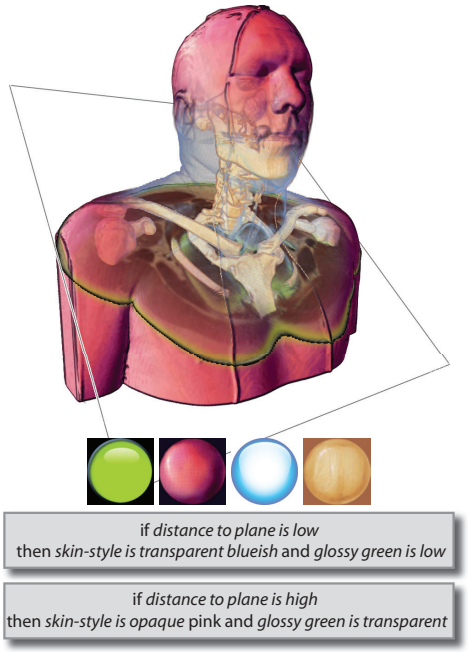


Figure 7: *Rendering of the visible human dataset. A slice plane of the histological data is shown. The CT-data is used for the volume rendering providing the context for the slice plane.*

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.

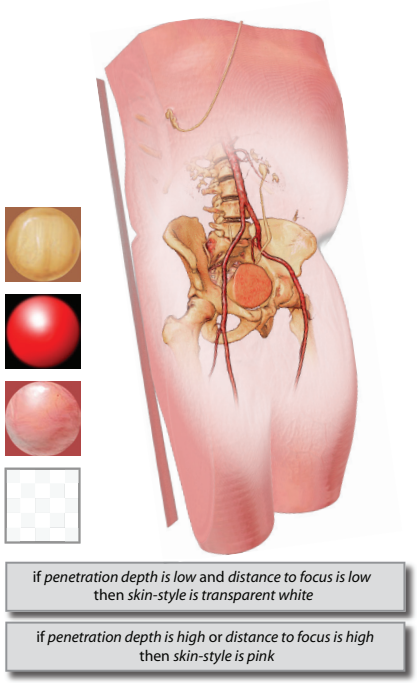


Figure 8: *The mouse cursor defines the user focus. Depending on the user focus the illustrative rendering is altered.*

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.

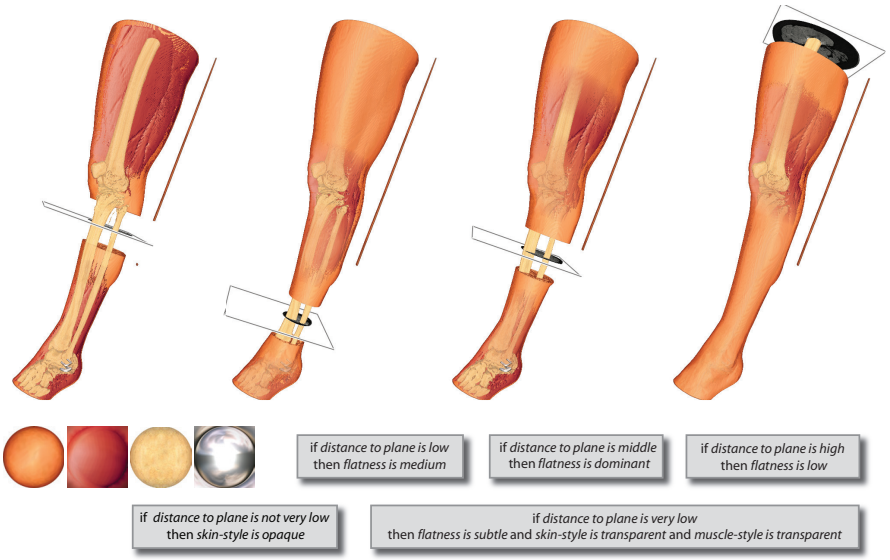


Figure 9: *Rendering of a CT-scan of a human leg. The distance to the slicing plane influences the rendering styles and the flatness parameter. The left rendering shows the flat rendering mode fully applied ignoring the spatial relations.*

[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.

2 LiveSync++ - Enhancements of an Interaction Metaphor

The LiveSync interaction metaphor allows an efficient and nonintrusive integration of 2D and 3D visualizations in medical workstations. This is achieved by synchronizing the 2D slice view with the volumetric view. The synchronization is initiated by a simple picking on a structure of interest in the slice view. In this paper we present substantial enhancements of the existing concept to improve its usability. First, an efficient parametrization for the derived parameters is presented, which allows hierarchical refinement of the search space for good views. Second, the extraction of the feature of interest is performed in a way, which is adapting to the volumetric extent of the feature. The properties of the extracted features are utilized to adjust a predefined transfer function in a feature-enhancing manner. Third, a new interaction mode is presented, which allows the integration of more knowledge about the user-intended visualization, without increasing the interaction effort. Finally, a new clipping technique is integrated, which guarantees an unoccluded view on the structure of interest while keeping important contextual information.

The following paper appears in its original format published as:

P. Kohlmann, S. Bruckner, A. Kanitsar, and M. E. Gröller. LiveSync++: Enhancements of an interaction metaphor. In *Proceedings of Graphics Interface 2008*, pages 81–88, 2008.

LiveSync++: Enhancements of an Interaction Metaphor

Peter Kohlmann* Vienna University of Technology
Stefan Bruckner† Vienna University of Technology
Armin Kanitsar‡ AGFA HealthCare
M. Eduard Gröller§ Vienna University of Technology

ABSTRACT

The LiveSync interaction metaphor allows an efficient and non-intrusive integration of 2D and 3D visualizations in medical workstations. This is achieved by synchronizing the 2D slice view with the volumetric view. The synchronization is initiated by a simple picking on a structure of interest in the slice view. In this paper we present substantial enhancements of the existing concept to improve its usability. First, an efficient parametrization for the derived parameters is presented, which allows hierarchical refinement of the search space for good views. Second, the extraction of the feature of interest is performed in a way, which is adapting to the volumetric extent of the feature. The properties of the extracted features are utilized to adjust a predefined transfer function in a feature-enhancing manner. Third, a new interaction mode is presented, which allows the integration of more knowledge about the user-intended visualization, without increasing the interaction effort. Finally, a new clipping technique is integrated, which guarantees an unoccluded view on the structure of interest while keeping important contextual information.

Keywords: Smart Interaction, Linked Views, Medical Visualization, Viewpoint Selection.

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; J.3 [Life and Medical Sciences]: Medical Information Systems—

1 INTRODUCTION

In the clinical routine the available time to investigate large quantities of patient data, recorded by modern medical imaging modalities (e.g., computed tomography), is very limited. As today's medical datasets may contain several hundreds of slices, it is already very time-consuming to scroll through them. As soon as a potentially pathological area is detected on a slice it can be very helpful to visualize its three-dimensional context. The prime reason which prevents a broad usage of 3D visualizations in the clinical routine is the tedious work to set up all the needed parameters. For example, the rotation in a three-dimensional space to select a good viewpoint is often not very intuitive for people, who are not dealing with computer graphics. Also the definition of a transfer function, which shows the feature of interest and its anatomical context in an expressive way often is a time-consuming trial-and-error process. Another issue which appears by moving from 2D to 3D visualization is occlusion. Clipping planes can be defined to remove structures which occlude the feature of interest. A typical set of clipping planes allows an object-aligned removal along the x-, y-, and z-axis, as well as a near and far clipping along the view direction. Finally the zoom factor for the 3D view has to be adjusted manually.

The goal of the LiveSync interaction metaphor is to derive all the parameters which are needed to present a meaningful 3D view with minimal interaction effort. As the radiologist is usually examining the slices with a keyboard and mouse as input devices, pointing with the mouse on a structure of interest combined with pressing a hot-key is not very intrusive. In earlier work [10] we introduced the basic concept of LiveSync. To encode the quality of a viewpoint for different input parameters, viewing spheres are deformed. A viewing sphere surrounds the entire object and all possible viewpoints are located on the surface of the sphere. The viewing direction points to the center of the sphere. This concept presented the basic building blocks to achieve the live synchronization of a 2D slice view and a volumetric view. In this paper we improve and extend the previously presented techniques with the following contributions:

Sphere parameterization: An efficient sphere parameterization is used to encode the viewpoint quality for the input parameters. With a parameterization in polar coordinates, the distances between neighboring points vary a lot, depending on their distance to the poles. A multi-resolution approach which samples the sphere with uniformly distributed points is used. This allows hierarchical refinement of the sampling for efficient calculations of the overall viewpoint quality. Beside its efficiency regarding memory consumption and performance, it can be employed to find a good viewpoint that is surrounded by other viewpoints which are estimated as good ones.

Feature extraction: The extraction of a feature of interest is a crucial part for finding good viewpoints. Our region growing based segmentation automatically determines the size of the region which is necessary to specify the shape of the object of interest.

Transfer function: The manual setup of a transfer function is rather time-consuming and often not very intuitive. With increasing dimension of the transfer function space the definition becomes more and more complicated. For alleviation, the user can choose from a predefined set of transfer functions which are adjusted for typical examination procedures. With the knowledge about the distribution of scalar values within the feature of interest, it is possible to fine-tune a predefined transfer function. This is especially important when a structure is picked on the slice which is not visible in the 3D view with the currently defined transfer function.

Interaction modes: An informal evaluation indicated that a single click on the slice often results in a very good view on the interesting object in the 3D view. However, the size of the area of interest is hard to depict without further user interaction. As it is preferred to keep the user effort as low as possible, the time of keeping a hot-key pressed is taken to interactively increase the area of interest. The view is updated in short time intervals and the user can decide when an intended view is reached. In this mode a mask with the current segmentation result can be displayed to inspect the region growing process. An automatically generated history helps the user to restore and to keep previously generated views. This is especially helpful to review the intermediate states which are generated in this new interaction mode.

*e-mail: kohlmann@cg.tuwien.ac.at
†e-mail: bruckner@cg.tuwien.ac.at
‡e-mail: armin.kanitsar@agfa.com
§e-mail: groeller@cg.tuwien.ac.at

Feature-driven clipping: In our previous paper [10] only the setup of a view-aligned clipping plane was supported. The performed evaluation showed that often object-aligned clipping planes are preferred, especially if there is further interaction necessary to manually fine-tune the view. In the presented approach the user can choose if the view-aligned or the object-aligned clipping planes are set automatically to remove occluding structures, while preserving as much contextual information as possible. To increase the degree of preservation, importance-driven clipping techniques are integrated.

This paper is structured as follows: In Section 2, the relevant previous work is discussed. An overview of the LiveSync workflow and functionalities is given in Section 3. Section 4 describes the parameterization of the spheres. In Section 5, it is first shown how the extent of the region growing-based segmentation and the other growing parameters are derived automatically. In the following, an approach is presented to fine-tune a predefined transfer function by taking the distribution of scalar values within the feature of interest into account. Section 6 introduces the different interaction modes and Section 7 describes the feature-driven clipping. Section 8 gives numbers about the performance of the live synchronization and presents qualitative feedback from users. Finally, Section 9 concludes the paper.

2 Related Work

A major task of LiveSync is the computation of viewpoint quality. The selection of a good viewpoint is a well-investigated research area for polygonal scenes. Only recently research on viewpoint selection for volumetric data originated. Kohlmann et al. [10] presented the first attempt to combine optimal viewpoint estimation and synchronized views for the visualization of medical volume data. Mühler et al. [13] presented an interesting approach for the generation of animations for collaborative intervention planning and surgical education. For the generation of these animations the selection of good viewpoints based on polygonal data in a medical context is crucial.

In the scope of volumetric data Bordoloi and Shen [2] introduced an approach based on entropy known from information theory. With their technique it is possible to determine a minimal set of representative views for a given scene. For the viewpoint selection process they take the distribution of the underlying data, the transfer function, and the visibility of voxels into account. Takahashi et al. [16] proposed a feature-driven approach for the detection of an optimal viewpoint. As a first step they define locally optimal viewpoints for identified feature components. Taking these local viewpoints into account, they compute a viewpoint which fulfills global viewpoint quality criteria. An importance-driven approach to focus on objects of interest within segmented volume data is presented by Viola et al. [19]. An expressive view on the user-selected object is generated automatically by their system. Chan et al. [5] focused on the viewpoint selection for angiographic volume data sets. They define view descriptors for visibility, coverage, and self-occlusion of the vessels to determine a globally optimal view. The final view is selected by a search over a viewpoint solution space.

To define a volume of interest (VOI) in volumetric data several approaches were presented within the last years. Tory and Swindells [17] presented ExoVis for the definition of cutouts which can be displayed with different rendering styles or transfer functions. A sketching-based approach where the user defines the interesting region by painting 2D strokes along the contour of the interesting structure is presented by Owada et al. [14]. The RGVis techniques introduced by Huang and Ma [7] present a 3D region growing approach to assist the user in locating and defining VOIs. They perform partial region growing to generate a transfer function, which reveals the full feature of interest.

Another work on semi-automatic generation of transfer functions was presented by Kindlmann and Durkin [9]. They assume that the boundary regions between relatively homogeneous materials are the areas of interest in the scalar volume. After generating a three-dimensional histogram using the scalar values, the first and the second directional derivatives along the gradient direction they can generate the opacity transfer function. A cluster-space approach for the classification of volume data was presented by Tzeng and Ma [18]. In a preprocessing step they transform volumetric data into a cluster space representation. This provides an intuitive user-interface which allows the user to operate in this cluster space. Rezk-Salama et al. [15] presented an approach where high-level transfer function models designed by visualization experts can be controlled by a user interface which provides semantic information. The non-expert user only has to adjust sliders for a goal-oriented setup of a suitable transfer function.

There are various approaches for the rendering and accentuation of identified features. Zhou et al. [22] use a geometric shape like a sphere to divide the volume into a focal and a context region. For rendering they combine direct volume rendering (DVR) with non-photorealistic rendering (NPR) techniques. Marchesin et al. [12] introduced locally adaptive volume rendering to enhance features. They modify the traditional rendering equation to improve the visibility of selected features independently of the defined transfer function. Huang et al. [8] presented an automatic approach to generate accurate representations of a feature of interest from segmented volume data. They construct a mesh for the boundary of the volumetric feature to enable high-quality volume rendering.

Regarding sphere parameterization, there is various research on how to distribute points evenly on the surface of a sphere. Bourke [3] presented an approach which distributes an arbitrary number of points over the surface of a sphere based on the standard physics formula for charge repulsion. Leopardi [11] worked on the partition of the unit sphere into regions of equal area and small diameter. Górski et al. [6] presented HEALPix which is a framework for high-resolution discretization and fast analysis of data distributed on the sphere. A hierarchical equal area iso-latitude pixelization produces a subdivision of the sphere where each pixel covers the same surface area.

3 LiveSync Workflow

LiveSync aims to provide an optimal setup of the view parameters for the volumetric view with little additional user interaction. Manually specifying a good 3D view for structures detected in cross-sectional images is a time-consuming task. The user has to edit many parameters, such as camera position and clipping planes in order to get an expressive visualization. This process has to be repeated for each new structure of interest. LiveSync makes the process of diagnosis more efficient by automatically generating good 3D views for interactively picked structures. This functionality can be activated on demand by pressing a hot-key while pointing the mouse over a structure of interest on the 2D slice. Depending on the quality of the instantly generated result, the user can manually refine the view by adjusting the view parameters. This concept allows an efficient and non-intrusive integration of 2D and 3D visualizations in medical workstations.

Figure 1 gives an overview on the LiveSync workflow. Based on a picking action on a slice, a set of relevant view input parameters is extracted. In the original system these parameters are *patient orientation*, *viewpoint history*, *local shape estimation* and *visibility*. To get a unified representation of the parameters regarding viewpoint quality, a viewing sphere is deformed for each of them. Each of these deformed spheres contains information about the viewpoint quality. They are combined to encode an estimation about the overall view goodness. The combined sphere now contains information on how to set up the viewpoint for the best estimated view on the

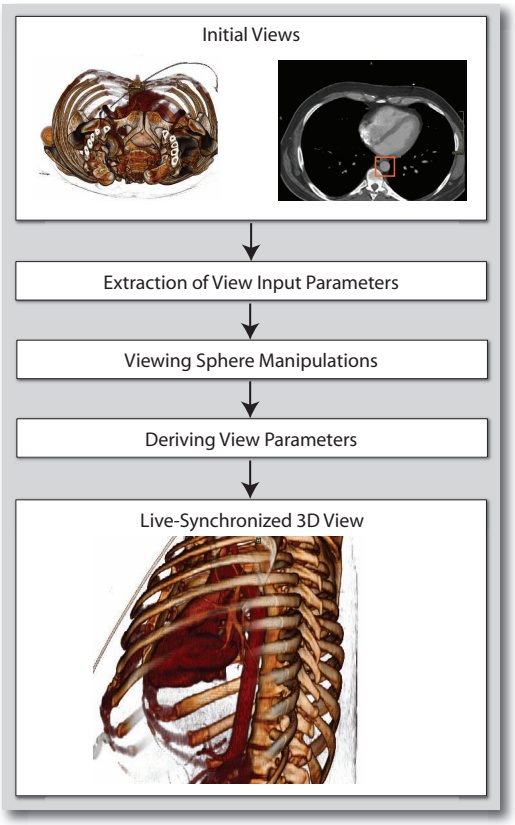


Figure 1: LiveSync workflow: The initial views are a 2D slice image and a volumetric view. A picking action on the structure of interest (surrounded by the rectangle) in the slice, starts the deformation of viewing spheres according to automatically derived view input parameters. These parameters are combined to derive the view parameters for setting up the volumetric view. As result an expressive live-synchronized 3D view is generated.

structure of interest. After the automatic setup of the clipping plane and the zoom factor, the live-synchronized view is provided.

The deformation of the spheres works as shown in Figure 2. For viewpoints which are indicated to be good ones based on one of the view input parameters, the radial distance of the surface point is increased. In Figure 2, the fully opaque eye indicates a good, the half-transparent one an average-rated, and the crossed out one a rather bad viewpoint.

The conceptual design of LiveSync is not limited to a certain number of input parameters. It can be easily extended by defining more parameters which influence the viewpoint quality. This is facilitated by the unified representation of the encoding of the viewpoint quality. Different weights can be assigned to the view input parameters to control their influence on the combined sphere. For more details on the general workflow of the LiveSync concept, we refer to previous work [10].

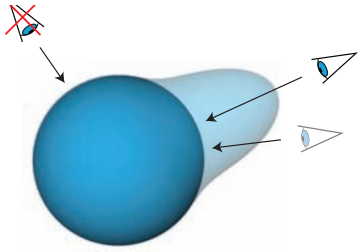


Figure 2: The quality of a viewpoint is encoded into the radial distances from a unit sphere. Good viewpoints are located at positions on the unit sphere which refer to high radial distances on the deformed sphere.

4 Sphere Parameterization

As the viewpoint quality of all view input parameters is encoded in a sphere parameterization, a smart way for accessing points on the surface of the sphere is important regarding performance and memory efficiency. In the original implementation there was the problem that the spheres for the view input parameters were sampled differently. Information about patient orientation, viewpoint history, and local shape information was analytically described. Due to expensive calculations, the visibility computation had to be performed in a discrete manner for a precomputed set of points on the surface of the sphere. This set of points was much smaller than for the other parameters. The computed radial distances for all spheres were stored in two-dimensional arrays with 360×180 elements via direct latitude-longitude mapping. This mapping results in a very uneven distribution of points on the surface of the sphere with much higher sampling close to the poles. Another issue has been the combination of differently sampled spheres.

4.1 Visibility Calculations

For a better understanding of the need for evenly distributed points, this section will provide a brief review of the performed calculations to generate the visibility viewing sphere. Good visibility is given if a ray from the picked point to the viewpoint exits the object of interest within few steps and if the distance until the structure of interest is occluded by other structures is high. In this case, there is high flexibility for positioning a clipping plane to remove occluding structures while preserving important anatomical context around the picked object.

Figure 3 illustrates how rays are cast from the picked point on the structure of interest to uniformly distributed points on the sphere. It has to be detected at which distance the ray exits the structure of interest. This is done by analyzing the distribution of scalar values and gradient magnitudes to decide if a voxel belongs to the structure. After a ray exits the structure of interest, opacities depending on the underlying transfer function are accumulated to detect occluding structures. As soon as a small opacity threshold is reached, the computation for a ray is terminated.

4.2 Sphere Partitioning

To achieve interactive performance the visibility calculations cannot be performed for 360×180 points. However, it is inaccurate to sample only a small number of points and to combine the sparsely sampled sphere with other, higher-sampled spheres. Moreover, taking the point with the highest radial distance after the sphere combination and before filtering, does not always lead to satisfying results. The structure of interest may be only visible through a small keyhole, and its larger part is hidden by occluding structures. In the ideal case it is very helpful for further interactions with the 3D view

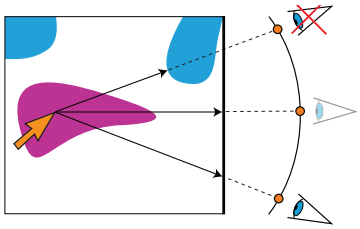


Figure 3: Starting from the picked position, visibility rays are cast to points which are equally distributed on the surface of the viewing sphere. Samples along the rays are analyzed to detect (a) when it exits the structure of interest and (b) at which position the object gets occluded by other anatomical structures.

if the provided viewpoint offers a good *view stability*. This criterion defined by Bordoloi and Shen [2] describes the maximum change in a certain view caused by small shifts of the camera position.

The capabilities of the HEALPix package [6] offer good properties for the existing needs if the techniques are applied in an elaborate way. Figure 4 (top left) shows the HEALPix base partitioning of the surface of a sphere into 12 equally sized areas with dots indicating their center positions. In each subdivision step a partition is divided into four new partitions. A nested indexing scheme can be utilized for the fast access of an arbitrary surface point at different partitioning resolutions.

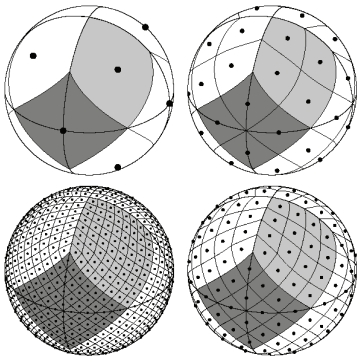


Figure 4: In its base partitioning a HEALPix sphere is divided into 12 equally sized areas. In each subdivision step an area is further divided into 4 areas (image courtesy of [6]).

Applied to the requirements of the LiveSync viewpoint quality encoding strategy, the whole sphere is sampled at an initial resolution for all view input parameters. Following, a partition of the sphere, holding the samples which indicate the best viewpoint quality is identified. This is done by summing up the radial distances of the sample positions for each partition. To achieve higher resolution, this peak partition is sampled more densely. The process of identifying a peak sub-partition with a successive hierarchical refinement can be repeated if higher resolution is required. To provide good view stability, a final filtering of the points in the target area is performed. The filtered peak is provided as the estimated best viewpoint regarding the view input parameters.

In the presented approach an initial resolution of 3076 points over the whole sphere is chosen, which corresponds to a 3.66 degrees angular distance between neighboring points. After hierarchical refinement, a peak area with 1024 sample points (0.55 degrees angular distance) is identified and low-pass filtering is performed. This approach allows high flexibility concerning the sampling of the sphere, to detect a partition which contains a collection of many good viewpoints.

5 Feature-Driven Transfer Function Tuning

A critical point in the LiveSync workflow is the definition or the extraction of the feature of interest. The shape of the feature is important to determine a good viewpoint. For example, if a blood vessel is picked, the user should be provided with a view which shows the course of the vessel and does not cut through it. This section first describes how the parameters for the region growing are controlled, before focusing on how the transfer function can be tuned with knowledge about the extracted feature.

5.1 Feature Extraction

A natural choice to segment the object around the picked point is region growing. The picked point is transformed to a voxel in 3D as seed position. Huang and Ma [7] presented with RGVis a cost function which determines if a visited voxel during the region growing process is a member of the region or not. In this approach the neighborhood of the seed point is analyzed regarding scalar value and gradient magnitude distributions to initiate the parameters of the cost function. If the seed is located close to a boundary, the growing captures the boundary of the object, whereas a seed point within a homogeneous area results in a more compact growing process.

Figure 5: The optional rendering of the segmentation mask enhances the structure of interest.

In previous work [10] the growing process was limited to a fixed region of $32 \times 32 \times 32$ voxels. In the presented approach the expansion is influenced by the spatial distribution of the points marked as region members so far. Growing progresses until the object-oriented bounding box (OOBB) of the included voxels reaches a certain limit. During the growing process the OOBB is updated at variable intervals to estimate how many more voxels may be added until the limit is reached. This strategy is superior to controlling the growing by the number of detected member voxels. The behavior of the growing process regarding the spreading of the points depends on various aspects. If just the boundary of a structure is segmented, a much lower number of voxels is sufficient to estimate

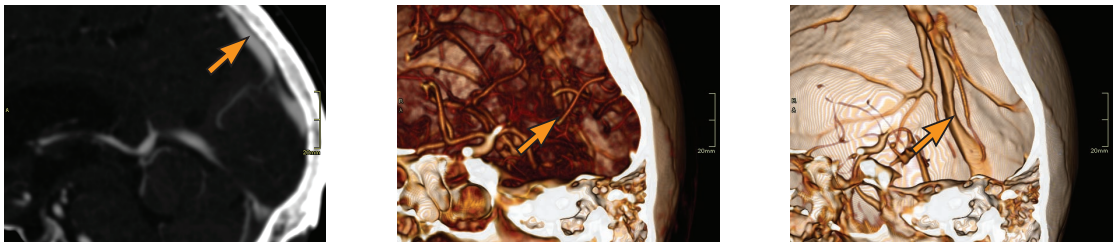


Figure 6: Left: The position of the picking on the slice. Middle: The structure of interest is hidden to a large extent by blood vessels. Right: Allowing LiveSync to fine-tune the transfer function automatically leads to an unoccluded view of the sinus vein.

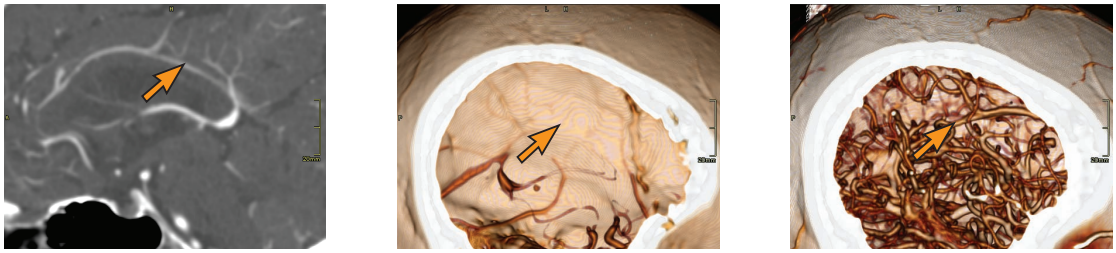


Figure 7: Left: The position of the picking on the slice. Middle: The structure of interest is not visible with the current setting of the opacity transfer function. Right: Allowing LiveSync to fine-tune the transfer function assigns opacity to the small vessels to make them visible.

Figure 8: Left: The position of the picking on the slice. Middle: Initial view with a default transfer function. Right: LiveSync-adjusted transfer function which enables a good view on the metatarsal.

the feature shape, than if the growing is performed in a very homogeneous area. Also with thin structures like blood vessels a small number of object voxels can already indicate the shape, whereas more voxels are needed for more compact areas.

As soon as the region growing stops, a principal component analysis (PCA) is performed on the member voxels to extract the three feature vectors and the corresponding eigenvalues. A metric of Westin et al. [21] is used to measure the local shape of the segmented feature from the relation between the eigenvalues. This metric allows to classify if a structure has an isotropic, a planar or linear shape. By labeling the segmented voxels with a segmentation mask, this information can be displayed in the live-synchronized volumetric view to highlight the structure of interest for a clear visual separation from its context. Figure 5 shows this enhanced visualization as result of a picking on the metatarsal in the slice view.

5.2 Transfer Function Tuning

It is frequently stated, that the definition of a transfer function to assign color and opacity to the data values, is a time-consuming and not very intuitive task. Even for the developer of a transfer function editor it often ends in a tedious trial-and-error process to generate the desired result. Usually, the necessary effort increases with the dimensions of the transfer function.

LiveSync aims to generate the 3D visualization without any additional interaction than the picking. Even the control of sliders for the transfer function design as presented by Rezk-Salama et al. [15] might lead to an unwanted distraction in the diagnosis process. The presented system allows the user to choose from a set of predefined transfer function templates, which are very well-tailored for different types of examinations. The transfer function is defined by a color look-up table and a simple ramp which assigns zero opacity to scalar values from zero to the start of the slope, increasing opacity along the slope, and full opacity from the peak of the slope to the end of the scalar range.

Inspired by the definition of a transfer function based on partial region growing in the work by Huang and Ma [7], knowledge about the distribution of scalar values within the extracted object is utilized to fine-tune an existing transfer function. This feature can be activated on demand and is especially helpful if the structure of interest is occluded to a large extent by densely surrounding objects. It is also helpful if a structure selected on the slice is hardly visible or not visibility at all applying the current opacity transfer func-

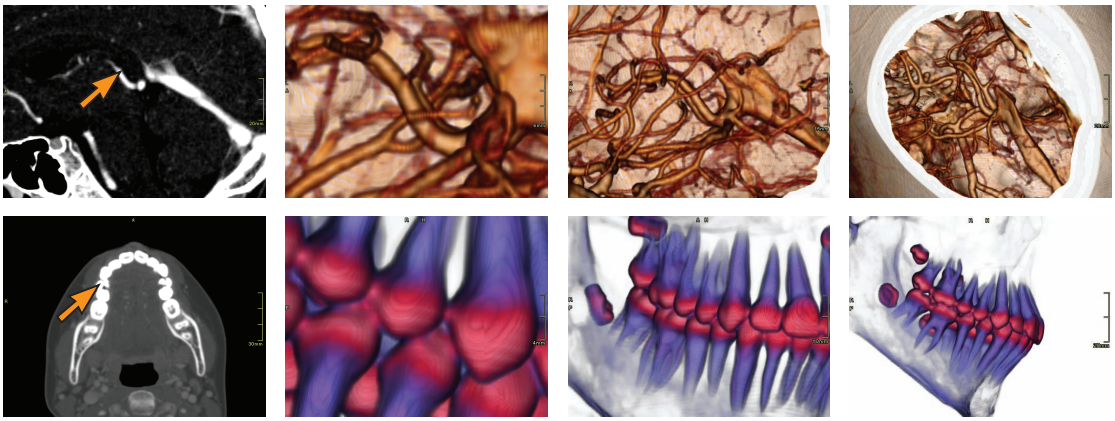


Figure 9: The slice images highlight the picking positions. Initiated by these pickings the 3D views are live synchronized at equal time intervals. Images are provided for increasing OOBB diagonal lengths from left to right.

tion. The adjustment is based on the mean value and the standard deviation of the feature's scalar values which were collected by the region growing. The center of the slope is set to the mean value and the width is adjusted to three times the standard deviation.

Figure 6 shows how the automatic setup of the ramp adjusts the transfer function. The structure of interest is visible to a much larger extent by making the surrounding blood vessels transparent. The opposite case is shown in Figure 7 as the current transfer function setup does not assign enough opacity to the smaller blood vessels. By moving the center and changing the width of the ramp which defines the opacity transfer function, the object of interest is clearly visible in the volumetric view. Figure 8 shows the result of automatic transfer function adjustment to provide a good view on the metatarsal.

6 Interaction Modes

The primary conceptual goal for the design of LiveSync was to interactively offer a synchronized 3D view. This shall be achieved without the need to manually adjust all the parameters to generate an expressive view. Therefore a non-intrusive user interaction technique has to be implemented which lets the user focus on his primary analysis goal. In this section, the value and the limitation of the LiveSync picking interaction will be described and extended to integrate more knowledge about the intentions of the user.

6.1 LiveSync Mode

The evaluation with a radiology technician in previous work [10] showed that the intuitive picking was sufficient to generate good results. One comment was that the handling and the mouse-over/hot-key interaction is very intuitive. However, it is rather difficult to derive in which part of an anatomical structure the user is interested. If the object is very small like a lung nodule or a polyp in the colon, the user will be provided with a view which shows the whole feature within its three-dimensional context. If the volumetric extent of the feature is rather large, like e.g., the sinus vein, then the user might either be interested in getting a close-up view of the structure around the picked point, or a view which shows the whole vein is preferable. The current zoom factor of the slice view controls the zoom factor of the volumetric view, as it indicates the size of the anatomical structure the user is interested in. To integrate more knowledge about the size of the structure, the goal was to extend the functionality of LiveSync without introducing new interaction methods to the user.

6.2 LiveSync++ Growing Mode

Taking into consideration that region growing is performed to segment the structure of interest, the natural choice was to let the user control the size of the area of interest by keeping the hot-key pressed. This strategy allows to control the volumetric extent of the region growing. As mentioned in Section 5.1, OOBBs are computed during the growing process to estimate the extent of the initial growing. This is used to derive a reliable estimation of the local feature shape. By keeping the hot-key pressed, the region keeps on growing and the 3D view is updated in fixed time intervals. Within each interval, the growing process is continued until the diagonal length of the OOBB of the so far segmented set of voxels reaches the next predefined target size.

Integrating the knowledge about the user-indicated size of the area of interest, this can be utilized to improve the live-synchronized volumetric view regarding the demands of the user. After each growing step, the viewpoint is optimized, the clipping planes are adjusted and the zoom for the 3D view is changed. Figure 9 shows intermediate steps of the growing process. These views are provided to the users while they keep the hot-key pressed. Whenever a satisfying result is reached a release of the key stops the update of the volumetric view. There is a smooth zoom out provided for each step as the user wants to examine a larger area of interest. The zoom factor is computed by projecting the OOBB of the currently segmented set of voxels on the screen, and allowing it to cover a certain amount of the available display area, e.g., 50%.

The growing mode generates live-synchronized views at short update intervals. The user releases the hot-key or continues with the mouse movement to stop the update process. To prevent that the users miss and lose a view they like, a history mode is implemented. All live-synchronized views, including the intermediate views, are stored automatically in a view history. This is achieved by storing the adjusted parameters for the current view. The user can select a previous view from a list and store the best ones permanently.

7 Feature-Driven Clipping

Viewpoint selection is critical for generating the live-synchronized 3D view. In most cases interesting structures are surrounded by other tissues. These can be removed by the definition of an adjusted

transfer function or, more straight-forwardly, by the definition of a clip geometry. In this section, first the automatic setup of conventional clipping planes is explained before focusing on clipping planes which enable a higher degree of context preservation.

7.1 LiveSync Clipping Strategies

Typically, a medical workstation offers view-aligned and object-aligned clipping planes to interact with. As LiveSync aims to minimize the user's effort, the clipping is performed automatically to provide an unoccluded view on the structure of interest. Based on the visibility calculations described in Section 4.1, a position in the volume to place the clipping planes is computed. The plane is placed along the visibility ray where the structure of interest is not yet occluded by other structures. In our previous work [10] only view-aligned clipping planes were supported. A plane which is orthogonal to the viewing direction is moved to the clipping position and the raycasting process starts where this plane intersects the volume. This strategy provides a good view on the structure of interest. Problems might occur if the user wants to perform further interactions with the 3D view after the live-synchronized view is generated. In the worst case, the structure of interest is clipped away by manual changes of the viewpoint.

Object-aligned clipping planes are moved along the three main patient axes. The six available clipping planes can clip from left, right, anterior, posterior, head and feet. To decide which of these planes has to be set to allow an unoccluded view on the structure of interest, the plane which is most perpendicular to the viewing direction has to be identified. Then this plane is placed at the appropriate clipping position along the visibility ray. An object-aligned clipping plane remains at its position when the viewpoint is changed manually and cannot clip away the structure of interest unintentionally.

7.2 LiveSync++ Smooth Importance-Driven Clipping

The drawback of conventional clipping strategies is that they often remove anatomical context which does not occlude the object of interest. Viola et al. [20] introduced importance-driven volume rendering to remedy this problem. Their strategy uses segmentation information to generate a clipping region which follows the shape of the focus object. This approach relies on a complete segmentation of the structure in question. As our method relies on incomplete information in order to permit parameter-less interaction, the method presented by Viola et al. can lead to misleading results as a full segmentation is not available.

To remedy this problem, while still providing more context information for the picked structure, we propose an extension of the context-preserving rendering model of Bruckner et al. [4]. The advantage of this approach is that it still follows the semantics of clipping planes, but retains salient features which are important for orientation.

We compute the opacity α at each sample point along a viewing ray in the following way:

$$\alpha = \begin{cases} \alpha_{tf}\left(|g|\left(1-|n\cdot v|\right)\right)^{(1-(d-c))^{\kappa}} & \text{if outside picked region} \\ \alpha_{tf} & \text{otherwise.} \end{cases}$$

where α_{tf} is the opacity as specified in the transfer function, $|g|$ is the gradient magnitude, n is the normalized gradient vector $(g/|g|)$, v is the view vector, d is the distance along the viewing ray, c is the location of the view-aligned clipping plane, and κ is a parameter which controls the transition between clipped and unclipped regions. The range of d and c is [0..1], where zero corresponds to the sample position closest to the eye point and one corresponds to the sample position farthest from the eye point. Opacity is only modulated in regions outside the focus structure identified by our local segmentation procedure. The adjustment is based on the idea of smooth clipping: Instead of having a sharp cut, it smoothly fades out structures that would be clipped otherwise. As the gradient magnitude indicates the "surfaceness" of a sample, its inclusion causes homogeneous regions to be suppressed. Additionally, since contours give a good indication of the shape of a three-dimensional structure, while being visually sparse (i.e., they result in less occlusion), the gradient magnitude is multiplied by $(1-|n\cdot v|)$. The parameter κ controls the transition between clipped and unclipped regions. In all our experiments, a value of 16 has proven to be most effective. Thus, this parameter does not require user adjustment and therefore does not introduce additional complexity.

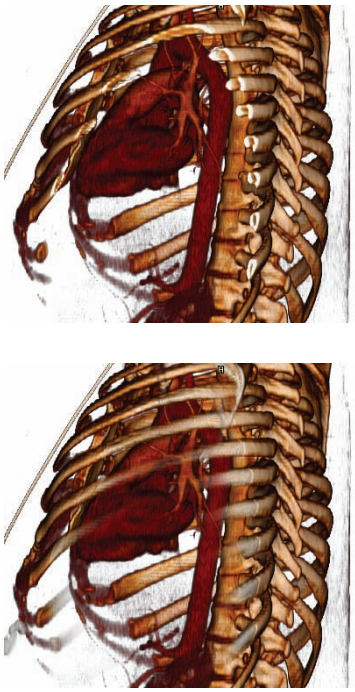


Figure 10: Top: Conventional clipping leads to a hard cut through the ribs. Bottom: Smooth clipping preserves anatomical context by reducing the opacity of the ribs which are not occluding important parts of the aorta.

While we already achieve satisfactory results using this context-preserving clipping model, pathological cases might still cause an occlusion of the picked structure. Therefore, we additionally reduce the previously accumulated color and opacity along a ray on its first intersection with the picked region by a factor of $0.5\left(1-\alpha_{tf}\right)$. This lets the picked region always shine through – similar to ghosted views commonly found in medical illustrations – even if it is occluded by other structures. An example of this clipping technique is shown in Figure 10. The picking was performed on the aorta in the slice view. In the top image there is a hard cut through the ribs, whereas the smooth clipping shows the ribs which are not occluding the interesting part of the aorta with decreased opacity.

8 Performance and Qualitative Results

All computations which are done to derive the parameters to set up a live-synchronized 3D view can be performed interactively. The performance was measured on a PC configured with an

AMD Athlon 64 Dual Core Processor 4400+ and 2 GB of main memory. Compared to previous work [10] where the region growing was fixed to a limit of $32 \times 32 \times 32$ voxels, the performance is more strongly influenced by the growing process. The size of the growing area is calculated adaptively. In the vast majority of the cases it is possible to stop the growing process much earlier than with a fixed growing area and the estimation of the local feature shape is more reliable.

Besides offering better view stability, the efficient and flexible sphere parameterization helps to eliminate the need for precomputing uniformly distributed points on the sphere, to reduce memory consumption, and to increase the overall performance. The LiveSync-related computations are averaging between 50 ms and 100 ms by measuring with different data sets and picking on various structures. In growing mode as described in Section 6.2, the time to compute the intermediate views is in the same range as for the instant updates.

In previous work [10] an informal evaluation with an experienced radiology assistant has been performed. The feedback about the handling of the picking interaction and the live-synchronized views was very positive. The extensions presented in this paper are designed based on comments for possible improvements. As the LiveSync feature is integrated into a real world medical workstation which is under development by our collaborating company partner, its functionality is frequently presented in demonstrations to radiologists. Their feedback is very positive as well, and they confirm the high practical value of this feature.

9 Conclusion

Live synchronization of the volumetric view from non-intrusive interaction with a slice view, is a powerful concept with the potential to improve the efficiency of diagnosis in clinical routine. In this paper we presented several enhancements of the existing concept. Our goal was to integrate more knowledge about the anatomical structure the user is interested in to provide an expressive 3D visualization. As the key of the LiveSync concept is to keep the user interaction as small as possible, we did not want to increase the interaction effort or to introduce new interaction techniques.

A more efficient parameterization for the derived view input parameters was integrated to allow a hierarchical refinement of the search space for the best estimated viewpoint. This parameterization is utilized to provide a view with a high degree of view stability. An adaptive region growing was presented that allows to extract the part of the feature which is needed to estimate its local shape in a reliable way. Information about the scalar value distribution within the feature of interest is used to fine-tune the existing opacity transfer function. This helps to enhance the visibility of the object the user is interested in. The growing interaction mode integrates knowledge about the volumetric extent of the area of interest without increasing the user interaction. Especially an appropriate zoom factor for the 3D view can be derived from this information. A new clipping technique was integrated to increase the preservation of valuable anatomical context while guaranteeing an unoccluded view on the structure of interest.

Acknowledgements

The work presented in this paper has been funded by AGFA HealthCare in the scope of the `DiagVis` project. We thank Rainer Wegenkittl, Lukas Mroz and Matej Mlejnek (AGFA HealthCare) for their collaboration and for providing various CT data sets. Additional data sets are courtesy of OsiriX's DICOM sample image sets website [1].

References

[1] OsiriX Medical Imaging Software. Available online at http://www.osirix-viewer.com, December 2007.

[2] U. D. Bordoloi and H.-W. Shen. View selection for volume rendering. In *Proceedings of IEEE Visualization 2005*, pages 487–494, 2005.

[3] P. Bourke. Distributing Points on a Sphere. Available online at http://local.wasp.uwa.edu.au/~pbourke/geometry/spherepoints/, December 2007.

[4] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569, 2006.

[5] M.-Y. Chan, H. Qu, Y. Wu, and H. Zhou. Viewpoint selection for angiographic volume. In *Proceedings of the Second International Symposium on Visual Computing 2006*, pages 528–537, 2006.

[6] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622:759–771, 2005.

[7] R. Huang and K.-L. Ma. RGVis: Region growing based techniques for volume visualization. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 355–363, 2003.

[8] R. Huang, H. Yu, K.-L. Ma, and O. Staadt. Automatic feature modeling techniques for volume segmentation applications. In *Proceedings of IEEE/Eurographics International Symposium on Volume Graphics 2007*, pages 99–106, 2007.

[9] G. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings of the IEEE Symposium on Volume Visualization 1998*, pages 79–86, 1998.

[10] P. Kohlmann, S. Bruckner, A. Kanitsar, and M. E. Gröller. LiveSync: Deformed viewing spheres for knowledge-based navigation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1544–1551, 2007.

[11] P. Leopardi. A partition of the unit sphere into regions of equal area and small diameter. *Electronic Transactions on Numerical Analysis*, 25:309–327, 2006.

[12] S. Marchesin, J.-M. Dischler, and C. Mongenet. Feature enhancement using locally adaptive volume rendering. In *Proceedings of IEEE/Eurographics International Symposium on Volume Graphics 2007*, pages 41–48, 2007.

[13] K. Mühler, M. Neugebauer, C. Tietjen, and B. Preim. Viewpoint selection for intervention planning. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization 2007*, pages 267–274, 2007.

[14] S. Owada, F. Nielsen, and T. Igarashi. Volume catcher. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2005*, pages 111–116, 2005.

[15] C. Rezk-Salama, M. Keller, and P. Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1021–1028, 2006.

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

[17] M. Tory and C. Swindells. Comparing ExoVis, orientation icon, and in-place 3D visualization techniques. In *Proceedings of Graphics Interface 2003*, pages 57–64, 2003.

[18] F.-Y. Tzeng and K.-L. Ma. A cluster-space visual interface for arbitrary dimensional classification of volume data. In *Proceedings of Joint Eurographics - IEEE TCVG Symposium on Visualization 2004*, pages 17–24, 2004.

[19] I. Viola, M. Feixas, M. Sbert, and M. E. Gröller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):933–940, 2006.

[20] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, 2005.

[21] C.-F. Westin, A. Bhalerao, H. Knutsson, and R. Kikinis. Using local 3D structure for segmentation of bone from computer tomography images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1997*, pages 794–800, 1997.

[22] J. Zhou, M. Hinz, and K. D. Tönnies. Focal region-guided feature-based volume rendering. In *Proceedings of the 1st International Symposium on 3D Data Processing, Visualization, and Transmission 2002*, pages 87–90, 2002.

3 Obscurance-based Volume Rendering Framework

Obscurances, from which ambient occlusion is a particular case, is a technology that produces natural-looking lighting effects in a faster way than global illumination. Its application in volume visualization is of special interest since it permits us to generate a high quality rendering at a low cost. In this paper, we propose an obscurancebased framework that allows us to obtain realistic and illustrative volume visualizations in an interactive manner. Obscurances can include color bleeding effects without additional cost. Moreover, we obtain a saliency map from the gradient of obscurances and we show its application to enhance volume visualization and to select the most salient views.

The following paper appears in its original format published as:

M. Ruiz, I. Boada, I. Viola, S. Bruckner, M. Feixas, and M. Sbert. Obscurance-based volume rendering framework. In *Proceedings of Volume Graphics 2008*, pages 113–120, 2008.

Obscurance-based Volume Rendering Framework

M. Ruiz [1], I. Boada [1], I. Viola [2], S. Bruckner [3], M. Feixas [1], and M. Sbert [1]

[1]Graphics and Imaging Laboratory, University of Girona, Spain
[2]Department of Informatics, University of Bergen, Norway
[3]Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

Abstract

Obscurances, from which ambient occlusion is a particular case, is a technology that produces natural-looking lighting effects in a faster way than global illumination. Its application in volume visualization is of special interest since it permits us to generate a high quality rendering at a low cost. In this paper, we propose an obscurance-based framework that allows us to obtain realistic and illustrative volume visualizations in an interactive manner. Obscurances can include color bleeding effects without additional cost. Moreover, we obtain a saliency map from the gradient of obscurances and we show its application to enhance volume visualization and to select the most salient views.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Global illumination is a well-known technique for producing realistic scenes. However, although it might play a decisive role in 3D volume visualization since it provides visual cues that enhance data interpretation, its application is still challenging in direct volume rendering. The main limiting factor is the computational cost of simulating global effects of light in a volume, making interactive exploration difficult [Max95]. On the other hand, illustrative methods aim at creating visualizations which convey information to the viewer as opposed to physically correct light interaction. Volume illustration enhances the expressiveness of volume rendering by highlighting important features within a volume while subjugating insignificant details and rendering the result in a way that resembles an illustration [ER00] . Ideally, a volume rendering system should be able to support both realistic and illustrative renderings.

Obscurances have been introduced by Zhukov et al. [ZIK98] and Iones et al. [IKSZ03] as an efficient technique that gives perceptually similar results to global illumination with a small fraction of the computational cost. Moreover, without adding computational cost, obscurances also allow us to compute *color bleeding*, which consists in the effect that the objects around another object with intense coloration are dyed with this color [MSC03]. The obscurance technique was first used in video-game environments. Its application to volume rendering, called vicinity shading, was introduced by Stewart [Ste03].

In this paper, we present an obscurance-based volume rendering system that allows to obtain realistic and illustrative volume visualizations in an interactive manner. One important aspect of our work shows that obsurances are not only useful for realistic depiction but also for illustrative rendering. As obscurances can be interpreted as general information about the neighborhood of a voxel, they can be used as a bias for the generation of more expressive illustrative depictions of a data set (see Figure 1).

Saliency typically arises from contrasts between items and their neighborhood [IK01, TIR05, vdWGB06] and it is considered that the most salient voxels in a 3D data set will attract the attention of the viewer. In our approach, voxel saliency is determined by the obscurance gradient, which measures the maximum variation of the obscurance field. Once the saliency of the volume is obtained, we implicitly have the saliency map of any structure contained in the volume. This saliency map can be applied to viewpoint selection and to enhance visualization. This can help to discover relevant characteristics of the model otherwise unnoticed by the observer.

(a) (b) (c)

Figure 1: *CT-human body data set rendered with the proposed obscurance-based volume rendering framework. The images have been obtained by modifying interactively the transfer function and the way in which obscurances are applied to the model.*

2. Background

In this section, obscurances, ambient occlusion, and related illumination models are described.

2.1. Obscurances and ambient occlusion

Zhukov et al. introduced ambient occlusion with the term obscurances [ZIK98, IKSZ03]. Roughly speaking, obscurance measures the part of the hemisphere obscured by the neighboring surfaces. For instance, a corner of a room is more obscured than the center. From the physics of light transport, obscurance expresses the lack of secondary (reflected) light rays coming to the specific parts of the scene, thus making them darker. Computation was done as a preprocess and the obscurance values were used as an ambient term during rendering. Since the obscurance computation was a property of the geometry and not of the lighting conditions, results could be combined with an arbitrary direct illumination. The method was also useful for interactive applications because the results were independent from the viewpoint. Landis detailed how ambient occlusion could be used to add realism to models [Lan02]. For a survey see [MFS08].

The obscurance O of a point p is defined as the integral

$$O(p) = \frac{1}{\pi}\int_{\Omega} \rho(d(p,\omega))\cos\theta d\omega, \qquad (1)$$

where ρ is a function of the distance $d(p,\omega)$ of the first intersection of a ray shot from point p with direction ω, p is a surface point, θ is the angle between the normal vector at p and direction ω, and the integration is over the hemisphere oriented according to the surface normal. We only consider a neighborhood of p, i.e. function ρ is set to 1 for distances greater than a maximum distance d_{max}. Therefore, the integral function $O(p)$ captures occlusion (or openness) information of the environment of point p. Considering extreme cases, an obscurance value 1 means that the point is completely open, i.e, not occluded and a value 0 means that it is completely occluded.

Ambient occlusion [Lan02] is a simplified version of the obscurances illumination model. Ambient occlusion

$$A(p) = \frac{1}{\pi}\int_{\Omega} V(p,\omega)\cos\theta d\omega, \qquad (2)$$

substitutes the ρ function in the obscurances equation (1) by the visibility function $V(p,\omega)$ that has value 0 when no geometry is visible in direction ω and 1 otherwise.

Color bleeding consists in the effect that the objects around another object with intense coloration are dyed with this color. To obtain color bleeding, Méndez et al. [MSC03] included in Equation (1) the diffuse reflectivity $R(q)$:

$$W(p) = \frac{1}{\pi}\int_{\Omega} R(q)\rho(d(p,\omega))\cos\theta d\omega, \qquad (3)$$

where q is the first point in direction ω that occludes p. When no occlusion is found within d_{max}, the average reflectivity is used. Observe that adding color bleeding to obscurances is almost free.

2.2. Volumetric shadowing

A volumetric version of the obscurances technique, called vicinity shading, was proposed by Stewart [Ste03]. Vicinity shading simulates illumination of isosurfaces by taking into account neighboring voxels. An occlusion volume is computed and stored in a shading texture that is accessed during rendering. This volume has to be re-computed each time that the rendering parameters are modified and the method does not support color bleeding. Since this first work, several models to illuminate the isosurfaces have been proposed. Wyman et al. [WPSH06] presented a method that supports the simulation of direct lighting, shadows and interreflections by storing pre-computed global illumination in an additional volume to allow viewpoint, lighting and isovalue changes. Despite the improvements achieved with these methods they still have a main limitation, they only allow to represent one of the surfaces of the volume. This limitation is overcome by Ropinski et al. [RMSD*08] and Hernell et al. [HLY07] using a local volumetric shadowing effect. Ropinski et al. compute a local histogram for each voxel from the voxel's neighbourhood, by accumulating intensities weighted by inverse squared distances. These local histograms can be combined interactively with the user defined transfer function to give an effect similar to local ambient lighting. Hernell et al. [HLY07] obtain the incident light intensity, arriving at a voxel, by integrating for each voxel and within a sphere surrounding it the attenuated transfer function density. This comes to compute, in the usual way, the visibility arriving at a voxel, using the opacities, averaged for all directions.

It is important to note at this point the twofold difference between these local volumetric shadowing effects and the classic obscurances (or ambient occlusion) used in our approach. Firstly, obscurances technique uses a ρ function (see discussion in Section 3.2) to modulate the effect of the occlusion with the distance. Secondly, obscurances compute

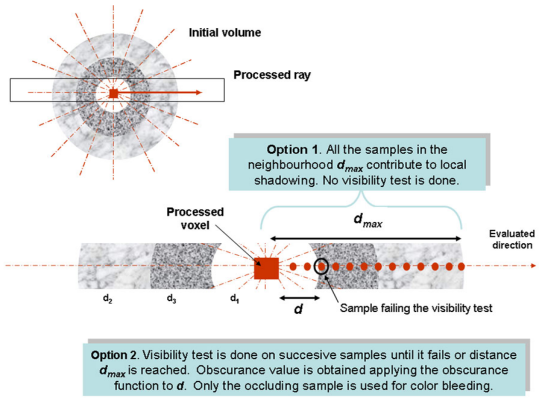


Figure 2: *Local volumetric shadowing (option 1) and obscurance computation (option 2) for a volume model consisting of three concentric spheres with densities $d_1 < d_2 < d_3$.*

explicitly visibility tests. This means that, although the test can be made up to a predefined maximum distance, if an occlusion is found, the rest of the environment in this direction is ignored, contrarily to local volumetric shadowing which integrates for the whole distance. In Figure 2, the essential difference between the local volumetric shadowing and the obscurances approach is illustrated. Observe that obscurances only take into account the distance from a voxel to the next occluding one, not what is in between. This is indeed different from Hernell's algorithm, which considers the accumulated visibility of the whole environment, being nearer to the physical realism (or at least more coherent with the transfer functions).

Obscurances (and later ambient occlusion) never claimed to be physically realistic, it was introduced as a fast photo-realistic approximation to indirect illumination. Local volumetric methods have a much higher cost, with complexity proportional to the square of the number of voxels, against the complexity of obscurances computation, proportional to the number of voxels times the number of directions. Thus, on a scale of physical realism (and cost) the different strategies can be sorted into increasing order as follows: ambient occlusion (the lowest), obscurances, Hernell's [HLY07] and Ropinski's [RMSD*08] approaches, and global illumination (the highest).

3. Obscurances for volume rendering

In this section we go further into the obscurance-based volume rendering by testing different distance functions for computing obscurances and providing discussion on quality assumptions.

3.1. Algorithm

We take as a basic implementation of the obscurance-based volume rendering approach the one proposed by Stewart in [Ste03]. First, the volume data set is centered in a sphere built from the recursive subdivision of an icosahedron and the lines from each vertex to the center of the volume are taken as the directions to consider (12, 42, and 162 directions have been taken in our experiments). Then, for each direction, the volume is swept using Bresenham's algorithm. This is equivalent to casting parallel rays covering all voxels. Obscurance computation for a given voxel is based on the presence (and the distance) of occluders within a certain radius along the processed direction. A visibility test compares the densities of two voxels which can be intensity values, which are independent of the transfer function, or opacity values assigned by the transfer function. In each case, we say that voxel v_i occludes v_{i-1} if the density of v_i is greater than that of v_{i-1}. To process the voxels we use a stack which stores the previously visited and yet unoccluded voxels in a density-based decreasing order. All the voxels in a ray are traversed and for each one we check if it is the nearest occluder to one of the previous unoccluded voxels (i.e. the ones stored in the stack), and in the occlusion test we check the distance. In each step, we start to check if the current voxel v_i occludes the one on the top of the stack v_s. If v_s is occluded then we can remove it from the stack so that it will not be processed anymore and continue applying the same procedure to v_{s-1}. If v_s is unoccluded, the rest of the stack voxels do not need to be processed since v_s is the voxel of the stack with lower density. Then, the next voxel of the ray is processed. This pre-computed obscurance is stored as vicinity shading values in a separate texture volume which is used during rendering.

In order to integrate color bleeding effects, we multiply the obscurance value by the color of the occluding voxel, and add an ambient color constant (in our case, white) to the unoccluded voxels. In this way, spectral obscurances are accumulated.

3.2. Analysis of ρ function

In this section, the meaning and shape of the ρ function in the obscurances definition (1) is discussed. First, this function should be a monotonically increasing function of d. Second, this function is bounded from above. This reflects the fact that normally ambient lighting of a given point is primarily affected by its neighborhood. This is especially true for scenes without bright light sources that may affect the illumination at large distances. From 0 to a determined value d_{max}, the function increases from 0 to 1, and for values greater than d_{max} the returned value is 1. This means that only a limited environment around the point p is considered and beyond this the occlusions will not be taken into account.

The shape of the ρ function is deduced from the fact that

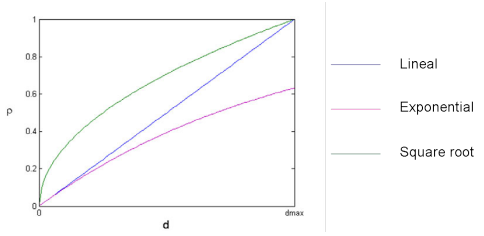


Figure 3: *Different* $\rho(d)$ *functions.*

we are interested in what happens in the vicinity of a point (or voxel). The nearer the occlusion, the higher the influence it should have. This influence diminishes with increasing distance. This is reinforced by interpreting the obscurance model with the $\rho(d) = (1 - e^{-\tau d})$ function as the illumination at the non reflecting boundaries of a nonscattering gas with opacity τ and constant volume emittance. If we consider the occlusions of the environment as having a similar damping opacity effect over the ambient light, we should use a function as similar as possible to $\rho(d) = (1 - e^{-\tau d})$ [IKSZ03]. Some candidate functions (see Figure 3) are: (a) All-or-nothing, $\rho(d) = 0$, used in the ambient occlusion approach (it does not allow color bleeding as the contribution of a hit is 0); (b) Linear, $\rho(d) = \frac{d}{d_{max}}$, used in the vicinity shading approach; (c) Exponential, $\rho(d) = 1 - e^{-\frac{d}{d_{max}}}$; and (d) Square root, $\rho(d) = \sqrt{\frac{d}{d_{max}}}$, introduced in [MSC03]. Note that we have considered the exponential function without normalization since the normalized exponential would become very similar to the linear function.

Different data sets have been used to analyze the effect of each function in the final visualization. The obtained images are shown in Figure 4 where each column represents a different ρ: (a) all-or-nothing, (b) linear, (c) exponential, and (d) square root. In the first row, we show the behavior of the above functions (including color bleeding) on a cubic phantom model with 128^3 voxels, where except for three adjacent faces with $opacity = 1$ (left wall: green, right wall: blue, floor: white), the rest of the voxels are transparent. The effects at the corner illustrate the behaviour of the distance function. The linear function ρ (column (b)) corrects in someway the non-smoothed effect of the all-or-nothing function (column (a)), but it considers a wider than necessary environment. The exponential ρ (column(c)) produces darker values, similar to the all-or-nothing case, due to the discontinuity leap at $d = d_{max}$. The square root function (column (d)) is a good compromise, as it considers a nearer environment and the darkness decays smoothly. This is appropriate to enhance the details of the model. We also study the results of applying the different ρ functions on a CT-human body of $256 \times 256 \times 415$ voxels. The obscurance volumes and the visualizations obtained using an obscurance-based illumination model (see Section 5.1) are shown, respectively, in the second and third rows of Figure 4. They have been obtained considering 162 viewing directions and a maximum distance equivalent to 64 voxels. Ambient occlusion (column (a)), linear (column (b)), and exponential (column (c)) become darker, and square root (column (d)) appears less dark and more pleasant to the eye.

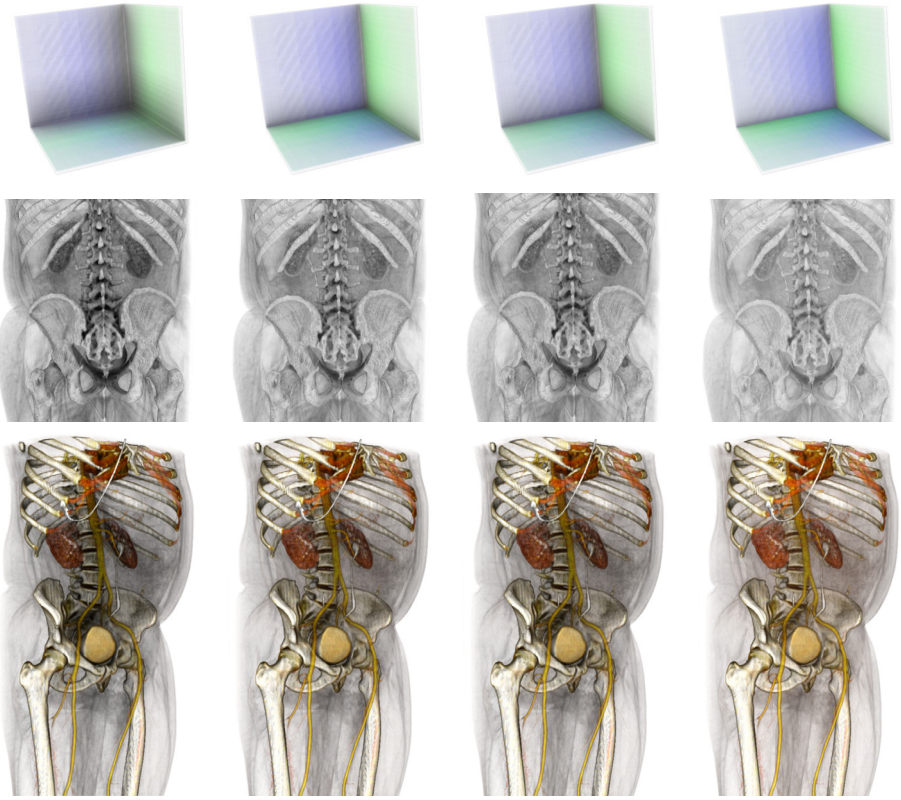
Data	Size	12	42	162
Aneurism	$256 \times 256 \times 256$	0:36	1:56	6:56
CT-body	$256 \times 256 \times 415$	1:14	4:09	15:02

Table 1: *Time cost (minutes:seconds) for computing obscurances using* 12*,* 42*, and* 162 *viewing directions.*

In the obscurances computation, d_{max} is also a key parameter to be considered since it controls the number of voxels in a determined direction that have to be taken into account to compute occlusions (see Equation 1). Therefore, if d_{max} has a high value, the probability of finding an occlusion increases, and hence the obscurance value, leading to darker images. Conversely, if d_{max} has a low value, the probability to be occluded decreases, leading to low obscurance values and hence lighter images. Figure 5 illustrates this effect in two data sets considering different d_{max} values (8, 64, and 256, respectively). As expected, we can observe how the darkness of the image increases when d_{max} increases. The other images in this paper have been computed using $d_{max} = 64$. In Figure 5, the effect of the number of directions in the obscurances values is also shown. Results for three different number of directions (12, 42, and 162) are given. Observe that although 12 directions could be considered for fast editing, we need at least 42 directions for a good quality final image. We have used the high quality obscurances given by 162 directions for the rest of the images shown in this paper. While the obscurances volume of the CT-body has been computed from its opacity (given by the transfer function), the obscurances of the aneurism have been computed from its intensity values. The time cost for computing obscurances for the CT-whole body and the aneurism is shown in Table 1. Times are given for a CPU Intel(R) Core(TM)2 Quad CPU Q6600 at 2.40GHz with 2 GB of memory. Note that, in accordance with the algorithm of Section 3.1, the time cost is proportional to the number of voxels times the number of directions. In the worst case, where densities are found in decreasing order, all the voxels in a ray would be pushed to the stack, and then each one would be popped, giving a cost proportional to the number of voxels in a ray. Thus, the cost of the algorithm is independent of d_{max}.

4. Volume saliency

The human visual system is able to reduce the amount of incoming visual data to a small but relevant amount of information for higher-level cognitive processing. Different com-



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

Figure 4: *From top to bottom, obscurances with color bleeding for a synthetic model, obscurances for the CT-human body data set, and a rendering of this data set using an obscurance-based illumination model, all of them obtained considering different* ρ *functions. From left to right: (a)* $\rho(d) = 0$*, (b)* $\rho(d) = \frac{d}{d_{max}}$*, (c)* $\rho(d) = 1 - e^{-\frac{d}{d_{max}}}$ *and (d)* $\rho(d) = \sqrt{\frac{d}{d_{max}}}$*.*

putational models have been proposed to interpret the selective visual attention. The biologically-inspired model of bottom-up attention of Itti et al. [IK01] permits us to understand our ability to interpret complex scenes in real time. The selection of a subset of available sensory information before further processing appears to be implemented in the form of a spatially circumscribed region of the visual field, called *focus of attention*, while some information outside the focus of attention is suppressed. This selection process is controlled by a *saliency map* which is a topographic representation of the instantaneous saliency of the visual scene and shows what humans find interesting in visual scenes.

Inspired by Itti's work, Lee et al. [LVJ05] introduced the concept of mesh saliency, a measure of regional importance for 3D meshes, computed using a center-surround mechanism that is able to identify regions that are different from their surrounding context. Mesh saliency is captured from surface curvatures and is used in graphics applications such as mesh simplification and viewpoint selection. Feixas et al. [FSG] defined a view-based saliency of a polygon as the average information-theoretic dissimilarity between this polygon and its neighbors. In the volume rendering field, Kim et al. [KV06] presented a visual-saliency-based operator to enhance human perception of the volume data by guiding the viewer's attention to selected regions. A definition of voxel saliency is not provided and it is assumed that a saliency value is assigned to each voxel by using a user specification, eye-tracking data, or feature computation. In different works on saliency, it has been shown that attention is attracted by changes in luminance, color, curvature, texture, shape, etc. [TIR05]. That is, salient features are generally determined from the local differential structure of images and different operators such as color or luminance gradient have

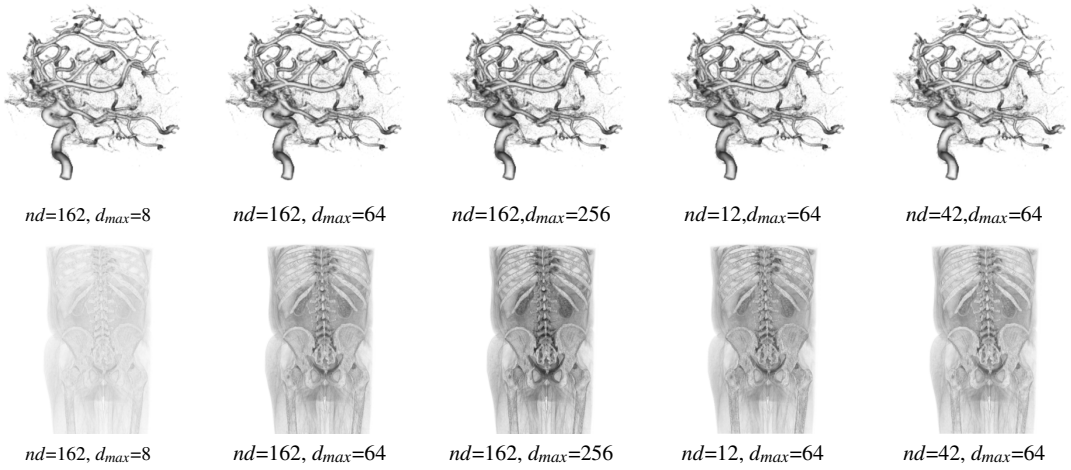


Figure 5: *The obscurance volumes of the aneurism (first row) and CT-human body (second row) are visualized considering different d_{max} values and number of directions nd. The square root function has been used in all the cases. Computation times are given in Table 1.*

been used [vdWGB06]. In Gonzalez et al. [GSF08], from an information theory perspective, ambient occlusion has been defined as the occlusion information associated with each polygon of the model.

In this paper, a definition of voxel saliency based on the gradient of obscurance field is proposed. Considering that obscurance represents occlusion information associated with a voxel, its variation with respect to its surround can indeed be perceptually salient, i.e. it can be considered as a salient feature of the volume. This saliency would be most noticeable at edges, occlusion variations, and corners. On the other hand, a smooth or uniform region would produce low saliency values, as is intuitively expected.

The voxel saliency is defined as the magnitude of the gradient of obscurances estimated by using the 4D linear regression method proposed in [NCKG00]:

$$S(p) = \sqrt[2]{A^2 + B^2 + C^2}, \tag{4}$$

where voxel p is located at the origin of the coordinate system, and A, B, and C are the components of the obscurance gradient $[A,B,C]$. These components are computed as $A = \sum_k w(k)O(k)x(k)$, $B = \sum_k w(k)O(k)y(k)$, and $C = \sum_k w(k)O(k)z(k)$, where k stands for the voxels in the neighborhood centered at voxel p, $w(k)$ is the distance between voxels p and k, $O(p)$ is the obscurance of voxel p, and $x(k)$, $y(k)$, and $z(k)$ are, respectively, the x, y, and z components of the vector from voxel p to voxel k. In our experiments, the neighborhood of p is given by a cube of 5^3 voxels, since smoother results are obtained than by using a cube of 3^3 voxels as in [NCKG00]. For each data volume, the saliency has been scaled ranging from 0 to 1. Analogously to mesh saliency [LVJ05], the gradient of obscurances is scale-dependent (i.e., the saliency value depends on the size of the neighborhood considered). We have to emphasize that our definition of saliency can be generalized to the local volumetric shadowing methods [HLY07, RMSD*08]. Figures 6(b), (d) and (f) show the color-coded saliency maps obtained for the CT-human body corresponding to Figure 6(a).

5. Applications

In this section, we describe how obscurances can be applied to volume rendering to interactively produce realistic and illustrative images. Two applications of saliency maps are also presented.

5.1. Realistic and illustrative rendering

To apply the obscurances to the visualization, we use the Blinn-Phong shading model where the color resulting from the local lighting of each voxel x is multiplied by its obscurance value:

$$I(x) = (k_d N(x) \cdot L + k_s (N(x) \cdot H)^n) O(x) \tag{5}$$

where k_d and k_s are the diffuse and specular lighting coefficients, $N(x)$ the normal of the voxel, L the light vector, H the half-angle vector between L and the direction to the viewer, n is the Phong exponent, and $O(x)$ the obscurance of voxel x which has been adjusted to the range $[0,1]$. Figure 4 (third row) illustrates the result of applying the obscurance-based Blinn-Phong model on the CT-human body data set.

We also introduce two parameters, *low* and *high*, such that

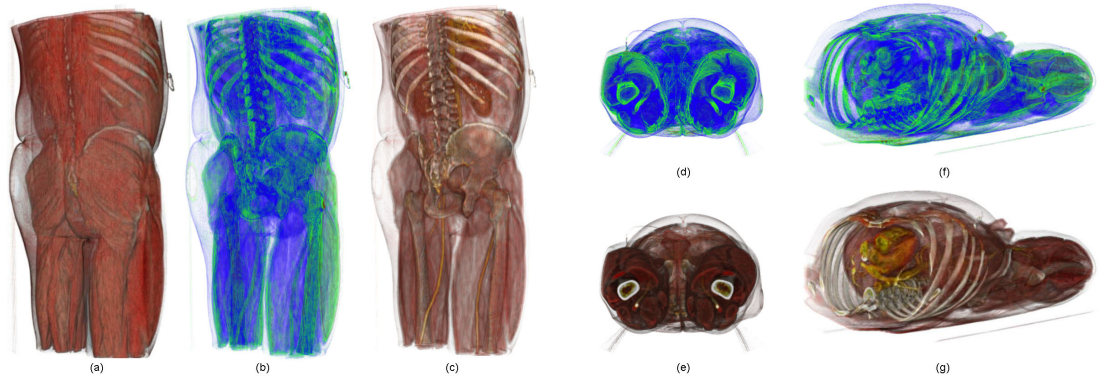


Figure 6: *(a) Original CT-body data set. (b, d, f) Color-coded (from blue to red) saliency maps corresponding to the most salient views. (c, e, g) Illustrative visualizations obtained with a saliency-based opacity modulation.*

from 0 to *low* obscurances are set to 0 (making the voxel completely black), from *low* to *high* they follow a linear distribution (preserving their original value), and from *high* to 1 their value is set to 1 (thus the voxel becomes completely unobscured). Increasing the *low* threshold will turn more voxels black, and this can be used to increase the contrast of the resulting image. Decreasing the *high* threshold means that more voxels are not darkened by their obscurance, and so become brighter, thus increasing slightly the contrast too. In the limit we could set *low* and *high* to the same value to have only some voxels with obscurance 0 and others with obscurance 1. Thus, there would be voxels with their own color (modified only by local lighting if applied) and the others would be black. The user can modify *low* and *high* parameters to obtain the desired effect interactively. Figure 1 shows different renderings of the CT-body data set. Figure 1(a) shows the visualization of the model resulting from the application of obscurances with $low = 0$ and $high = 1$. Figure 1(b) has been obtained with $low = 0.4$ and $high = 0.6$, and making the skeleton transparent. Finally, in Figure 1(c) all the structures have been set to white and the obscurances assignment has been adjusted with $low = 0.6$ and $high = 0.7$.

5.2. Saliency

As a measure of importance, the volume saliency is applied to obtain the most salient views and to enhance volume visualization by modifying the transfer function according to the computed saliency.

Similar to [LVJ05], where mesh saliency was used to select the best views, a method to calculate the saliency of a viewpoint is proposed. Given the saliency of all the voxels, we can find the viewpoint which maximizes the visible saliency. The *viewpoint saliency* is defined by

$$S(v) = \sum_{p \in P} S(p)V(p), \tag{6}$$

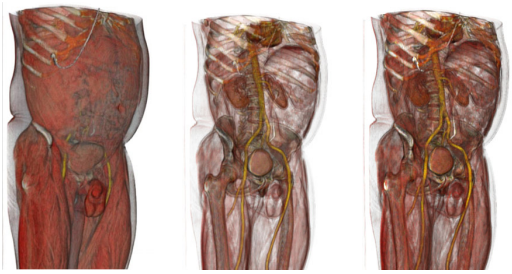
where v is a given viewpoint, P is the set of voxels of the volume data, $S(p)$ is the saliency of voxel p, and $V(p)$ is the visibility of voxel p from v.

We also present an automated technique to enhance volume visualization by emphasizing (increasing the opacity of) the most salient voxels and de-emphasizing (reducing the opacity of) the least salient ones. So, the viewer's attention is guided towards the most salient parts of the model.

In Figure 6, (a) the original CT-body data set, (b-c) the most salient view, (d-e) the least salient view, and (f-g) the most salient view per unit area are shown. Images (c), (e) and (g) have been obtained by multiplying the opacity by the saliency. Figure 7 shows (a) the original CT-body data set and (b-c) two different renderings obtained by scaling the opacity according to the saliency values. In Figure 7(b), voxels with saliency lower than 0.2 have been made transparent and the opacity of the most salient ones has been preeserved. In Figure 7(c), the voxels with saliency lower than 0.2 have been made transparent while the opacity of the most salient ones has been doubled.

6. Conclusions

In this paper, we have analyzed obscurance-based volume rendering by evaluating the main parameters involved in its computation, such as the obscurance function and the number of viewing directions. From this study, we conclude that the square root function gives better results than other analyzed functions and that 42 directions are enough to obtain obscurances of a certain quality, although for high quality results we have used 162 directions. In addition, we have introduced two new applications of obscurances. The first is a technique to obtain illustrative renderings and the second is a method to compute the saliency map as the gradient of obscurances. Saliency has been used to enhance visualization



(a) (b) (c)

Figure 7: *(a) Original CT-human body data set. (b, c) Images obtained by scaling the opacity according to the saliency values.*

and to select the most salient views. All our proposals have been integrated in a common framework and tested on several volume data sets. As future work, we plan to programme a GPU version of the obscurances algorithm to obtain real-time or interactive obscurance computation.

Acknowledgements

This work has been supported by TIN2007-68066-C04-01 and TIN2007-67982-C02 of the Ministry of Education and Science (Spanish Government), by the MedViz Initiative in Bergen (medviz.uib.no), and by the Austrian Science Fund (FWF) grant no. P18322.

References

[ER00] EBERT D., RHEINGANS P.: Volume illustration: non-photorealistic rendering of volume models. In *VIS '00: Proceedings of the conference on Visualization '00* (Los Alamitos, CA, USA, 2000), IEEE Computer Society Press, pp. 195–202.

[FSG] FEIXAS M., SBERT M., GONZALEZ F.: A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Transactions on Applied Perception*. In press.

[GSF08] GONZALEZ F., SBERT M., FEIXAS M.: Viewpoint-based ambient occlusion. *IEEE Computer Graphics and Applications 28*, 2 (2008), 44–51.

[HLY07] HERNELL F., LJUNG P., YNNERMAN A.: Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *Eurographics/IEEE-VGTC Symposium on Volume Graphics 2007* (2007), pp. 1–8.

[IK01] ITTI L., KOCH C.: Computational modeling of visual attention. *Nature Reviews Neuroscience 2*, 3 (2001), 194–203.

[IKSZ03] IONES A., KRUPKIN A., SBERT M., ZHUKOV S.: Fast, realistic lighting for video games. *IEEE Comput. Graph. Appl. 23*, 3 (2003), 54–64.

[KV06] KIM Y., VARSHNEY A.: Saliency-guided enhancement for volume visualization. *IEEE Trans. Vis. Comput. Graph. 12*, 5 (2006), 925–932.

[Lan02] LANDIS H.: Production-ready global illumination. In *Course 16 notes, SIGGRAPH 2002* (2002).

[LVJ05] LEE C. H., VARSHNEY A., JACOBS D. W.: Mesh saliency. *ACM Trans. Graph. 24*, 3 (2005), 659–666.

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108.

[MFS08] MENDEZ-FELIU A., SBERT M.: From obscurances to ambient occlusion: A survey. *The Visual Computer* (2008). In press, published online 15 February 2008.

[MSC03] MÉNDEZ A., SBERT M., CATÁ J.: Real-time obscurances with color bleeding. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics* (2003), ACM, pp. 171–176.

[NCKG00] NEUMANN L., CSÉBFALVI B., KÖNIG A., GRÖLLER E.: Gradient estimation in volume data using 4d linear regression. *Comput. Graph. Forum 19*, 3 (2000), 351–358.

[RMSD*08] ROPINSKI T., MEYER-SPRADOW J., DIEPENBROCK S., MENSMANN J., HINRICHS K. H.: Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum (Eurographics 2008) 27*, 2 (2008), 567–576.

[Ste03] STEWART A. J.: Vicinity shading for enhanced perception of volumetric data. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (2003), IEEE Computer Society, pp. 355–362.

[TIR05] TSOTSOS J. K., ITTI L., REES G.: A brief and selective history of attention. In *Neurobiology of Attention*, Itti L., Rees G., Tsotsos J. K., (Eds.). Elsevier, San Diego, CA, 2005.

[vdWGB06] VAN DE WEIJER J., GEVERS T., BAGDANOV A. D.: Boosting color saliency in image feature detection. *IEEE Trans. Pattern Anal. Mach. Intell. 28*, 1 (2006), 150–156.

[WPSH06] WYMAN C., PARKER S. G., SHIRLEY P., HANSEN C. D.: Interactive display of isosurfaces with global illumination. *IEEE Trans. Vis. Comput. Graph. 12*, 2 (2006), 186–196.

[ZIK98] ZHUKOV S., IONES A., KRONIN G.: An ambient light illumination model. In *Proceedings of Eurographics Rendering Workshop '98* (1998), Springer Wien, pp. 45–56.

4 Information-based Transfer Functions for Multimodal Visualization

Transfer functions are an essential part of volume visualization. In multimodal visualization at least two values exist at every sample point. Additionally, other parameters, such as gradient magnitude, are often retrieved for each sample point. To find a good transfer function for this high number of parameters is challenging because of the complexity of this task. In this paper we present a general information-based approach for transfer function design in multimodal visualization which is independent of the used modality types. Based on information theory, the complex multi-dimensional transfer function space is fused to allow utilization of a well-known 2D transfer function with a single value and gradient magnitude as parameters. Additionally, a quantity is introduced which enables better separation of regions with complementary information. The benefit of the new method in contrast to other techniques is a transfer function space which is easy to understand and which provides a better separation of different tissues. The usability of the new approach is shown on examples of different modalities.

The following paper appears in its original format published as:

M. Haidacher, S. Bruckner, A. Kanitsar, and M. E. Gröller, Information-based transfer functions for multimodal visualization. In *Proceedings of Visual Computing for Biomedicine 2008*, pages 101–108, 2008.

Information-based Transfer Functions for Multimodal Visualization

Martin Haidacher[1], Stefan Bruckner[1], Armin Kanitsar[2], and M. Eduard Gröller[1]

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

Abstract

Transfer functions are an essential part of volume visualization. In multimodal visualization at least two values exist at every sample point. Additionally, other parameters, such as gradient magnitude, are often retrieved for each sample point. To find a good transfer function for this high number of parameters is challenging because of the complexity of this task. In this paper we present a general information-based approach for transfer function design in multimodal visualization which is independent of the used modality types. Based on information theory, the complex multi-dimensional transfer function space is fused to allow utilization of a well-known 2D transfer function with a single value and gradient magnitude as parameters. Additionally, a quantity is introduced which enables better separation of regions with complementary information. The benefit of the new method in contrast to other techniques is a transfer function space which is easy to understand and which provides a better separation of different tissues. The usability of the new approach is shown on examples of different modalities.

Categories and Subject Descriptors (according to ACM CCS): I.4.10 [Image Processing and Computer Vision]: Volumetric, Multidimensional

1. Introduction

Volume visualization is a technique which enables physicians and scientists to gain insight into complex volumetric structures. Currently, the trend towards information acquisition using data sets from multiple modalities is increasing in order to facilitate better medical diagnosis. As different modalities frequently carry complementary information, our goal is to combine their strengths providing the user with a consistent interface.

Normally a side-by-side view is provided in medical applications for the inspection of the different modalities. A physician can simultaneously scroll through both registered modalities. This practice has two main drawbacks. One is the missing direct visual combination of the data. A physician has to mentally overlap the two images to get the corresponding points of one modality in the other one. A second drawback is the restriction to a 2D visualization. These drawbacks can be eliminated by the fused display of both data sets together in a 3D multimodal visualization. The challenge for such a visualization is the density of information in space. For each sample point at least two values from the different modalities are present. To reduce the density a transfer function can be used which defines optical properties, such as color and opacity, for certain values. The transfer function can be controlled by the user to change the appearance of the result image. The more input values are taken to classify a sample point and assign optical properties to it, the harder it is for the user to find a good transfer function. This is the main problem of multimodal visualization because there are at least two values involved.

In this paper, we introduce a novel concept for defining transfer functions in multimodal volume visualization. Our method aims to reduce the complexity of finding a good transfer function. A new transfer function space is provided which can be controlled by the user in an intuitive and familiar way. This is done by using the information contained in the distribution of values in both modalities. Based on this information, the values of both modalities are fused. This results in a fused transfer function space with a single value and a single gradient magnitude as parameters. A measure for the complimentary information of both modalities is used

as additional parameter for more user control and a better separation of different tissues.

In Section 3 the new approach is described in detail. We show how the retrieved information of the value distribution can be used to generate the transfer function space. Section 4 briefly describes an efficient implementation of the new method. The usability of the new method is shown in Section 5 with some results. Conclusions and ideas for further work are given in Section 6. First an overview over related works on this topic is given in the following section.

2. Related Work

All different methods for multimodal visualization can be classified - as described by Cai and Sakas [CS99] - according to the level in the rendering pipeline in which they are applied. In the illumination-model-level intermixing optical properties are assigned to a combination of values from the different modalities. The accumulation-level intermixing fuses the values after optical properties are assigned to each modality individually. In the image-level intermixing the fusion is done after the 2D images have been rendered.

The image-level intermixing is the simplest way for the fusion of two modalities, but it has the disadvantage that the 3D information is lost. Therefore this fusion technique is typically just applied on single slices of the volume. Several techniques have been developed for this purpose, such as alternate pixel display or linked cursor [SBS*87, SZHV94].

Due to the increasing speed of computers and graphics hardware volume rendering became more popular and, therefore, also the multimodal fusion could be done in the volume space. The first methods were based on surface models. Levin et al. [LHT*89] generated a surface model from an MRI scan and mapped the PET-derived measurement onto this surface. Evans et al. [EMT*91] generated an integrated volume visualization from the combination of MRI and PET. These works are mainly focused on the combination of anatomical and functional images. A more general approach for the fusion of modalities was introduced by Zuiderveld and Viergever [ZV94]. For this method an additional segmentation of the volumes is necessary to decide which one to show at a given sample point. A more recent work by Hong et al. [HBKS05] describes how fusion techniques in this intermixing level can be efficiently implemented using the graphics hardware.

More sophisticated but more complex methods for multimodal visualization are directly applied in the illumination-model-level. The intermixing in this level directly generates optical properties from the combination of the values and additional properties of the two volumes at a single sample point. A case study for the rendering of multivariate data where multiple values are present at each sample point was done by Kniss et al. [KHGR02]. In this work the idea of multi-dimensional transfer functions to assign optical properties to a combination of values was used. Akiba and Ma [AM07] used parallel coordinates for the visualization of time-varying multivariate volume data. Multimodal visualization of medical data sets by using multi-dimensional transfer functions was shown by Kniss et al. [KSW*04]. The classification is done on the basis of the dual histogram. Kim et al. [KEF07] presented a technique which simplifies the transfer function design by letting the user define a separate transfer function for each modality. The combination of them defines the two-dimensional transfer function. The problem with this technique is the loss of information by reducing the multi-dimensional transfer function to two 1D transfer functions.

As mentioned before, the assignment of optical properties in multimodal visualization is dependent on more than one value. If the whole information space is used then a multi-dimensional transfer function is needed. In general it is a non-trivial task to define a multi-dimensional transfer function because of its complexity. Nevertheless, multi-dimensional transfer functions are commonly used for volume visualization. 2D transfer functions were first introduced by Levoy [Lev88]. In addition to the data value the gradient magnitude was used as second dimension to classify a sample point. Due to the fact that the design of a 2D transfer function is non-trivial, methods were developed, to support this task. Kindlmann and Durkin [KD98] introduced a semi-automatic approach for the visualization of boundaries between tissues. Pfister et al. [PBSK00] gave an overview on existing techniques to support the design task of transfer functions. The direct manipulation widgets introduced by Kniss et al. [KKH01] can be used to find regions of interest in the multi-dimensional transfer function space in an intuitive and convenient way. In other work, Kniss et al. [KPI*03] describe a way to efficiently represent multi-dimensional transfer functions by Gaussian functions instead of storing a multi-dimensional lookup table.

For the definition of the multi-dimensional transfer functions, in addition to the values from the two volumes, further properties can be used to better distinguish between tissues. In this paper, these additional properties are retrieved by methods from information theory founded by Shannon [Sha48]. He described how the probability of occurrence of a signal can be used to define the information content of the signal. In imaging, information theory is used in different areas. Image registration is one of these areas. Wells et al. [WVA*96] maximized the mutual information to find a good registration position for two images or volumes. This idea is the basis for the information-based part of the new approach in this paper.

Rezk-Salama et al. [RSKK06] employed PCA to assist the generation of more effective transfer functions based on semantics. Our approach provides additional derived quantities for evaluating the joint information of multiple modalities. In future work, a combination of both methods could lead

to even more intuitive user control for multimodal volume visualization.

3. Information-based Transfer Functions for Multimodal Visualization

In this section we introduce a novel transfer function space for multimodal visualization. The aim of all steps described here is the design of a transfer function space which is as simple as possible but still able to separate different tissues. The main contribution of the new approach is the use of methods from information theory for the design of this transfer function space. Figure 1 shows all necessary processing steps to classify a tuple of input values (f_1, f_2) in this new transfer function space with optical properties. The further sections describe these processing steps in detail.

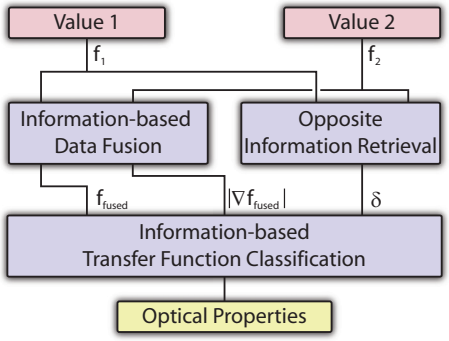


Figure 1: *Processing pipeline for the classification of sample points in a multimodal visualization by an information-based transfer function.*

In Section 3.2, we describe how the input values can be fused to get just a single value for each pair of input values. Section 3.3 introduces an additional property which is used to refine the classification of different tissues through the transfer function. Finally, Section 3.4 describes how the fused values are used to define the new transfer function space and how the additional property is used to influence the classification. First of all we describe how the probabilities are estimated for the further approach.

3.1. Probabilities in Volume Data

To estimate the probabilities within the volume we first assume the volume is given as a set of regularly arranged grid points. The simplest way to estimate the probability of a certain value in such a volume is done by counting its occurrence in the whole data set and by dividing this number by the total number of points in the volume. To do this for all values a histogram is generated. In a histogram the count of a bin is increased if a value falls in the range of this bin. When the counted numbers for all bins are divided by the total number of points in the volume, we get a probability distribution $P(f)$ which returns a probability of occurrence for each value f.

For retrieving the information content of the joint occurrence of two values from two modalities another probability distribution is needed. It returns a probability $P(f_1, f_2)$ for each tuple of values f_1 from modality 1 and f_2 from modality 2, also referred to as joint probability. Equally to the probability for the occurrence of only one value this probability distribution can also be estimated by a histogram. Due to the dependency of two values, the histogram is defined in 2D. This histogram is often referred to as dual histogram.

In the context of the joint probability $P(f_1, f_2)$ the probability of just a single value $P(f_1)$ is referred to as marginal probability. These two types of probabilities are further used in the following sections to generate a new transfer function space based on the methods of information theory.

3.2. Information-based Data Fusion

At some point in a multimodal visualization pipeline the information from both data sets has to be combined, as each sample point can only have one color and opacity. The idea behind the information-based data fusion is a fusion which loses as little as possible information. Information can be measured based on the quality or the quantity of the data. To be measured by the quality, user interaction would be necessary to decide which region is important in which modality. This would be a good measurement but it is a time-consuming process and has to be repeated for each new data set.

A second way to measure the information is based on the quantity, i.e. frequency, of the data. For this measurement the methods of information theory are used. The idea behind this measurement is that values which occur very often have less information than values which occur not so often. For medical data sets this can be interpreted that larger regions with the same value, such as the background, contain less information than smaller regions, such as border areas or small tissues. The information content can be expressed by the following equation:

$$I(f) = -log_2(P(f)) \tag{1}$$

where $P(f)$ is the probability of occurrence for a certain value f. Through the log_2 function the information $I(f)$ is high for values with a low probability. The fusion should then be done in a way to weight the value with more information content higher than the value with less information content. To formalize this weighting we want to introduce the following equation:

$$\gamma(f_1, f_2) = \frac{I(f_2)}{I(f_1) + I(f_2)} \tag{2}$$

The γ value is 0 when the second modality has no information. It is 1 if the first modality has no information. For a value of 0.5 both modalities contain the same amount of information for a given pair of values.

With Equation 2 we get a number for each pair of values which can directly be used for the weighting in the fusion step. The fusion of two values, f_1 and f_2, is simply done by the following equation:

$$f_{fused} = (1-\gamma) * f_1 + \gamma * f_2 \tag{3}$$

The fused value f_{fused} is close to the value of one modality when this modality contains more information than the other modality. Therefore, points with high information content in just one modality are only slightly modified in contrast to their original value. This property makes it easier to find such points in the new transfer function space because they have almost the same value as they would have in volume visualization of this modality alone. For points with a γ around 0.5 the fused value is a mixture of both values and, therefore, is distinguishable from points with high information content in one modality.

The gradients of both modalities are fused in the same manner as the values to get an appropiate fused gradient according to the values:

$$\nabla f_{fused} = (1-\gamma) * \nabla f_1 + \gamma * \nabla f_2 \tag{4}$$

The fusion of the gradients is needed for the shading calculation as well as for classification by the transfer function based on gradient magnitude. The result of the fusion is a single value for each sample point like for the visualization of a single volume. This fused value together with the magnitude of the fused gradient can be used for the classification by a transfer function. Unfortunately some tissues are overlapping in this fused transfer function space. Therefore an additional parameter is introduced in the following section which supports the transfer function design for a better separation of different tissues.

3.3. Opposite Information Retrieval

In the previous section a quantity was calculated which indicates which of the two values has more information. In this section we will define a quantity which indicates the information contained in the joint occurrence of two values rather than the information contained in the occurrence of a single value. This new quantity will be used as another attribute for the classification of a sample point. It allows for a better separation of different tissues.

For image and volume registration the maximization of the mutual information is a common tool to find a good registration position. In this context the best registration position is found when the mutual information is at a maximum. This means that in this position both data sets contain the lowest possible opposite information. The mutual information is a quantity for the whole data set. In contrast the pointwise mutual information (PMI) is a quantity for the mutual information for a certain combination of points. It is defined by the following equation:

$$PMI(f_1, f_2) = log_2\left(\frac{P(f_1, f_2)}{P(f_1) * P(f_2)}\right) \tag{5}$$

The PMI is 0 when a pair of values occurs exactly as frequently as one would expect by chance. This is the case when both values are statistically independent from each other and the joint probability $P(f_1, f_2)$ is exactly the product of both marginal probabilities $P(f_1)$ and $P(f_2)$. If they occur together more frequently as one would expect by chance then the result of the calculation is greater than 0. Conversely, the value is lower than 0 if a pair of values occurs less frequently as one would expect by chance. By the definition of Shannon this case contains more information than a result value greater than 0 because the occurrence is less frequent. For a joint probability $P(f_1, f_2)$ of 0 the PMI is by definition 0. For all other probabilities the PMI can be normalized to a value between 0 and 1 by subtracting the lower bound ($P(f_1) = 1$ and $P(f_2) = 1$) from the PMI and dividing it by the difference between the upper bound ($P(f_1) = P(f_1, f_2)$ and $P(f_2) = P(f_1, f_2)$) and the lower bound:

$$PMI_{norm}(f_1, f_2) = \frac{PMI(f_1, f_2) - log_2(P(f_1, f_2))}{log_2(\frac{1}{P(f_1, f_2)}) - log_2(P(f_1, f_2))} \tag{6}$$

The value of PMI_{norm} approaches 0 if the information carried by the pair of values is high. Values close to 1 represent low information content. To get a high value for pairs of values with high information content we define a new quantity δ as an inversion of PMI_{norm}:

$$\delta(f_1, f_2) = 1 - PMI_{norm}(f_1, f_2) \tag{7}$$

Figure 2 illustrates the behavior of δ. The different regions, labeled with capital letters, have different colors to symbolize regions of different values in both modalities. The red crosses are sample points for which the δ value should be calculated. For the sample point S_1 the involved marginal probabilities ($P(f_1)$ and $P(f_2)$) are rather low because only a small area (C_1 and C_2) has the same value in both modalities. For the sample point S_2 the marginal probability in the second modality is higher because the sample point lies in a larger area B_2. The joint probability $P(f_1, f_2)$ is the same for both sample points because the combination of C_1 and C_2 occurs exactly as often as the combination of D_1 and B_2. By calculating the δ values with these probabilities we, however, get a smaller value for the sample point S_1 than for the sample point S_2.

This example can be interpreted in a way that for sample point S_1 both modalities contain correlated information whereas for S_2 modality 1 complements the information of modality 2 because the region D_1 is only represented in

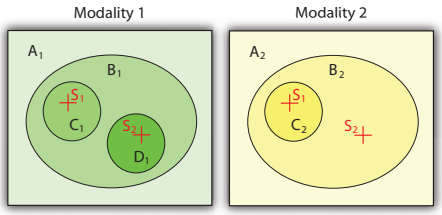
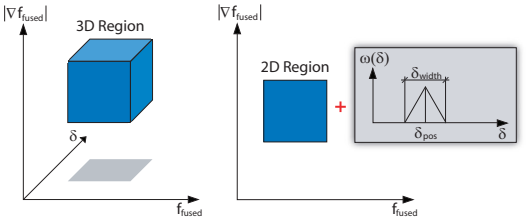


Figure 2: *Example of slices of two different modalities to explain how the δ value is affected by the value distribution. S_1 and S_2 are sample points for which the δ value is calculated.*

modality 1. This means that the δ value responds with a high value for regions with high opposite information content. So this value can be used to separate tissues which only show up in one modality from tissues which are present in both modalities. It can be seen as a quantity which indicates the difference of information content in both modalities at each sample point. Noise in the data sets does not influence the δ value. It flattens the probability distribution function but the relation between the probabilities does not change and, therefore, the δ value is not affected. The following section describes how this property can be integrated in the classification process.

3.4. Information-based Transfer Function Classification

In the previous two sections we described how methods from information theory can be used to generate a fused value and fused gradient as well as an additional property δ which indicates the opposite information. These values together will be used now for the assignment of optical properties.



(a) 3D transfer function space (b) 2D transfer function space and δ windowing function

Figure 3: *Transfer function space is converted from 3D (a) to 2D (b). Additionally, a simple windowing function for the δ value is used to modify the optical properties of each 2D region.*

Due to the existence of three values (f_{fused}, $|\nabla f_{fused}|$, δ) for each sample point the classification could be done in a 3D space. For every triple of values optical properties would be assigned. This approach is shown in Figure 3(a). The problem with this approach is the complexity of the transfer function design and, therefore, it is hard to find a good transfer function. To avoid this we reduce the degree of freedom by defining a region only in the 2D transfer function space (f_{fused}, $|\nabla f_{fused}|$). The design task in this space is easier because the 2D space is already well-known from volume visualization of only one volume. Additionally, for each region a simple windowing function is defined for the δ value. The selection of a windowing function for this task results from the fact that the δ values for points of one tissue in anatomical modalities or a level of activity in functional modalities are in a certain value range. To extract such parts only points with a δ value in this range should be selected. A windowing function is easy to adjust to a certain value range and, therefore, is perfect for this purpose. The windowing function can be expressed by the following equation:

$$\omega(\delta) = max\left(\left|1 - \frac{\delta - \delta_{pos}}{0.5 * \delta_{width}}\right|, 0\right) \tag{8}$$

The parameters δ_{pos} and δ_{width} define the position and shape of the windowing function $\omega(\delta) \in [0,1]$. The original opacity α, assigned according to a 2D region in the transfer function space, is multiplied with this value to fade out points with a low value of this windowing function. In Figure 3(b) the separation in a 2D region and a corresponding windowing function is shown.

4. Implementation

For a fast and efficient volume rendering it is necessary to do as many calculations as possible in a pre-process. The most time-consuming part of the whole process is the generation of the dual histogram and the two individual histograms of both modalities for the estimation of the probabilities. This can be done before the rendering because the histograms are static for two given volume data sets and do not change during the rendering process. The histograms are used to calculate the γ and δ values as described in the previous section. Each of these values can be stored in a 2D lookup table. They also do not change for two given volume data sets.

Figure 4 shows the processing steps for each sample point during the rendering process. The processing steps with sharp corners are lookups and the processing steps with round corners are calculations. As first step lookups in the a priori generated γ and δ lookup tables are done. The γ value is used to fuse the two input values as described in Section 3.2. With the fused value and the magnitude of the fused gradient a lookup in the lookup tables of the transfer function is done. One lookup table stores the color c and opacity α for each point in the transfer function space. The second lookup table stores the parameters δ_{pos} and δ_{width} of the windowing function. The color c of the 2D transfer function is directly used for further processing steps, such as shading. The opacity α is modified by the windowing function according to the parameters δ_{pos} and δ_{width} as well as the δ

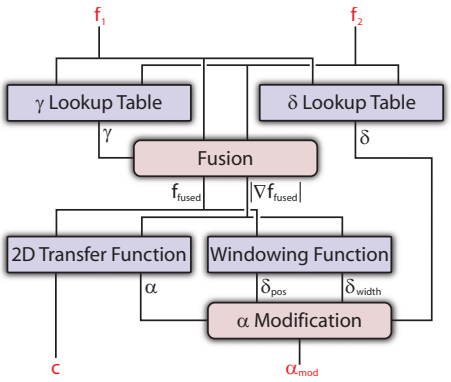


Figure 4: *Overview over the processing steps for each sample point during the rendering process. Nodes with round corners are calculation steps and nodes with sharp corners are lookups.*

value. As output of this calculation step we get a modified opacity α_{mod} which is further used in the rendering process. The speed of the implementation is no big issue because all processing steps can be executed quite fast on the graphics hardware to achieve real-time frame rates.

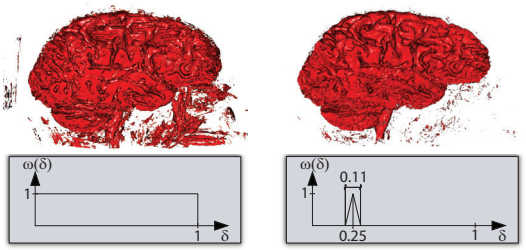
5. Results

Modalities can be generally classified into two groups: functional and anatomical modalities. The most common anatomical modalities are CT and MRI. CT is typically used to show bone structures. Soft tissues have a higher contrast in MRI. In Figure 5(a) a visualization of a CT scan is shown and in Figure 5(b) the visualization of an MRI scan. Both visualizations can be useful for special examinations but it can also be seen that both data sets contain some joint information. Furthermore some regions with less information, such as the tissue around the brain in the MRI scan, are hiding regions with more information, such as the brain itself.

The goal of a multimodal visualization is to combine relevant tissues from both modalities and show them together to provide additional context. The relevance of a tissue is dependent on the kind of examination. In a combination of CT and MRI of a head the brain could be the relevant part of the MRI scan and the bones could be the relevant parts of the CT scan. Figure 5(c) shows the rendering results of a multimodal visualization based on the dual histogram. Both relevant tissues, the brain and the bones, are visible but also a lot of artifacts are visible in the result. This follows from the fact that the brain cannot be better separated in the transfer function space based on the dual histogram. Figure 5(d) shows the result generated by the new method. In comparison to the result generated with the traditional multimodal visualization technique the brain is clearly separated from other tissues and only a few artifacts are visible.

Figures 5 (e) to (h) show the corresponding histograms for the visualizations in Figures 5 (a) to (d). The regions which were used to classify sample points with optical properties, such as color and opacity, are also shown on top of these histograms. It can be seen that the regions for classifying the brain tissue and the bones in the new fused transfer function space, as shown in Figure 5(h), are highly related to the individual regions in the single modality visualizations, as shown in Figure 5(e) and Figure 5(f). The regions for the multimodal visualization, based on the dual histogram, are shown in Figure 5(g). The position and shape of the regions in this transfer function space are completely different in comparison to the regions for the single modality visualization. This makes it much harder for the user to define regions for the transfer function because the knowledge from the single modality visualization cannot be used.

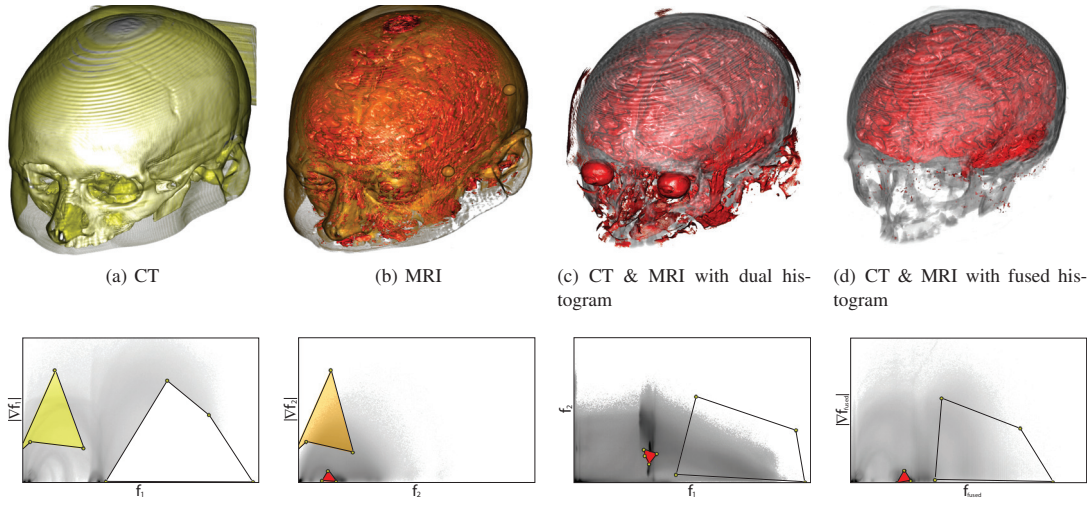
As described in Section 3.4 the definition of a transfer function is done in two steps. In Figure 5(h) only the regions are shown which assign a color and non-zero opacity to sample points. Furthermore for each of these regions a windowing function for the δ value is defined. This function is used to refine the separation by the transfer function. In Figure 6(a) the rendering result is shown which is generated without the usage of a windowing function for δ. The region which is used to assign optical properties to the brain is the same as used for Figure 5(d). It can be seen that the result contains a lot of artifacts. In comparison to that, Figure 6(b) shows a result which is generated by the additional usage of a windowing function for δ to modify the opacity. Through the refinement of the classification with the windowing function most of the artifacts are gone and the brain is clearly separated.



(a) No δ windowing function (b) With δ windowing function

Figure 6: *The two results show the effect of the usage of* δ *to modify the optical properties of a 2D region in the transfer function space.*

Besides the reduction of artifacts the strength of the additional δ value is the ability to find regions with high differences in both data sets. This can be very helpful for several

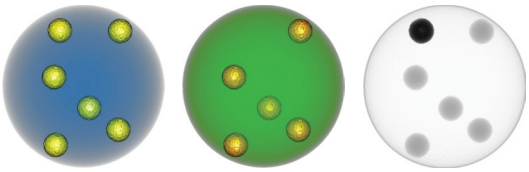


(a) CT (b) MRI (c) CT & MRI with dual histogram (d) CT & MRI with fused histogram

(e) CT histogram (f) MRI histogram (g) Dual histogram (h) Fused histogram

Figure 5: *The images show single volume visualizations of CT data (a) and MRI data (b) in contrast to multimodal visualizations by using the dual transfer function space (c) and the fused transfer function space (d). Histograms (e)-(h) with the colored 2D regions for the assignment of optical properties correspond with the above visualizations.*

applications, such as the finding of a tissue which only shows up in one modality. Due to the properties of δ as described in Section 3.3 regions with opposite information in both data sets have a high δ value. Figure 7 shows the response of the δ value for the combination of two example data sets. In Figure 7(a) and Figure 7(b) two data sets are shown which only differ at one region where in modality 1 a sphere exists and in modality 2 not. Figure 7(c) shows the corresponding distribution of δ values for the two modalities. In the region where the sphere is represented in only one modality the δ value is the highest due to complementary information.



(a) Modality 1 (b) Modality 2 (c) δ distribution

Figure 7: *The image in (c) shows the distribution of* δ *in volume space. It is highest in regions with the largest difference. In this case the largest difference occurs where in modality 1 (a) a sphere exists and in modality 2 (b) not.*

Figure 8 shows a result of a multimodal visualization for the combination of a CT scan and a PET scan generated by the new approach. The regions of high acitivity inside the brain and in the tumor on the neck are shown more opaque. This example proves that the method also works with the combination of anatomical and functional modalities and, furthermore, with different spatial resolutions.

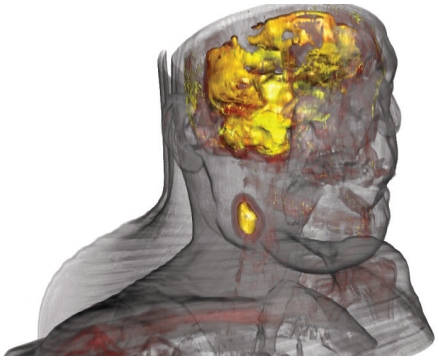


Figure 8: *Multimodal visualization of a CT and PET scan. The more opaque regions symbolizes regions of high activity such as in the brain and in the tumor on the neck.*

6. Conclusion and Future Work

In this paper we introduced a novel approach for the definition of transfer functions for multimodal visualization. The initial idea was to define a user-friendly transfer function

space, which makes it easy to find an expressive transfer function in order to visualize certain tissues of both modalities. Through the fusion of the data values, based on the information content, a 2D transfer function space is defined which is similar to the well-known 2D transfer function space of single volume visualization with value and gradient magnitude as the two dimensions. Therefore, the distribution of points in this transfer function space is easier to understand by the user. An additional δ value, which describes the complementary information contained in a pair of values, is used for a better separation of different tissues. In the result section we have shown how the new transfer function space can be used to show relevant parts of both modalities together.

In comparison to other approaches, which are used for multimodal visualization, the benefit of the new approach is the conversion of the classification problem to a problem which is already known from classification in single volume rendering. A penalty of the new method is that more information does not always mean more importance. So it can happen that, e.g., artifacts can have high information content while other, more important parts have lower information content. But anyway, the user can control this by defining a transfer function which has low opacity for such unimportant parts.

An idea for future work is the extension of the method to more than two modalities. For this reason the approach can be modified to generate a single fused value and fused gradient as a combination of all values and gradients from all modalities. The same modification can be done with the calculation of the opposite information as well. With these modifications we still get a transfer function space with the same dimensionality as for two modalities. The question is if different tissues are still separable in this transfer function space. The answer to that question will be part of the new research.

References

[AM07] AKIBA H., MA K.-L.: A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization* (2007), pp. 115–122.

[CS99] CAI W., SAKAS G.: Data intermixing and multi-volume rendering. In *Computer Graphics Forum* (1999), vol. 18, pp. 359Ű–368.

[EMT*91] EVANS A., MARRETT S., TORRESCORZO J., KU S., COLLINS L.: MRI-PET correlation in three dimensions using a volume-of-interest (VOI) atlas. *Journal of Cerebral Blood Flow and Metabolism 11*, 2 (1991), A69–A78.

[HBKS05] HONG H., BAE J., KYE H., SHIN Y.-G.: Efficient multimodality volume fusion using graphics hardware. In *International Conference on Computational Science (3)* (2005), pp. 842–845.

[KD98] KINDLMANN G., DURKIN J. W.: Semi-automatic generation of transfer functions for direct volume rendering. In *VVS '98: Proceedings of the 1998 IEEE Symposium on Volume Visualization* (1998), pp. 79–86.

[KEF07] KIM J., EBERL S., FENG D.: Visualizing dual-modality rendered volumes using a dual-lookup table transfer function. *Computing in Science and Engineering 9*, 1 (2007), 20–25.

[KHGR02] KNISS J., HANSEN C., GRENIER M., ROBINSON T.: Volume rendering multivariate data to visualize meteorological simulations: a case study. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002* (2002), pp. 189–195.

[KKH01] KNISS J., KINDLMANN G., HANSEN C.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *VIS '01: Proceedings of the 12th IEEE Visualization 2001* (2001), pp. 255–262.

[KPI*03] KNISS J., PREMOZE S., IKITS M., LEFOHN A., HANSEN C., PRAUN E.: Gaussian transfer functions for multi-field volume visualization. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003* (2003), pp. 65–72.

[KSW*04] KNISS J., SCHULZE J. P., WÖSSNER U., WINKLER P., LANG U., HANSEN C.: Medical applications of multi-field volume rendering and VR techniques. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization* (2004), pp. 249–254.

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3 (1988), 29–37.

[LHT*89] LEVIN D., HU X., TAN K., GALHOTRA S., PELIZZARI C., CHEN G., BECK R., CHEN C., COOPER M., MULLAN J.: The brain: integrated three-dimensional display of MR and PET images. *Radiology 172* (1989), 783–789.

[PBSK00] PFISTER H., BAJAJ C., SCHROEDER W., KINDLMANN G.: The transfer function bake-off. *VIS '00: Proceedings of the 11th IEEE Visualization 2000* (2000), 523–526.

[RSKK06] REZK-SALAMA C., KELLER M., KOHLMANN P.: High-level user interfaces for transfer function design with semantics. In *VIS '06: Proceedings of the 17th IEEE Visualization 2001* (2006), pp. 1021–1028.

[SBS*87] SCHAD L., BOESECKE R., SCHLEGEL W., HARTMANN G., STURM V., STRAUSS L., LORENZ W.: Three dimensional image correlation of CT, MR, and PET studies in radiotherapy treatment planning of brain tumors. *Journal of Computer Assisted Tomography 11*, 6 (1987), 948–954.

[Sha48] SHANNON C. E.: A mathematical theory of communication. *Bell System Technical Journal 27* (1948), 379–423,623–656.

[SZHV94] STOKKING R., ZUIDERVELD K. J., HULSHOFF POL H. E., VIERGEVER M. A.: SPECT/MRI visualization for frontal-lobe-damaged regions. *Visualization in Biomedical Computing 1994 2359*, 1 (1994), 282–290.

[WVA*96] WELLS III W. M., VIOLA P., ATSUMI H., NAKAJIMA S., KIKINIS R.: Multi-modal volume registration by maximization of mutual information. *Medical Image Analysis 1* (1996), 35–51.

[ZV94] ZUIDERVELD K. J., VIERGEVER M. A.: Multi-modal volume visualization using object-oriented methods. In *VVS '94: Proceedings of the 1994 IEEE Symposium on Volume Visualization* (1994), pp. 59–66.

5 Contextual Picking of Volumetric Structures

This paper presents a novel method for the interactive identification of contextual interest points within volumetric data by picking on a direct volume rendered image. In clinical diagnostics the points of interest are often located in the center of anatomical structures. In order to derive the volumetric position which allows a convenient examination of the intended structure, the system automatically extracts contextual meta information from the DICOM (Digital Imaging and Communications in Medicine) images and the setup of the medical workstation. Along a viewing ray for a volumetric picking, the ray profile is analyzed for structures which are similar to predefined templates from a knowledge base. We demonstrate with our results that the obtained position in 3D can be utilized to highlight a structure in 2D slice views, to interactively calculate centerlines of tubular objects, or to place labels at contextually-defined volumetric positions.

The following paper appears in its original format published as:

P. Kohlmann, S. Bruckner, A. Kanitsar, and M. E. Gröller. Contextual picking of volumetric structures. In *Proceedings of IEEE Pacific Visualization 2009*, pages 185–192, 2009.

Contextual Picking of Volumetric Structures

Peter Kohlmann* Vienna University of Technology
Stefan Bruckner† Vienna University of Technology
Armin Kanitsar‡ AGFA HealthCare
M. Eduard Gröller§ Vienna University of Technology

ABSTRACT

This paper presents a novel method for the interactive identification of contextual interest points within volumetric data by picking on a direct volume rendered image. In clinical diagnostics the points of interest are often located in the center of anatomical structures. In order to derive the volumetric position which allows a convenient examination of the intended structure, the system automatically extracts contextual meta information from the DICOM (Digital Imaging and Communications in Medicine) images and the setup of the medical workstation. Along a viewing ray for a volumetric picking, the ray profile is analyzed for structures which are similar to predefined templates from a knowledge base. We demonstrate with our results that the obtained position in 3D can be utilized to highlight a structure in 2D slice views, to interactively calculate centerlines of tubular objects, or to place labels at contextually-defined volumetric positions.

Keywords: Contextual Visualization, Smart Interaction, Linked Views, Medical Visualization, Feature Selection.

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; J.3 [Life and Medical Sciences]: Medical Information Systems—

1 INTRODUCTION

Current computer hardware allows to display volume data with different rendering techniques simultaneously and in real time. For a certain medical diagnostic task in the clinical routine, a digital hanging protocol defines how the data is reformatted and arranged on the screen. For some examinations, e.g., in mammography, the hanging protocol is highly standardized, whereas, e.g., in vascular examinations, more often customized hanging protocols are preferred. Frequently, Multi-Planar Reformatting (MPR) is the technique of choice to provide sectional renderings. With a Curved Planar Reformation (CPR) the whole extent of a tubular structure is displayed within a single image. These 2D renderings are often accompanied by a Direct Volume Rendering (DVR) technique like raycasting. The examination of a structure in its three-dimensional setting often provides better insights into contextual information.

A typical hanging protocol arranges a side-by-side presentation of different views of the volumetric data. The physician performs various interactions during the examination of the data. Examples of frequently recurring interactions are scrolling through slices (2D images), zooming, panning, labeling, windowing (2D/3D images), or viewpoint selection and clipping (3D images). The synchronization of the different views is quite challenging because it is not trivial to determine if an interaction in one view leads to changes within another view. In earlier work [7, 8] we presented the LiveSync interaction metaphor, a solution for the live synchronization of a 2D slice view and a 3D volumetric view. The viewing parameters for the 3D view are derived automatically from a picking of the anatomical structure of interest on the 2D slice. In that work the viewing parameters are viewpoint, zoom factor, clipping planes, and transfer function setup.

This paper presents a new approach to handle the picking of a structure which is directly performed on the 3D volumetric view in a context-sensitive way. Therefore, the first step is the identification of the 3D position of interest within the volumetric data. For each point on a 2D slice its exact 3D position can be calculated easily, whereas the picking of a point in the 3D volumetric view is not well defined. This is due to the fact that for each pixel on the screen ray casting is performed, where opacity and color is accumulated along a ray from the eye point through the volume. Potentially each position along this ray might be the desired volumetric position. A simple solution to this problem is the definition of a *first-hit position* as the location along the ray where a certain opacity threshold is exceeded. For some cases this might be sufficient, but often a different, contextually-defined position is of interest. We propose ray-profile templates which are designed to represent anatomical structures like, e.g., a vessel, the aorta, the airway, or a vertebra. The ray profile which is calculated for each picking on the 3D volumetric view is then scanned and analyzed to find similarities to the defined ray-profile templates in a knowledge base.

To narrow down the number of anatomical structures which are of relevance for a certain examination, the presented method extracts contextual information automatically from the DICOM data. The DICOM standard [16] provides a detailed specification of a format for medical images which includes meta information like parameters of the scanner and patient information. Further context is provided by the setup of the medical workstation and the selection of clinical tools. For the examination of anatomical structures, frequently center points are of special interest. The *first-hit* solution cannot provide these positions because of self-occlusion or occlusion by other structures. Our proposed method can either return the first hit of the determined structure of interest or its center along the analyzed ray.

When the 3D position of interest is located, the next important step is the presentation of the result to the physician. This paper depicts the highlighting of the results by synchronized 2D slice views as a straight-forward solution. Also the placement of labels at the appropriate 3D positions is demonstrated. For instance, the whole spine can be labeled by picking each vertebra directly in the 3D volumetric view. Further it is shown that approximate centerlines can be calculated interactively by tracing along tubular structures in the 3D view. These centerlines could be utilized to display CPR renderings of the structure or to guide a segmentation process.

2 RELATED WORK

In medical visualization some techniques have been developed to ease the interaction with multiple views of a certain data set. Kohlmann et al. [7, 8] presented the first attempt to combine optimal viewpoint estimation and synchronized views for the visualization of medical volume data. Götzelmann et al. [6] presented an approach where 3D visualizations are linked with textual descriptions, e.g., from medical textbooks. Their approach focuses on an educational purpose and supports students to learn the terminology and to understand textual descriptions of complex objects.

*e-mail: kohlmann@cg.tuwien.ac.at
†e-mail: bruckner@cg.tuwien.ac.at
‡e-mail: armin.kanitsar@agfa.com
§e-mail: groeller@cg.tuwien.ac.at

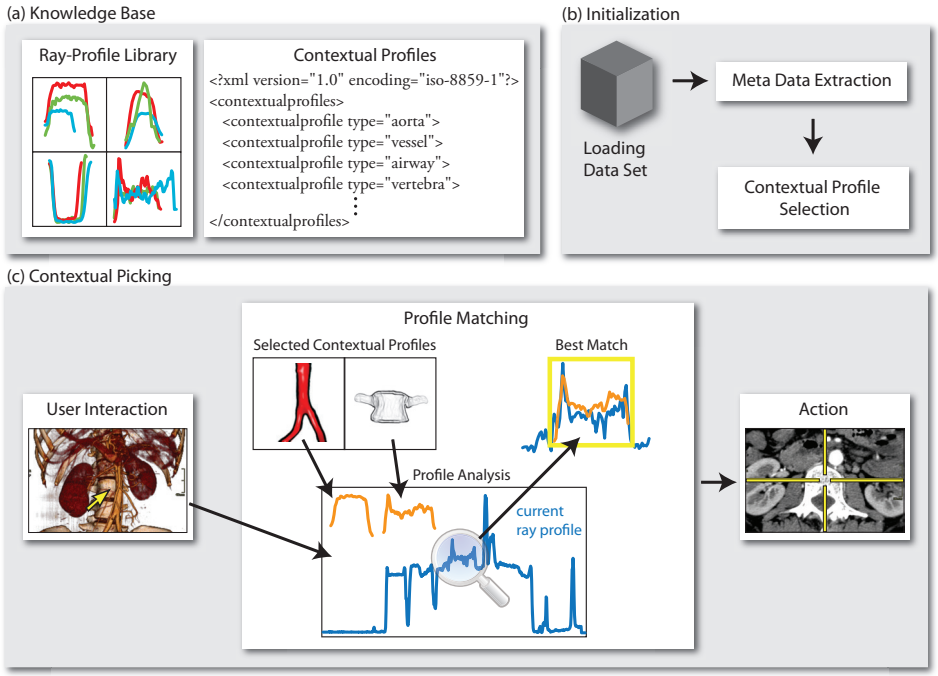


Figure 1: Contextual picking overview: (a) A knowledge base provides a library of ray profiles of anatomical structures together with an XML description of available contextual profiles. (b) An initialization step is performed during data set loading. Meta information is extracted to automatically select contextual profiles from the knowledge base. (c) For a picking on the volume rendered image, the current ray profile is analyzed to detect anatomical structures which are represented by the selected contextual profiles. If a good match is given, then this can be utilized for instance to highlight an interest point (e.g., the center of a vertebra) in the slice view.

Related to this work, IBM is currently developing the Anatomic and Symbolic Mapper Engine [3]. This technology uses a 3D model of the human body which is linked to medical records. Whenever the doctor clicks on a certain part of the body, a search of the medical records is triggered to extract the relevant information. The Medical Exploration Toolkit presented by Tietjen et al. [18] bundles various concepts for loading, visualizing, and exploring segmented medical data sets. Critical distances to pathological structures are computed, and displayed in a synchronized manner in 2D and 3D. The integrated LIFTCHART [17] displays the overall extents of structures within the volume in a narrow frame as color bars. A structure can be selected in the LIFTCHART and the corresponding slice is displayed in the slice viewer. Another clinical application they integrated is the NECKSURGERYPLANNER [19]. Segmented structures can be enabled and disabled by a textual representation. They are synchronously highlighted in the 3D and 2D views to support operation planning for neck dissections.

A lot of research has concentrated on the extraction of certain anatomical structures. Multi-scale filtering approaches are very popular for the detection of curvilinear structures. Vessel enhancement filters based on eigenvalue analysis of the Hessian matrix have been proposed, e.g., by Sato et al. [13] and Frangi et al. [5]. Tek et al. [14] presented an approach which focuses on the segmentation of vessel cross sections. A single click inside the vessel on a slice initiates mean shift-based ray propagation to detect the boundary of the vessel. Tschirren et al. [20] presented an airway segmentation algorithm based on fuzzy connectivity. Their method uses small adaptive regions of interest which follow the airway branches during the segmentation process. Kovács et al. [9] developed a system for automatic segmentation of the entire aorta without any user interaction for treatment planning of aortic dissections. The segmentation is based on a Hough transformation to detect the approximate circular shape of the aorta. To fit this shape more closely to the actual contour of the aortic lumen an elastic mass-spring deformable model is utilized. An interesting concept for the detection of tubular objects in medical imaging is the *Gradient Vector Flow (GVF) snake* introduced by Xu and Prince [21]. This method first calculates a field of forces (GVF forces) over the image domain. These forces drive the snake to fit to the boundaries of an object. Bauer and Bischof [1, 2] utilize the properties of the GVF for the detection of tubular objects and the extraction of curve skeletons, e.g., for virtual endoscopy. They argue that conventional tube detection or line filters which use local derivatives at multiple scales have problems with undesired diffusion of nearby objects. Their GVF-based method allows an edge-preserving diffusion of gradient information. Malik et al. [10] presented a rendering algorithm called feature peeling. They analyze peaks and valleys of intensity ray profiles for a given viewpoint to detect features inside the volume data. By classifying a number of feature layers it is possible to scroll through the layers to inspect various structures.

Sketch-based techniques are employed to classify and segment volume data by painting directly on the volume rendering. Owada et al. [11] presented a sketching interface which allows an intuitive segmentation of volumetric regions. The user has to draw 2D strokes along the contour of the 3D target region to initiate a constraint segmentation process. Related to this approach, Chen et

al. [4] applied sketching to seeded region growing for volume segmentation. Initially, the user specifies a region of interest by placing a closed free-form sketch on the volume rendering. Then a region of interest is extruded to facilitate the definition of seed points for the region growing. Ropinski et al. [12] proposed an interface for the design of 1D transfer functions which is based on the drawing of strokes on the volume rendered image. Features of interest can be identified by strokes which are close to their silhouettes. Based on a histogram analysis, component transfer functions are automatically generated for the identified features. The user can then decide which of the selected features should be integrated into the final transfer function.

3 Contextual Picking Overview

The overview of related works indicates that often highly specialized methods are used to detect various anatomical structures within the volumetric data. This paper presents a generalized system which is not limited to a certain type of structure. An example illustrates one potential application area for the presented system: A frequently needed task during an orthopedic examination of the spine is the labeling of individual vertebrae. The vertebral column consists of 7 cervical, 12 thoracic, 5 lumbar, 5 sacral, and 3 to 5 coccygeal vertebrae. If the labeling is performed in 2D slice views only, quite some scrolling through the slices is involved to navigate to a meaningful 3D position where a single label is placed. A good position for the label is the center of the vertebral body. By picking on the vertebrae in the 3D view, contextual picking allows a convenient labeling. If the label for the first picked vertebra and the labeling direction is given, then a single contextual picking on the following vertebrae is sufficient to add the appropriate labels at a central position within the vertebral body.

Figure 1 shows the building blocks to achieve contextual picking. A knowledge base consists of a ray-profile library and contextual profiles. The ray-profile library holds ray-profile samples (intensities and gradient magnitudes) of various anatomical structures. A contextual profile for a certain structure bundles the needed information to react to a contextual picking operation. In the XML format it describes the following components: The type of the structure, a list of keywords, minimal and maximal extent of a structure, a representative mean ray profile built from the samples in the ray-profile library, and the default reaction to a picking operation. An initialization step is performed whenever a new data set is loaded into the workstation. The DICOM header as well as the workstation environment is analyzed to extract the relevant meta data for selecting the applicable contextual profiles.

Contextual picking is initiated by positioning the mouse cursor on the 3D view and pressing a hot-key. Whenever the physician picks on a structure of interest in the 3D view, the following steps are performed: First, information from the current picking, which includes the intensity and gradient magnitude values along the viewing ray, as well as accumulated opacities and positions of clipping planes, are collected. Second, the representative mean ray profiles of the selected contextual profiles are compared to the current ray profile. Finally if a good match is detected, this result is utilized to highlight the anatomical structure of interest in an appropriate way, e.g., in MPR views.

4 Knowledge Base

Most approaches to automatically detect features in medical volume data need a considerable amount of user interaction to set up the needed parameters. Besides, they are often very specialized on a certain type of anatomical structure as well as on a specific extent of the feature. Two important preconditions for the presented system are that it has to be as generic as possible and easy to extend. For these reasons a ray-profile library was set up which is supported by an easy-to-use interface for the generation of new ray-profile samples. Together with the contextual profiles this knowledge base provides all information to react to a contextual picking.

4.1 Ray-Profile Library

The ray-profile library is implemented as an XML file which stores ray-profile samples for various anatomical structures. A ray-profile sample for a certain structure consists of a sample id, a textual description, the spacing along the ray in mm, the extent of the structure in mm, and a list of intensity and gradient magnitude values. The system provides a convenient user interface to add new ray-profile samples to the ray-profile library. Figure 2 illustrates the generation of a ray-profile sample of a contrast-enhanced blood vessel. After picking on a vessel in the 3D view, a ray profile is generated and displayed. A plot of the ray profile shows the intensity values (blue) and the values of the gradient magnitude (green) along the ray. To ease the selection of a ray-profile sample, the color transfer function for the corresponding scalar values is displayed in a small horizontal bar below the profile plot.

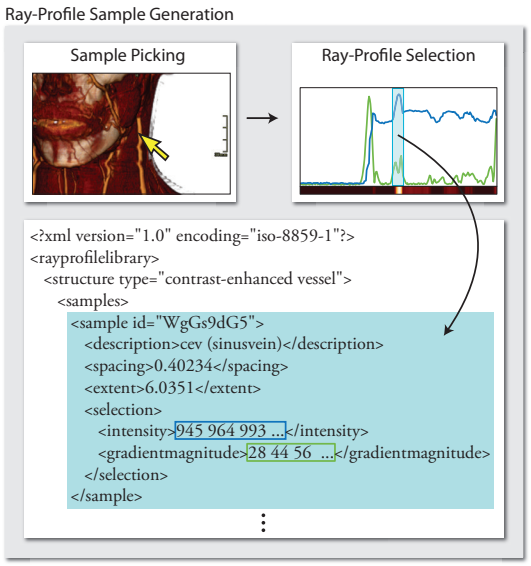


Figure 2: The generation of a ray-profile sample for a contrast-enhanced vessel is initiated by picking on the structure in the 3D view. A ray profile is displayed which shows the intensity and the gradient magnitude values along the viewing ray. From a selected subset of this ray profile, which represents the picked vessel, a sample is generated and stored in the ray-profile library.

By dragging the mouse with the left button pressed a semi-transparent window is painted over the ray profile which represents a selection along the ray. By confirming this selection, a sample for a new or an already existing structure is written to an XML file. For a certain anatomical structure it is recommended to generate several representative samples. The intensity values of an anatomical structure can vary slightly, e.g., because of the patient's age or sex. Further the size of the structures varies because of the mentioned factors. A set of multiple samples in a contextual profile for example enables to detect vessels of a certain diameter range. The generation of ray-profile samples has to be done only once for each anatomical structure. They are added to the library by a domain expert. This step is not visible to the physician who just uses the contextual picking.

4.2 Contextual Profiles

The contextual profiles are stored in an XML file which contains all needed instructions to react to a contextual picking. Listing 1 shows the XML skeleton of the contextual profile for an anatomical structure. First of all it has a *type* entry which has to match with the corresponding *structure types* in the ray-profile library. A list of keywords describes the type of data sets and/or the setup of the medical workstation in which the defined structure is of interest. The *extent* defines a range of the typical extent of the anatomical structure. In the *meanrayprofile* entry, a mean ray profile is stored which is generated from the available ray-profile samples in the ray-profile library. Finally, *return* defines which position will be returned (e.g., the center of the structure) and which default action shall be performed (e.g., highlighting of the obtained position in MPR views).

Listing 1: XML skeleton of a contextual profile

```
<contextualprofile type="">
    <keywords>...</keywords>
    <extent>...</extent>
    <meanrayprofile>
        <spacing>...</spacing>
        <intensity>...</intensity>
        <gradientmagnitude>...</gradientmagnitude>
    </meanrayprofile>
    <return>
        <position>...</position>
        <reaction>...</reaction>
    </return>
</contextualprofile>
```

Mean ray-profile generation: The generation of a mean ray profile is motivated by the variation of intensities and extents of anatomical structures. To obtain a good representation of a structure, all samples from the ray-profile library which correspond to the type of the contextual profile are collected. Figure 3 (left) shows three intensity ray-profile samples of the aorta which were all captured using different data sets. The similarities of the three samples are clearly visible. The intensity values are in a range of Hounsfield Units between about 880 and 1440 (shown on the y-axis of the plots) and the extents of the samples differ in a range between about 17.5 and 28.5 mm (shown on the x-axes of the plots). In general the samples start with a steep intensity ascent followed by a plateau and a steep intensity descent. The generated mean sample (right) shows the same characteristics with the advantage that outliers are filtered out. The algorithm for the generation of the mean ray profile first calculates the mean extent of the available samples. Then the ray-profile samples are scaled horizontally. The scaling factor is determined by dividing the mean extent by the extent of the current sample. Afterwards the mean of the corresponding intensity values is calculated at uniformly distributed positions along the x-axis. Analogously, a mean ray profile is also generated for the gradient magnitudes. Taking the mean extent of the ray-profile samples seems to be an appropriate approach because we assume that there is an approximate Gaussian distribution of the extents for multiple samples of a single structure.

The described algorithm for the generation of the mean ray profiles is well suited to preserve slopes which are characteristic for a certain structure. This is due to the fact that the steepness of slopes in shorter samples is decreased and the steepness of slopes in longer samples is increased by the horizontal scaling. Taking the mean of the intensity values results in a mean ray profile which represents the anatomical structure accordingly. Mean calculations have to be performed only when a new sample is added to the ray-profile library.

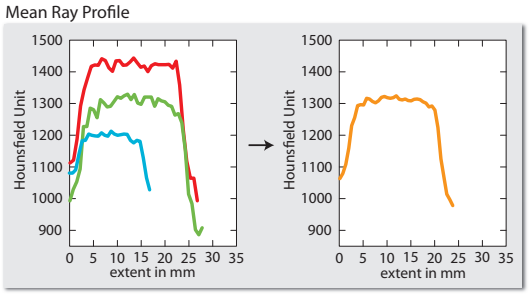


Figure 3: Three samples of the aorta (left) with varying extent and intensity range are collected from the ray-profile library. These samples are utilized to construct a representative mean ray profile (right).

5 Initialization

The major challenge for a system which allows contextual picking of anatomical structures in the 3D view is to gain as much information as possible about the structure of interest. Typically for a certain type of examination only a small number of structures is relevant for the diagnosis. For instance in vascular examinations veins and arteries are of special interest whereas in an orthopedic examination the spine and bones are more important structures. There are two reasons to narrow down the number of target structures in a certain examination. First, our approach is based on a matching between ray-profile samples from a knowledge base and the ray profile extracted for the current picking. In volume data, structures are occluding each other. Thus the analysis of the current ray profile with respect to all structures given in the contextual profiles, might lead to ambiguous results. For example a vessel in front of a bone is usually not relevant for an orthopedic examination. In such cases the contextual picking has to respond appropriately. Second, the analysis of the current ray profile through matching with a lot of structures of varying extents can lead to high computational costs.

As this work aims to minimize the overhead of the user interaction, the presented system extracts valuable meta information directly from the DICOM headers and the current setup of the medical workstation. The DICOM format is widely used in hospitals as a standard for handling, storing, printing and transmitting information in medical imaging. DICOM files contain the image pixel data of the scanned slices combined with header information. A *Patient* entity contains relevant data about the patient like name, age, and sex. Data concerning the examination like the name of the physician or the description of the examination is stored in a *Study* entity. A *Series* entity represents a logical unit of information about the imaging modality (*Equipment*) and information about the spatial relations of images within a series (*Frame of Reference*). Finally, the image itself and meta data about the image are stored in an *Image* entity. Grouping the image and the meta data prohibits that the image data gets separated from the meta information [16]. To narrow down the number of structures the physician might be interested in, we identified a suitable set of entries in the DICOM header:

- (0018, 0015) - Body Part Examined
- (0008, 1030) - Study Description
- (0008, 103E) - Series Description
- (0040, 0254) - Performed Procedure Step Description
- (0018, 1030) - Protocol Name

After a new data set is loaded, our system extracts the textual description which is stored for these entries. Entries which contain details about the patient like sex, age, and weight are further candidates which could be utilized to gather information about the size and the intensity range of a specific structure. The system also considers if the medical workstation is only used for certain examinations. For instance often a medical workstation with reduced functionality is available, e.g., as a vascular, orthopedic, cardiac, or mammography workstation.

The automatically extracted information is used to select the suitable contextual profiles from the knowledge base. Each contextual profile contains a list of classified keywords to decide if the represented structure is relevant for the currently loaded data set with the current setup of the workstation. Listing 2 shows exemplary keywords in a contextual vertebra profile.

Listing 2: Keywords of the contextual vertebra profile

```
<keywords>
    <strong>
        Workstation=Orthopedic
        BodyPartExamined=*SPINE ...
    </strong>
    <medium>
        BodyPartExamined=ABDOMEN ...
    </medium>
    <weak>
        BodyPartExamined=HIP ...
    </weak>
    <kickout>
          Workstation=Cardiac
          Workstation=Vascular ...
    </kickout>
</keywords>
```

A list of keywords is categorized into the classes *strong*, *medium*, *weak*, and *kickout*. The information which is extracted during loading the data set is compared with these entries to decide if a certain anatomical structure is relevant for the current examination. In the presented example a vertebra is strongly relevant when using an orthopedic workstation and/or when the examined body part is the spine. Within a cardiac or a vascular workstation a vertebra is typically not a structure of interest. With this approach it is possible to select from a ranked list the contextual profiles which are suitable in the given environment. The ranking is given by a comparison of the extracted information with the classified keywords.

6 Contextual Picking

After one or more contextual profiles are automatically selected, the physician can perform the picking directly on the 3D view and the system provides an immediate feedback. For each contextual picking, the current ray profile is analyzed to find close similarities in the selected contextual profiles. This analysis is done by a profile-matching algorithm which evaluates a cost function to measure the degree of similarity. As our main focus was to provide a framework for contextual visualization we did not deeply investigate the applicability of established pattern classification methods from fields like signal processing, computer vision, or neurocomputing. Our empirical approach gives satisfactory results and the comparison to other methods has to be done in future work. Based on the outcome of the profile matching the respective action is taken. Three actions have been implemented so far. The default action is the highlighting of the center of the picked anatomical structure in MPR views. Further, contextual picking is integrated into a spine labeling system to demonstrate its potential to place labels at meaningful 3D positions. Finally the system allows the calculation of approximate centerlines of picked tubular structures.

6.1 Profile Matching

The profile matching to detect the anatomical structure of interest is repeated for each of the selected contextual profiles. A contextual profile provides a mean ray profile which represents a structure together with the information about a minimal and a maximal extent of the structure. To allow the search for structures within this range along the current ray profile, a non-uniform scaling of the mean ray profiles is performed. In Figure 4 the minimal (left) and the maximal scaling (right) is shown for the mean ray profile (middle) of the aorta to cover a structure extent from about 16 to 32 mm. The scaling algorithm only performs a scaling on positions where the gradient magnitude is low. This avoids an undesirable change of the steepness of the major slopes.

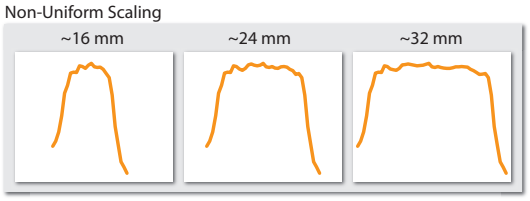


Figure 4: The mean ray profile of the aorta (middle) is non-uniformly scaled between a minimal (left) and a maximal range (right) to match the corresponding structure in the current ray profile.

The implementation of the profile matching is described in Algorithm 1. In this algorithm the mean ray profile and the current ray profile are represented by intensities as well as gradient magnitudes. Thus, the summed Euclidean distances are calculated for the corresponding intensities and gradient magnitudes. The matching

Algorithm 1 Structure detection by profile matching.

```
1:  set pos = 0, width = 0, minCost = MAX_VALUE
2:  for each scaling of the mean ray profile m do
3:      for each step s along the current ray profile c do
4:          if s + m.length < c.length then
5:              sum up squared Euclidean distances (SEDs) at the corresponding positions along m and c (s to s + m.length)
6:          end if
7:      end for
8:      set profile matching cost to sum_of_SEDs/m.length
9:      if cost < minCost then
10:         set pos = s, width = m.length, minCost = cost
11:     end if
12: end for
```

algorithm detects the section along the current ray profile which is most similar to the mean ray profile at different scales. A length normalization of the fitting costs allows the comparison between different scales, as well as the comparison of responses between different contextual profiles. This is especially important if multiple contextual profiles are selected and thus, the ray profile is scanned for different structures. For instance, if an abdominal data set is loaded the contextual aorta profile and the contextual vertebra profile might be selected. To decide which structure is detected, a trade off between the cost function and the suitability of the contextual profile for the current environment has to be calculated. In the following, low costs of the profile matching are equivalent to a good or high response of a contextual profile.

Optimizations: To decrease ambiguities of the contextual profile response and to increase the performance of the system several

optimizations were applied to the algorithm. First of all, the positions of the clipping planes are considered before the profile matching is performed. Only the part of the volume which is not clipped away is analyzed for matching regions. On the one hand this lowers the computational cost and on the other hand it reduces the chances of ambiguities. For example in the case of the contextual vessel profile, a vessel which is located in front of another vessel along the ray but clipped away could lead to unintended results. A second optimization which is applied for similar reasons takes the opacity transfer function into account. The analysis of the ray profile starts only at a sample position where a small opacity threshold is exceeded. From this position on the remaining part of the ray (if not clipped away) is considered. Third, the cost function for the profile matching is implemented so that the costs slightly increase along the ray. With this adjustment, structures which are closer to the viewer, e.g., vessels which are in front of other vessels, return lower costs. To apply this modified cost function line 8 in Algorithm 1 is replaced by the equation

$$cost = (sum\_of\_SEDs/m.length) \times (1 + 0.5 \times s/c.length), \quad (1)$$

where sum_of_SEDs are the summed-up squared Euclidean distances, $m.length$ is the sample size of the current matching profile, s is the current sample position along the ray profile, and $c.length$ is the total sample size of the ray profile. Our experiments have shown that the multiplication of the costs with a penalty factor of up to 1.5 for distant structures leads to good results for features which appear multiple times along the viewing ray. Despite of the improvements achieved by Equation 1 two overlapping target structures which are very close to each other might still be problematic. An example is if the aorta due to the current viewpoint of the volumetric view is right in front of some vertebrae and contextual profiles for both structures are active.

If the user is continuously tracing along the aorta it would be quite disturbing if the contextual vertebra profile and the contextual aorta profile alternate in generating the better response. The proposed solution to this problem is to deactivate all but one contextual profile as long as a continuous tracing goes on. As soon as the hot-key is released the other contextual profiles are re-activated. Finally, a default contextual profile is implemented which returns the *first-hit position*. The opacity is accumulated along the ray until a certain opacity threshold is reached. This contextual profile becomes active if the cost function of the profile matching returns too high values, which means that no contextual profiles are detected along the current ray profile.

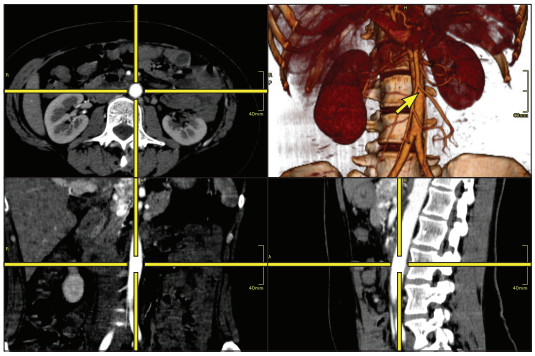
6.2 Contextual Picking Action

The implemented default action to react to a contextual picking is the highlighting of the detected interest point in the MPR views. For each picking on the volumetric view, a three-dimensional position within the data set is computed. This position can be the center of the target structure along the current viewing ray or the first-hit position if no target structure is detected. The structure's center can be calculated easily as the start and the extent of the structure along the viewing ray is determined by the profile-matching algorithm. To show the obtained position in the MPR views, the axial, coronal, and sagittal cross sections for this volumetric position are displayed. The centering of this position on the slices, as well as the overlay of crosshairs is used to highlight the target structure.

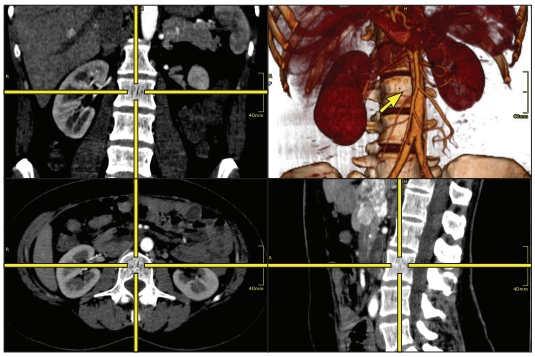
Another proposed action following a contextual picking is the labeling of anatomical structures. Often the labeling is performed in the slice views alone although the volumetric view can be very well suited for this task. For instance by utilizing the contextual vertebra profile, the physician gets more flexibility in the spine labeling process. The whole spine can be easily labeled in the 3D view. A single contextual picking on each vertebra determines the exact three-dimensional position of each label. Finally, the estimation of feature center points during the continuous tracing along a structure can be utilized to calculate approximate centerlines of tubular structures like vessels. If the obtained approximation is not sufficient it can be a helpful initial input for more accurate centerline-detection algorithms.

7 Performance and Results

The contextual picking is implemented in Java and the contextual profiles as well as the ray-profile library are stored in the XML format. For parsing and manipulation of the XML files within the Java classes the JDOM API [15] is used. The contextual picking is integrated into a real-world medical workstation which is under development by our collaborating company partner. All contextual picking-related computations are performed interactively. For the generation of the ray-profile samples in our ray-profile library eight different CT data sets were used. Three samples were taken for each anatomical structure from suitable data sets. The data sets shown in the result images of this section are different from the data sets which were used to establish the ray-profile library.



(a)



(b)

Figure 5: Contextual picking on a thoracic-abdominal CT data set. The 3D position which is returned by the contextual profile with the best response is used to provide meaningful MPR views of the picked aorta (a) and the picked vertebra (b).

In Figure 5 the contextual picking is illustrated for a thoracic-abdominal CT data set. After the data set is loaded into the workstation, meta information is extracted according to the description in Section 5. Based on this information, the contextual vertebra

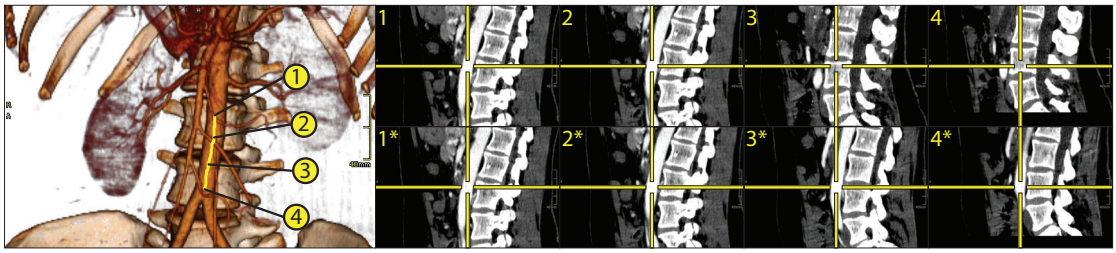


Figure 6: Continuous trace path along the aorta (left). This might lead to unintended responses for the vertebra if the contextual profiles of the aorta and the vertebra are selected (top-right [1-4]). The automatic temporary deactivation of the contextual vertebra profile during the tracing leads to a continuous capturing of the aorta (bottom-right [1*-4*]).

profile and the contextual aorta profile are selected automatically. When the aorta is picked as in Figure 5(a), the contextual aorta profile gives the best response. The detected center position of the aorta along the viewing ray is utilized to set up the axial, coronal, and sagittal MPR views. Four different views on the picked anatomical structure are provided to the physician. The picking of a close-by vertebra in Figure 5(b) leads to analogous results.

Figure 6 (left) shows the path (indicated by the yellow line) of a continuous tracing along part of the aorta. If the contextual aorta profile and the contextual vertebra profile are active, the contextual vertebra profile has the better response at some positions along the trace path although the user is interested in the examination of the aorta. Figure 6 (top-right [1-4]) shows the resulting sagittal slice views when only the best response is taken into account. The vertebra is captured as the prominent structure at the positions 3 and 4. Whenever a continuous tracing is performed, the assumption can be made that the user currently examines a single anatomical structure. Thus, just a single contextual profile is active during the continuous tracing and all the others are deactivated temporarily. This leads to the results shown in Figure 6 (bottom-right [1*-4*]). Along the trace path, the aorta is always captured as the prominent structure and jerky leaps in the MPR views between the aorta and a vertebra are avoided. The tracing along a tubular structure allows the computation of its approximate centerline.

Figure 7 depicts the result for the contextual picking of the airway in the 3D view (left) using a head CT data set. The contextual airway profile gives a better response than the contextual vessel profile which is also active. A highlighting of the corresponding position is performed in a 2D slice view (right). Occluding structures do not impede the detection of a central point within the airway.

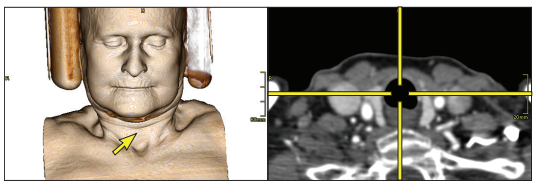


Figure 7: Contextual picking of the airway (left). The identified 3D position is used to provide a meaningful MPR view (right).

Figure 8 shows some results when the contextual picking is integrated into a spine labeling tool. With this tool the user has to specify the label for the first picked vertebra and a labeling direction (head-to-feet or feet-to-head). Then a single picking on each vertebra leads to the results shown in Figure 8(a). Figure 8(b) shows the labeling from another viewpoint if just the first-hit position is taken for the 3D placement of the labels. If the placement is done by taking the positions determined by the contextual picking, the labels are in the center of the vertebral body as shown for the same viewpoint in Figure 8(c). The exact positions of the labels are depicted on the axial, coronal, and sagittal slices for the first-hit approach in Figure 8(d) and for the contextual picking approach in Figure 8(e).

8 Conclusion and Future Work

In this paper we presented a novel method for the interactive identification of contextual interest points within volumetric data by picking on a direct volume rendered image. We built a knowledge base which holds characteristic ray-profile samples for different anatomical structures. New ray-profile samples can be added with an easy-to-use interface by domain experts. A contextual profile bundles information like for which kind of data sets it should be selected, the extent of the target structure, or the action which has to be performed if there is a high response to the contextual profile. Based on this knowledge base, the contextual profiles which are applicable in the current environment are selected automatically. Our profile-matching algorithm analyzes the viewing ray for each contextual picking on the 3D volumetric view. It returns a position within the volume with the best response to one of the selected contextual profiles. Based on this result, certain actions are performed interactively. In the simplest case, the obtained position is the center of the selected structure and is highlighted by crosshairs in MPR views. Because of the interactivity of the underlying computations, contextual picking is well suited to continuously trace the mouse pointer along volumetric structures. This allows to simultaneously examine the selected structure in multiple views. We have also demonstrated that the contextual picking can be easily integrated into a conventional spine labeling framework to increase its flexibility.

Classifying structures only along ray profiles is not a limitation of the contextual picking framework. Other local classification schemes could be added as well. More research is necessary to investigate if this could help to further improve the detection of a structure's interest point. Until now, our method is to a certain degree dependent on the chosen viewpoint of the 3D view. While this could be a problem in some cases, there are often default viewpoints for 3D diagnostic examination procedures. For this reason the viewpoint dependency was not a big issue in the application scenarios which are presented in this paper. Shape-based methods could be integrated into our framework to be more flexible in the selection of appropriate viewpoints. The challenge thereby will be to ensure interactivity. Alternatively, multiple contextual profiles could be provided for different viewpoints on a structure. The integration of techniques to display uncertainty information about the currently detected structure is another interesting direction for further research.

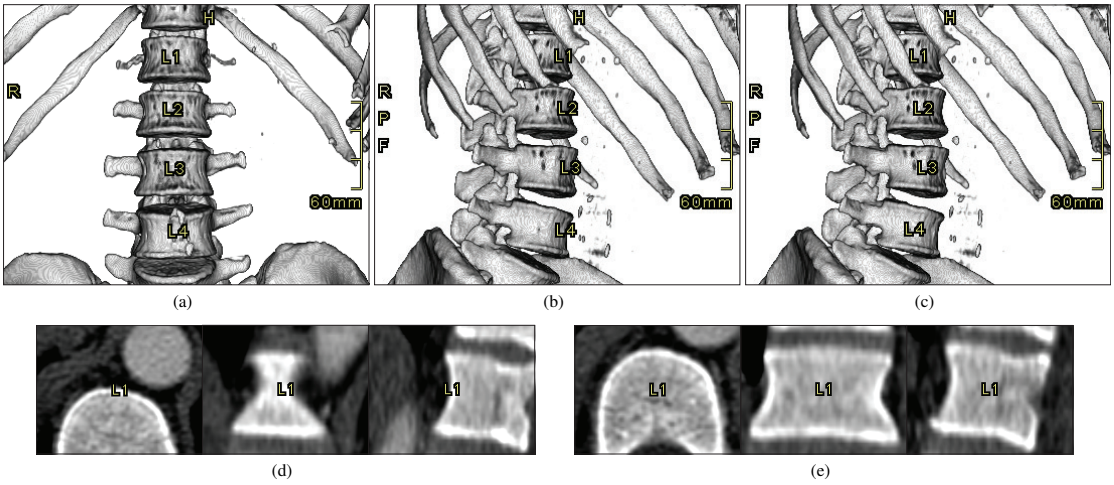


Figure 8: Labeling of lumbar vertebrae. (a) A single picking on each of the four lumbar vertebrae is performed for their labeling. (b) The result from another viewpoint when the first-hit position is taken for label placement. (c) The result for the same viewpoint when the contextual picking result is taken for label placement. (d) The exact labeling positions of L1 with just the first-hit approach. (e) The exact labeling positions of L1 with the contextual picking approach.

ACKNOWLEDGEMENTS

The work presented in this paper has been funded by AGFA HealthCare in the scope of the `DiagVis` project. We thank Rainer Wegenkittl, Lukas Mroz and Matej Mlejnek (AGFA HealthCare) for their collaboration and for providing various CT data sets. Additional data sets are courtesy of the OsiriX Foundation.

REFERENCES

[1] C. Bauer and H. Bischof. Extracting curve skeletons from gray value images for virtual endoscopy. In *Proceedings of the 4th International Workshop on Medical Imaging and Augmented Reality 2008*, pages 393–402, 2008.

[2] C. Bauer and H. Bischof. A novel approach for detection of tubular objects and its application to medical image analysis. In *Proceedings of the 30th DAGM Symposium on Pattern Recognition 2008*, pages 163–172, 2008.

[3] R. N. Charette. Visualizing electronic health records with "Google-Earth for the body". IEEE Spectrum Online, Jan. 2008. Available online at http://www.spectrum.ieee.org/jan08/5854/, September 2008.

[4] H.-L. J. Chen, F. F. Samavati, M. C. Sousa, and J. R. Mitchell. Sketch-based volumetric seeded region growing. In *Proceedings of the Eurographics Workshop on Sketch-Based Interfaces and Modeling 2006*, pages 123–129, 2006.

[5] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. In *Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention – MICCAI 1998*, pages 130–137, 1998.

[6] T. Götzelmann, P.-P. Vázquez, K. Hartmann, T. Germer, A. Nürnberger, and T. Strothotte. Mutual text-image queries. In *Proceedings of Spring Conference on Computer Graphics 2007*, pages 181–188, 2007.

[7] P. Kohlmann, S. Bruckner, A. Kanitsar, and M. E. Gröller. LiveSync: Deformed viewing spheres for knowledge-based navigation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1544–1551, 2007.

[8] P. Kohlmann, S. Bruckner, A. Kanitsar, and M. E. Gröller. LiveSync++: Enhancements of an interaction metaphor. In *Proceedings of Graphics Interface 2008*, pages 81–88, 2008.

[9] T. Kovács, P. C. Cattin, H. Alkadhi, S. Wildermuth, and G. Székely. Automatic segmentation of the vessel lumen from 3D CTA images of aortic dissection. In *Proceedings of Bildverarbeitung für die Medizin 2006*, pages 161–165, 2006.

[10] M. M. Malik, T. Möller, and M. E. Gröller. Feature peeling. In *Proceedings of Graphics Interface 2007*, pages 273–280, 2007.

[11] S. Owada, F. Nielsen, and T. Igarashi. Volume catcher. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games 2005*, pages 111–116, 2005.

[12] T. Ropinski, J. Praßni, F. Steinicke, and K. Hinrichs. Stroke-based transfer function design. In *Proceedings of the IEEE/EG Symposium on Volume and Point-Based Graphics 2008*, pages 41–48, 2008.

[13] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Tissue classification based on 3D local intensity structures for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):160–180, 2000.

[14] H. Tek, D. Comaniciu, and J. P. Williams. Vessel detection by mean-shift based ray propagation. In *Proceedings of IEEE Workshop on Mathematical Methods in Biomedical Image Analysis 2001*, pages 228–235, 2001.

[15] The JDOM API Project Website. Available online at http://www.jdom.org, September 2008.

[16] The National Electrical Manufacturers Association (NEMA). The DICOM Standard. Available online at http://medical.nema.org/, September 2008.

[17] C. Tietjen, B. Meyer, S. Schlechtweg, B. Preim, I. Hertel, and G. Strauß. Enhancing slice-based visualizations of medical volume data. In *Proceedings of IEEE/Eurographics Symposium on Visualization 2006*, pages 123–130, 2006.

[18] C. Tietjen, K. Mühler, F. Ritter, O. Konrad, M. Hindennach, and B. Preim. METK - The medical exploration toolkit. In *Proceedings of Bildverarbeitung für die Medizin 2008*, pages 407–411, 2008.

[19] C. Tietjen, B. Preim, I. Hertel, and G. Strauß. A software-assistant for pre-operative planning and visualization of neck dissections. In *CURAC 2006*, pages 176–177, 2006.

[20] J. Tschirren, E. A. Hoffman, G. McLennan, and M. Sonka. Intrathoracic airway trees: Segmentation and airway morphology analysis from low-dose CT scans. *IEEE Transactions on Medical Imaging*, 24(12):1529–1539, 2005.

[21] C. Xu and J. L. Prince. Gradient vector flow: A new external force for snakes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition 1997*, pages 66–71, 1997.

6 A Multidirectional Occlusion Shading Model for Direct Volume Rendering

In this paper, we present a novel technique which simulates directional light scattering for more realistic interactive visualization of volume data. Our method extends the recent directional occlusion shading model by enabling light source positioning with practically no performance penalty. Light transport is approximated using a tilted cone-shaped function which leaves elliptic footprints in the opacity buffer during slice-based volume rendering. We perform an incremental blurring operation on the opacity buffer for each slice in front-to-back order. This buffer is then used to define the degree of occlusion for the subsequent slice. Our method is capable of generating high-quality soft shadowing effects, allows interactive modification of all illumination and rendering parameters, and requires no pre-computation.

The following paper appears in its original format published as:

V. Šoltészová, D. Patel, S. Bruckner, and I. Viola. A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum*, 29(3):883–891, 2010.

A Multidirectional Occlusion Shading Model for Direct Volume Rendering

Veronika Šoltészová[1] Daniel Patel[2] Stefan Bruckner[3] Ivan Viola[1]

[1]University of Bergen, Norway
[2]Christian Michelsen Research, Norway
[3]Simon Fraser University, Canada

Abstract

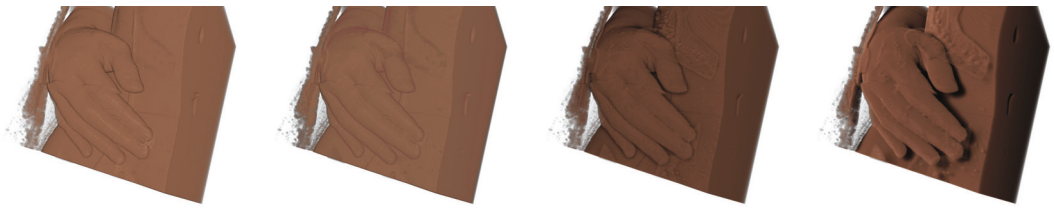
In this paper, we present a novel technique which simulates directional light scattering for more realistic interactive visualization of volume data. Our method extends the recent directional occlusion shading model by enabling light source positioning with practically no performance penalty. Light transport is approximated using a tilted cone-shaped function which leaves elliptic footprints in the opacity buffer during slice-based volume rendering. We perform an incremental blurring operation on the opacity buffer for each slice in front-to-back order. This buffer is then used to define the degree of occlusion for the subsequent slice. Our method is capable of generating high-quality soft shadowing effects, allows interactive modification of all illumination and rendering parameters, and requires no pre-computation.

Categories and Subject Descriptors (according to ACM CCS):
I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism —Color, shading, shadowing, and texture

1. Introduction

Local illumination models, such as the Phong model, are suited for conveying shape cues for well-defined structures in volume data. However, they are generally not suitable for visualization when the main goal is to emphasize three-dimensional structural arrangements. In such a scenario, it is important to convey information about relative positions and distances between individual features. The human visual system is adapted to inferring three-dimensional information from illumination. Soft shadows, in particular, are effective monocular depth cues. Not only do they provide occlusion information, but the size and shape of the penumbra can be used to judge distances. Global illumination models provide these cues at high computational costs, especially for volume rendering. Visualization research has therefore focused on the development of new global illumination approximations for volume data that limit the complexity and allow for real-time image synthesis. For this purpose, precomputation or parameter constraint strategies are frequently employed. Both suffer from limited flexibility which can be problematic when interactive exploration is required. The directional occlusion shading model introduced by Schott et al. [SPH*09] is a forward scattering approximation based on a conical phase function. While the method is capable of generating realistic illumination at interactive frame rates, it requires that the view and the light directions have to coincide. In this paper, we introduce a multidirectional occlusion model, which removes this constraint.

The importance of illumination in 3D object perception has been well-studied [BBC83,KMK94,BLK00]. To find out how to best improve volume rendering, we have been conducting studies with medical illustrators. During our demonstrations of state-of-the-art visualization techniques to experienced medical illustrators, their first critique point was the positioning of the light in the scene and the choice of non-standard colors. While visualization researchers often carelessly define the light vector parallel to the view vector, this is considered a novice mistake in the domain of illustration. The resulting image is *flat*, akin to photos taken with built-in front flash. To give depth to an image, as a rule, medical illustrators use illumination from the top left. To further op-



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

Figure 1: *Visualizations of a human hand using raycasting (a), and sliced-based volume rendering (b), both using the Phong illumination. Directional occlusion shading model with a headlamp illumination setup (c) and illumination from the top left (d). Illumination with the top-left light position causes that the fingers cast soft shadows on the body and evoke strong depth-perception cues.*

timize the appearance of the depicted structures, manual fine tuning is required.

The directional occlusion shading model for interactive direct volume rendering takes the advantage of a headlight setup for performance reasons: by placing the light source at the eye position, the samples required for illumination can be reused for compositing, allowing the method to perform both operations in a single pass for a view-aligned slice through the volume. Our approach uses elliptic occlusion footprints computed from the light position, instead of the symmetric spherical footprints which were used in the original paper. We achieve the same performance with the additional possibility to position the light source anywhere within the hemisphere defined by the view vector. An example of the significant improvement of depth perception compared to the previous method is shown in Figure 1. A visualization of a human hand rendered with different techniques is compared to the headlight and top-left shading. Both a professional illustrator and a user study confirmed our subjective assessment which favored the rendering result 1(d).

The remainder of this paper is structured as follows: In Section 2, we review related work. Our multidirectional occlusion model is derived in Section 3. Section 4 provides additional implementation details. Results are presented and discussed in Section 5. Conclusions are drawn in Section 6.

2. Related Work

Levoy [Lev87] proposed the use of gradients in volume rendering for evaluating a surface-based local illumination model. While this common approach is effective in accentuating material boundaries, it suffers from noise. In particular, gradient-based shading fails to provide useful results in nearly homogenous regions. Illumination models which exploit the volumetric nature of the data can therefore provide additional cues. Max [Max95] gives a comprehensive overview of different optical models for volume rendering.

Yagel et al. [YKZ91] employed recursive ray tracing which allows for effects such as specular reflection and shadows. Behrens and Ratering [BR98] added shadows to texture-based volume rendering by using an additional shadow volume. The model presented by Kniss et al. [KKH02, KPH*03] captures volumetric light attenuation effects including volumetric shadows, phase functions, forward scattering, and chromatic attenuation using half-angle slicing. Hadwiger et al. [HKSB06] presented a GPU-accelerated algorithm for computing deep shadow maps for volume rendering. Rezk-Salama [RS07] proposed a semi-interactive approach for GPU-based Monte Carlo volume raytracing.

Ambient occlusion as described by Zhukov et al. [ZIK98] inverts the principle of light-exposure of a point in space to obscurance by its close environment. Dachsbacher et al. [DSDD07] refer to obscurance as antiradiance. They treat visibility implicitly while propagating antiradiance as an additional quantity. The advantage of these approaches is that they are view-independent: for fixed geometry, occlusion information only has to be computed once and can then be applied efficiently during rendering, for example using texture mapping. Several fast techniques which utilize this concept have been presented [Bun05, SA07]. Knecht [Kne07] and Méndez-Feliu [MS09] provide comprehensive overviews of rendering techniques based on ambient occlusion and obscurances.

In the context of volume visualization, the radiance at a point is determined by shooting rays in all directions from the point and averaging its degree of occlusion by other parts of the volume. The result is an approximation of global diffuse illumination. It produces soft shadowing effects which give a good indication of spatial relationships. However, the

art [Ste03] introduced vicinity shading, a variation of ambient occlusion to enhance perception of volume data by darkening depressions and crevices. To reduce evaluation costs, occlusion calculations are reused. The approach of Ropinski et al. [RMSD*08] relied on local histogram clustering to precompute occlusion information for all possible transfer function settings. However, high frequency data, in particular the presence of noise, reduces the effectiveness of their clustering approach and can lead to artifacts. Additionally, their precomputation process is very time and memory consuming. Hernell et al. [HYL07] used a local approximation of ambient occlusion in volumes to limit computation times. In subsequent work [HLY08, HLY09], they utilized local piecewise integration to approximate global light propagation. This approach still requires ambience data for each voxel to be recomputed when changing the transfer function, but their method is able to run interactively by limiting the number of rays shot for evaluating the ambience and by subsampling the rays using adaptive compression. In recent work, Ropinski et al. [RDR10] described a volumetric lighting model which simulates scattering and shadowing. They use slice-based volume rendering from the view of the light source to calculate a light volume and raycasting to render the final image.

View-dependent approaches do not require extensive precomputation and therefore allow fully interactive transfer function modification. This is frequently achieved by limiting light evaluation from spherical local neighborhoods to conical neighborhoods. Desgranges et al. [DEP05] use incremental blurring to achieve shading effects without the use of a gradient. The approach by Bruckner and Gröller [BG07] employed non-photorealistic shadowing and emission effects for the purpose of illustration. Finally, as stated in the previous section, our method is an extension of the model by Schott [SPH*09].

3. Multidirectional Occlusion Shading

The **D**irectional **O**cclusion **S**hading model by **M**athias **S**chott et al. (MS-DOS) [SPH*09] describes an approximation of light scattering in particles of a volume. This simple method generates soft a shadow effect and hence provides important shape and depth-perception cues. Although the approximation of the light transfer delivers slightly different results compared to reference images from a raytracer, it provides visually compelling shading effects at interactive frame-rates and with no precomputation. However, the light transfer approximation in the MS-DOS model constrains the light direction to the viewing direction. In this section, we derive an approximation which does not limit the light to this fixed direction.

transport of light energy L in a medium. Every point in the environment receives a portion of energy, i.e., radiance composed by background radiance L_b and medium radiance L_m. The medium radiance consists of the emitted radiance L_e and in-scattered radiance L_i. The emitted radiance at a point $\mathbf{x}$ depends only on the local environment of $\mathbf{x}$. Unlike L_e, the in-scattered radiance L_i integrates over global features:

$$L_i(\mathbf{x},\omega) = \int_{4\pi} L(\mathbf{x},\omega_i)\Phi(\omega,\omega_i)d\omega_i \qquad (1)$$

where $\Phi(\omega,\omega_i)$ denotes the phase function for two light-ray directions ω and ω_i. L_i quantifies the total radiance incident to point $\mathbf{x}$ from all directions ω_i. From Equation 1, it can be seen that L_i requires an expensive recursive evaluation. The MS-DOS shading model and our model (multidirectional OS) simplify the evaluation which considerably reduces the computational costs.

We assume that the medium emits light only in directions within a specific cone. The phase function from Equation 1 can be therefore replaced by a simple cone-shaped phase function $\Phi_{\theta,\alpha}(\omega,\omega_i)$ where θ is the aperture angle and α the tilt angle of the cone. A schematic illustration of this scenario is depicted in Figure 2. A particle at a point $\mathbf{x}$ scatters light which is received by particles inside the cone. The in-scattering term L_i is conceptually related to the fractional visibility which is equivalent to the opacity and cumulates information about ambient occlusion.

Like the original model, we use a slice-based volume renderer with an additional opacity buffer. Slices are composited in the front-to-back order and the opacity buffer is incrementally filtered and used to determine the accumulated opacity for the next slice as shown in Figure 3. MS-DOS

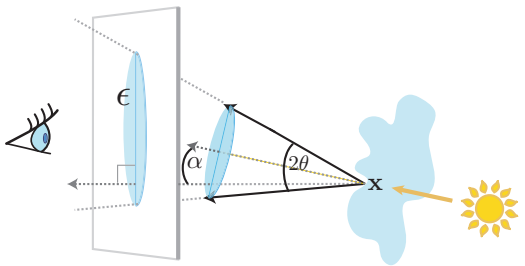


Figure 2: *Conical phase function setup: a selected point in space* $\mathbf{x}$ *scatters light which we approximate by a tilted cone* (α = *tilt*, θ = *aperture*). *The axis of the cone is parallel to the light direction. The projection of the light energy leaves an elliptical footprint* ε *on a selected viewing plane.*

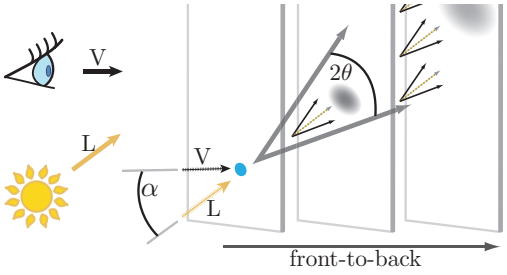


Figure 3: *Incremental blurring of the opacity buffer. We use a view-aligned slice stack composited in the front-to-back order.*

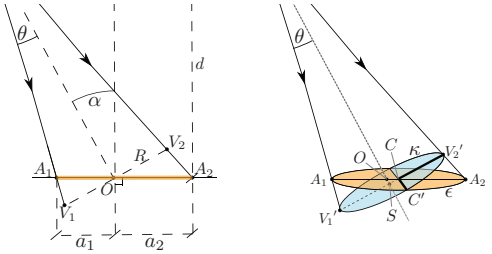


Figure 4: *A geometrical description of the cone-shaped phase function: the elliptical cone-section defines a circle* κ *centered in S and intersecting the center C of the ellipse* ε*. A side view (a) and a 3D view (b) of the planar cone-section.*

operates on view-aligned slices and assumes that the direction of the light source is aligned to the viewing direction. As a consequence, the opacity buffer can be convolved with a symmetrical disc-shaped Gaussian kernel. To enable interaction with the light source, we change the symmetrical disc-shaped kernel to an elliptical kernel. The ellipse ε is defined by the intersection of a tilted cone which represents the phase function $\Phi_{\theta,\alpha}(\omega,\omega_i)$ and the slice plane. The cone-shaped phase function is tilted by an angle α which is limited to $[0,\frac{\pi}{2}-\theta)$. This restricts the cone-section from degenerating into hyperbolas or parabolas. Figure 4 describes this geometrical situation.

3.2. Analysis of the Geometrical Model

This section describes the analytical computation of the elliptical kernel, namely, the derivation of its major and minor axes $A=|A_1A_2|$ and $B=|CC'|$ from a known tilt α, a cone aperture θ and a slice distance d. According to Figure 4(a), we derive R from d, θ and α as:

$$R = d\frac{\tan\theta}{\cos\alpha} \tag{2}$$

The axis of the cone intersects the plane at the point O. When the tilt angle $\alpha=0$, the cone section is a circle, and $a_1 = a_2 = A$. With a known R, we turn to the law of sine in the triangles $\triangle A_1V_1O$ and $\triangle OA_2V_2$. With α, θ, and R given, Equations 3 and 4 yield a_1 and a_2:

$$\frac{a_1}{\sin(\frac{\pi}{2}-\theta)} = \frac{R}{\sin(\frac{\pi}{2}+\theta-\alpha)} \tag{3}$$

$$\frac{a_2}{\sin(\frac{\pi}{2}+\theta)} = \frac{R}{\sin(\frac{\pi}{2}-\theta-\alpha)} \tag{4}$$

$$A = \frac{a_1+a_2}{2} \tag{5}$$

With known a_1 and a_2, we use Equation 5 to calculate A which is the major axis of the ellipse.

The center of the ellipse ε is in C with $\|OC\| = \|\frac{a_2-a_1}{2}\|$. We define a circular cone section κ which intersects the point C. Consequently, the axis of the cone intersects κ in its center S. This scenario is illustrated in Figure 4(b). The intersection line $\varepsilon\cap\kappa$ is perpendicular to A_1A_2 and intersects the center C of ε. Consequently, $\varepsilon\cap\kappa$ is collinear with the minor axis of ε. Figure 5 illustrates the side view of $\varepsilon\cap\kappa$. In Figure 6, we focus on the triangles $\triangle XV_2'V_2$ and $\triangle SCO$, and on the circle κ. Basic analysis implies Equations 6, 7, 8, and 9. Solving them, we determine B - the minor axis of ε.

$$\sin\alpha = \frac{d'}{\|OC\|} \tag{6}$$

$$\tan\theta = \frac{dR}{d'} \tag{7}$$

$$R' = R + dR \tag{8}$$

$$B = \sqrt{R'^2 - \|OC\|^2 + d'^2} \tag{9}$$

3.3. Weighting Function

Light rays collinear to the cone axis hit the slice with the highest intensity. We revisit Figure 4(b): O is the point with the highest incident energy. We define a weighting function as follows:

$$W_L(x,y) = 1-k \tag{10}$$

with k defined implicitly by:

$$\frac{(x-(1-k)\|OC\|)^2}{A^2} + \frac{y^2}{B^2} = k^2 \tag{11}$$

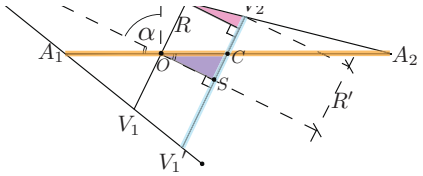


Figure 5: *A detailed side view of the intersection of ellipse* ε *and circle* κ.

Equation 11 results in a quadratic equation with two real roots from which we take the maximum. A kernel with a linear fall-off from O towards the border of the ellipse is illustrated in Figure 7(a). Additionally, we apply the Gaussian function to smooth the fall-off of the weights as illustrated in Figure 7(b).

3.4. On-the-fly Convolution

We apply an incremental convolution of the opacity buffer O_i and the elliptical kernel G_ε for each slice i. As the light direction $\mathbf{L}$ changes, G_ε has to be aligned respectively. We project the light vector to the viewing plane which yields a 2D-vector **L'** and rotate the kernel so that its major axis is aligned with **L'**:

$$\frac{\vec{OC}}{\|OC\|} = \frac{\mathbf{L'}}{\|\mathbf{L'}\|} \tag{12}$$

The weight-distribution in G_ε depends only on the tilt, aperture, light direction, and slice distance. Therefore, an update is triggered only if one of these parameters changes. In practice, we render the kernel G_ε to a texture when an update is triggered. First, we uniformly scale the ellipse so that it fits into a unit-square. Second, we set-up the texture coordinates so that G_ε is aligned correctly. During the volume rendering pass, we apply inverse scaling operation to regenerate G_ε of the correct size. In Figure 8, we visualize the gecko dataset with different tilts and apertures.

Based on the incremental convolution ($*$), we calculate a modulation factor λ_i for each sample on the slice i which determines the visibility of the current slice:

$$\lambda_i = \frac{1}{1 + G_\varepsilon * O_{i-1}} \tag{13}$$

In addition to the opacity buffer O_i, we use a color buffer C_i for each slice. The opacity buffer for the next slice combines the opacity buffer of the previous slice with the opacity of the current slice α_i:

$$O_i = G_\varepsilon * O_{i-1} + \alpha_i \tag{14}$$

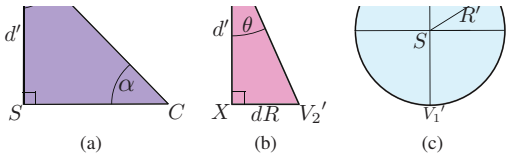


Figure 6: *We introduce new literals for selected primitives: (a) the triangle* $\triangle$ *SCO, (b) the triangle* $\triangle$ $XV_2'V_2$ *and (b) the circle* κ. *These primitves are defined in Figures 4 and 5 by the same color encoding. Note, that* $||CC'|| = B$ *which is the minor axis of the ellipse* ε.

The color contribution c_i is multiplied by λ_i. The color c_i and opacity α_i propagate to C_{i+1} using traditional alpha-blending with the over operator. Our method requires no pre-computation and performs at interactive frame-rates. Due to incremental blurring of the opacity buffer, shadows cast by highly occlusive regions fade-out smoothly with distance. Compared to the state-of-the-art model, we thereby add a movable light source with negligible performance penalty.

4. Implementation Details

Our new model was implemented as a plugin to VolumeShop [BG05] using C++ and OpenGL/GLSL. Using the ARB_draw_buffers OpenGL extension, two render targets are written for each slice: the intermediate rendered image and the occlusion buffer. The elliptical blurring kernel is stored in an additional texture which is updated whenever the light source parameters change. For all examples in the paper, we use a texture size of 128×128. When the lighting parameters change, we recompute the footprint. The major axis of the ellipse is aligned with the projection of the light vector to the viewing plane by multiplying GL_TEXTURE matrix stack by a rotation matrix. In case the ellipse grows or moves out of the texture, we apply translation and scaling to

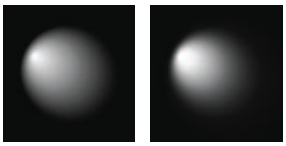


Figure 7: *Elliptical kernels used for incremental blurring of the opacity buffer: with linear fall-off (a) and Gaussian fall-off of the weighting function (b).*

(a) (b) (c)

Figure 8: *Visualizations of the gecko CT dataset with different setup of aperture* θ *and tilt angle* α*:* $\theta = 10^\circ$ *and* $\alpha = 37^\circ$ *(a),* $\theta = 40^\circ$ *and* $\alpha = 37^\circ$ *(b), and* $\theta = 40^\circ$ *and* $\alpha = 5^\circ$ *(c). Light is coming from the right for all images.*

Figure 9: *Visualizations of the carp CT dataset using the directional occlusion shading model with a headlamp illumination setup (a) and using illumination setup conventional to medical illustrations (b).*

fit it into the bounding box of the texture. During rendering, the inverse transformation is applied to access the kernel at correct positions. However, massive downscaling of the coordinate system may lead to a loss of precision. Users can interactively adjust the tilt, the aperture, and the XY-rotation of the light source. This gives the user full control to set the light source arbitrarily in the hemisphere defined by the view vector. The parameters aperture, tilt, and rotation are set by sliders in the user-interface.

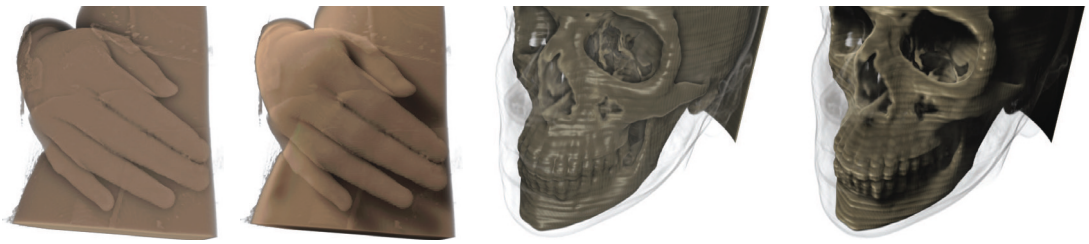
5. Results and Discussion

In this section, we provide case studies and comparisons to other volume rendering approaches and analyze the performance of our new method.

5.1. Case Studies

Medical illustrators generally place the light source in the top left corner to improve depth perception. Figure 9 depicts the carp CT dataset visualized under different illumination conditions. While in Figure 9(a), the image lacks depth, Figure 9(b) emphasizes the detailed structure of the skeleton through shadows. Similarly, Figure 10 shows cases where illumination leads to better perception of structures. In Figure 10(a), the hand seems to directly contact the body. In reality, there is a small gap which is visible in Figure 10(b). Similarly for Figures 10(c) and 10(d): in Figure 10(d), the eye sockets of the skull appear deeper than in Figure 10(c). We consulted a certified medical illustrator with over 25 years of professional experience who affirmed that the visualizations generated using this kind of illumination yield stronger perception cues. We presented her visualizations using different lighting settings. Her task was to choose which lighting conditions suit medical illustrations the best. She consistently preferred image such as those depicted in Figures 10(b) and 10(d). The illustrator further confirmed that interactive fine-tuning of the exact light placement is necessary in many cases, in order to avoid excessive darkening of focus objects. In volume data, regions with high gradient magnitude correspond to surface-like structures. Using the gradient magnitude to add an additional local specular component to these objects can further improve perception. Figure 11 presents a computer tomography of a human foot generated with different illumination models and varying light source positions: Figures 11(a) and 11(b) use the multidirectional OS model enhanced by specular highlights, and Figures 11(c) and 11(d) use the pure multidirectional OS model.

To gain a coarse impression on the impact of our technique on non-professionals, we also conducted a small user study on a group of 42 participants with different backgrounds. We presented them two series of result images: the human hand and the human thorax which are shown in Figures 1 and 12. Their task was to choose an image which in their opinion yields the strongest depth cues. From the series of different renderings of the hand, 39 participants (92.86%) favored the top-left illumination in Figure 1(d), 2 participants (4.76%) preferred the raycasting in Figure 1(a) and 1 participant (2.38%) preferred the headlamp illumination in Figure 1(c). A majority of 41 (97.62%) participants also pre-



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

Figure 10: *Visualizations of computer tomography data using the directional occlusion shading model with a headlamp illumination setup (a) and (c) using the illumination setup conventional to medical illustrations (b) and (d).*

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

Figure 11: *Visualizations of a human foot acquired by computer tomography using the directional occlusion shading model: using the Phong illumination model with the headlamp illumination setup (a) and with the top-left lighting (b). Visualizations (c) and (d) use the diffuse illumination model with the headlamp and the top-left light source setup respectively.*

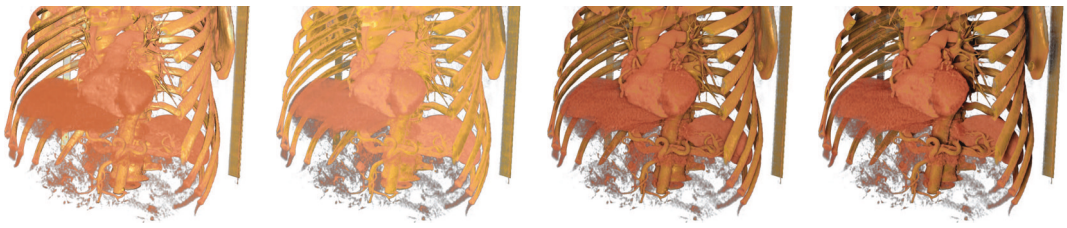
ferred the top-left illumination of the thorax in Figure 12(d) and only one participant (2.38%) selected the raycasted image in Figure 12(a).

Local surface-based illumination of volume data employs the gradient to substitute for the surface normal. However, gradients estimation frequently performs poor in the presence of noise which can lead to distracting artifacts. Thus, for modalities such as ultrasound, unshaded volume rendering is commonly employed. This makes the structures in the data difficult to interpret even for experienced medical professionals. The directional occlusion shading model as a gradient-free shading method can be used to improve perception. Interactive light source modification enables the user to inspect and understand the structures better. Figure 13 shows different visualizations of 3D cardiac ultrasound: 2D slices and 3D volume renderings. The clipping plane reveals the inside of the heart chambers. During examination, physicians see the ultrasound visualizations on their workstations as in Figure 13(a). We used a transfer function which shows the heart in a similar fashion. Figure 13(b) shows that gradient-based shading is not well-suited for ultrasound data. Multidirectional occlusion shading, on the other hand, reveals the structure, and interaction with light source enables the user to better perceive the depth of the cavities.

We described a shading model which does not require precomputation and storage of additional data, unlike deep shadow maps [HKSB06] or light volumes [RDR10], and which allows arbitrary light position within the hemisphere defined by the view vector. Half-angle slicing, introduced in the work of Kniss et al. [KKH02], generates shadows by using a slicing direction halfway between view and light direction. However, choosing a slicing direction which is non-parallel to the viewing directions leads to visible artifacts, especially when the light source tilt angle surpasses 60°. Figure 14 clearly demonstrates a situation when such artifacts are visible when half-angle slicing is used. In the original half-angle slicing approach, the order of the slices is reverted if the light source is located in the hemisphere opposite to the viewer. Reverting the traversal of the slice-stack is a possible extension of our approach which would not limit the light vector to the hemisphere defined by the view vector.

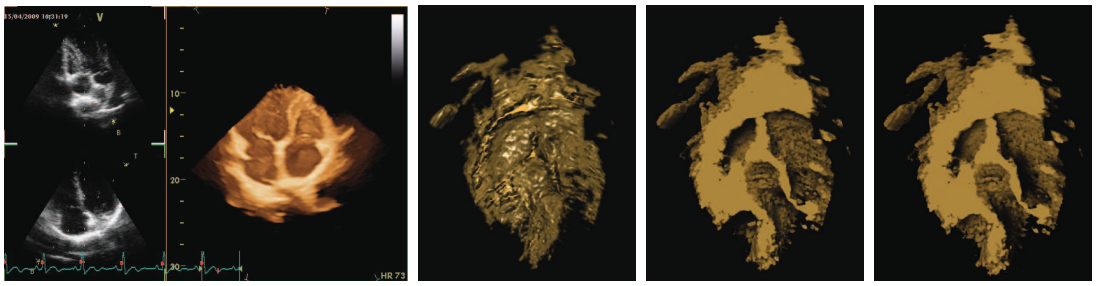
5.2. Performance Analysis

We tested the described method on a workstation equipped with an NVIDIA GeForce 295 GTX GPU with 1.7GB graph-



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

Figure 12: *Visualizations of a human thorax we used for user study: using raycasting (a), sliced-based volume rendering (b), both using the Phong illumination followed by the directional occlusion shading model with the headlamp illumination setup (c) and illumination from the top left (d).*



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

Figure 13: *Visualizations of 3D ultrasound of cardiac data: user interface of a 3D cardiac ultrasound workstation (a), a clipped 3D cardiac ultrasound visualization using direct volume rendering and Phong illumination model, rendered with a raycaster (b), clipped 3D cardiac ultrasound visualization using the multidirectional occlusion shading model with light coming from the top left (c) and the bottom left (d).*

(a) (b)

Figure 14: *Visualizations of the bonsai dataset: slice-based volume rendering using a view-aligned slice stack (a) and using a half-angle-aligned slice stack (b).*

ics memory, an Intel®Core i7 CPU with 3.07GHz and 12GB of RAM. We measured the performance of our implementation using the gecko dataset of resolution $512 \times 512 \times 88$ voxels, 0.5 voxels sampling distance and viewport-resolution 768×407 pixels. We achieved interactive frame-rates of 19Hz with using the MS-DOS and 18Hz with multidirectional OS using a 37° angle of aperture while interacting with the viewing parameters. During interaction with the light source, which required update of the kernel, we achieved 14Hz frame-rates. For comparison, using the same framework, a simple slice-based renderer with no shadowing and Phong illumination achieved 25Hz and a high-quality raycaster with Phong illumination and no shadowing achieved 21Hz. We performed the same test with the foot dataset of resolution $256 \times 256 \times 256$ voxels, 0.5 voxels sampling distance and viewport-resolution 531×311 pixels. We achieved 15Hz while using the original MS-DOS approach, 14Hz using our new method, and 12Hz during light source interaction. In this case, a simple slice-based renderer performed at 25Hz and a raycaster at 22Hz. These tests prove that the interactive light source placement is a valuable extension of the original approach traded for a negligible performance penalty.

high-quality soft shadow effects without the need for precomputation. Our method extends a previous technique to enable interactive placement of the light source. Using elliptic instead of circular footprints, we achieve almost the same performance while greatly improving the flexibility of the method. Additionally, we discussed several applications of such a shading model and consulted a professional illustrator to confirm the importance of freely modifying the light direction.

7. Acknowledgments

This work was carried out within the IllustraSound research project (# 193180), which is funded by the VERDIKT program of the Norwegian Research Council with support of the MedViz network in Bergen, Norway. The authors wish to thank the certified medical illustrator Kari Toverud for consulting, numerous respondents for their feedback and anonymous reviewers for their comments.

References

[BBC83] BERBAUM K., BEVER T., CHUNG C. S.: Light source position in the perception of object shape. *Perception 12*, 5 (1983), 411–416. 1

[BG05] BRUCKNER S., GRÖLLER M. E.: VolumeShop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization* (2005), pp. 671–678. 5

[BG07] BRUCKNER S., GRÖLLER M. E.: Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1344–1351. 3

[BLK00] BRAJE W., LEGGE G., KERSTEN D.: Invariant recognition of natural objects in the presence of shadows. *Perception 29*, 4 (2000), 383 Ű 398. 1

[BR98] BEHRENS U., RATERING R.: Adding shadows to a texture-based volume renderer. In *Proceedings of IEEE Symposium on Volume Visualization 1998* (1998), pp. 39–46. 2

[Bun05] BUNNEL M.: *GPU-Gems*, vol. 2. Addison Wesley, 2005, ch. Dynamic Ambient Occlusion and Indirect Lighting. 2

[DEP05] DESGRANGES P., ENGEL K., PALADINI G.: Gradient-free shading: A new method for realistic interactive volume rendering. In *Proceedings of Vision, Modeling, and Visualization 2005* (2005), pp. 209–216. 3

[DSDD07] DACHSBACHER C., STAMMINGER M., DRETTAKIS G., DURAND F.: Implicit visibility and antiradiance for interactive global illumination. *ACM Transactions on Graphics 26*, 3 (2007), 61.1–61.10. 2

[HKSB06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K.: GPU-accelerated deep shadow maps for direct volume rendering. In *Proceedings of SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware* (2006), pp. 49–52. 2, 7

[HLY08] HERNELL F., LJUNG P., YNNERMAN A.: Interactive global light propagation in direct volume rendering using local piecewise integration. In *Proceedings of IEEE/EG International Symposium on Volume and Point-Based Graphics* (2008), pp. 105–112. 3

[HYL07] HERNELL F., YNNERMAN A., LJUNG P.: Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *Proceedings of Volume Graphics 2007* (2007), pp. 1–8. 3

[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. 2, 7

[KMK94] KERSTEN D., MAMASSIAN P., KNILL D.: *Moving cast shadows and the perception of relative depth*. Tech. Rep. 6, Max-Planck-Institut für biologische Kybernetik, Tübingen, Germany, 1994. 1

[Kne07] KNECHT M.: *State of the art report on ambient occlusion*. Tech. rep., Technische Universität Wien, Vienna, Austria, 2007. 2

[KPH*03] KNISS J., PREMOZE S., HANSEN C., SHIRLEY P., MCPHERSON A.: A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (2003), 150–162. 2

[Lev87] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8* (1987), 29–37. 2

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108. 2

[MS09] MÉNDEZ-FELIU À., SBERT M.: From obscurances to ambient occlusion: A survey. *"The Visual Computer" 25*, 2 (2009), 181–196. 2

[RDR10] ROPINSKI T., DÖRING C., REZK-SALAMA C.: Interactive volumetric lighting simulating scattering and shadowing. In *Proceedings of IEEE Pacific Visualization* (2010), pp. 169–176. 3, 7

[RMSD*08] ROPINSKI T., MEYER-SPRADOW J., DIEPENBROCK S., MENSMANN J., HINRICHS K. H.: Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum 27*, 2 (2008), 567–576. 3

[RS07] REZK-SALAMA C.: GPU-based Monte-Carlo volume raycasting. In *Proceedings of Pacific Graphics 2007* (2007), pp. 411–414. 2

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings of Symposium on Interactive 3D Graphics and Games* (2007), pp. 73–80. 2

[SPH*09] SCHOTT M., PEGORARO V., HANSEN C., BOULANGER K., STRATTON J., BOUATOUCH K.: A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum 28*, 3 (June 2009), 855–862. 1, 3

[Ste03] STEWART A. J.: Vicinity shading for enhanced perception of volumetric data. In *Proceedings of IEEE Visualization 2003* (2003), pp. 355–362. 3

[YKZ91] YAGEL R., KAUFMAN A., ZHANG Q.: Realistic volume imaging. In *Proceedings of IEEE Visualization 1991* (1991), pp. 226–231. 2

[ZIK98] ZHUKOV S., IONES A., KRONIN G.: An ambient light illumination model. In *Rendering Techniques* (1998), pp. 45–56. 2

7 Seismic Volume Visualization for Horizon Extraction

Seismic horizons indicate change in rock properties and are central in geoscience interpretation. Traditional interpretation systems involve time consuming and repetitive manual volumetric seeding for horizon growing. We present a novel system for rapidly interpreting and visualizing seismic volumetric data. First we extract horizon surface-parts by preprocessing the seismic data. Then during interaction the user can assemble in realtime the horizon parts into horizons. Traditional interpretation systems use gradient-based illumination models in the rendering of the seismic volume and polygon rendering of horizon surfaces. We employ realtime gradient-free forward-scattering in the rendering of seismic volumes yielding results similar to high-quality global illumination. We use an implicit surface representation of horizons allowing for a seamless integration of horizon rendering and volume rendering. We present a collection of novel techniques constituting an interpretation and visualization system highly tailored to seismic data interpretation.

The following paper appears in its original format published as:

D. Patel, S. Bruckner, I. Viola, and M. E. Gröller. Seismic volume visualization for horizon extraction. In *Proceedings of IEEE Pacific Visualization 2010*, pages 73–80, 2010.

Seismic Volume Visualization for Horizon Extraction

Daniel Patel*
Christian Michelsen Research, Bergen, Norway

Stefan Bruckner†
Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

Ivan Viola ‡
Department of Informatics, University of Bergen, Norway

Eduard M. Gröller §
Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

ABSTRACT

Seismic horizons indicate change in rock properties and are central in geoscience interpretation. Traditional interpretation systems involve time consuming and repetitive manual volumetric seeding for horizon growing. We present a novel system for rapidly interpreting and visualizing seismic volumetric data. First we extract horizon surface-parts by preprocessing the seismic data. Then during interaction the user can assemble in realtime the horizon parts into horizons. Traditional interpretation systems use gradient-based illumination models in the rendering of the seismic volume and polygon rendering of horizon surfaces. We employ realtime gradient-free forward-scattering in the rendering of seismic volumes yielding results similar to high-quality global illumination. We use an implicit surface representation of horizons allowing for a seamless integration of horizon rendering and volume rendering. We present a collection of novel techniques constituting an interpretation and visualization system highly tailored to seismic data interpretation.

Keywords: Seismic interpretation, Seismic horizons, Volume rendering, Ambient occlusion, GPU acceleration.

1 INTRODUCTION

Energy is one of the most important resources in today's societies. Over half of the world-energy needs are covered by oil and gas [13]. This results in high demands for hydrocarbon resources and makes their identification and extraction economically valuable. To identify hydrocarbon reservoirs, subsurface volumetric data is obtained by sending sound waves into the ground and receiving and processing their echoes. The resulting seismic reflection volume then goes through a complex and time consuming manual interpretation for identifying subsurface structures which may hint to where hydrocarbons are trapped. In this paper we present a system for rapid interpretation of seismic reflection volumes.

The earth subsurface consists of material layers with distinct mineral densities and porosity characteristics. The interfaces between material layers are called horizons. They are indicated as high and low valued bands in seismic reflection data and are central structures for interpretation. Other structures such as faults, channels, salt bodies, and gas chimneys are mainly identifiable by their interaction with horizons. Faults are generally sub-vertical fractures which have shifted horizons on either sides; they are thus identified as discontinuities in horizons. Salt bodies are homogeneous units of salt. Due to the high seismic wave velocity of homogeneous salt, such structures can have strong reflections at their boundaries and low or no internal reflections. Areas void of horizons can thus indicate the presence of salt bodies having the property of shadowing the underlying seismic data. Subsurface leakages of gas, called gas chimneys, can be indicated by the up-bulging of horizons around them and the fragmentation of the horizons in their path. A reservoir in a porous rock formation can be accurately pinpointed by identifying its upper and lower boundary. The different rock materials at the boundaries give rise to horizons in the reflection data. Therefore horizons can be central in delineating reservoirs.

In addition to these descriptive properties of horizons, they are perhaps the most easily identifiable objects in the seismic volume and thus among the most important objects during interpretation. Compared to most other seismic structures, horizons can directly be identified with image processing techniques such as ridge and valley detection. For these reasons we focus on horizon identification in this work. We present an approach for automatically identifying horizons and visualizing them.

Visualizing 3D seismic data is challenging due to the dense nature of the data, the high presence of noise, and the difficulty to identify, manipulate and select structures in 3D. Consequently, interpretation of 3D seismic data is nowadays split into lengthy 2D interpretations of a stack of selected cross-sections throughout the 3D volume. In this paper we present a novel approach that facilitates rapid 3D interpretation using interactive computer-assisted 3D visualization. The basic idea is to precompute horizon candidates from the dataset and partition them into small building blocks. The building blocks are then selected by the geoscientist during a rapid interpretation process and assembled together into correct horizons.

As opposed to the accurate but slow interpretation process currently used, ours is quick but less accurate. However the interpreter is aware of the reduced accuracy since the seismic ground-truth data is provided for context. Our approach creates early overviews for identifying areas to later perform accurate interpretation in, instead of performing accurate and time consuming interpretation of all the data. The currently used interpretation workflows do not support attaining overviews. We propose rapid sketching of 3D interpretations. This is similar to 3D sketch systems [10] in architectural CAD systems which have become superior during idea prototyping over traditional 2D floor-based modeling software [2]. For subsequent detailed modifications, interpreted horizons can be imported into existing interpretation systems. In such detail-oriented systems, tasks like identifying horizon-splitting structures such as faults and unconformities can be performed.

A major challenge for providing useful 3D interactive visualization, is the choice of an appropriate 3D rendering algorithm. Gradient-based shading [17] is effective for depicting volumetric data having clear material boundaries. The gradient vectors are then used as normals in the Phong illumination model [24]. Gradient-based methods, whether based on central differences or more complex seismic dip and azimuth estimations [19], are in general sensitive to high-frequency noise. As seismic acquisition is based on acoustics, data are typically of noisy nature and lack distinct material boundaries. Gradient-based shading on seismic data introduces distracting artifacts which makes interpreting 3D seismic data renderings difficult. Other approaches, such as unshaded direct volume rendering tend to depict seismic data as a homogeneous cloud without distinct features [33]. Common approaches are frequently unsuitable for visualizing seismic data. In this paper we present a gradient-free ambient-occlusion like method for shaded volume rendering of seismic data that reduces the abovementioned problems.

*e-mail:daniel@cmr.no
†e-mail:bruckner@cg.tuwien.ac.at
‡e-mail:ivan.viola@uib.no
§e-mail:groeller@cg.tuwien.ac.at

Related work is reviewed in Section 2. The high-level concept is described in Section 3. Sections 4 and 5 describe the horizon extraction and visualization stage. A demonstration of the proposed technology is presented in Section 6 and conclusions are drawn in Section 7.

2 Related Work

Several research works and commercial solutions present interpretation and visualization algorithms for 3D seismic data. Pepper and Bejarano [23] gives an overview of computer-assisted interpretation algorithms. Commercial software used in oil companies include HydroVR [18] and Petrel [30]. Interpretation algorithms for horizons are typically semi-automatic and require a detailed and time consuming user involvement. User steered horizon growing is a standard method. The user manually places a seed point on a horizon in a seismic slice and adjusts growing parameters before starting a growing process of the horizon. This method is not fully interactive due to the need to set parameters and to wait for the growing to finish. To our knowledge there are no commercial solutions that support fully interactive and thereby rapid horizon interpretation of seismic data. For generality, the interpretation software often presents the user with many parameters to specify for the growing process. Typically the user changes parameters and re-grows until a satisfactory result is obtained. The parameters to be set are related to the internal growing/image processing algorithms and can be difficult to understand for a geologist performing interpretation. One might argue that too much low level control is given to the user. In our system we aim at minimizing the need for parameter tweaking.

Growing algorithms based on local waveform matching are common. The local waveform is defined as the vertical 1D signal of the seismic values in a neighborhood above and below a sample. Castanie et al. [7] propose user-specified seeding followed by growing in areas that have local waveforms similar to the seedpoint. The method builds on the fact that horizons typically have characteristic local waveforms. The method requires setting parameters such as the length of the neighborhood to match. Interpretation software [30] performs growing in voxels that have been thresholded or in zero crossings or extrema of the local waveform. We employ the latter method as it requires no parameters to be set.

There are completely automatic approaches for horizon interpretation [4]. In their work, voxels are mapped to points in an n-dimensional space based on their local waveforms. Clusters are then identified in the n-dimensional point cloud. Probably due to the low control and low predictability of the outcome and long cycles of setting parameters and waiting periods before the results are available, such methods have not gained popularity.

Automatic horizon interpretation as preprocessing before rapid interpretation of 2D seismic slices has been proposed by Patel et al. [21]. An extension of their technique into 3D would be difficult as their horizon analysis produces just a set of line segments for each 2D seismic slice. A 3D horizon preprocessing method was presented in the work by Faraklioti and Petrou [9]. However their connected component analysis was only able to grow planar and well-defined horizons.

In our work we present the concept of rapid horizon interpretation by focusing on the analysis of seismic horizons, high quality 3D visualization and quick interaction techniques. We automatically extract surfaces that, with high likelihood, coincide with horizons. We subdivide these surfaces into smaller surfaces using an existing mesh clustering algorithm [1]. An overview of mesh clustering algorithms can be found in the work by Shamir [32]. Shamir categorizes clustering methods into greedy quick methods, global slow methods, growing methods, hierarchical methods and spectral analysis methods. We chose a hierarchical greedy method due to the generation of hierarchy information which we use during interpretation. As opposed to growing methods the approach does not require initial seed faces as input that affect the resulting subdivision and is therefore deterministic.

For a seamless integration of horizon-surface visualization with seismic-volume visualization, we represent the surfaces as a distance volume having segmentation masks around each surface-part. Existing distance-transform techniques have been surveyed by Jones et al. [15]. Since we perform the distance transform in a preprocessing step, we are not dependent on speed. We create a computationally expensive but analytically accurate distance transform by in essence considering the analytical distance to all triangles from each voxel [14].

The basic concept of our approach is to carry out horizon interpretation directly in 3D. Several aspects for 3D visualization of seismic volumes have been investigated in earlier works. Plate et al. [26, 25] and Castanie et al. [7] discuss handling multiple large seismic volumes in the rendering pipeline. Ropinski et al. [29] discuss volume rendering of seismic data in VR with volumetric cutouts. Illustrative rendering techniques employing textures have been proposed for presentation of interpreted seismic data [22].

Often gradient-based illumination is used to render volumes and calculate the gradient directly from the seismic data. One exception is the work by Silva et al. [33]. They observe that seismic horizons are not isosurfaces of the volumetric data, therefore seismic gradients are not optimal to use. Instead they calculate gradients from a derived phase volume which gives better results. However, gradient-based illumination models lack depth cues and display a strong visual dominance of noise making it difficult to identify subtle horizon structures. For these reasons the rendering approaches currently used for seismic data are not ideal for horizon display.

Depth cues can be added to volume rendered images by employing more realistic illumination models. Yagel et al. [35] employ recursive ray tracing which allows effects such as specular reflection and shadows. Behrens and Ratering [3] add shadows to texture-based volume rendering. The model presented by Kniss et al. [16] captures volumetric light attenuation effects including volumetric shadows, phase functions, forward scattering, and chromatic attenuation. Rezk-Salama [27] presents an approach for GPU-based Monte Carlo raytracing. Max [20] gives a comprehensive overview of different optical models for volume rendering. A problem of increasing the physical realism is, however, lack of speed and lack of control over the specific appearance of certain structures of interest. As they are based on actual physical laws, it is difficult to control individual visualization properties separately.

Other approaches employ quicker, visually plausible approximations of realistic lighting effects. Stewart [34] introduces vicinity shading, a view-independent model to enhance perception of volume data. It is based on occlusions in the local vicinity of a sample point resulting in shadows in depressions and crevices. Similarly, Hernell et al. [12] use a local approximation of ambient occlusion. Real-time global illumination for volumes has been done by Hernell et al. [11] by local piecewise integration and subsampling using adaptive compression. Desgranges et al. [8] use incremental blurring to achieve shading effects without the use of a gradient. The approach by Bruckner and Gröller [6] is able to generate shadowing and emission effects in a view-dependent manner while still allowing interactive transfer-function modifications. We use an interactive gradient-free illumination model inspired by Schott et al. [31] to visually reduce the noise and to provide depth cues.

3 Rapid Horizon-Interpretation Pipeline

An overview of the pipeline of this paper is depicted in Figure 1. The pipeline is divided into two parts. The first part is a preprocessing step that extracts horizon candidates in advance of the visualization so the user can perform rapid interpretation. This is covered in Section 4. The second part, the realtime use of the system, is discussed in Section 5 and covers visualization of volumes (5.1) and horizons (5.2) and user interaction for interpretation (5.3).

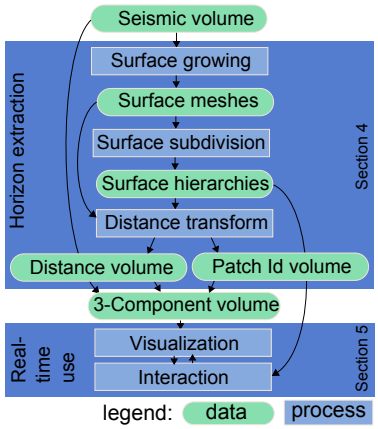


Figure 1: An overview of our pipeline. All preprocessing is performed in the horizon-extraction phase, followed by visualization and interaction for assembling correct horizons. The horizon extraction is further described in Figure 2 and Figure 4.

4 Horizon Extraction and Representation

Performing automatic feature extraction on seismic data is difficult due to the degree of uncertainty and noise in seismic data. Even domain experts may come up with different interpretations which demonstrates the under-defined nature of seismic data.

We design our horizon extraction with this uncertainty in mind by providing the user with a large collection of precalculated horizon surfaces. A set of plausible horizon surface-parts based on a horizon-growing and a surface-subdivision algorithm is precalculated and presented to the user to pick from. The user can then build horizons by assembling these surface-parts. The user is now freed from being repeatedly interrupted during interpretation by having to explicitly seed horizons followed by waiting for the horizons to grow. Explicit methods have advantages when it comes to fine-tune the horizons. However we focus on rapid interpretation for getting a first overview of the data. This overview can complement existing workflows. Later, a detailed interpretation can be focused on areas that have been identified as important with our method.

We enable the user to choose which level of horizon granularity she wishes to operate on. For each picked horizon surface she can change the granularity level. For this to work we first grow all possible horizons in the volume. Then we perform a hierarchical subdivision of the horizons into smaller surfaces. The hierarchy is stored in a tree structure with the smallest surfaces as leaf nodes. The user can navigate in the hierarchy and work with surfaces of the size and shape which reflect the interpretation best.

Horizon identification methods that do not rely on growing, such as performing clustering of the voxels based on their local neighborhood [4], result in segmentation masks that are not directly transferable to polygonal surfaces. Incremental growing methods are better suited as they can be made topology aware. This allows for constrained growing and for producing connectivity information during the polygonization of the horizons. We seed and grow each voxel in the volume. Growing is only performed in voxels that are on positive extrema of the vertical waveform, i.e. voxels that have a lower valued voxel directly above and below it. This gives a robust tracing of horizons of varying intensities. However other growing criteria can easily be used instead. We achieve subvoxel precision on the horizon growing by fitting a higher order polynomial to the trace signal and using the exact position of the analytical maximum. This avoids staircase artifacts aligned with the voxels.

Voxels are visited only once to avoid intersecting and overlapping surfaces. An example of growing from one voxel is shown in Figure 2. The width of each growing frontier, defined as the number of neighboring points it consists of, must be above a certain size or else the growing of that border is terminated. The larger the minimum width is, the less surface bifurcations are created. For instance, a minimum width of 1 will create surfaces that branch out in lines which in turn may become surfaces. The width parameter heuristically controls the minimum radius a surface can have. We achieved a good balance between undetected horizons and oversegmentation by using a width of four voxels. To create surfaces, we triangulate the frontier during growing. We achieve this by triangulating the strips consisting of the points in the current borders (green points in Figure 2) and the points in the borders from the previous step (red points in Figure 2). The grown surface is represented as a vertex list of the coordinates of the grown voxels followed by an index list of triangle triples with indices to these vertices.

Frequently during surface growing, parts which an interpreter would consider as separate horizons, are grown into a single surface. This typically happens for horizons vertically close to each other. Due to noise or unconformities, a growing frontier might jump to the horizon directly above or below. It will then start growing on that horizon while other frontiers continue on the original horizon. Erroneous growings result in surfaces consisting of multiple merged horizons instead of isolated correct ones. An example of such an incorrect growing originating from one seed point and resulting in five overlapping horizons can be seen in Figure 3. To address overgrowing, we subdivide each grown surface into smaller pieces for the user to select. The user can in realtime pick such surfaces, subdivide them further if necessary and thereby assemble correct horizon surfaces.

For subdividing the triangulated surfaces into smaller pieces we are using the method described by Attene et al. [1]. This greedy method creates a hierarchical surface segmentation by starting with single-triangle surfaces and iteratively merging neighboring surfaces into larger ones by choosing the neighbors that maximize the flatness of the joined surface. This process continues until all surfaces are merged into one. See Figure 3 for an illustration of the subdivision process. The merging operations can be described as the interior nodes of a binary tree having single triangles as leaf nodes. We store this binary surface tree for each grown surface. The binary tree is used later during interpretation. It lets the user work with subsurfaces at a user-defined granularity level by moving up and down in the hierarchy. After selecting a leaf node, the user can navigate up in the tree and select surfaces of appropriate sizes for a quick horizon assembly.

The subdivisions of a binary surface tree are depicted in Figure 3. For each consecutive image, a jump of four levels up in the tree is performed. Distinct colors represent distinct nodes in the corresponding tree level. In the top image, the tree leaves are seen. Each color represents a single triangle of the surface. However, single triangles have approximately the size of a voxel and are prohibitively small to work with. Therefore we prune the tree so that during interaction, the child nodes are not single triangles, but groups of triangles of a predefined minimum size. We chose to prune the tree into having leaf nodes with a minimum of 500 triangles. Each leaf node consists of a unique leaf-id followed by a list of indices to the triangle triples of the surface as described earlier. Each interior node consists of pointers to two child nodes and the number of triangles in its subtree.

For an integrated representation of seismic data and surface trees, we perform a distance transform on all leaf surfaces which together constitute all grown surfaces. In each voxel of the seismic volume we calculate and store the distance to the closest point on a leaf surface together with the unique id of that leaf surface. Then, during picking, the picked voxel's id is a link to the leaf node in the surface

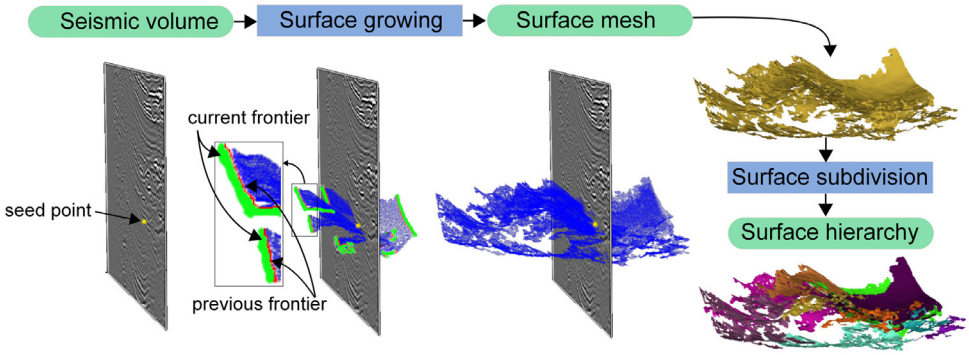


Figure 2: The process of growing from one specific seedpoint (yellow point). The green borders show the growing frontier and the red regions show the previous frontier of the blue triangulation. The triangulation is then subdivided.

tree of the closest surface. This link enables 3D picking of horizons followed by a hierarchical navigation in the surface tree from the leaf node to the largest valid horizon surface. We combine the seismic volume, the distance field volume and the leaf surface-id into one 3-component volume. This facilitates fast lookups during interaction. See Figure 4 bottom for an illustration of the distance component on a slice. The distance component combined with the leaf surface-id component allows for several advanced volume-rendering modes elaborated in Section 5.2.

Hierarchies with leaf surface-ids and geometric surface definitions associated with each leaf surface-id are stored on file. After the user has built horizons by selecting leaf surface-ids, the interpreted horizons can be geometrically reassembled and exported into commercial seismic interpretation systems for further processing.

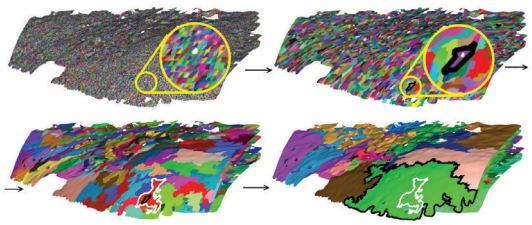


Figure 3: Increasing levels of a surface hierarchy made from one grown surface. The top-left image shows the initial one-triangle surfaces. Each successive image shows the resulting merging after four merge levels in the binary hierarchy tree. One triangle is magnified in white at the top image and its successive higher level surfaces are shown delineated with alternating white and black borders.

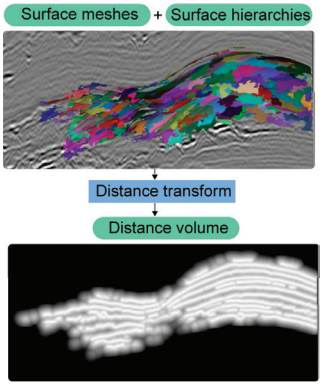


Figure 4: The distance transform. One grown and subdivided surface mesh is shown in 3D at the top intersecting a slice of the seismic data. The colors represent surfaces at the lowest level in the pruned hierarchy. The bottom image shows a slice through the resulting distance transform with white representing distance zero and darker colors representing higher distances.

5 Volume and Horizon Visualization with Interaction

During interactive interpretation, high quality volumetric rendering is performed, horizons are visualized and the user can interact with the data. These three topics are covered respectively in the three following subsections.

5.1 Gradient-Free Volume Visualization

Conventional surface-based shading using the gradient direction as a substitute for the surface normal is most effective when there are clearly identifiable material boundaries. An example is medical volume data acquired by computed tomography (CT). For seismic data, however, this approach commonly results in noisy images which are difficult to interpret. Unshaded visualization, on the other hand, suffers from a lack of clearly delineated features. In our approach we want to identify horizons and then visualize the extracted horizon information in conjunction with the original seismic data to enable verification of the extraction process. Thus, a volume visualization technique is required which gives a good indication of the overall structure of the data set.

A candidate to fulfill these requirements is ambient occlusion. The incoming radiance at a point is determined by shooting rays in all directions from the point for finding its degree of occlusion by other parts of the volume. The result is an approximation of global diffuse illumination. It produces soft shadowing effects which give a good indication of spatial relationships. In volume visualization the opacity at any point is determined by the transfer function. Ambient occlusion therefore requires an expensive computation step every time the transfer function is modified. The

approach of Ropinski et al. [28] relies on local histogram clustering to precompute this information for all possible transfer function settings. The precomputation is extremely time and memory consuming. In addition, the high frequency and noisy seismic data makes a bad candidate for local histogram clustering. We therefore choose to employ a view-dependent method which generates similar shadowing effects but still allows interactive transfer function manipulation.

Inspired by the approach of Bruckner et al. [6], Desgranges et al. [8] as well as recent work by Schott et al. [31], we use a slice-based volume rendering method where an incremental blurring operation is applied to the accumulated opacity buffer. This buffer is then used to determine the degree of shadowing applied to subsequent slices. The volume is traversed in front-to-back order using view-aligned slicing. Two buffers C_i, the color buffer, and O_i, the occlusion buffer, are written for each slice i. For each sample on the slice, we apply a Gaussian low-pass filter G to the occlusion buffer O_{i-1} of the previous slice and combine it with the opacity contribution α_i of the current slice using additive blending:

$$O_i = G * O_{i-1} + \alpha_i$$

Based on this incremental filter operation, for each sample on the slice, a modulation factor λ_i is computed which determines the degree of visibility:

$$\lambda_i = \frac{1}{1 + G * O_{i-1}}$$

The color contribution c_i is then multiplied by this factor and c_i and α_i are used for conventional alpha-blending using the over operator to generate the color buffer C_i.

This simple approach generates soft shadowing effects akin to ambient occlusion but can be performed interactively on current graphics hardware. Due to the incremental filtering of the occlusion buffer, darkening from a highly occlusive region will reduce smoothly with distance leading to a visual appearance as if the volume was illuminated by a large area light source located at the camera position. By setting ambient values in the occlusion buffer for certain objects in the volume one achieves the effect of having objects cast light instead of shadows. Examples of such light emitting objects will be shown in the subsequent section.

In Figure 5 we give a comparison between a Phong-shaded seismic volume and our rendering approach. As can be seen, our rendering is less susceptible to high frequency noise and gives a better depth perception of the data. Both images use the same transfer function for comparison reasons with a simple black to white color ramp. The lower values of the reflection data are set to transparent while the higher values are set to be opaque.

5.2 Horizon Visualization

Horizons are represented implicitly by the distance component and the leaf surface-id component. As opposed to a geometric model, this representation enables a seamless co-rendering of horizons and seismic data in one rendering pass. Implicitly representing the surfaces removes the need to store and update visualization parameters of the surface geometries in memory. It enables a single-pass integrated rendering, however with the penalty of an increased volume size. The distance volume is of the same resolution as the seismic volume. Due to trilinear interpolation, the implicit surfaces will therefore be smooth with an error distance from the original polygon of less than one voxel.

Having a volumetric segmentation mask with distance information for horizons opens up the possibility for more advanced rendering techniques. The picked leaf surface-ids and the distance volume constitute a segmentation mask around a horizon. This segmentation mask can be used to render horizon surfaces with a user defined appearance and thickness. A mapping from leaf surface-ids to RGBA and emission values defines the color, opacity and emission of each leaf-surface in the volume. During the slice-based volume rendering, the distance and leaf surface-id component of the pixels of each slice that is composited, is looked up in the volume. If the distance is less than the user defined horizon thickness, the pixel is modulated with the color and transparency assigned to the leaf surface-id. Initially all horizon leaf surface-ids are set to fully transparent. Thus the mapping defines the selection and appearance of horizons. Interpreted horizons are then easily visualized with different transparencies, colors and emission values.

Volume rendering of the seismic data only, can be restricted to take place in the vicinity of selected horizons and having full transparency everywhere else for verification reasons. Alternatively in an inverse mode, full transparency can be applied in the segmentation mask. This removes interpreted structures from the data for focusing on uninterpreted areas. Other seismic attributes from the same seismic study, such as fault probability, can be opaquely displayed on the horizons. This would express the fault probability along a horizon surface.

Advanced light effects can be achieved by extending the incremental filtering used for creating shadows during volume rendering. For instance, instead of casting shadows, we can let the horizons cast their colors, thereby simulating emissively glowing horizons. Emissiveness can be used to expressively highlight important horizons embedded in the seismic volume or to distinguish them from other horizons (Figure 7g-i). Compared to other focus+context techniques, our rendering method gives focus by using inherent and natural properties of the lighting model.

5.3 Interaction

We designed the system with the focus on simple interaction that supports easy and rapid interpretation of horizons. Intuitive interaction with the seismic volume is achieved by using the leaf surface-id component. This gives the ability to select leaf-surfaces by clicking directly on the volume rendered image. When a leaf-surface is selected, its surface hierarchy is available and can be navigated. This allows the user to iteratively select higher-level surfaces until the optimal size is found.

Volumetric horizon selection is achieved by first storing the 2D screen-space coordinate of the point the user selects. While performing front-to-back compositing of volume slices during volume rendering, the first hit with a nontransparent voxel is detected at the 2D coordinate on the compositing slice. The corresponding leaf surface-id is fetched from the leaf surface-id volume. This id identifies the selected leaf-surface and its color mapping is changed from full transparency to a color indicating it is selected.

Due to the dense nature of seismic data, we have defined a roaming box which confines the volume rendering. The user can easily move the roaming box through the volume and select horizons by clicking on the sides of the roaming box. Section 6 describes such an interaction scenario.

6 Results

In this section we present a scenario of navigating and interacting for rapid horizon interpretation. In the scenario, two horizons are interpreted by using surface picking and hierarchical compositing of surface-parts into larger surfaces.

We use a white single-colored transfer function with low to high values mapped from transparent to opaque. Dark areas are due to shadows cast by opaque horizons through (semi)transparent space onto other horizons. Typically transfer functions for seismic-volume rendering use a black to white or red-white-blue gradient to easier discern horizons. With shadowing the color gradient specification is not necessary. This simplifies the transfer function setup thus supporting rapid interpretation.

To identify a horizon for interpretation, the side of the roaming box is moved through the volume. An interesting horizon is

identified on the front-side of the roaming box (Figure 6a) and the horizon is selected using point-and-click (red arrow in Figure 6a). This results in a selected leaf-surface shown in pink. Instead of continuing to select small leaf-surfaces along the horizon, the hierarchy information is utilized. By using the keyboard up-arrow all surfaces in the hierarchy level above the current one are colored in pink. The user can navigate up to the highest hierarchy level that yields a correct horizon and use it as a starting point of the horizon assembly. Figures 6b-f show increasing surface-hierarchy levels. Going from the hierarchy level in Figure 6e one level up as shown in Figure 6f, erroneous horizons above the current one are included. The error is revealed by the irregular and overlapping structures that do not exist in horizons. Overlapping wrong horizons are easy to spot by our method due to the shadows they cast on horizons directly below them. Therefore the user goes back to the previous level shown in Figure 6e and uses this as a starting point. He has now selected the horizon at the highest level in the hierarchy without errors. This surface constitutes a horizon with holes. The holes might consist of unselected surface-parts. The user attempts to fill these holes by selecting potentially segmented surface-parts located in the holes. To select these surfaces, the roaming-box side is moved to intersect the holes (Figure 7a). The red arrow shows where the user selects the horizons. In Figure 7b the resulting selected surface-part is shown in green. The process of moving the roaming-box side over the horizon and filling out holes where possible is performed over the entire horizon. The result can be seen in Figure 7c with five green surface-parts representing filled-in holes indicated with yellow arrows. Now one horizon has been interpreted. The user selects another horizon color (yellow) and repeats the procedure (Figure 7d). Two interpreted horizons can be seen in Figure 7e. Different rendering types are shown in Figures 7f-k. In Figure 7f the top horizon is shown semitransparently so the underlying data can be seen. In Figure 7g the top horizon with an emissive factor is shown. Emissiveness can be used for bringing attention to a certain horizon. Figure 7h shows the bottom horizon emissively. In Figure 7i both horizons are shown emissively. On the side face of the roaming box one can see how the emissiveness interacts with the volume rendering by casting light from the emissive object onto its surroundings. In Figures 7j and k the horizons extruding from an opaque cube are shown using a black to white transfer function. Ambient occlusion-like shading of volumes and horizons gives these renderings a quality comparable to a manually shaded illustration.

In Figure 7k a scenario with seismic data around a hypothetical boring well and its relation to the two horizons is shown. The interpretation just presented could easily be performed in less than ten minutes. This short interpretation time underlines the potential of our approach.

Implementation Details

The software has been developed using the VolumeShop [5] framework. For preprocessing the dataset of size 256^3 used in this article, about 2000 surfaces were grown and subdivided, constituting altogether about 2 million triangles. The hierarchy for each surface is stored in memory in a downstripped version containing only leaf surface-ids in the leaf nodes without the list of triangle-indices defining the surface-part. The size is then less than 1MB.

With unoptimized preprocessing, it took 1 hour to calculate the distance transform using a maximal distance of 10 voxels. The unoptimized brute-force surface-growing from each voxel in the volume took five hours and it took one hour for the hierarchy creation of the grown surfaces. We ran the preprocessing on a 1.8 Ghz AMD Athlon 64, Dual core processor 3800. Preprocessing could be sped up by using existing optimized distance transform algorithms running on GPUs. Growing could be parallelized by reimplementation in CUDA. However, the preprocessing is very labor intensive and seismic volumes are increasing in size. We do not expect to be able to perform preprocessing on the order of minutes in the near future. A realistic scenario would be to settle for processing times taking less than one night, thus having the data ready the next day.

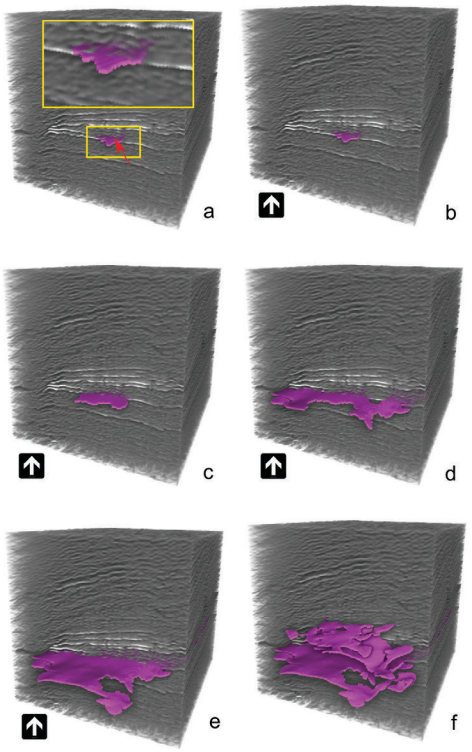


Figure 6: Selecting a leaf-surface (a) and navigating stepwise up in the hierarchy (b-f) until an erroneous surface is shown (f).

For rendering we used an Nvidia GeForce 275. The 3-component volume was represented with one byte for the seismic value, one byte for the distance and two bytes for the leaf surface-id taking 64MB, thus 4 times larger than the original seismic volume. During interaction we achieved 25 FPS in a 500 by 500 window. The high-resolution images of size 1000×1000 used in this article were taken with 8 samples per voxel resulting in a speed of 7-14 FPS.

7 Conclusion

We have presented a system for rapid interpretation and expressive visualization of seismic horizons by carefully combining appropriate technologies. Our main contributions are horizon growing with surface subdivision, implicit horizon representation enabling single-pass advanced rendering and using an illumination model supporting emission for clearer visualization of noisy acoustic seismic reflectance data.

8 Acknowledgements

Data used in this article is by courtesy of Norske ConocoPhillips AS, ExxonMobil Norway AS, Total E&P Norge AS, Statoil AS and Petoro AS. The work was performed partly under the Geoillustrator project sponsored by Statoil and Christian Michelsen Research.

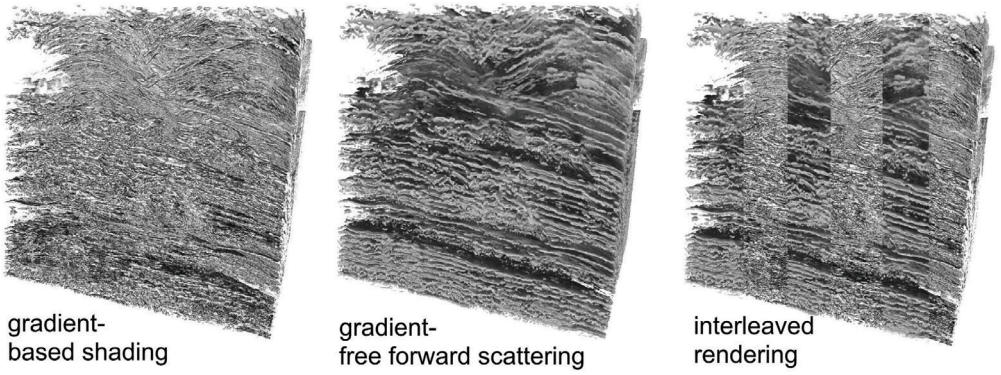


Figure 5: A comparison of gradient (left) and gradient-free shading (middle). In the right image, the two left images are interleaved and one can see how 3D and depth perception is improved and high frequency noise reduced in the gradient-free method.

REFERENCES

[1] M. Attene, M. Spagnuolo, and B. Falcidieno. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, March 2006.

[2] Autodesk AutoCAD, http://www.autodesk.com/, 2009.

[3] U. Behrens and R. Ratering. Adding shadows to a texture-based volume renderer. In *Proc. of IEEE Symposium on Volume Visualization 1998*, pages 39–46, 1998.

[4] H. Borgos, O. Gramstad, G. Dahl, P. L. Guern, and L. Sonneland. Extracting horizon patches and geo-bodies from 3d seismic waveform sequences. *SEG/New Orleans Annual Meeting*, pages 1595–99, 2006.

[5] S. Bruckner and M. E. Gröller. Volumeshop: An interactive system for direct volume illustration. In H. R. C. T. Silva, E. Gröller, editor, *IEEE Visualization 2005*, pages 671–678, Oct. 2005.

[6] S. Bruckner and M. E. Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1344–1351, 2007.

[7] L. Castanie, B. Levy, and F. Bosquet. Volumeexplorer: Roaming large volumes to couple visualization and data processing for oil and gas exploration. *Proc. of IEEE Visualization '05*, pages 247–254, 2005.

[8] P. Desgranges, K. Engel, and G. Paladini. Gradient-free shading: A new method for realistic interactive volume rendering. In *Proc. of Vision, Modeling, and Visualization 2005*, pages 209–216, 2005.

[9] M. Faraklioti and M. Petrou. Horizon picking in 3D seismic data volumes. *Machine Vision and Applications*, 15(4):216–219, 2004.

[10] Google SketchUp, http://sketchup.google.com/, 2009.

[11] F. Hernell, P. Ljung, and A. Ynnerman. Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 99(2), 2009.

[12] F. Hernell, A. Ynnerman, and P. Ljung. Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *Proc. of Volume Graphics 2007*, pages 1–8, 2007.

[13] A. Iske and T. Randen, editors. *Atlas of 3D Seismic Attributes, Mathematics in Industry, Mathematical Methods and Modelling in Hydrocarbon Exploration and Production*. Springer, 2006.

[14] M. W. Jones. 3d distance from a point to a triangle. *Technical Report CSR-5-95, Department of Computer Science, University of Wales Swansea*, February 1995.

[15] M. W. Jones, J. A. Baerentzen, and M. Sramek. 3d distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG)*, 12(4):581–599, 2006.

[16] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson. A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):150–162, 2003.

[17] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8:29–37, 1987.

[18] E. M. Lidal, T. Langeland, C. Giertsen, J. Grimsgaard, and R. Helland. A decade of increased oil recovery in virtual reality. *IEEE Computer Graphics and Applications*, 27(6):94–97, 2007.

[19] K. Marfurt. Robust estimates of 3D reflector dip and azimuth. *Geophysics*, 71:P29, 2006.

[20] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[21] D. Patel, C. Giertsen, J. Thurmond, J. Gjelberg, and M. E. Gröller. The seismic analyzer: Interpreting and illustrating 2d seismic data. *IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG)*, 14(6):1571–1578, Oct 2008.

[22] D. Patel, C. Giertsen, J. Thurmond, and M. E. Gröller. Illustrative rendering of seismic data. In *Proc. of Vision Modeling and Visualization*, pages 13–22, 2007.

[23] R. Pepper and G. Bejarano. Advances in seismic fault interpretation automation. In *Search and Discovery Article 40170, Poster presentation at AAPG Annual Convention*, pages 19–22, 2005.

[24] B. T. Phong. Illumination for computer generated pictures. *ACM Commununications*, 18(6):311–317, 1975.

[25] J. Plate, T. Holtkaemper, and B. Fröhlich. A flexible multi-volume shader framework for arbitrarily intersecting multi-resolution datasets. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1584–1591, 2007.

[26] J. Plate, M. Tirtasana, R. Carmona, and B. Fröhlich. Octreemizer: a hierarchical approach for interactive roaming through very large volumes. *Proc. of VISSYM '02*, pages 53–64, 2002.

[27] C. Rezk-Salama. Gpu-based monte-carlo volume raycasting. In *Proc. of Pacific Graphics 2007*, pages 411–414, 2007.

[28] T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. H. Hinrichs. Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum*, 27(2):567–576, 2008.

[29] T. Ropinski, F. Steinicke, and K. H. Hinrichs. Visual exploration of seismic volume datasets. *Journal Proc. of WSCG '06*, 14:73–80, 2006.

[30] Schlumberger and Petrel. Petrel seismic interpretation software, schlumberger information solutions (sis).

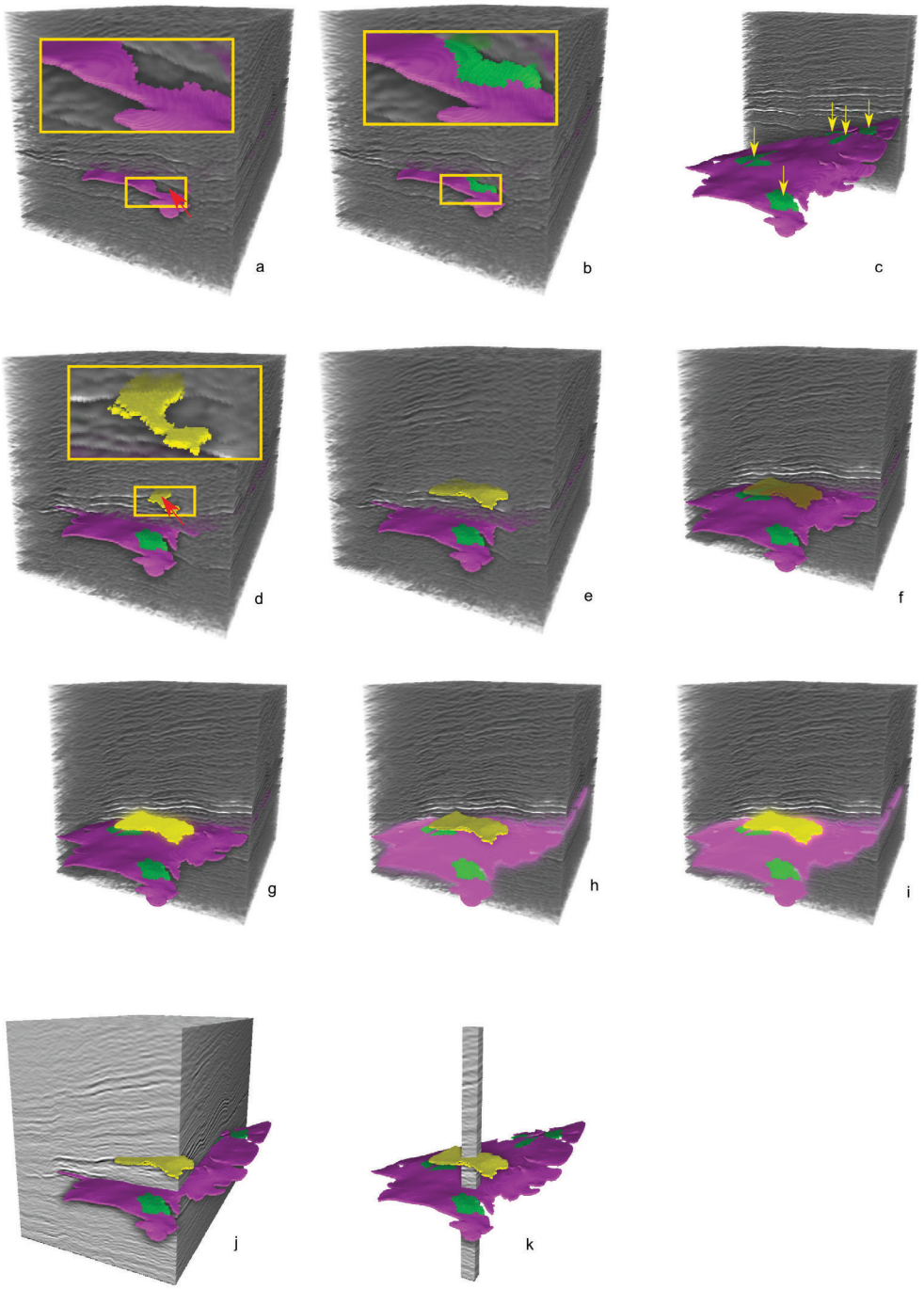
[31] M. Schott, V. Pegoraro, C. Hansen, K. Boulanger, J. Stratton, and K. Bouatouch. A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum*, 28(3):855–862, June 2009.

[32] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[33] P. Silva, M. Machado, and M. Gattass. 3d seismic volume rendering. *8th Intn. Congress of The Brazilian Geophysical Soc.*, 2003.

[34] A. J. Stewart. Vicinity shading for enhanced perception of volumetric data. In *IEEE Visualization 2003*, pages 355–362, 2003.

[35] R. Yagel, A. Kaufman, and Q. Zhang. Realistic volume imaging. In *IEEE Visualization 1991*, pages 226–231, 1991.


a
b
c
d
e
f
g
h
i
j
k

8 Volume Visualization based on Statistical Transfer-Function Spaces

It is a difficult task to design transfer functions for noisy data. In traditional transfer-function spaces, data values of different materials overlap. In this paper we introduce a novel statistical transferfunction space which in the presence of noise, separates different materials in volume data sets. Our method adaptively estimates statistical properties, i.e. the mean value and the standard deviation, of the data values in the neighborhood of each sample point. These properties are used to define a transfer-function space which enables the distinction of different materials. Additionally, we present a novel approach for interacting with our new transferfunction space which enables the design of transfer functions based on statistical properties. Furthermore, we demonstrate that statistical information can be applied to enhance visual appearance in the rendering process. We compare the new method with 1D, 2D, and LH transfer functions to demonstrate its usefulness.

The following paper appears in its original format published as:

M. Haidacher, D. Patel, S. Bruckner, A. Kanitsar, and M. E. Gröller. Volume visualization based on statistical transfer-function spaces. In *Proceedings of IEEE Pacific Visualization 2010*, pages 17–24, 2010.

Volume Visualization based on Statistical Transfer-Function Spaces

Martin Haidacher* Institute of Computer Graphics and Algorithms Vienna University of Technology, Austria

Daniel Patel† Christian Michelsen Research Bergen, Norway

Stefan Bruckner‡ Institute of Computer Graphics and Algorithms Vienna University of Technology, Austria

Armin Kanitsar§ AGFA HealthCare Vienna, Austria

M. Eduard Gröller¶ Institute of Computer Graphics and Algorithms Vienna University of Technology, Austria

ABSTRACT

It is a difficult task to design transfer functions for noisy data. In traditional transfer-function spaces, data values of different materials overlap. In this paper we introduce a novel statistical transfer-function space which in the presence of noise, separates different materials in volume data sets. Our method adaptively estimates statistical properties, i.e. the mean value and the standard deviation, of the data values in the neighborhood of each sample point. These properties are used to define a transfer-function space which enables the distinction of different materials. Additionally, we present a novel approach for interacting with our new transfer-function space which enables the design of transfer functions based on statistical properties. Furthermore, we demonstrate that statistical information can be applied to enhance visual appearance in the rendering process. We compare the new method with 1D, 2D, and LH transfer functions to demonstrate its usefulness.

Keywords: Transfer function, statistics, classification, noisy data, shading.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture; I.4.10 [Image Processing and Computer Vision]: Image Representation—Volumetric

1 INTRODUCTION

Volume classification is a major issue in volume visualization. The goal of classification is to enhance different materials or features which are important for further analysis of the data. Transfer functions have been proven to be a powerful tool for classification. Nevertheless, in most cases it is a non-trivial task to find a good transfer function which is able to achieve the desired classification.

A transfer function is a general concept. Concrete implementations use one or more properties, derived from the data, to define a transfer-function space. A transfer function is then designed in this space. How easily different materials can be distinguished by the transfer function, depends on the data set as well as on the used properties in the definition of the transfer-function space.

Noise in the measured data is a typical problem, which complicates the classification process. The most frequently observed noise in measured data is Gaussian white noise. White noise has a mean value of zero and a symmetric variance. For different materials in a volume data set, the noise causes variations of the intensity values around an average value. Therefore, it is difficult to assign intensity values of sample points to a certain material, especially if average values of other materials are close by. This intermixing of materials in the intensity space cannot be resolved in the transfer-function space when only local properties, such as the intensity and gradient magnitude, are used to define the transfer-function space.

In this paper we present a method which considers a larger area around each sample point to derive properties for the transfer-function space. By this, we are able to estimate the distribution of noise around the average value of a material. The statistical properties of this distribution are used to describe the material at a certain sample position. Since different materials can be distinguished by their distributions of intensity values, we are able to separate them.

For the estimation of the statistical properties, we employ an adaptive growing approach at each sample point. The extent of the growing is dependent on the local neighborhood of a sample point. The estimated properties are used to define the statistical transfer-function space. Sample points from separate materials can be seen as separate clusters in this space. We introduce special transfer-function regions which are adapted for this space to design a transfer function. Furthermore, we demonstrate how the statistical properties can be used to steer visual properties such as shading. This results in higher quality visualizations, especially for noisy data.

We use the new statistical transfer-function space to generate images for medical MR and industrial CT data. We show for these data sets, that our method classifies different materials better than other state-of-the-art transfer-function spaces.

2 RELATED WORK

The work presented in this paper spans several research fields. Since we are dealing with noise in the data, the analysis of noise in image processing is related to this work. The growing of regions around each sample point is similar to the scale-space analysis where data is analyzed on different scales. In this work we introduce a new transfer-function space. Therefore, most of the related work is dedicated to other transfer-function spaces.

Image Processing. Noise in data is a well investigated field in image processing. In this work we assume the noise in the data to be Gaussian white noise. This is specifically true for CT data sets [12, 27]. For other data sets, the Gaussian distribution is at least a good approximation of the noise distribution. In MRI, e.g., the real distribution is a Rician distribution, but for a low signal-to-noise ratio the difference to a Gaussian distribution is very small [6].

Scale-Space Analysis. Early works, such as Lindeberg [14], analyzed images on different scales. Over the years, different scale-spaces were investigated. The most common scale space is the linear scale space, which is generated by progressive Gaussian smoothing. In this scale-space Lindeberg [15] introduced a technique for feature detection and automatic scale selection. Due to the complexity of the scale-space generation for volume data, alternatives, such as Laplacian pyramids [5] or Wavelet transforms [20], were developed for an easier and faster representation of different scales. A method to improve the classification of features, based on the pyramid representation, was introduced by Lum *et al.* [17]. In contrast to scale-space analysis, our method uses different scales for each sample point because the growing is terminated depending on local properties of the neighborhood region.

*e-mail:haidacher@cg.tuwien.ac.at
†e-mail:daniel@cmr.no
‡e-mail:bruckner@cg.tuwien.ac.at
§e-mail:armin.kanitsar@agfa.com
¶e-mail:groeller@cg.tuwien.ac.at

Transfer Function Spaces. In an early work, Levoy [13] used the data value alone to define a transfer function space. Kniss *et al.* [9] employed the data value and the gradient magnitude for the classification of different materials and borders between them. Since they only consider single data values and a very small neighborhood for the gradient magnitude, this technique is not well suited for the classification of noisy data. Hladůvka *et al.* [8] proposed curvature as an additional property for the classification. With this method special features, like ridges and valleys, could be extracted. An extension to multi-dimensional transfer functions was introduced by Roettger *et al.* [23]. The method includes spatial information in the transfer-function space. They simplified the transfer-function design-process by using the spatial information to color the transfer-function space. However, for noisy data different materials overlap in this transfer-function space.

In the work of Lum and Ma [16], a larger region is considered for the definition of the transfer-function space. Besides the data value at a sample point, a data value along the gradient direction is used as well. In data sets with sharp transitions, such as CT data, this technique can be used to highlight border areas. An extension to this method was introduced by Sereda *et al.* [26] named LH histograms. This method looks for homogeneous regions along both directions of the gradient streamline. The detected low and high data values are used to define the transfer-function space. This method provides good results for data sets with little noise. For noisy data sets, values in homogeneous regions have a high variance. Therefore, the clusters representing homogeneous regions are getting larger and overlap each other in the LH histogram space.

A method which also uses a larger neighborhood for the classification was presented by Hadwiger *et al.* [7]. They use region growing to detect features of different sizes in industrial CT data. In a 3D transfer-function space these different features can be classified. In the work of Correa and Ma [3], a multi-scale approach is used to detect the size of features in a data set. The feature size is then used as an additional parameter for the definition of a transfer function. In both approaches the shape of a feature in the data set is the main criterion for the classification. Instead in our method, the statistical properties of materials are used for the classification. These properties are independent of object shapes.

Lundström *et al.* [18] introduced a method to classify different tissues by the local histograms in the neighborhood around a sample point. Caban and Rheingans [2] used textural properties to differentiate between materials, possibly with similar data values. These methods are able to separate materials but they use a neighborhood with a fixed size for the extraction. Thus, these approaches do not differentiate between homogeneous and inhomogeneous regions.

Laidlaw *et al.* [11] use Bayes' theorem on a small neighborhood of a voxel to classify mixed materials. Tenginakai *et al.* [24, 25] introduced a method to extract salient iso-surfaces based on statistical methods. A different classification based on statistics was introduced by Kniss *et al.* [10]. For the estimation of the statistical characteristics certain properties of the different materials have to be known. For our approach no prior knowledge of material properties is necessary. Lundström *et al.* [19] used the variance in a neighborhood of a voxel to separate materials. In comparison to our work, they used a fixed neighborhood size to estimate the variance. In our previous work (Patel *et al.* [21]) we used statistical properties to manually classify materials for differently sized neighborhood regions. In this work we extract the statistical properties for the best suited neighborhood size semi-automatically. Furthermore, we use these statistical properties to define a transfer-function space and to enhance the visual appearance of the resulting rendering.

3 Statistical Transfer-Function Space

The idea behind the statistical transfer-function space is that materials are distinguishable according to their statistical properties. Since the data is not segmented, we are not able to calculate the statistical properties for different materials in general. Therefore, we introduce a technique which extracts statistical properties for the neighborhood of each sample point individually. We expect that sample points from the same material get similar statistical properties. In the new transfer-function space this leads to clusters for different materials, which makes it possible to design meaningful transfer functions. In this section we describe all steps which are necessary to generate the statistical transfer-function space.

Figure 1 shows an overview of the workflow. To generate a visualization based on statistical transfer functions, different processing steps have to be applied on a volume data set. For the generation of the transfer-function space, statistical properties, i.e. the mean value and the standard deviation, are extracted first. This is done in a pre-processing step. The user defines a confidence level for this step. This confidence level is a quantity for the tolerance in the extraction step. It is further explained in Section 3.1.

The properties for each sample point are then depicted in the transfer-function space. They serve as a clue for the user to design a transfer function. The transfer function together with the statistical properties drives the successive visualization step. Additionally the statistical properties are used to enhance the shading.

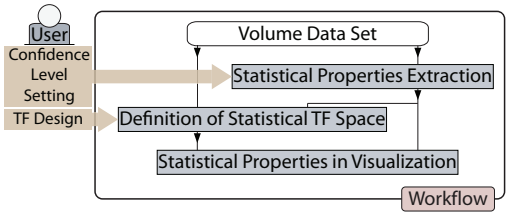
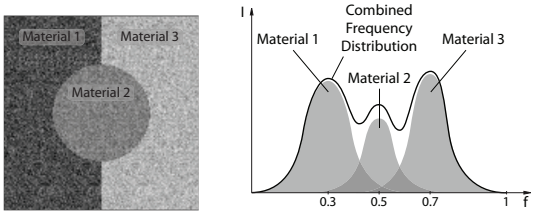


Figure 1: Statistical transfer-function workflow.

To exemplify our new method, we generated a synthetic data set of size $128 \times 128 \times 128$. The data set contains three different materials. In Figure 2(a) a slice through the center of the data set is shown. Material 2 in the center of the data set is a sphere, embedded between material 1 and material 3. Gaussian white noise has been added to all three materials. As mentioned before this is a realistic noise model for most data sets especially for CT and MRI. In Figure 2(b) the histograms of the materials are shown. On the horizontal axis the data values $f(x,y,z)$ of the sample points are mapped. The vertical axis holds the frequency of occurrences I for each data value. The Gaussian distributions of all three materials have high standard deviations ($\sigma_1 = 0.14$, $\sigma_2 = 0.09$, and $\sigma_3 = 0.11$), consequently the distributions considerably overlap each other. The black line gives the frequency distribution of all three materials together.



(a) Slice view (b) Frequency distribution

Figure 2: Slice through the synthetic data (a) and the frequency distribution of the data values (b).

The synthetic data set exemplifies noisy data. The overlapping of the distributions for different materials is a problem which often occurs in real-world data sets [26]. In the further explanation of the method, the synthetic data set is used to show the effects of the different processing steps.

3.1 Statistical Properties Extraction

The extraction of the statistical properties is essential for our statistical transfer-function space. For sample points within a certain material, the extracted statistical properties should be close to the statistical properties of the entire material. We achieve this by investigating a neighborhood region around each sample point. To keep the neighborhood within the same material, we introduce an adaptive growing which is dependent on the local properties.

The distribution of data values of a single material in real-world data sets, such as MRI or CT, can be approximated very well by the Gaussian white noise model. Therefore, we consider only the Gaussian distribution as basis for the calculation of statistical properties. A Gaussian distribution is described by its mean value and standard deviation. Hence, we use these two parameters as our statistical properties.

For the extraction of the statistical properties, we iteratively grow a spherical neighborhood by increasing the radius by one voxel in each step. We compare for each growing step if the newly grown hull still belongs to the same material. Figure 3 shows a cross section of such a neighborhood. In the following explanation we use two different notations for the statistical properties. The mean value μ_r and the standard deviation σ_r for a certain radius r, are the estimations for the statistical properties of all points within a sphere of radius r. $\dot{\mu}_r$ and $\dot{\sigma}_r$ are the statistical properties of the points in the outer hull of the sphere (see Figure 3).

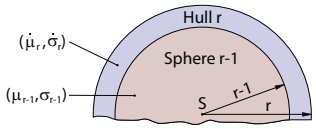


Figure 3: The standard deviation and mean for the sphere and the outer hull.

In each growing step several calculations are done to decide if the growing should be terminated or not. Figure 4 shows the processing steps for each growing step. The goal of these steps is to detect whether the sphere grows into another material. In such a case the loop is terminated.

As initial parameters for the loop, the mean value and the standard deviation of a sphere with a radius of one are used. The data values of points in the sphere may not be normally distributed. This can happen when the sphere intersects two materials. Such a situation should be detected in an early stage so growing can be terminated. Therefore, we apply a normal-distribution test, as will be described later in this section, to check if the points in the initial sphere are normally distributed. Only if this test is passed, the loop is started with the initial statistical properties. Otherwise μ_1 and σ_1 of the initial sphere are used as statistical properties for the actual sample point.

In the loop depicted in Figure 4, the first step is the calculation of the statistical properties $\dot{\mu}_r$ and $\dot{\sigma}_r$ for points in the hull at a radius r. In the next step it is tested if the distribution in the hull is normally distributed. If this is the case, the properties are compared with μ_{r-1} and σ_{r-1}. In the case the statistical properties are similar, the statistical properties of the hull are merged with the statistical properties of the sphere $r-1$. If a sample point lies in the center of a large homogeneous area, the loop is terminated when the maximum radius r_{max} is reached.

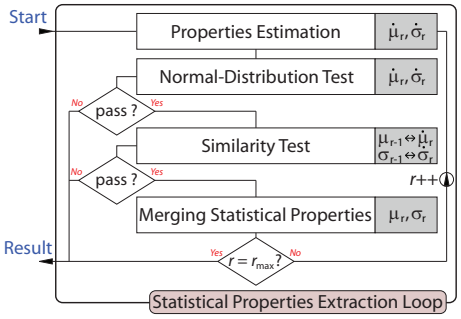


Figure 4: Calculation loop for the extraction of the statistical properties.

For the extraction of the statistical properties, a confidence level ω has to be set. This confidence level expresses the general confidence in the distribution of data values in a data set with respect to the general noise level. It can be set differently to adapt the model for various data types such as MRI or CT. In the following part of this section, the usage of ω and the other processing steps are described in more detail.

Properties Estimation

In each successive cycle of the extraction loop, the statistical properties of a larger region are considered. Since the statistical properties for sphere $r-1$ are already known, we are interested in the statistical properties $\dot{\mu}_r$ and $\dot{\sigma}_r$ of the additional points in the hull r (see Figure 3).

As the distribution is considered to be Gaussian, we estimate the mean value and the standard deviation for the points in the hull r [4]. The mean value $\dot{\mu}_r$ is the average and $\dot{\sigma}_r$ is the biased standard deviation of all points:

$$\dot{\mu}_r = \frac{1}{\dot{N}_r}\sum_{i=1}^{\dot{N}_r} f_i, \quad \dot{\sigma}_r = \sqrt{\frac{1}{\dot{N}_r}\sum_{i=1}^{\dot{N}_r}(f_i - \dot{\mu}_r)^2} \tag{1}$$

$\dot{N}_r$ is the number of points included in the hull r. f_i denotes the data value of a sample point in the hull. With these estimations for the mean value and standard deviation we expect to get values which are close to the real statistical properties of the material in the hull.

Normal-Distribution Test

Before we apply a similarity test with the derived properties of the hull and the statistical properties of the inner sphere, it must be ensured that the distribution is normally distributed. This is necessary because the similarity test is based on normal distributions. When two materials are intersecting the hull, the distribution is not normally distributed. In such a case the distribution would have two peaks. With the normal-distribution test we want to detect such situations and terminate the loop.

In statistics, several normality tests exist. We chose the Jarque-Bera test (JB). This test uses the third-order moment, i.e. the skewness $\dot{S}_r$, and the fourth-order moment, i.e. the kurtosis $\dot{K}_r$, of the points in the hull at radius r for the calculation of JB:

$$JB = \frac{\dot{N}_r}{6}\left(\dot{S}_r^2 + \frac{(\dot{K}_r - 3)^2}{4}\right) \tag{2}$$

The benefit of using this test is that it is not necessary to sort the point values of the distribution. Therefore, it can be efficiently implemented on the graphics hardware.

The parameter JB is used to test for the null hypothesis with a test level $1-\alpha$ of 99.9%. This results in a test value of 13.82 according to statistical lookup-tables:

$$H_0 : JB < 13.82_{(1-\alpha=0.999)} \tag{3}$$

With the high test level, only distributions with a high divergence to a normal distribution are declined by the null hypothesis. This is necessary, due to the low number of samples which are used for the test. Only distributions which are very different from a normal distribution will fail the test.

If the null hypothesis is declined, the loop is terminated and μ_{r-1} and σ_{r-1} of sphere $r-1$ are taken as statistical properties for the sample point S. If the test passes the null hypothesis, we continue with the similarity test.

Similarity Test

In this processing step, we measure the similarity between the statistical properties in the hull r and the statistical properties of the sphere $r-1$. The goal is to detect whether the hull is still part of the same material as the sphere $r-1$.

Through the properties-estimation step, we get the statistical properties of the hull. We have calculated the statistical properties of the sphere $r-1$ in the preceding loop cycle. Since all the values are estimations and the numbers of points which are involved in the estimation is rather low, we use a variant of the student's t-test for the similarity test. This test is best suited for Gaussian-distributed populations with small sample sizes [4].

In our case, we have two independent samples which were used for the estimations. Since the mean values as well as the standard deviations of both estimations can vary, we use a generalized form of the student's t-test also known as the Welch's t-test [28]. As primary parameter for the similarity test, a t_r parameter is calculated:

$$t_r = \frac{\mu_{r-1} - \dot{\mu}_r}{\sqrt{\frac{\sigma_{r-1}^2}{N_{r-1}-1} + \frac{\dot{\sigma}_r^2}{\dot{N}_r-1}}} \tag{4}$$

The t_r parameter is dependent on the mean values, the standard deviations, and sample sizes of both distributions. Additionally, a degree-of-freedom δ_r has to be calculated:

$$\delta_r = \frac{\left(\frac{\sigma_{r-1}^2}{N_{r-1}-1} + \frac{\dot{\sigma}_r^2}{\dot{N}_r-1}\right)^2}{\frac{\sigma_{r-1}^4}{(N_{r-1}-1)^3} + \frac{\dot{\sigma}_r^4}{(\dot{N}_r-1)^3}} \tag{5}$$

The degree-of-freedom δ_r is only dependent on the standard deviations and the sample sizes, but not on the mean values. The δ_r value together with the confidence level ω are used as parameters to get a reference $t_\omega(\delta_r)$ value in a t-test lookup-table. This value is used to test for the null hypothesis H_0:

$$H_0 : |t_r| < t_\omega(\delta_r) \tag{6}$$

If the null hypothesis is true, it is assumed that both Gaussian distributions are the same. If the null hypothesis is declined then both distributions are expected to be different with a probability of $1-\omega$. Therefore, a small confidence level ω results in a high probability that both distributions are not similar if the null hypothesis is declined. On the other hand, the reference $t_\omega(\delta_r)$ value for a small ω is high which makes the similarity test less selective.

As in the step earlier μ_{r-1} and σ_{r-1} of sphere $r-1$ are taken as statistical properties if the test is failed. Otherwise we continue with the next growing step.

Merging Statistical Properties

If the statistical properties have passed the normal-distribution test and the similarity test, we assume that the material in the outer hull still is the same as in the sphere $r-1$. Therefore, the statistical properties of both areas can be merged together.

This step results in a new μ_r and σ_r. These statistical properties represent the distribution of all points in the sphere r. The merged statistical properties are used in the successive cycle of the loop to do the similarity test with the next larger hull: $r+1$.

The loop is terminated when the normal-distribution test or the similarity test fails or when the maximum radius is reached. In the first two cases we store μ_{r-1} and σ_{r-1} as statistical properties μ and σ for the sample point S. In the third case we take the statistical properties μ_r and σ_r after the merging step.

Additionally, we store the radius r_{break} at which the loop is terminated. The closer r_{break} is to r_{max} the more significant the statistical properties are at this point, because the population of points for the estimation is larger. In the next section, r_{break} is used to highlight statistical properties with a higher significance.

Figure 5 shows the statistical properties μ and σ for the synthetic data set at different confidence levels ω. For a low ω of 0.1%, the similarity test is more easily passed. Therefore, the sphere grows larger and results in more consistent values for μ and σ. For the transfer-function space this means that the clusters of materials are smaller. If the confidence level ω is high, e.g. 30%, the loop has a higher probability of being terminated. Since in this case a smaller area is used to estimate the statistical properties, the result for μ and σ is less smooth compared to a low ω but details, such as borders, are preserved better.

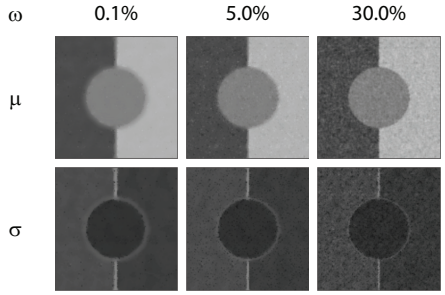


Figure 5: Statistical properties (μ and σ) of the synthetic data for three different confidence levels $\omega = \{0.1\%, 5.0\%, 30.0\%\}$. The brighter a point is, the higher is its value.

Congruent to the characteristic in this example, ω should be chosen according to the type of data. If a modality is very noisy and the distribution of points does not exactly follow a normal distribution, the confidence value should be chosen rather low. Therefore, some details get lost but the clusters for different materials in the transfer-function space are smaller. MRI is an example of such data. In such a case, ω should be set to a low value, such as 0.1%. For less noisy data types with a Gaussian-like distribution of the data values, e.g. CT, ω can be set to a higher value, such as 20%. Then details are better preserved and the clusters are small enough due to the low noise level. The synthetic data set is rather noisy but the distribution of the points follows exactly a normal distribution. Therefore, a confidence level of 5% was chosen. The values for the confidence

levels were found through experiments, where we tried to achieve the best balance between noise reduction and detail preservation. This has to be done only once for a certain type of data source. In Section 5 we demonstrate that our presented values work well for different MRI and CT data sets.

3.2 Definition of Statistical Transfer-Function Space

By extracting the statistical properties we get a mean value μ and a standard deviation σ for each sample point. In the next step we use this information for the design of a transfer function. First we have to define a transfer-function space, which is used for the design of a transfer function. The intent of this statistical transfer-function space is to separate different materials in the presence of noise.

For the transfer-function space we use the original data value f of each sample point together with the mean value μ and the standard deviation σ. We follow the convention of using the horizontal axis of a transfer-function space for the data value. The horizontal axis is, however, also used to depict the mean μ. In the new transfer-function space the data value f on the horizontal axis is considered as starting point of a line segment. The statistical properties μ and σ for each sample point define a second point, where the standard deviation is given on the vertical axis.

In Figure 6 the transfer-function space is shown. On the left side the properties of a sample point S are drawn as a line segment in this space.

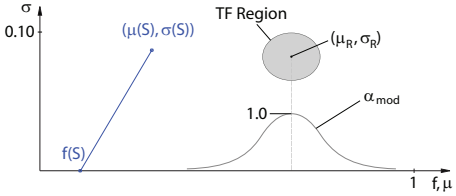


Figure 6: Schematic representation of the statistical transfer-function space. On the left side the properties of a sample point are shown as a line segment. On the right side a transfer-function region in the new space can be seen.

Since the clusters for each material are expected to show up around a certain mean value and standard deviation, an elliptical area is used to define a transfer-function region. On the right side of Figure 6 an example of such a region is shown. The center of the region is at (μ_R, σ_R). A sample point is only classified by this region when its statistical properties (μ, σ) lie within the elliptical area.

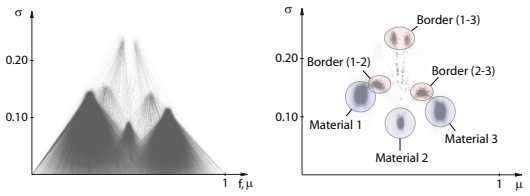
The optical properties for each region are represented with a color c_R and an opacity α_R. The position of the transfer-function region also defines the corresponding Gaussian distribution of data values along the horizontal axis. The opacity of sample points is manipulated according to this distribution:

$$\alpha_{mod} = \alpha_R * e^{-\frac{1}{2}\left(\frac{f-\mu_R}{\sigma_R}\right)^2} \tag{7}$$

α_R is the opacity defined for the respective transfer-function region. f is the data value of the sample point and (μ_R, σ_R) is the center point of the region. Equation 7 is represented as a curve along the horizontal axis in Figure 6, below the transfer-function region. The opacity for data values f close to the center μ_R of the region are modified less in contrast to the opacity of data values f, which are more different to μ_R. By this modification, we reduce the influence of sample points which are less likely to be part of the desired distribution.

Figure 7 shows the sample points of the synthetic data set in the statistical transfer-function space. In Figure 7(a) each sample point is represented as a line segment. Since the line segments of points with high σ values might occlude points with low σ values, an alternative representation is shown in Figure 7(b). In this representation only a single dot at (μ, σ) is drawn for each sample point. In both representations three large clusters for the different materials and three smaller clusters for the borders can be seen. The border between material 1 and 3 actually consists of two small clusters. This results from the fact that the mean values of material 1 and 3 are very different. Therefore, it makes a difference if the origin sample point of a neighborhood-region lies in material 1 or 3, because more points from one or the other material are then used for the estimation of the mean value.

If the difference between the mean values of two materials is smaller, e.g. between material 1 and 2, then only one cluster shows up in the transfer-function space.



(a) Line representation (b) Dot representation

Figure 7: Line and dot representation in the statistical transfer-function space of the synthetic data.

To enhance points with higher significance, we weight the opacity of line segments or dots by the termination radius r_{break} of the calculation loop. Thus, statistically well defined interior regions are emphasized.

3.3 Statistical Properties in Visualization

In addition to the definition of a transfer function we use the statistical properties as input for the shading process. In volume visualization gradient-based techniques are most common for shading. They are computational less expensive and faster as gradient-free shading techniques. For noisy data the gradient-based techniques have the penalty that the noise in the data deteriorates the estimation of the gradient and, therefore the shading [1]. Especially in homogeneous regions, where the gradient magnitude is rather low, noise has a high impact on the estimation of the gradient direction.

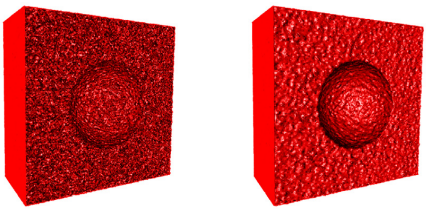
To reduce the influence of noise on the gradient, we use the mean values of the sample points to estimate the gradient direction. The mean values are smoother than the original data values and, therefore, the gradient direction in nearly homogeneous regions is estimated better in comparison to the widely used central-difference method. With this gradient we calculate a color c_{shaded} based on Phong shading [22].

For the observer, shading is important for the perception of surfaces. To avoid visual clutter resulted when shading all parts of the volume, we apply shading only for border regions. For this purpose the standard deviation σ can be used. σ is higher in border regions and it is less affected by noise than the gradient magnitude. Therefore, σ is used to interpolate between the shaded color c_{shaded} and the unshaded color $c_{unshaded}$:

$$c = (1-\sigma)c_{unshaded} + \sigma c_{shaded} \tag{8}$$

The lower σ is, the less shading is applied. This leads to a visualization where the border areas are shaded more as opposed to

the rest. The synthetic data with traditional shading in Figure 8(a) is shown in contrast to the shading based on statistical properties in Figure 8(b). With statistical shading the influence of noise is clearly reduced without modifying the data itself from filtering or other techniques.



(a) Traditional shading (b) Statistical shading

Figure 8: Visual difference between traditional shading and statistical shading. With statistical shading the influence of noise is reduced.

4 Implementation

The steps which are described in Section 3 can be divided into two parts of computation. The first part is the extraction of statistical properties. This is a pre-processing step which has to be recalculated only when the confidence level ω is changed. The other part is the visualization with the usage of the statistical transfer function. This part is done in real-time.

Since the extraction of the statistical properties is a highly parallel process, we use graphics hardware. Nevertheless, the extraction is expensive because many lookups have to be performed to estimate the statistical properties.

The maximum number of lookups per sample point is dependent on the maximum radius r_{max} of the calculation loop. We decided to set the maximum radius to a value of 6. With this radius, a maximum of 925 lookups are done per sample point. This leads to the following 99% confidence limits:

$$\mu = \overline{\mu} \pm (0.0424\sigma) \tag{9}$$

This means that the estimated μ is, with a confidence of 99%, not farther away from the real mean value $\overline{\mu}$ than 0.0424 times σ. Even for a large σ the interval for μ is small enough for our purpose of estimating the statistical properties.

In addition to the maximum radius r_{max}, the confidence level ω influences the speed of the calculation. The lower ω, the less likely it is that the calculation loop is terminated. Therefore, more lookups have to be done for the estimation of the statistical properties. In Figure 9 the termination level at different radii is shown. The graph shows the percentage of all sample points for which the calculation loop is already terminated at a certain radius. For the synthetic data set different confidence levels are used. It can be seen that the termination level at the maximum radius of 6 is much lower for an ω of 0.1% in comparison to an ω of 30%. For the low ω the loop is only terminated for points close to the border. A high ω causes terminations also for large variations in homogeneous regions. As shown in Figure 5, the visual difference of various ω values can be seen in the smoothness.

Apart from the synthetic data set, Figure 9 also shows the termination level for two real-world data sets with according confidence levels for each type of data. In the case of CT data, we get a high termination level even at a low radius. This results from the fact that areas with zero variation, e.g. air, do not pass the initial normal-distribution test. For MRI data, the curve is more linear, due to the fact that all parts contain at least some noise.

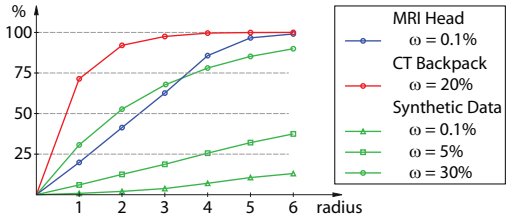


Figure 9: Termination level of the calculation loop for different data sets and confidence levels ω.

Table 1 shows the time measurements for the estimation of the statistical properties for the data sets of Figure 9 on a GeForce GTX 260. The different settings for the level ω have only a small influence on the calculation time. This results from the fact that most time is used to initialize the graphics hardware for the calculation.

MRI Head 256×256×128	CT Backpack 256×256×186	Synthetic Data 128×128×128		
ω=0.1%	ω=20%	ω=0.1%	ω=5%	ω=30%
2.386*sec*	2.330*sec*	0.654*sec*	0.649*sec*	0.646*sec*

Table 1: Time measurement for the estimation of statistical properties for different data sets.

After the extraction of the statistical properties, we use raycasting for the visualization. The statistical properties μ and σ are stored in additional channels of the volume. To classify a sample point with the designed transfer function, the parameters of all transfer-function regions are handed over to the graphics card. There it is tested if the statistical properties of a sample point lie within a transfer-function region. If so, the color and opacity is assigned to the sample point as described in Section 3.2. This classification can be implemented efficiently on the graphics card. We get interactive rendering rates for data sets of size $256 \times 256 \times 256$ on a GeForce 8800 GT graphics card.

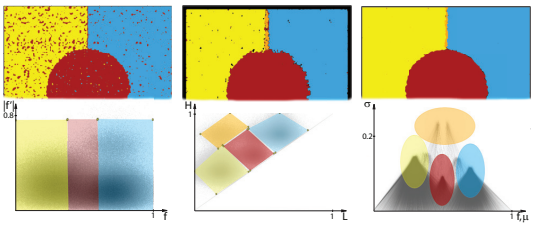
5 Results and Discussion

In this section we show some results generated with our new method and compare them with other techniques.

5.1 Synthetic Data Set

For the explanation of the method we have introduced a synthetic data set, as shown in Figure 2. The three different materials in this data set are rather noisy. Therefore, it is difficult to separate the materials in common transfer-function spaces. In Figure 10(a) the 2D transfer-function space with axis f and $|f'|$ and in Figure 10(b) the LH histogram-space were used to classify the different materials. With the 2D transfer function we were not able to classify all points correctly, because of the density overlapping. Especially at the border between material 1 and material 3 (blue and yellow) points are classified as material 2 (red). In the LH histogram-space it is easier to separate the different materials and the border but transitions are very ragged.

In Figure 10(c) the result of our method is shown for an ω of 5%. Since the different materials have different statistical properties they can be clearly seen as clusters in the statistical transfer-function space. For the synthetic data, smoothing techniques would be able to reduce the cluster sizes in the 2D transfer-function space and in the LH histogram-space. However, the smoothing only clusters the data values. Our approach uses the standard deviation for



(a) 2D TF (b) LH histogram (c) Statistical TF

Figure 10: Classification results of the materials in the synthetic data set with different methods.

the classification as well. Therefore, we are better able to classify different materials especially if they differ by their standard deviation, such as border regions.

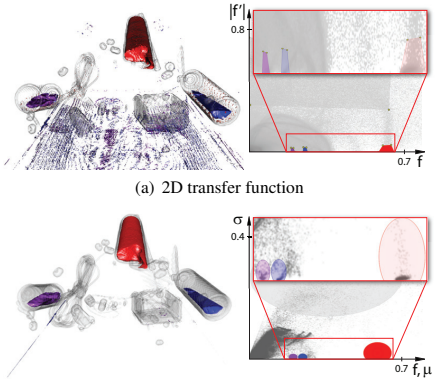
5.2 Real-World Data Sets

In the real world, noise is typically present in measured data sets. The amount of noise varies between acquisition techniques. In MRI data sets, the noise level is rather high. Therefore, it is especially difficult to classify different materials in such data sets. A common problem is the visualization of the brain in an MRI scan of the head. Figure 11 shows different results of this task for different classification techniques. Below each rendering result, the settings of the transfer function for each of these spaces are shown. In comparison to the 1D (Figure 11(a)) and 2D transfer function (Figure 11(b)) as well as the LH histogram-space (Figure 11(c)), we can better separate the brain from other tissues with our method (Figure 11(d)). In the statistical transfer-function space it is also easier to design a transfer function because the brain tissue has different statistical properties than other tissues in this data set and is more tightly clustered. As can be seen in Figure 11(d) the cluster is rather large due to the different matters in the brain but it is distinguishable from other clusters of other tissues. In comparison to this, in the 2D transfer-function space and in the LH histogram-space no cluster for the brain tissue can be seen. Additionally, Figure 11 shows that the shading based on statistical properties (Figure 11(d)) results in a smoother surface in comparison to normal gradient-based shading (Figure 11(a)-(c)).

The ability of the new method to separate different materials by their statistical properties can be used for many applications. An example is the detection of certain materials in CT scans for security checks. Figure 12 shows the scan of a backpack containing three different fluids. With a 2D transfer function, as shown in Figure 12(a), the different fluids cannot be classified without classifying also other parts of the data set. It is also hard to detect the fluids in the transfer-function space because they do not show up as clusters. In comparison, our method can clearly classify the fluids, as shown in Figure 12(b). It can also be seen that the fluids show up as clusters with very low standard deviation in the transfer-function space. This makes it much easier to define a transfer function. Furthermore, the shading with our method is slightly smoother.

Figure 13 shows a result of an MRI scan. The data set contains a tumor inside the brain. The statistical properties of the tumor are actually different from the rest of the brain which is captured in our transfer-function space. This can be seen in Figure 13. Since the tumor is rather small, only a few sample points show up in the area of the classification region of the tumor (red region). However, with other methods, such as 1D, 2D, and LH transfer functions, we were not able to clearly separate the tumor from the brain.

The results show that the new method can be used for various data sets and different tasks. The main reason for this is the confidence level ω, which can be set according to the type of data. For MRI, e.g., where the material distributions slightly differ from a Gaussian distribution, we set ω to a low level in comparison to CT data. We are able to classify different materials even if the data type is different. This is not as easily possible in common other transfer-function spaces.



(a) 2D transfer function

(b) Statistical transfer function ($\omega = 20\%$)

Figure 12: Detection of different fluids in a CT scan of a backpack.

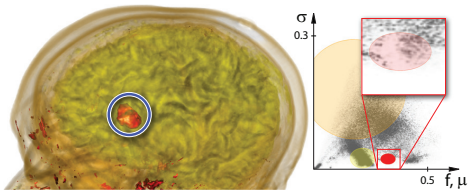
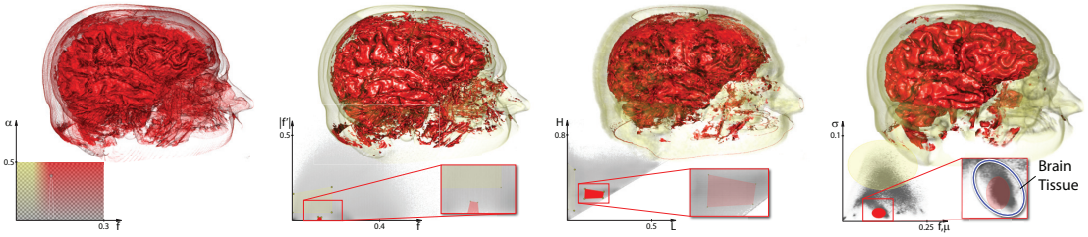


Figure 13: Detection of a brain tumor. For the extraction of the statistical properties ω was set to 0.1%.

There are some limitations of the new technique. One drawback is the rather high memory consumption because for each sample point two different statistical properties have to be stored together with the data value. Thus the data size is tripled. For large data sets this could exceed the memory of a graphics card. Another penalty can occur for noise distributions very different from Gaussian white noise. In such cases the test methods have to be adapted to the given frequency distribution in the data sources. For this work we concentrated on measured data, where the distributions of data values are similar to a Gaussian distribution.

Although the confidence level is the only parameter which has to be set by the user, we want to do further research to automatically define this parameter. For this task we want to use the termination level in our future work. With the termination level at different radii it is probably possible to detect if ω is either too high or too low for a given data set. Furthermore, we want to develop algorithms for the automatic detection of clusters in the statistical transfer-function space. By using the exit radius r_{break} of the calculation loop, we should be able to automatically find significant clusters for different materials. This additional step should accelerate the design process for transfer functions because an initial setting can be provided.



(a) 1D transfer function (b) 2D transfer function (c) LH histogram (d) Statistical TF ($\omega = 0.1\%$)

Figure 11: Comparison of the statistical transfer-function space with the 1D and 2D transfer function space as well as the LH histogram-space. The task for the generation of the results was to classify the brain in the different spaces.

6 Conclusion

In this paper we introduced a novel transfer-function space, based on statistical properties. With these statistical properties from the neighborhood of each sample point we are able to better separate different materials in comparison to other widely used transfer-function techniques. Even for noisy data sets the materials are still distinguishable by their statistical properties.

Additionally, we use the statistical properties in the shading calculations. Through this approach, the influence of noise on the shading is reduced.

In our experiments, the novel transfer-function space has proven to produce better results for different tasks than other common transfer-function techniques. Therefore, we believe the statistical transfer-function space can be used for classifying different materials in a volume data set. Through the user-specified confidence level it can be employed for data sets from various modalities.

Acknowledgements

The work presented in this paper has been done in the scope of the SmartCT (FFG, no. 818108/16647) project and the ScaleVS (WWTF, no. ICT08-040) project.

References

[1] S. Bruckner and M. E. Gröller. Instant volume visualization using maximum intensity difference accumulation. *Computer Graphics Forum*, 28(3):775–782, 2009.

[2] J. J. Caban and P. Rheingans. Texture-based transfer functions for direct volume rendering. *IEEE TVCG*, 14(6):1364–1371, 2008.

[3] C. Correa and K.-L. Ma. Size-based transfer functions: A new volume exploration technique. *IEEE TVCG*, 14(6):1380–1387, 2008.

[4] B. S. Everitt. *The Cambridge Dictionary of Statistics*. Cambridge University Press, 1998.

[5] M. H. Ghavamnia and X. D. Yang. Direct rendering of Laplacian pyramid compressed volume data. In *VIS '95: Proceedings of the IEEE Visualization '95*, pages 192–199, 1995.

[6] H. Gudbjartsson and S. Patz. The Rician distribution of noisy MRI data. *Magnetic Resonance in Medicine*, 34(6):910–914, 1995.

[7] M. Hadwiger, L. Fritz, C. Rezk-Salama, T. Höllt, G. Geier, and T. Pabel. Interactive volume exploration for feature detection and quantification in industrial CT data. *IEEE TVCG*, 14(6):1507–1514, 2008.

[8] J. Hladůvka, A. König, and M. E. Gröller. Curvature-based transfer functions for direct volume rendering. In *Spring Conference on Computer Graphics 2000 (SCCG 2000)*, volume 16, pages 58–65, 2000.

[9] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE TVCG*, 8(3):270–285, 2002.

[10] J. Kniss, R. V. Uitert, A. Stephens, G.-S. Li, T. Tasdizen, and C. Hansen. Statistically quantitative volume visualization. In *VIS '05: Proceedings of the IEEE Visualization '05*, pages 287–294, 2005.

[11] D. H. Laidlaw, K. W. Fleischer, and A. H. Barr. Partial-volume bayesian classification of material mixtures in MR volume data using voxel histograms. *IEEE Transactions on Medical Imaging*, 17(1):74–86, 1998.

[12] T. Lei and W. Sewchand. Statistical approach to X-ray CT imaging and its applications in image analysis part I: Statistical analysis of X-ray CT imaging. *IEEE Transactions on Medical Imaging*, 11(2):53–61, 1992.

[13] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[14] T. Lindeberg. Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:234–254, 1990.

[15] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116, 1998.

[16] E. B. Lum and K.-L. Ma. Lighting transfer functions using gradient aligned sampling. In *VIS '04: Proceedings of the IEEE Visualization '04*, pages 289–296, 2004.

[17] E. B. Lum, J. Shearer, and K.-L. Ma. Interactive multi-scale exploration for volume classification. *The Visual Computer*, 22(9–11):622–630, 2006.

[18] C. Lundström, P. Ljung, and A. Ynnerman. Local histograms for design of transfer functions in direct volume rendering. *IEEE TVCG*, 12(6):1570–1579, 2006.

[19] C. Lundström, P. Ljung, and A. Ynnerman. Multi-dimensional transfer function design using sorted histograms. In *Proceedings Eurographics/IEEE Workshop on Volume Graphics 2006*, pages 1–8, 2006.

[20] S. Muraki. Volume data and wavelet transforms. *IEEE Computer Graphics and Applications*, 13(4):50–56, 1993.

[21] D. Patel, M. Haidacher, J.-P. Balabanian, and M. E. Gröller. Moment curves. In *Proceedings of the IEEE Pacific Visualization Symposium 2009*, pages 201–208, 2009.

[22] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[23] S. Roettger, M. Bauer, and M. Stamminger. Spatialized transfer functions. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization*, pages 271–278, 2005.

[24] S. Tenginakai, J. Lee, and R. Machiraju. Salient iso-surface detection with model-independent statistical signatures. In *VIS '01: Proceedings of the IEEE Visualization '01*, pages 231–238, 2001.

[25] S. Tenginakai and R. Machiraju. Statistical computation of salient iso-values. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, pages 19–24, 2002.

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

[27] J. Wang, T. Li, H. Lu, and Z. Liang. Penalized weighted least-squares approach to sinogram noise reduction and image reconstruction for low-dose X-ray computed tomography. *IEEE Transactions on Medical Imaging*, 25(10):1272–1283, 2006.

[28] B. L. Welch. The generalization of "student's" problem when several different population variances are involved. *Biometrika*, 34(1–2):28–35, 1947.

9 Hybrid Visibility Compositing and Masking for Illustrative Rendering

In this paper, we introduce a novel framework for the compositing of interactively rendered 3D layers tailored to the needs of scientific illustration. Currently, traditional scientific illustrations are produced in a series of composition stages, combining different pictorial elements using 2D digital layering. Our approach extends the layer metaphor into 3D without giving up the advantages of 2D methods. The new compositing approach allows for effects such as selective transparency, occlusion overrides, and soft depth buffering. Furthermore, we show how common manipulation techniques such as masking can be integrated into this concept. These tools behave just like in 2D, but their influence extends beyond a single viewpoint. Since the presented approach makes no assumptions about the underlying rendering algorithms, layers can be generated based on polygonal geometry, volumetric data, point-based representations, or others. Our implementation exploits current graphics hardware and permits real-time interaction and rendering.

The following paper appears in its original format published as:

S. Bruckner, P. Rautek, I. Viola, M. Roberts, M. C. Sousa, and M. E. Gröller. Hybrid visibility compositing and masking for illustrative rendering. *Computers & Graphics*, 34(4):361–369, 2010.

Technical Section

Hybrid visibility compositing and masking for illustrative rendering ☆

Stefan Bruckner [a,*], Peter Rautek [a], Ivan Viola [b], Mike Roberts [c], Mario Costa Sousa [c], M. Eduard Gröller [a]

[a] Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria
[b] Department of Informatics, University of Bergen, Norway
[c] Department of Computer Science, University of Calgary, Canada

ARTICLE INFO

Keywords:
Compositing
Masking
Illustration

ABSTRACT

In this paper, we introduce a novel framework for the compositing of interactively rendered 3D layers tailored to the needs of scientific illustration. Currently, traditional scientific illustrations are produced in a series of composition stages, combining different pictorial elements using 2D digital layering. Our approach extends the layer metaphor into 3D without giving up the advantages of 2D methods. The new compositing approach allows for effects such as selective transparency, occlusion overrides, and soft depth buffering. Furthermore, we show how common manipulation techniques such as masking can be integrated into this concept. These tools behave just like in 2D, but their influence extends beyond a single viewpoint. Since the presented approach makes no assumptions about the underlying rendering algorithms, layers can be generated based on polygonal geometry, volumetric data, point-based representations, or others. Our implementation exploits current graphics hardware and permits real-time interaction and rendering.

1. Introduction

Digital compositing was arguably one of computer graphics' first mainstream commercial applications. Areas such as motion picture post-production greatly benefited from automated processing. The ability to flexibly combine multiple sources ultimately lead to the widespread adoption of digital special effects which are now ubiquitous in the film industry. Today, virtually every image editing software package has the ability to arrange elements in layers, modify alpha channels, control blending, and apply effects to individual layers.

In this paper, we focus on the compositing of dynamic 3D content. Instead of combining static elements such as images, movie sequences, or pre-rendered 3D animations, an interactive environment allows the modification of properties such as the viewpoint for individual layers which are rendered on-the-fly. Common software tools such as Adobe Photoshop have recently introduced the ability to embed dynamically generated layers based on 3D models. However, for the purpose of composing, the layer content is still treated as a 2D image even though additional information would be available. One reason for this choice is the fact that the conventional layered compositing approach, which assumes 2D layers, is deeply incorporated into these software packages and the workflow of their users. In this paper, we present a concept for the integration of 3D layers which preserves this intuitive notion, but allows artists to take advantage of 3D information by extending the operator set of traditional compositing approaches. We deliberately make minimal assumptions about the algorithms and data structures used to generate layer content to facilitate non-invasive integration into conventional image manipulation software.

One of our target applications is the generation of technical, medical, and scientific illustrations which frequently make use of selective occlusion overrides and blending in order to produce an expressive image. Using our approach, an illustrator can modify 3D properties of the content layers without having to go through the entire compositing process again. However, the presented approach is general and not restricted to this particular scenario. While recent work in illustrative visualization has put special emphasis on the role of methods employed by graphics artists and illustrators, many approaches are limited to specific types of scientific data (e.g., scalar volume data) and/or application domains. Although our approach can handle dynamically changing 3D layers, it does not require knowledge about the underlying rendering algorithms. This enables the flexible integration of different data representations such as polygonal meshes, volumetric-, and point-based data as well as different rendering algorithms such as painterly, photorealistic, or illustrative rendering.

The remainder of this paper is structured as follows: Section 2 reviews related work. In Section 3, we discuss the concepts behind our hybrid visibility compositing approach. Section 4 presents a technique for masking of dynamic 3D layers. Section 5 details our implementation and Section 6 presents further results. We discuss our approach in Section 7 and conclude the paper in Section 8.

☆ Funded by: FWF.
* Corresponding author. Tel.: +43 1 58801 18643; fax: +43 1 58801 18698.
E-mail address: bruckner@cg.tuwien.ac.at (S. Bruckner).

2. Related work

The work presented in this paper is related to several fields. Our approach is based on the large body of research in the area of digital compositing. Masking of 3D layers is related to image-based rendering techniques which attempt to generate novel views of 3D objects based on partial information. Furthermore, we were also inspired by illustrative and non-photorealistic rendering techniques which aim to reproduce the aesthetic quality of manually generated artwork using computer graphics.

Digital compositing: Digital compositing dates back to the early days of computer graphics as a scientific discipline. Alvy Ray Smith and Ed Catmull combined two images using a third image of coverage values which subsequently lead to the notion of the alpha channel [1]. Wallace [2] extended the approach to recursive blending allowing layers to be composited in any order that obeys associativity. Porter and Duff [3] introduced the concept of pre-multiplied alpha and formulated the compositing algebra which is in widespread use today. For the purpose of anti-aliased combination of 3D rendering results, Duff [4] proposed the rgbαz representation which, in addition to color and alpha channels, also includes a depth value for every pixel. Recent work by McCann and Pollard [5] extends the flexibility of traditional compositing by enabling stacking decisions on a per-overlap basis. However, their approach is designed to provide more control over the compositing of 2D layers, while the concept presented in this paper aims at facilitating the integration of 3D content in a consistent manner.

Image-based rendering: The idea of avoiding expensive rendering passes by using compositing to combine parts of a scene gave rise to the area of image-based rendering. Image-based rendering approaches attempt to synthesize novel views which closely approximate correct visibility from information captured during the generation of a single image. Lengyel and Snyder [6] proposed a factorization of 3D scenes into independent 2D sprites which could have different update rates. While their approach attempts to identify independent 2D layers, our method makes use of the available depth information and therefore allows full control over intersecting objects. Nailboards additionally store a depth value for every pixel of a sprite to enable the rendering of interpenetrating 3D objects with correct visibility [7]. Layered depth images contain multiple pixels along each line of sight to enable the generation of novel views with higher fidelity [8]. These approaches use layers with additional spatial information to combine parts of a scene with correct visibility. However, the focus of our work is different: we want to provide the user with the ability to selectively override occlusion relationships as it is common in technical, medical, and scientific illustrations.

Illustrative rendering: Several systems for the generation of illustrations using computer graphics have been developed. Dooley and Cohen [9,10] presented approaches for the automatic generation of semi-transparent line and surface illustrations from 3D models. Pioneering work by Seligman and Feiner [11–13] first treated the topic of visibility constraints. Their work employed cutaways and ghosting to resolve visibility conflicts. Preim et al. [14] presented Zoom Illustrator, a semi-interactive tool for illustrating anatomic models. Their approach focuses on the integration of three-dimensional graphics and textual representations. Diepstraten et al. [15,16] proposed rendering algorithms for ghosting and cutaway effects. Owada et al. [17,18] developed a system for modeling and illustrating volumetric objects. They semi-automatically generate artificial cutting textures based on surface models. Viola et al. [19] introduced the notion of importance-based rendering for improved visualization of features in volume data. Extending this approach, Bruckner and Gröller [20] presented a flexible interactive direct volume illustration system. Rautek et al. [21,22] proposed the use of semantic layers defined using a fuzzy logic rule base. Cole et al. [23] proposed a technique for generating architectural illustrations featuring a stylized focus area through local variations in shading effects and line qualities. Kalkofen et al. [24] used stylized overlays for focus+context visualization in augmented reality applications. Li et al. [25,26] presented geometric methods for generating high-quality cutaway and exploded view diagrams. Raman et al. [27] discussed a system which uses layer-based effects to enhance the visualization of volume data. Similar to our approach, the ClearView system presented by Krüger et al. [28] uses layered rendering to generate a number of different transparency effects inspired by traditional illustrations. However, their approach relies on a globally defined layer order, for instance nested isosurfaces of a volume dataset. Furthermore, their method only allows the use of a single spherical focus region.

In this paper we contribute with a new approach to combining interactively rendered 3D output based on the communication goals and stylization requirements of technical, medical and scientific illustrations. We introduce the notion of hybrid visibility compositing which allows integration of layered 2D compositing with 3D visibility operations in a flexible and intuitive manner. Additionally, we propose a new method for performing common masking operations based on this concept. The resulting framework enables the interactive generation of 3D illustrations featuring effects and techniques typically only available in 2D compositing software.

3. Compositing

Duff [4] was the first to propose the rgbαz representation for compositing 3D rendered images. In such a representation each pixel stores, in addition to its color and alpha value, a depth value. In a way, such an rgbαz image is a generalization of a 2D sprite [29]—points with color, transparency, and depth, but without any thickness information. Even though rgbαz layers are not a complete description of a general 3D object, they are a useful extension of conventional 2D layers. One of the main reasons why we choose this representation is that it requires minimal information about the actual data structures and algorithms used to provide layer content. A layer may be generated through ray tracing, rasterization of polygonal models, point-based rendering, or virtually any other technique capable of producing color and depth information.

We therefore choose this representation as one of the basic building blocks of our compositing framework. Each 3D layer is bound to a renderer instance and captures its output as an rgbαz image at any time. The content of such a layer may change dynamically, e.g., due to user interaction or animation. The compositing engine then decides how these layers are combined to form the final image. Since current graphics hardware allows us to easily access its color and depth buffers, one advantage of employing an rgbαz representation is that no modifications to the rendering stage are required. This means that the compositing engine can be used to combine layers produced by a variety of different rendering algorithms.

3.1. Implicit visibility

In contrast to 2D compositing where the stacking order of layers is solely specified by the user, rgbαz layers have an *implicit visibility* defined by the relative depth values of their pixels. The general technique for compositing multiple rgbαz layers with correct visibility is through a per-pixel application of the painter's

algorithm: for each pixel, the corresponding depth values of all layers are sorted and then blended together using the *over* operator, i.e., each layer overdraws the layers located behind to a degree specified by its alpha channel. While such a compositing algorithm permits the combination of many different rendering techniques, it does not provide the same level of flexibility as 2D compositing in which the user has full control over layer order and blending operators.

3.2. Explicit visibility

Another approach to compositing rgbαz layers is to employ *explicit visibility* by ignoring the per-pixel depth values and defining a stacking order in which blending operators are applied. This means that the layers are treated as flat images. Their operators are applied in the same order for all pixels. Employing explicit visibility for 3D content can be useful for creating illustrations when a particular layer should be emphasized by overlaying over layers depicting occluding structures. However, it also completely discards the additional information provided by the depth values.

3.3. Hybrid visibility

One of the main complications of implicit visibility compositing is that there is no consistent layer order. Using explicit visibility, layers can be moved in the stacking order to control which structures appear in front of each other and this relationship remains true for all pixels of an image. For implicit visibility, however, there is an inherent layer order which may be different for each pixel. Ignoring the depth information sacrifices all the advantages of 3D layers while relying on the implicit visibility severely limits the range of possible operations. In order to provide the user with a more intuitive interface based on familiar 2D compositing metaphors while preserving the ability to render with correct occlusion, we use a *hybrid visibility* approach which represents a flexible combination between implicit and explicit visibility.

As illustrated in Fig. 1, our framework allows the user to specify a stacking order for the input layers and group them hierarchically. Just like in conventional 2D approaches, each layer and group can be assigned a blending operator. Additionally, an optional layer mask, discussed in detail in Section 4, can be specified. Compositing is performed by traversing the layer hierarchy starting with the bottommost layer and blending the layers using their associated operator. The depth value of an intermediate image pixel always corresponds to the last layer which makes a visible contribution to it. For the purpose of integrating hybrid visibility into this familiar setup we provide a special set of blending operators which take into account spatial relationships.

Fig. 2 illustrates the advantages of hybrid visibility for the generation of illustrations. In the first row, a manually generated illustration of a sports car is depicted in the left column. The center column shows the implicit visibility of a similar 3D model. In the right column, four individual layers of the car (chassis, tires, interior, and details) are shown. The second row shows an example of explicit visibility using the following stacking order from bottom to top: chassis, tires, interior, details. Even though a result similar to the manual illustration can be generated by employing explicit visibility, it does not translate to other viewpoints. The third row depicts results generated using our hybrid visibility approach which allows us to closely mimic the essential features of the manually generated image. Interior and tires form a *visibility chain* which uses implicit visibility. The result is combined with the chassis and the details using *occlusion-based blending*. These concepts are discussed in detail in the following sections.

3.3.1. Visibility chains

A visibility chain is simply a group of layers where, for each pixel, compositing is performed with implicit visibility. It is specified using the *visibility* operator. The chain starts with a layer that has its operator set to *visibility* and ends with the first subsequent layer in stacking order which uses a different operator—this layer terminates the chain. The compositing result of the visibility chain is combined with the intermediate image using the operator specified for the terminating layer. Compositing then proceeds normally with the next layer. The advantage of visibility chains is that they allow groups of layers to exhibit correct occlusion relationships among themselves while still being embedded in the specified layer hierarchy.

For each pixel within a visibility chain, our algorithm first performs a depth sort of its input layers. Compositing is then performed by blending the individual layers in visibility order using the *over* operator. For additional control, we use a smooth distance-based weight similar to the blurred *z*-buffering approach proposed by Luft et al. [30]. The color rgbα used for compositing a layer L_i is a distance-weighted sum of the color of all layers in the visibility chain:

$$\mathrm{rgb}\alpha = L_i \cdot \alpha \frac{\sum_j (1-\Delta z(L_i \cdot z, L_j \cdot z, L_j \cdot \omega)) L_j \cdot rgb\alpha}{\sum_j (1-\Delta z(L_i \cdot z, L_j \cdot z, L_j \cdot \omega)) L_j \cdot \alpha} \quad (1)$$

Note that in Eq. (1) each layer's color is pre-multiplied by its alpha value and that the result will also be an opacity-weighted color. The function $\Delta z(z_0, z_1, \omega) \in [0,1]$ is a user-selectable function which controls the nature of the depth transition. We require the function to be monotonically increasing with the absolute difference between its first two arguments. The third argument $\omega \in [0,1]$ allows additional control over the particular shape of this function—increasing ω should lead to a sharper transition. Different types of such transition functions are possible, similar

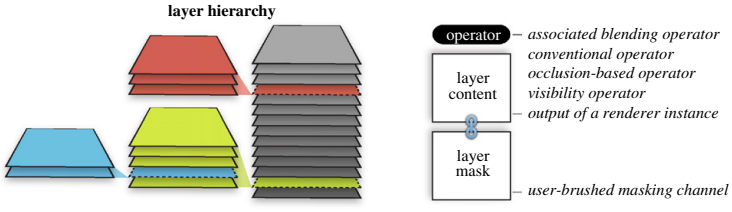


Fig. 1. Conceptual overview of our compositing setup. A layer definition is comprised of the layer content in form of an rgbαz image, an additional layer mask, and an associated blending operator. Layers are arranged in a hierarchical blending tree as exemplified in the figure: the upper highlighted layer in the stack consists of three sublayers and the lower highlighted layer consists of five sublayers one of which is itself composed of two sublayers.

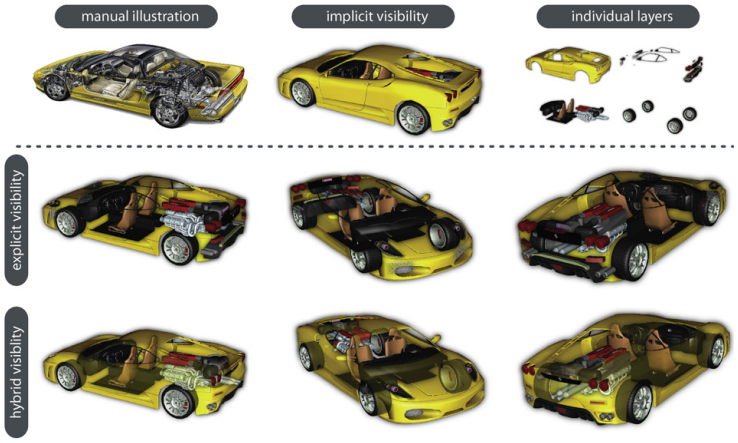


Fig. 2. Comparison of implicit, explicit, and hybrid visibility approaches to compositing. *Top row*: Left—manually generated illustration of a sports car. Center—implicit visibility of a similar 3D model. Right—four individual layers of the model. *Middle row*: Explicit visibility of the layers from three different viewpoints. *Bottom row*: Hybrid visibility of the layers from three different viewpoints. While implicit visibility alone does not capture the subtle effects used in the manual illustration, explicit visibility leads to distracting results when changing the viewpoint. Hybrid visibility avoids the drawbacks of both approaches. *Manual illustration courtesy of ©Kevin Hulsey Illustration, Inc.*

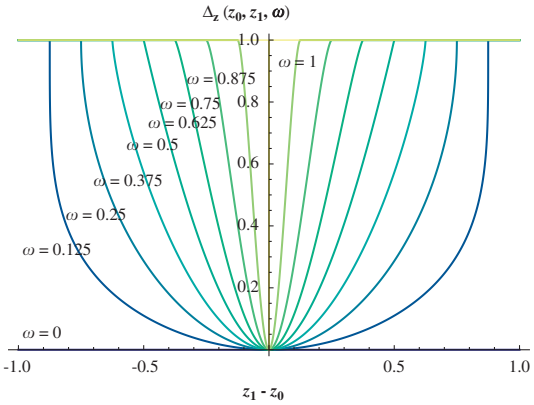


Fig. 3. Graphs of the Δz- function for different values of ω.

to easing curves in animation. In our current implementation, we use the following definition:

$$\Delta z(z_0,z_1,\omega) = 1-(\text{smoothstep}(\omega,1,1-|z_1-z_0|))^{\omega} \quad (2)$$

where smoothstep (a,b,x) is OpenGL's smoothstep function typically implemented as $u^2(3-2u)$ with $u = \text{clamp}(0,1,(x-a)/(b-a))$.

If $\omega = 0$, the value of Δz is always zero. For $\omega = 1$, the function value is zero only if $z_0 = z_1$ and one otherwise. Fig. 3 depicts graphs of the Δz-function for different values of ω.

If a pixel has the same depth in two layers, the resulting color will be the opacity-weighted average of the two layers' colors. Conversely, if the depth of the pixel in both layers is sufficiently different, their influence on each other will be zero. The user can control the distance weighting for each layer in a visibility chain by modifying its ω parameter. The approach can be used to effectively combat z-fighting, but it also offers an additional degree of artistic freedom. For instance, the weight may be altered on a per-layer basis to give a better indication of spatial relationships or to suggest the softness of a particular object. Fig. 4 shows an example. The teapot's body, handle, lid, and spout are rendered into separate layers and ω is globally set to 1, 0.5, 0.25, and 0.

3.3.2. Occlusion-based blending

In addition to the *visibility* operator, we provide a simple but powerful extension of conventional blending operators which allows them to make use of the additional spatial information. This includes the operators of the Porter–Duff algebra, such as *over*, *atop*, *in*, and *out*, as well as further operators typically present in image manipulation software (e.g., *multiply*, *screen*, or *overlay*). Our framework allows the use of all these operators in combination with a blending weight based on the distance between the layer's depth z and the current depth of the intermediate composite z_f. The layer's opacity is multiplied by the blending weight w_o which is computed by

$$w_o = \begin{cases} 1 & \text{if } \beta z < \beta z_f \\ 1-\Delta z(z,z_f,|\beta|) & \text{otherwise} \end{cases} \quad (3)$$

where $\beta \in [-1,1]$ is a user-controlled parameter of the operator. If β is zero (the default value), the operator will behave exactly like its two-dimensional counterpart. If $\beta > 0$, the parts in front of the intermediate image will be shown and parts behind it will decrease in opacity with increasing distance. Conversely, if $\beta < 0$ parts behind the current depth of the intermediate composite will be shown with full opacity and parts in front of it will decrease in opacity with increasing distance. This enables smooth fading of layers based on occlusion relationships. For instance, two layers containing different representations of the same object can be used to make it shine through an occluding layer with a different appearance. This effect is demonstrated in Fig. 10 where an occlusion-based *plus* operator is used to show an X-ray style representation of the hand where it is occluded by the lens of the magnifying glass.

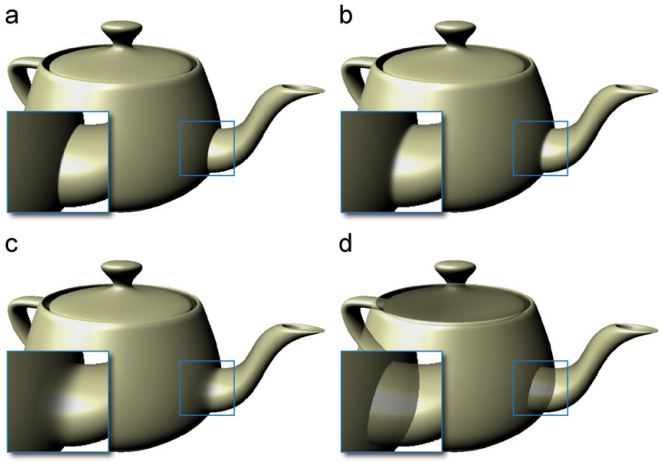


Fig. 4. Soft depth compositing using different values for ω: (a) $\omega = 1$; (b) $\omega = 0.5$; (c) $\omega = 0.25$; and (d) $\omega = 0$.

4. Masking

A common technique frequently employed for the compositing of 2D images is masking. A layer mask enables the artist to modify visibility independent of layer content. It can be utilized to make structures semi-transparent using smooth transitions, give selective emphasis to certain objects, or to remove unwanted parts. Common software packages feature flexible brushing tools to perform these manipulations (e.g., Adobe Photoshop's eraser).

When dealing with 3D layers which are generated on-the-fly and allow interactive modification of the viewpoint, however, the extension of such tools is not straight-forward. A purely 2D approach would be invariant to any camera changes and therefore frequently lead to undesired results when the viewpoint is modified. When we attempt to operate in object-space, on the other hand, we face the problem that a rgbαz layer is not a complete representation of a 3D object. The only 3D information available is the first visible surface of the object for the current viewpoint. While the renderer has complete information about the structure of the object, leaving the task of masking to each renderer would be prone to much duplication and potential inconsistencies as well as requiring modification of each rendering algorithm. For instance, a layer generated using volume rendering would need to handle masking operations in an entirely different manner than a layer generated by rendering polygonal geometry.

Our approach for masking represents a hybrid image-space/object-space approach which does not require additional information other than that provided by rgbαz layers. It can therefore be used for any kind of layer, irrespective of layer content. The user simply selects the target layer for the masking operation and can then paint on it to establish the mask. As soon as a stroke is placed by brushing over an area, the depth value of the layer is read. Together with the image-space position this gives us the 3D location of the stroke under the current viewing transformation. Applying the inverse viewing transformation transforms the stroke location into object space. This position, together with the current brush settings, is stored in a list associated with the selected layer. Whenever a layer has been updated by the corresponding renderer (e.g., due to a change of the viewing transformation) the list of strokes for the layer is traversed and rendered using splatting [31]. Each stroke is rasterized as a view-aligned impostor under the current viewing transformation. For each fragment of the stroke, we now have its intensity i and depth z. The intensity i is determined by the brush parameters and can be, for example, a 2D Gaussian with its peak centered at the stroke's image-space position. The depth z is simply the depth of the impostor fragment.

Then, for each fragment of the stroke, the depth z_l of the layer at the fragment's position is read. As this value is the first intersection of the viewing ray with the three-dimensional object represented by the layer, we can use it to estimate how much influence this fragment of the stroke should have for the current viewpoint. For instance, if the surface point we originally placed our stroke on is now occluded by another part of the surface, the difference between z and z_l will be high. Conversely, if the same point on the surface we placed the stroke on is still visible in the novel viewpoint, the difference will be zero at that location. Fig. 5 illustrates this behavior. It depicts two stroke centers rendered from two different viewpoints. From view 1, both stroke centers lie on the surface, i.e., $z = z_l$. For view 2, stroke 2 still lies on the visible surface. The position of stroke 1, however, is occluded by another part of the object, i.e., the difference between z and z_l is large.

As we want our strokes to vary smoothly in intensity when the view is changed, we choose to modulate the stroke intensity i using a weight w_b based on the difference between z and z_l. This weight is computed using the previously discussed Δz-function:

$$w_b = 1 - \Delta z(z, z_l, \gamma) \tag{4}$$

where $\gamma \in [0,1]$ is a user-controlled parameter of the brush. An intuitive feature of this approach is that the brush is sensitive to the properties of the visible surface. If a large brush is chosen the rendered imposter will be a large flat disc centered at the stroke position. As the distance between the disc's depth and the surface depth modifies the brush intensity, depth discontinuities will tend to be preserved. Since we store brush strokes using a point-based representation there are other advantages of our approach: as the brush strokes are rasterized for every novel view, aliasing is avoided. Furthermore, parameters such as brush intensity, size, or shape can be modified after the strokes have been placed.

Typically, the masking channel is used to modulate layer opacity, i.e., it is multiplied with the α value of the corresponding layer pixel. To enable further control over the effect of masking,

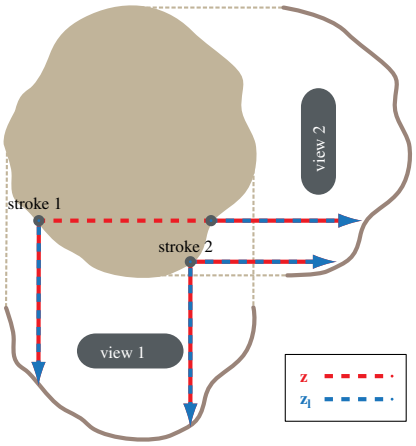


Fig. 5. Example of distance-based weighting for brush strokes. Two brush strokes (stroke 1, stroke 2) generated from the viewpoint view 1 are shown. In view 1, $z = z_l$ for both strokes, i.e., both strokes receive the maximum weight. When a novel viewpoint (view 2) is chosen, stroke 1 has $z \neq z_l$ due to occlusion, i.e., it receives a lower weight, while stroke 2 remains visible.

we provide an additional parameter s in the range $[-1,1]$ which the user can modify independently for each layer. If $s \geq 0$, the α value for each pixel is additionally multiplied by $1-si$, where i is the brushed intensity for the layer pixel. If s is negative, α is multiplied by $1-si+s$. If s is positive, the layer becomes more transparent with higher brush intensity. Negative values of s invert the effect: the layer is transparent where the brush intensity is zero and becomes more opaque with increasing values. Setting s to zero disables any effect of masking.

5. Implementation

The techniques described in this paper were implemented as an extension to an existing rendering framework written in C++ and OpenGL/GLSL. In order to make use of all available renderers of the framework, the basic display routine was modified to supply each renderer instance (which represents a layer) with an offscreen buffer for color and depth instead of the visible framebuffer. This is possible using the EXT_framebuffer_object OpenGL extension. For the renderers, this change was completely transparent—the rendering code did not require any changes. In fact, the framework now allows the compositing engine to be switched at runtime. The offscreen buffers are allocated as an array texture (an array of 2D textures which can be accessed similar to a 3D texture) provided by the EXT_texture_array extension. When a renderer instance needs to update itself, it simply renders a new image into its offscreen buffer–all other images are left unmodified. This also means that when a user interaction occurs, only those renderers which are affected by the change need to re-execute.

The compositing algorithm then uses the current values stored in the array texture. It is executed for every frame. The significant improvements concerning conditionals and loops introduced by the latest generation of graphics hardware allowed us to implement the whole compositing algorithm in a single-pass fragment program. This fragment program first reads colors, depth values, and masking information for every layer. Completely transparent pixels can be culled during this early stage which serves as a great source for performance improvement as they do not have to be considered in the subsequent steps. Compositing is then performed corresponding to the specified stacking order and grouping hierarchy. For visibility chains, depth sorting is performed in the fragment program. An additional offscreen buffer is kept for each layer where masking has been applied. When the appropriate input event occurs, the current masking parameters together with the determined object-space stroke locations are stored in a list associated with the selected layer. Every time a renderer has updated itself, this information is used to execute the algorithm described in Section 4. For rendering the strokes as imposters, we employ OpenGL's ARB_point_sprite extension which allows for textured as well as analytically defined brush tips. One advantage of having the result of several renderers available as separate layers is that different effects can be applied selectively. Again, we draw inspiration from 2D image manipulation software which offers a wide variety of layer effects. As our layers also store depth information even more options are available. Our framework provides a flexible interface for integrating these effects. For example, we employ the depth-based image enhancement approach presented by Luft et al. [32] which has proven to be a natural extension of common two-dimensional glow or drop shadow filters.

Although the current implementation of our compositing algorithm was not optimized for performance, it performs at frame rates above 20 frames/s for up to eight layers with a window size of 800×600 pixels on a GeForce 8800 GTX GPU. The overall performance is heavily dependent on the algorithms and models used to generate the layer content. For all the results shown in this paper the frame rate was over 5 frames/s for re-rendering all layers and effects. As the modification of a layer mask only requires a re-execution of the compositing pass, it is independent of layer content.

6. Results

In order to evaluate the utility of the presented techniques, we consulted a professional medical illustrator with over 25 years of experience in the field. We attempted to recreate effects and techniques commonly found in scientific and technical illustrations using 3D models.

The illustration shown in Fig. 6(a) depicts the female reproductive system. The purpose of the illustration was to clearly show internal reproductive organs while indicating their placement within the body. The illustrator used 2D renditions of the individual elements which were combined in Adobe Photoshop. First, the body contours were placed on the bottommost layer and a drop shadow was added to lift the image off the background. The pelvis was then added as a second layer, its opacity was lowered, and a drop shadow filter was applied. Additionally, a mask layer was added to preserve the contour of the body around the genital area. Reproductive organs were added as a third layer and a layer mask was employed to lower the opacity of the uterus as it passes behind the pelvis. Instead of completely masking out the structures behind the pubic symphysis, the artist chose to keep this area slightly visible while still indicating to the viewer that these regions are located behind the pelvis.

Using our approach the process of creating a similar illustration, as shown in Fig. 6(b), is analogous. Given a suitable 3D model, the user assigns the respective objects to individual layers. The same three layers are used: body contours, pelvis, and internal reproductive organs. The opacity of the pelvis layer is adjusted and our masking tool is applied, just like in the 2D workflow. However, instead of manually specifying a mask to achieve the desired see-through effect for the internal structures, pelvis and reproductive organs simply form a visibility chain

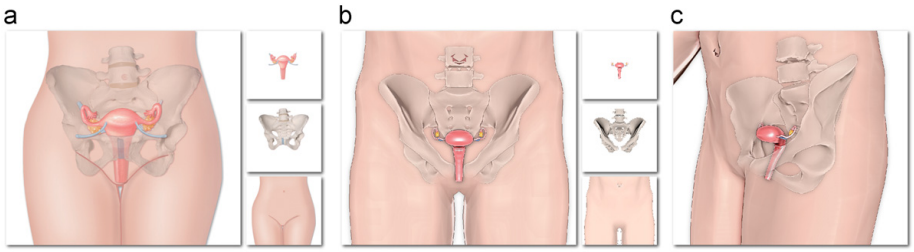


Fig. 6. Female reproductive system: (a) 2D illustration generated using Adobe Photoshop; (b) 3D illustration generated using our compositing approach; and (c) different viewpoint of the 3D illustration. *Illustrations courtesy of ©Kari C. Toverud, MS, CMI.*

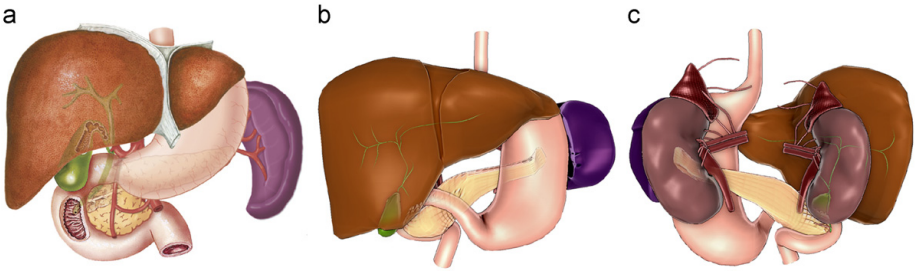


Fig. 7. Upper gastrointestinal tract: (a) 2D illustration generated using Adobe Photoshop; (b) 3D illustration generated using our compositing approach; and (c) different viewpoint of the 3D illustration. *Illustrations courtesy of ©Kari C. Toverud, MS, CMI.*

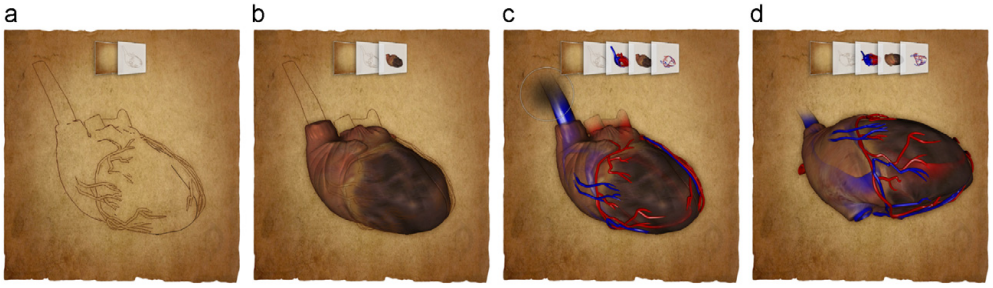


Fig. 8. Generation of a 3D illustration of the human heart: (a) line drawing layer; (b) combination with muscle layer; (c) applying masking; and (d) after rotation.

which is combined with the body layer using the *over* operator. This enables the generation of novel views without requiring any changes, as shown in Fig. 6(c). Fig. 7 depicts a further example of how our approach allows the user to employ the same effects as in 2D illustrations (see Fig. 7(a) and (b)), but enables the easy generation of novel views with the same hybrid visibility order. Additional illustrations showing the same topic are therefore easily created, as demonstrated in Fig. 7(c).

In Fig. 8(a), we show an example of a human heart model rendered as a line drawing. Then, in Fig. 8(b), a layer depicting the pericardium (heart muscle) is added. In Fig. 8(c), additional layers depicting arterial and venous system are enabled. As no visibility overrides are required, all heart layers form a visibility chain which terminates with the *over* operator. The ω parameter of the pericardium is adjusted to make the inner structures of the heart close to the surface shine through. Masking is then applied to the line drawing and vascular layers making them fade into the background. Fig. 8(d) demonstrates that the brushed mask smoothly translates to other viewpoints.

Fig. 9 depicts an illustration of the human vascular system. There are five layers: background, skin, skeleton, arterial system, and venous system. The background layer resides at the bottom, the skin layer uses the *over* operator. Skeleton, arteries, and veins form a visibility chain. The vein layer terminates the chain using the *over* operator. Masking has been applied to make these layers smoothly fade into the skin. Additionally, the ω parameter of the skeleton layer has been adjusted to show blood vessels passing closely behind bones.

The illustration depicted in Fig. 10 demonstrates that our approach can also be used to easily generate interactive effects such as magic lenses. This setup contains two layers generated by volume rendering of a human hand CT dataset. The first one uses non-photorealistic isosurface rendering of the skin while the second one uses maximum intensity projection to achieve an X-ray effect. The magnifying glass model is split into two layers: lens and body. The X-ray layer resides in a separate layer group with the lens and uses the *plus* operator and a negative β parameter so only the parts of the layer located behind the lens are added. The result forms a

visibility chain with skin and body which is combined with the background layer using the *over* operator.

7. Discussion

One goal of the work discussed in this paper was to provide a practical way of incorporating illustrative rendering techniques into the workflow of illustrators and artists. Many high-quality illustrative techniques have been presented in recent years. However, these methods frequently rely on particular data structures and algorithms which makes their integration into professional software tools difficult. While it is possible to generate many of the effects presented in this paper using specialized algorithms, our contribution is a general concept which allows seamless integration of 3D layers into 2D software. Traditional compositing tools see the increasing need to provide 3D integration. The demand for this kind of functionality is evidenced by its recent incorporation into widely popular applications such as Adobe Photoshop. However, in current implementations 3D layers behave as 2D images with respect to other layers—there is no way to make use of the visibility information between two 3D layers. Using our approach, this functionality could be greatly extended in a non-invasive manner while still covering the full range of 2D operations.

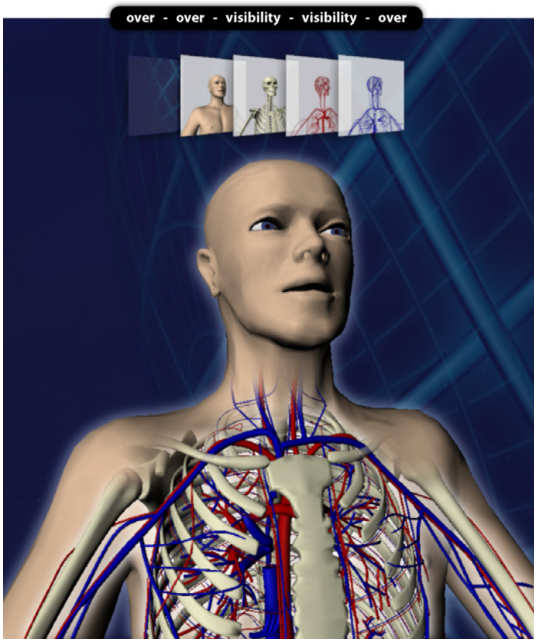


Fig. 9. Interactively generated 3D illustration of the human vascular system.

In a 2D workflow, artists frequently employ manually drawn layer masks to emulate visibility information for the generation of see-through and ghosting effects. Visibility chains and occlusion-based blending have shown to be effective tools to reduce the number and complexity of manually specified layer masks by taking advantage of the existing spatial information. Based on the artist's intent, however, additional masking is still useful in many cases. Our hybrid image-space/object-space method for brushing layer masks closes this gap by also exploiting spatial information for this operation. During experiments we found that our masking approach is very similar in behavior to analogous tools in 2D applications. One has the impression to be manipulating an image, but masking information smoothly transfers to nearby viewpoints in a consistent manner. However, it is impossible to predict the intent of the user in all cases. For example, if a user paints on one side of a radially symmetric object, it might be desirable to automatically have the object appear transparent from all viewing directions along the axis of symmetry. As our system is completely interactive, these cases can be easily resolved by rotating the object and placing new strokes. Our general approach also allows easy integration of additional specialized tools for this purpose.

We received positive feedback on the utility of our prototype implementation and the general concept of hybrid visibility for generating illustrations. As shown in the examples in Section 6, our approach is capable of closely mimicking the traditional 2D compositing workflow. The ability of being able to alter an

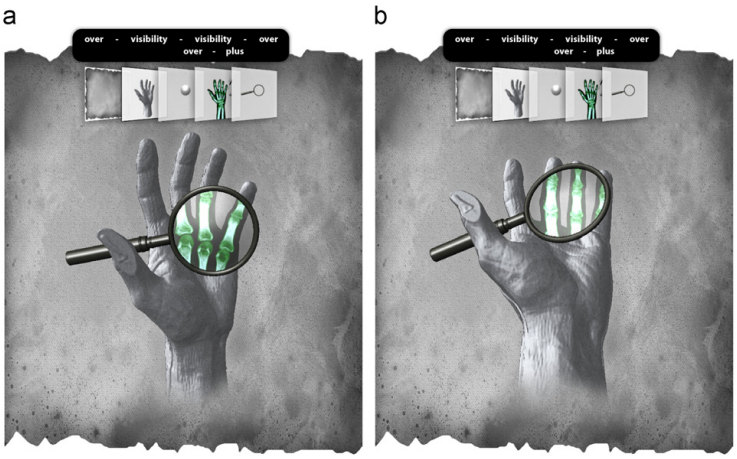


Fig. 10. Two viewpoints for a magic lens effect generated using our compositing framework.

existing illustration by modifying the viewpoint was greatly appreciated and, given the availability of suitable 3D models, considered to have a high potential of speeding up the production process. However, as our research prototype does not encompass the full range of functionality featured in standard software packages, only the integration of the proposed concepts into a commercial product is likely to facilitate widespread adoption.

8. Conclusion

In this paper we presented a simple concept for illustrative compositing of dynamic 3D layers in an interactive environment. Our approach enables a wide variety of different effects such as selective occlusion overrides commonly employed in the generation of scientific and technical illustrations. With our method these operations can be performed in 3D using an extension of the familiar layer metaphor. We also proposed a novel technique for masking of 3D layers. It enables the generation of opacity transitions which smoothly extend beyond a single viewpoint. The presented framework makes minimal assumptions about the underlying algorithms used for rendering the individual layers. By exploiting the performance of current graphics hardware, high-quality illustrations of 3D objects can be generated interactively.

Acknowledgments

The work presented in this publication was supported by the Austrian Science Fund (FWF) Grant no. P21695—ViMaL: The Visualization Mapping Language, the Norwegian Research Council (Project no. 193170/S10) and by the MedViz Initiative in Bergen, as well as the iCORE/Foundation CMG Industrial Research Chair in Scalable Reservoir Visualization and the Discovery Grants Program from the Natural Sciences and Engineering Research Council of Canada.

We want to thank CF Lietzau 3D Special (http://www.anatomium.com) for permission to use the P1 human anatomy model depicted in Figs. 6–9. Furthermore, we want to express our gratitude to Kari C. Toverud for providing her time and expertise.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version of 10.1016/j.cag.2010.04.003.

References

[1] Smith AR. Alpha and the history of digital compositing. Technical Memo 7, Microsoft Corporation, Redmond, Washington.
[2] Wallace BA. Merging and transformation of raster images for cartoon animation. ACM Computer Graphics 1981;15(3):253–62.
[3] Porter T, Duff T. Compositing digital images. ACM Computer Graphics 1984;18(3):253–9.
[4] Duff T. Compositing 3-D rendered images. ACM Computer Graphics 1985; 19(3):41–4.
[5] McCann J, Pollard NS. Local layering. ACM Transactions on Graphics 2009;28(3):1–7.
[6] Lengyel J, Snyder J. Rendering with coherent layers. In: Proceedings of ACM SIGGRAPH '97, 1997. p. 233–42.
[7] Schaufler G. Nailboards: a rendering primitive for image caching in dynamic scenes. In: Proceedings of the Eurographics Workshop on Rendering '97, 1997. p. 151–62.
[8] Shade J, Gortler S, He L-W, Szeliski R. Layered depth images. In: Proceedings of ACM SIGGRAPH '98, 1998. p. 231–42.
[9] Dooley D, Cohen MF. Automatic illustration of 3D geometric models: lines. In: Proceedings of the symposium on interactive 3D graphics '90, 1990. p. 77–82.
[10] Dooley D, Cohen MF. Automatic illustration of 3D geometric models: surfaces. In: Proceedings of IEEE Visualization '90, 1990. p. 307–14.
[11] Seligmann DD, Feiner SK. Automated generation of intent-based 3D illustrations. In: Proceedings of ACM SIGGRAPH '91, 1991. p. 123–32.
[12] Feiner SK, Seligmann DD. Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. The Visual Computer 1992;8(5 & 6): 292–302.
[13] Seligmann DD, Feiner SK. Supporting interactivity in automated 3D illustrations. In: Proceedings of the international conference on intelligent user interfaces '93, 1993. p. 37–44.
[14] Preim B, Ritter A, Strothotte T. Illustrating anatomic models—a semi-interactive approach. In: Proceedings of the international conference on visualization in biomedical computing '96, 1996. p. 23–32.
[15] Diepstraten J, Weiskopf D, Ertl T. Transparency in interactive technical illustrations. Computer Graphics Forum 2002;21(3):317–25.
[16] Diepstraten J, Weiskopf D, Ertl T. Interactive cutaway illustrations. Computer Graphics Forum 2002;22(3):523–32.
[17] Owada S, Nielsen F, Nakazawa K, Igarashi T. A sketching interface for modeling the internal structures of 3D shapes. In: Proceedings of the international symposium on smart graphics '03, 2003. p. 49–57.
[18] Owada S, Nielsen F, Okabe M, Igarashi T. Volumetric illustration: designing 3D models with internal textures. ACM Transactions on Graphics 2004;23(3): 322–8.
[19] Viola I, Kanitsar A, Gröller ME. Importance-driven feature enhancement in volume visualization. IEEE Transactions on Visualization and Computer Graphics 2005;11(4):408–18.
[20] Bruckner S, Gröller ME. VolumeShop: an interactive system for direct volume illustration. In: Proceedings of IEEE visualization '05, 2005. p. 671–8.
[21] Rautek P, Bruckner S, Gröller ME. Semantic layers for illustrative volume rendering. IEEE Transactions on Visualization and Computer Graphics 2007; 13(6):1336–43.
[22] Rautek P, Bruckner S, Gröller ME. Interaction-dependent semantics for illustrative volume rendering. Computer Graphics Forum 2008;27(3): 847–54.
[23] Cole F, DeCarlo D, Finkelstein A, Kin K, Morley K, Santella A. Directing gaze in 3D models with stylized focus. In: Proceedings of the eurographics symposium on rendering '06, 2006. p. 377–87.
[24] Kalkofen D, Mendez E, Schmalstieg D. Interactive focus and context visualization for augmented reality. In: Proceedings of the IEEE international symposium on mixed and augmented reality '07, 2007. p. 1–10.
[25] Li W, Ritter L, Agrawala M, Curless B, Salesin D. Interactive cutaway illustrations of complex 3D models. ACM Transactions on Graphics 2007; 26(3):3:11–31:11.
[26] Li W, Agrawala M, Curless B, Salesin D. Automated generation of interactive 3D exploded view diagrams. ACM Transactions on Graphics 2008;27(3): 1–7.
[27] Raman S, Mishchenko O, Crawfis R. Layers for effective volume rendering. In: Proceedings of the international symposium on volume and point-based graphics '08, 2008. p. 81–8.
[28] Krüger J, Schneider J, Westermann R. Clearview: an interactive context preserving hotspot visualization technique. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):941–8.
[29] Smith AR. A sprite theory of image computing. Technical Memo 5, Microsoft Corporation, Redmond, Washington.
[30] Luft T, Deussen O. Real-time watercolor illustrations of plants using a blurred depth test. In: Proceedings of the international symposium on non-photorealistic animation and rendering '06, 2006. p. 11–20.
[31] Westover L. Footprint evaluation for volume rendering. In: Proceedings of ACM SIGGRAPH '90, 1990. p. 367–76.
[32] Luft T, Colditz C, Deussen O. Image enhancement by unsharp masking the depth buffer. ACM Transactions on Graphics 2006;25(3):1206–13.

10 Unified Boundary-Aware Texturing for Interactive Volume Rendering

In this paper we describe a novel approach for applying texture mapping to volumetric data sets. In contrast to previous approaches, the presented technique enables a unified integration of 2D and 3D textures and thus allows to emphasize material boundaries as well as volumetric regions within a volumetric data set at the same time. One key contribution of this paper is a parametrization technique for volumetric data sets, which takes into account material boundaries and volumetric regions. Using this technique, the resulting parametrizations of volumetric data sets enable texturing effects which create a higher degree of realism in volume rendered images. We evaluate the quality of the parametrization and demonstrate the usefulness of the proposed concepts by combining volumetric texturing with volumetric lighting models to generate photorealistic volume renderings. Furthermore, we show the applicability in the area of illustrative visualization.

The following paper appears in its original format published as:

T. Ropinski, S. Diepenbrock, S. Bruckner, K. Hinrichs, and M. E. Gröller. Unified Boundary-Aware Texturing for Interactive Volume Rendering. To appear in *IEEE Transactions on Visualization and Computer Graphics*, 2012.

Unified Boundary-Aware Texturing for Interactive Volume Rendering

Timo Ropinski, *Member, IEEE,* Stefan Diepenbrock, Stefan Bruckner, Klaus Hinrichs, *Member, IEEE* and Eduard Gröller, *Member, IEEE*

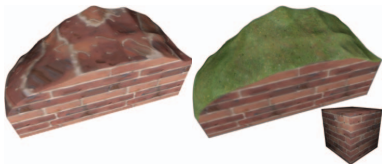
Abstract—In this paper we describe a novel approach for applying texture mapping to volumetric data sets. In contrast to previous approaches, the presented technique enables a unified integration of 2D and 3D textures and thus allows to emphasize material boundaries as well as volumetric regions within a volumetric data set at the same time. One key contribution of this paper is a parametrization technique for volumetric data sets, which takes into account material boundaries and volumetric regions. Using this technique, the resulting parametrizations of volumetric data sets enable texturing effects which create a higher degree of realism in volume rendered images. We evaluate the quality of the parametrization and demonstrate the usefulness of the proposed concepts by combining volumetric texturing with volumetric lighting models to generate photorealistic volume renderings. Furthermore, we show the applicability in the area of illustrative visualization.

Index Terms—volumetric texturing, interactive volume rendering.

1 Introduction

In the past years, much effort has been undertaken in the field of volume rendering to generate more compelling images. Many of the current advances could be achieved by transferring or adopting techniques which have been proven useful when rendering polygonal models. Texture mapping, however, which is heavily used when rendering polygonal data, has received only limited consideration in the area of volume rendering. Only little effort has been undertaken so far to unleash the full potential of texture mapping in the context of volume graphics. One commonly used technique assigns 3D textures to a volumetric data set according to the current voxel's position (e. g., Lu et al. [38] or Satherley and Jones [50]). While 3D texturing is sufficient for altering the overall appearance of an object, it is not suitable for controlling the display of surface details. As illustrated in Figure 1, 3D textures can be assigned to volumetric regions and give the impression that the object has been carved out of a block of material (a), while 2D textures can be assigned to surfaces in order to display information on material boundaries (b). Since volumetric regions and material boundaries are both considered as equally important features within a volumetric data set [25], [33], in many cases a unified 2D and 3D texturing approach is desired.

In this paper we present a volumetric parametrization model as well as an interactive rendering approach in order to allow a unified integration of 2D and 3D texture mapping into the volume rendering process. By enabling 2D texturing, the whole spectrum of 2D texturing effects as known from polygonal rendering can be used for volume rendering (see Figure 2). To do so, we had two face two challenges. First, a meaningful parametrization of volumetric objects has to be found. While useful 2D surface parametrization algorithms have been developed (e. g., see Floater and Hormann [14]), almost no efforts have been undertaken to parametrize volumetric data sets. Since in most cases, users are not only interested in the volumetric nature given by homogeneous regions, but also in material boundaries, a straightforward parametrization neglecting the topology and only considering the voxel coordinates would not be sufficient. While the overall structure of material boundaries contained in a volumetric data set is given by the data, the actual position may shift depending on the selected rendering parameters, e. g., the transfer function. Thus, a parametrization has to be appropriate for the potentially shifting surfaces of interest and still has to be meaningful for nearby structures. The second major challenge when texture mapping volumetric data sets is the actual rendering. In contrast to polygonal models, volume objects are composed of several nested layers. Each layer may have its own texture. Sometimes, it may be desirable to



(a) (b)

Fig. 1. A volume data set rendered with a 3D brick texture (a), and with an additional 2D grass texture (b). While 3D texturing results in a carved-out effect, 2D texturing can be used to depict surface details.

- *T. Ropinski is with the University of Linköping, E-mail: timo.ropinski@liu.se.*
- *S. Diepenbrock and K. Hinrichs are with the University of Münster, E-mail: {diepenbrock|khh}@math.uni-muenster.de.*
- *Stefan Bruckner and Eduard Gröller are with the Technical University Vienna, E-mail: {bruckner|groeller}@cg.tuwien.ac.at.*

interpolate between the textures assigned to these layers, and sometimes, when distinct material boundaries are preferred, no interpolation is necessary.

In this paper we present a novel concept for the parametrization of volumetric data sets of genus 0. When exploiting the resulting parametrization, the unified application of 2D and 3D texturing at the same time becomes possible. Our approach is different in spirit to existing approaches, which aim at local parametrizations of specific surfaces within a volume data set [4], [48] or exploit two-part mapping techniques [1], [54]. It is, to the best of our knowledge, the first boundary-aware method which has been explicitly developed for real-world volumetric data sets and therefore allows to entirely capture their volumetric nature. Throughout this paper, we refer to a boundary as the interface between two volumetric materials, which can be either extracted through segmentation or classification, e.g., by using 2D transfer functions. To support interactive unified 2D and 3D texture mapping for volumetric objects, we additionally propose GPU-based rendering techniques which exploit the features of current graphics hardware. By using GPU-friendly data structures and access functions, we are able to assign multiple layers to each parametrized object interactively. We will show that due to the interactive frame rates it becomes possible to apply interaction techniques working with textures, e. g., sculpting and carving.

Due to the simplicity of the described rendering techniques, the proposed concepts can be integrated into existing volume rendering pipelines and thus open up new avenues in the quest for illustrative as well as photorealistic volume graphics. At this point we would like to reemphasize that by exploiting the proposed algorithms, we are able to transfer most of the concepts known from surface texturing to volumetric objects without requiring an intermediate polygonal surface extraction. Hence, all renderings shown throughout this paper are volume renderings, and no tessellations have been generated except for comparison reasons.

2 Related Work

Volumetric texturing so far has only been applied to volume rendering by using 3D textures or by exploiting local parametrizations, i. e., only selected surfaces in a data set are partially parametrized. Satherly and Jones applied hypertexturing to volumetric data sets [50]. Miller and Jones have extended these concepts and realized hypertexturing of volumetric objects in real-time by exploiting modern graphics hardware [41]. They perform isosurface rendering while applying texturing or hypertexturing. Shen and Willis proposed a data-dependent triangulation method combined with a two-part mapping in order to map 2D textures to isosurfaces extracted from volumetric data [54]. They also addressed how to use 2D textures in volume rendering. Instead of exploiting the layered nature for more advanced rendering effects, they use an interpolation of the texture in-between two isosurfaces. In a follow-up paper Shen and Willis described how to interactively position textures on the extracted surfaces [53].

Fig. 2. Application of different texture mapping techniques to a volumetric cylinder data set (a). Color mapping in combination with Phong shading (b), bump mapping (c), bump mapping and color mapping in combination with specular lighting (d), displacement mapping with diffuse shading (e), displacement mapping with color mapping and bump mapping in combination with specular lighting (f) and the application of tonal art maps to achieve a non-photorealistic effect (g).

Some papers describe how to exploit texturing in the context of volume rendering to achieve non-photorealistic rendering effects. Treavett and Chen have proposed a technique based on mathematically derived, non-photorealistic solid textures [58]. Baer et al. adopted a cube-map parametrization approach to allow non-photorealistic volume rendering through stippling textures [1]. A similar approach is the TileTree data structure, which employs an octree to manage the texture to be mapped [32]. Bruckner and Gröller used style transfer functions based on sphere maps for illustrative volume rendering [3], which allows stylized lighting. Similar to Treavett and Chen [58], Lu et al. used synthesized 3D textures based on Wang Cubes to achieve an illustrative style when rendering volume data [38]. A similar approach was employed by Kabul et al. [23], who generate anisotropic solid textures based on 2D examples. While these approaches allow to transfer the overall appearance from medical text books, they do not allow to integrate 2D textures to depict surface details. The same is true for the texture transfer-function approach presented by Manke and Wuensche, which allows to assign 3D textures to intensity ranges within a volumetric data set [39], or the work by Dong and Clapworthy which attempts to orient anisotropic structures along the main object axis [10].

Volume parametrization is necessary to allow the same richness of texturing effects as known from polygonal rendering. Kurzion et al. [30] developed an approach for mapping textures to volumetric isosurfaces and parametric surfaces. Another recent approach uses tri-planar texture mapping to map textures onto surfaces found in volumetric data sets [29]. Besides other techniques directly developed for polygonal rendering [19], Zwicker et al. proposed a parametrization for point-based models [63]. In recent years, concepts have been proposed which explicitly address the parametrization of volumetric data. Patel et al. [44] developed a parametrization for seismic data sets, which allows to apply 2D textures to illustrate seismic horizons. In a follow-up paper, they improved their technique to allow more sophisticated texturing effects on seismic slices [43]. To map textures with textual annotations onto boundary materials within a volumetric data set, Ropinski et al. proposed an image-based technique exploiting Bézier patches [48]. This approach as well as the interactive volume editing technique presented by Bürger et al. [4] only provides a local parametrization. Li et al. presented harmonic volumetric mapping, which establishes a

bijective correspondence of two solid shapes having the same topology [35], [36]. They also demonstrated the use of their technique for 3D texture synthesis. However, they assume that a $< u,v >$ parametrization of the outer surface is already given, which is propagated towards the interior. Unlike our approach, their method does not consider multiple surfaces of interest contained in a volumetric data set and is therefore not boundary-aware. In a recent extension [37], they showed how to compute the correspondence between two given objects of the same topology by considering multiple surfaces, which is more efficient and accurate and supports adaptive refinement. While this technique is similar in spirit to our approach, the resulting parametrizations are not suitable for the texturing effects presented in this paper. Since our approach is a true volumetric algorithm which is based on volumetric cuts, we are able to generate parametrizations with consistent $< u,v >$ mappings. In contrast, Li et al.'s approach expects decoupled $< u,v >$ parametrizations as input for all considered surfaces and thus does not support a $< u,v >$ *synchronization* which is essential for many texturing effects, e. g., the interactive cutaways shown in Figure 10. Ju et al. [22] and Martin et al. [40] presented more general approaches which interpolate information in the interior of closed triangular meshes. Owada et al. [42] introduced a technique which allows to synthesize textures for the interiors of polygonal models. In contrast to our approach their focus is mainly on the texture synthesis and the user interface. A similar technique was presented by Pietroni et al., who synthesize internal textures for polygonal objects based on special cross-sectional input photographs [45].

Our parametrization technique is motivated by the layered structure exhibited by many objects. This observation has also been exploited in the context of polygonal techniques. Cutler et al. presented a scripting-based modeling approach for generating layered objects [8]. The shell map approach allows to obtain a parametrization of a thin layer around the surface of a polygonal object by generating a tetrahedral mesh, which is parametrized using barycentric coordinates [46]. Zhou et al. proposed an alternative low-distortion shell parametrization allowing mesh quilting, which is based on a surface aligned mesh synthesis [62]. Takayama et al. presented the concept of lapped volumetric textures, where 3D textures are mapped to a mesh to capture the outer and the inner appearance [57]. In contrast to our approach, the technique utilizes a tetrahedral mesh and relies on 3D textures as input. Although these can be synthesized with recent approaches [28], [11], [56], the fine-grain control of surface details on different boundaries is not possible with lapped volumetric textures.

3 Parametrizing Volume Data Sets

In this section, we present the concepts necessary to achieve a boundary-aware volume parametrization, which allows unified 2D and 3D texturing. Since nowadays many volume data sets are acquired using medical scanners, the presented parametrization model is inspired by anatomical models found in medicine. Many organs and tissues as described in anatomic atlases consist of several nested layers [18]. Human skin, for instance, consists of the hypodermis, dermis and epidermis layers. For an extremity such as the arm, the existing layers are nested around the corresponding bone. Another example for the layer concept found in anatomy is the bone structure. Each bone consists of several layers – the endosteum, different lamellae layers and the periosteum – which are nested around the bone marrow. Similar analogies can be found for many other natural objects. As shown in Figure 3, the nested layers of an object of genus zero are arranged in the $< u,v,w >$ parameter space, where the $< u,v >$ coordinates correspond to the surface parametrization of this layer, while the w coordinate depicts the layer's *depth*. Obviously, it is not possible to generate a distortion-free parametrization for all parts of the object. Instead, the goal is to focus on features of interest, which are given either automatically by material boundaries [25], [33] or which are extracted manually through segmentation. Thus, it becomes possible to generate a global parametrization, which is optimized for these features, while still achieving meaningful results for the remaining parts of the object. The latter is important, since in volume rendering parameter changes can have a drastic influence on the visualized structures. When for instance changing the transfer function or the iso-value, these changes may result in shifting boundaries. It is not possible to consider all these boundaries during parametrization, and just computing parametrizations for a selected set of isosurfaces would not be sufficient. Within this paper we refer to a global volume parametrization as a parametrization which assigns 3D texture coordinates to each voxel within the volume data set. Such a parametrization preserves the volumetric nature of a data set and thus also allows to incorporate homogeneous regions. There are three major differences, when comparing a volumetric parametrization to a surface parametrization. First, in volume data no knowledge about the connectivity along a surface is present, and thus coherence is harder to achieve. Second, in contrast to polygonal rendering, surfaces in volume data are not fixed. They may change depending on the current rendering parameters. Third, volumetric cuts are needed, which penetrate the whole volume in order to be able to map it into the cube-shaped parameter space. For polygonal models sophisticated algorithms for surface cutting exist, e. g., the seamster algorithm [52], but no algorithms have been proposed yet for generating volumetric cuts.

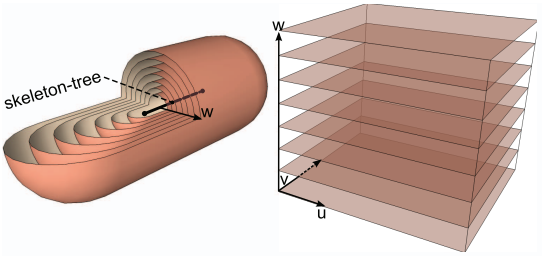


Fig. 3. Our parametrization model is characterized by nested layers, which are aligned around its skeleton-tree (left). In the cube-shaped parameter space, layers are stacked (right).

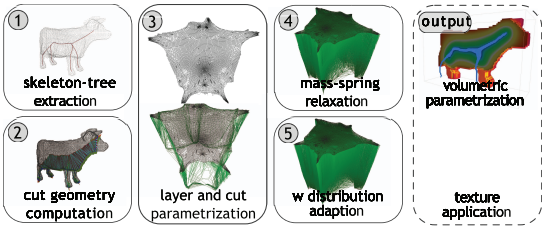


Fig. 4. The workflow of our approach is divided into five subsequent steps. First, a skeleton-tree is computed, which serves as the basis for the innermost parametrization layer (1). Based on the skeleton-tree, we compute the cut geometry (2), used to enable unfolding. After the volume has been cut, a parametrization is generated for the boundary features and the voxels adjacent to the cut (3). Then the interior of the volume is parametrized using mass-spring relaxation (4). Finally, we correct the initial w coordinate distribution, which results from a higher number of springs towards the outer surface (5). This processing results in a parametrization volume, which can be for instance exploited to achieve texturing effects.

Figure 4 illustrates the workflow of the presented parametrization approach. In step 1 we compute a skeleton-tree, which is used to represent the innermost layer's interior of the nested layer model introduced above. To be able to unfold the nested layers coherently and fit them together with the volumetric regions into the parametrization volume as shown in Figure 3 (right), we perform a volumetric cut in step 2 by using the shown cut geometry. The $< u,v >$ parametrization is performed in step 3 by exploiting a mass-spring model. Using a mass-spring model has two main advantages. First, it enables us to use the same approach for boundaries and volumetric regions which permits the incorporation of the same parametrization constraints. Second, the parametrizations of all boundaries are synchronized, which is essential for our nested layer model. Thus, after we have generated the initial $< u,v >$ parametrization in step 3, the resulting model is augmented by inserting masses and springs in the interior, which are relaxed in step 4 to achieve the parametrization of volumetric regions in-between the layers of interest. To ensure that the boundary parametrizations remain planar, their w coordinate is not affected during this relaxation. In general, the outermost surface of a volume contains more voxels than the center-line. This results in an uneven distribution of springs, which directly influences the achieved w distribution. To prevent this, we apply a w distribution adaption in step 5. The output, shown in the rightmost box in Figure 4, is the volumetric parametrization. The color coding in the top inset applies a subset of the rainbow color map to the w coordinate, where blue/green is associated with small w, while red is associated with large w. The bottom inset shows the $< u,w >$ parametrization by applying a checkerboard texture to the outermost surface at $w = 1.0$.

Thus, the workflow depicted in Figure 4 allows us to consider several boundaries, which can have been extracted through segmentation or classification, as well as volumetric regions in order to support unified 2D and 3D texturing. In the following subsections we will discuss all steps shown in Figure 4 in greater detail.

3.1 Skeleton-Tree Extraction

As shown in Figure 4, we first need to derive a center representation for our nested layer model in step 1. A 3D curve-skeleton of an object is a stick-like figure or centerline representation of the object [7]. This definition makes the curve-skeleton an appropriate candidate to be used as the *center* of the nested layers. Cornea et al. list the following properties of curve-skeletons as desirable: homotopic, invariant under isometric transformations, allowing reconstruction of original, thin, centered, reliable, component-wise differentiation, robust, efficient to compute, and hierarchical [6]. For the application case described in this paper, we only need a subset of these relevant properties. The curve-skeleton should be thin, i. e., one voxel thick (except at joints). Furthermore, in order to improve parametrization quality, the curve-skeleton should be reliable, i. e., every surface point is visible from at least one curve-skeleton location. Since we deal with real-world volume data, which is often subject to noise, the curve-skeleton should also be robust, i. e., a centerline of a noise-free object and the same object with noise should be similar. Because the parametrization is a pre-processing step, efficiency is not crucial. From the sophisticated curve-skeleton algorithms known, we have chosen to exploit the potential-field approach proposed by Cornea et al. [7], since it best complies with the properties listed above [6]. The algorithm places point charges on the boundary of the object, to calculate a repulsive force field over the volume data. *Sinks* within this field are then connected, using a force following algorithm. By considering topological characteristics of the resulting vector field, such as critical points and critical curves, a hierarchy of increasingly detailed curve-skeletons can be extracted. The four subsequent steps of the algorithm can be summarized as follows. First, identify the boundary voxels, place a charge at each boundary voxel and calculate the resulting force field for each inner voxel. We allow to define this boundary manually, by exploiting a step transfer function. This mask is then used to derive the force field as described in [7]. In the second step, a level 0 skeleton is obtained by connecting critical points in the force fields through path-lines which are *seeded* at saddle points and, following the direction of the force field, moved in small steps until they reach other critical points or path-lines. In the third step, the level 0 skeleton is transformed into a level 1 skeleton by attaching path-lines from points with low divergence. Because divergence measures the rate of flow leaving a point, points with low divergence indicate a *sink*. Since this step can be computed interactively, it is usually controlled manually, enabling the user to interactively increase or decrease the number of points added. This is the step in the skeleton extraction, where the level of detail can be controlled. To show the impact of this step on the level of detail, we provide a visual comparison of different divergence thresholds in Subsection 5.1. In the fourth step, points on the boundary with high curvature are detected and connected to the existing skeleton in order to obtain a level 2 skeleton.

The four steps described above reflect the curve-skeleton algorithm as it has been originally proposed by Cornea et al. [7]. However, since in our case the requirements slightly vary, we also employ some modifications to the algorithm. For our skeleton-tree data structure, we incorporate the computed level 1 curve-skeleton segments as well as the critical points. Since these level 1 curve-skeletons are already abstractions of the center line, no pruning is necessary. Level 2 skeletons include several additional connections to the outer surface, and are therefore inappropriate for our cuttings algorithm described in the next subsection. Furthermore, since we work on voxel data for rendering, sub-voxel precision is not needed. Additionally, in order to be able to generate a single cut geometry we need to build a skeleton-tree from the critical points and the path-lines, which are originally treated as separate entities. To merge them into one tree structure, we use a modified version of Prim's minimal spanning tree algorithm. Therefore, segments and critical points are interpreted as vertices in a fully connected graph using the Euclidean metric as edge weights. A starting segment is chosen and the closest segment or critical point is added until the tree is complete. If one end of a segment is closest to the incomplete tree it is added as a whole, adding two new leafs.

Skeleton-tree computation is an active field of research, and while the curve skeleton seems to be the most promising approach for our algorithm this might change in the future when new skeleton extraction algorithms are proposed. However, in our approach the skeleton extraction portion of the pipeline can be easily replaced in order to incorporate future advances in this area.

3.2 Volumetric Cuts

When dealing with surface parametrizations, cutting is required to be able to embed an arbitrary surface into the plane. Similarly, a volume has to be cut in order to embed it into the cube-shaped parameter space as shown in Figure 3. To generate the required volumetric cut in step 2 of our algorithm, we take into account the computed skeleton-tree, and generate a seam-tree on the outermost surface of the volumetric object. This outermost surface is the same one as used for the skeleton-tree extraction.

Intuitively, the cut-geometry is generated by introducing planar geometry between each corresponding segment of the skeleton-tree and the seam-tree. Therefore, as seen in Figure 5 (left), the seam-tree having green nodes and the skeleton-tree having red nodes have to be isomorphic in order to generate the cut-geometry. Therefore, the generation of the seam-tree raises two problems to be solved. First, the placement of one seam-node for each leaf and fork of the skeleton-tree, and second, connecting this seam-nodes to obtain a seam-tree which is isomorphic to the skeleton-tree. The whole process of generating a volumetric cut starts at one leaf of the skeleton-tree and generates cut-geometry breadth-first along the skeleton-tree. During each step a seam-node is calculated, before the cut-geometry is generated to connect the new seam-node to the already existing cut-geometry. In the following paragraphs, we first describe how to determine the seam-nodes,

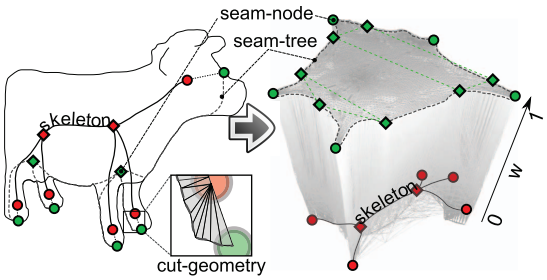


Fig. 5. An illustration of the volumetric cutting. Based on the skeleton-tree (red nodes), we compute a seam-tree on the surface (green nodes). The volume is cut based on the dotted cutting lines derived from the two tree structures (left) and unfolded into the parameter cube (right).

before explaining how to connect them in order to obtain the seam-tree and finally generate the cut-geometry, which is used to cut the mass-spring model before relaxation.

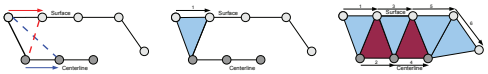
Seam-node placement. To obtain a seam-tree which is isomorphic to the skeleton-tree, one seam-node is placed on the outermost surface for each node in the skeleton-tree. The procedure for finding these seam-nodes is different for inner nodes and outer nodes of the skeleton-tree. For the outer nodes (depicted by red discs in Figure 5 (left)), we extend the adjacent skeleton-tree segment towards the outer surface and choose the nearest intersection with the surface as seam-node. For the inner nodes of the skeleton-tree (depicted by red diamonds in Figure 5 (left)), we choose those points on the outer surface as seam-nodes, which are close to the corresponding inner skeleton-node with the goal that the length of the seam-tree connecting all seam-nodes becomes minimal. This does not mean that the cuts along the surface have to fulfill this minimality property. As discussed further below, different metrics can be used to select a appropriate cut.

When placing the seam-nodes, three different cases need to be distinguished: placing the first seam-node, placing seam-nodes for outer nodes of the skeleton-tree and for inner nodes of the skeleton-tree. We start with the first seam-node, which is always associated with an outer node of the skeleton-tree. Surface voxels are only suitable to be chosen as seam-node when they are visible from the corresponding skeleton-tree node, i. e., there is no other surface voxel between the centerline and this voxel. By using two or more voxels from the end of a skeleton-tree branch, the direction in which the seam-node should be located can be estimated. Placing seam-nodes for other outer nodes of the skeleton-tree works the same way as the first seam-node, except that the connectivity with the previous seam-node is taken into account as described above. Due to this dependency the seam-tree changes slightly when another starting skeleton-tree node is selected. Placing seam-nodes associated with inner nodes of the skeleton-tree is done by using the distance to the inner node as well as the distance to the previous seam-node again. In the special case where the skeleton-tree is degenerated to only one node representing the center-point of a segment, e. g., of a spherical

segment, we choose two surface points lying across from each other as seam-nodes. These nodes are then connected as any other two adjacent seam-nodes.

Seam-tree generation. To generate a seam-tree based on the chosen seam-nodes, several considerations have to be made. As in surface parametrizations, long cuts should be avoided. Especially when dealing with semi-transparent transfer functions, potentially revealing the whole volume, longer cuts could be more easily spotted. Therefore, we connect the seam-nodes by using Dijkstra's shortest path algorithm on the surface voxels (stippled gray line in Figure 5 (left)). However, the length of the seam is not the only criteria, and we modify the edge weights to prevent cutting through voxels with low curvature or high visibility, which both would result in unwanted visual effects. To prevent self intersection of the seam-tree surface, voxels already traversed during the seam-tree generation can only be traversed once. In contrast to the rest of the parametrization, the seam-tree is fixed on one particular outer surface, while all internal surfaces in volumetric data sets might shift based on the chosen rendering parameters. Choosing these voxels for the seam generation is necessary to obtain a parametrization for the entire data set. In cases where this outermost surface is subject to noise, morphological operators can be applied to adapt it accordingly.

Cut-geometry generation. The actual cut-geometry is represented by a triangle strip between all skeleton-tree segments and the corresponding seam-tree segments. Segments of the skeleton-tree and the corresponding segments of the seam-tree typically do not have the same length. To account for this some voxels on the curve-skeleton are connected to multiple voxels on the cut to form a triangle fan with them and vice versa. Our algorithm used to generate the triangle strip can be illustrated by two cursors, one marching along the skeleton-tree and one along the seam-tree. The cursors start at one end of the segment and at each iteration one of them moves one voxel forward until both reach the other end. To synchronize their movement, a move which results in a shorter distance between the cursors is preferred. Figure 6(a) shows the initial position of both cursors and illustrates the alternatives for the first move. Moving the cursor along the seam-tree results in a shorter distance (red) than moving the cursor along the skeleton-tree (blue). If both alternatives result in an equal distance, the cursor associated with the seam-tree is moved forward. Figure 6(b) shows the result of the first iteration. The seam-tree cursor is moved one step forward and a triangle (light blue) is created from the old and the new position as well as the position of the skeleton-tree cursor. As soon as one cursor reaches the end of its segment, the geometry is finalized by a triangle fan. After step 4 in Figure 6(c) the skeleton-tree cursor has finished and steps 5 and 6 create a triangle fan around this end of the centerline.



(a) a) Initial positions with movement alternatives (red, blue).
(b) b) Cursor positions after first move.
(c) Complete triangle strip with movement order.

Fig. 6. Generation of triangle strips representing the cut-geometry is achieved marching along the seam-tree and the skeleton-tree in a synchronized manner.

3.3 Volume Parametrization

To compute the actual parametrization, we exploit a mass-spring model, one of the first approaches used for surface parametrization [19]. We have chosen this model, because it can be applied to volume data without major changes. While more sophisticated surface parametrization techniques may be also extend able to volume data, this approach has the benefit that it can be used for both 2D and 3D parametrizations without any modification. Thus, we can parametrize boundary features as well as volumetric regions with the same parameters. This enables a coherent transition of $< u, v >$ coordinates between the incorporated layers, which allows to achieve a nested layer effect when applying textures also to intermediate layers not considered during the parametrization. To our knowledge our approach is the first to allow this coherency. Furthermore, mass-spring models enable us to easily integrate the constraints required for a boundary-aware parametrization.

Within our mass-spring model, each mass represents a voxel of the original volume data set and is connected to the 18 neighbors (the direct neighbors plus the 12 diagonal neighbors lying in the same plane) by rest-length zero springs, and all masses have the same constant weight. The volumetric cut is performed by removing all springs intersecting the cut-geometry. To comply with our layer model (see Figure 3), we place the masses representing skeleton-tree voxels in the $w = 0$ plane, and those representing voxels of the outermost surface in the $w = 1$ plane, which results in a layout as shown in Figure 5 (right). While both trees are still isomorphic with respect to the outer nodes (green and red discs), this does not hold for the inner nodes (green and red diamonds). Since each inner node of the seam-tree lies directly on the cut geometry, it is split into several nodes. The dotted green lines in Figure 5 (right) indicate, which inner nodes of the deformed seam-tree are associated with the same original inner node. As it can be seen, the number of result nodes is given by the number of segments meeting in the original inner node.

The actual mass-spring relaxation is performed in steps 3 to 5 as depicted in the workflow shown in Figure 4. In step 3, the $< u, v >$ coordinates are computed for the outermost surface as well as the voxels adjacent to the volumetric cut. Next, based on the thus achieved mass-spring setup, $< u, v >$ coordinates are computed for the interior as well as w coordinates for the voxels adjacent to the volumetric cut in step 4. Finally, in step 5 the w distribution is adapted to deal with the fact, that the mass-spring system contains more masses for $w = 1$ than $w = 0$. In the following we will provide details on these three steps.

For the $< u, v >$ boundary parametrization computed in step 3, we exploit the virtual boundary approach [31], which is known to result in a mass-spring parametrization with reduced distortion properties. This results in a parametrization hull as

shown in step 3 in Figure 4.

After step 3 is completed, the masses of all inner voxels are still at their original position. To distribute them appropriately within the regions defined by the layers, and to synchronize the $< u,v >$ parametrizations across layers, the parts of the mass-spring system between layers of interest is activated in step 4. During this relaxation step, we set some of the masses associated with already computed coordinates to infinity in order to keep the results. While for the masses representing the outer surface, we fix all $< u,v,w >$ coordinates, we fix only the $< u,v >$ coordinates for those masses representing voxels adjacent to the cut. One benefit of the mass-spring parametrization approach is that layers of interest can be integrated easily. When incorporating layers of interest, we need to exploit a technique which assigns w coordinate values to the layers of interest. One approach is of course to assign these w coordinate values manually. However, in order to automate the whole parametrization process, the w coordinate values can be derived from the distance d to the skeleton-tree. Therefore, the w coordinate values can be set to $w = \frac{d_f}{d_o}$, where d_f is the average distance of all voxels on the layer of interest and d_o is the average distance of all voxels on the outermost surface. To identify the boundary features different approaches are possible. One approach could be to use isosurfaces which have a large gradient magnitude. These are typically considered as material boundaries and thus more likely to be present in a visualization of the data [26]. However, in most cases a manual extraction will be used. Within our implementation, we support the selection based on isovalues or on a segmentation. It should be pointed out that the extracted layers of interest are still assumed to be of genus 0 and comply to with the nested onion peel model, in order to obtain parametrizations of good quality. Once a w coordinate for the layers of interest has been chosen, this stays fixed during the relaxation in step 4. Thus, for the boundary of layers of interest we have fixed $< u,v,w >$ coordinates, while the interior of such a layer has fixed w coordinates only. This behavior allows us to achieve a synchronized parametrization of adjacent layers, which are connected through interior springs. During the relaxation, Hooke's law would be the physically correct way to model most springs. However, using other relationships between elongation and force can lower the distortion. The idea is to penalize extension over the normal length more than by using Hooke's law and to reduce the force exerted by springs shorter than their normal length. This can be achieved by using a polynomial function of a higher degree instead of a linear function. Thus, the large forces generated by heavily distorted springs modeled with the cubical function favor an even distribution of the distortion over all springs. After step 4 is finished, we have a $< u,v,w >$ parametrization for all voxels.

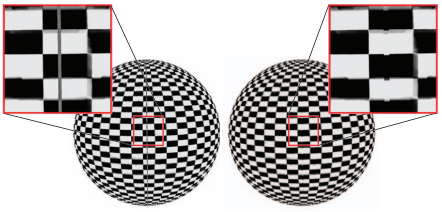
In step 5 we then adapt the output of the mass-spring model to the distribution of the masses, which is of higher density towards the outermost surface. Since the distribution of the springs is directly related to the distribution of voxels, we are able to suppress this effect. Therefore, we take into account that the presented parametrization has been developed for objects of genus zero. Our w distribution adaption is based on the observation that for objects of genus zero the density of masses is roughly distributed according to the ratio of a sphere's radius to its surface, i. e. quadratically. Therefore, we can apply the square root function to every w coordinate in order to achieve a uniform distribution.

4 Volume Texturing

Boundary-aware parametrizations of volumetric data enable the interactive application of several effects which previously were impossible or very difficult to achieve in the context of volume rendering. According to our nested layer model (recall Figure 3), we have a parametrization which assigns an $< u,v,w >$ coordinate to each voxel in the volume. The $< u,v >$ values can be used to perform a lookup in a 2D texture, while the w coordinate can be used for specifying the texture layer to be fetched. For all texture coordinates standard texture coordinate transformations can be applied. Thus, it becomes not only possible to change the size and orientation within one $< u,v >$ plane, but also control the density of texturing layers along the w axis. Since these transformations are highly application dependent, they need to be user-controlled, which can be done interactively during rendering. Alternatively, it is possible to discard the w coordinate and assign textures to intensity ranges through a texture transfer-function. At the same time, the w coordinate can be exploited to extract information about the distance to a boundary of interest.

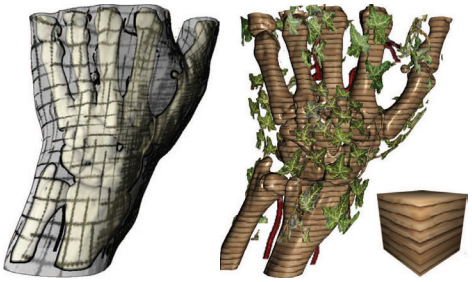
Texturing functionality. Figure 2 shows the application of different texturing effects as commonly used for polygonal rendering. While classical direct volume rendering (DVR) has been applied in Figure 2 (a), Figure 2 (b) shows the application of a color texture, which modifies the diffuse color as fed into the Phong illumination model. As shown in Figure 2 (c), bump mapping now also becomes possible in the context of volume rendering, which has been combined with color mapping and specular highlights in Figure 2 (d). Finally, Figure 2 (e) and Figure 2 (f) illustrate the use of displacement mapping to increase the degree of realism. The displacement is realized by adding the displacement vector to the 3D texture coordinate used to access the volume. This idea was proposed before. Kniss et al. have used 3D noise volumes to modify the location of the data access [27]. In their case three components of the noise volume form a vector which is added to the volume texture coordinates. However, due to the lack of a parametrization, the perturbation volume is repeated for the whole volume. In contrast, in our case the perturbation can be altered at specific locations. To demonstrate the usefulness of our approach in the context of non-photorealistic rendering, we have also integrated tonal art maps [47] into our volume rendering framework (see Figure 2 (g)).

All techniques just need one additional 3D and one additional 2D texture fetch for each sample, with the exception of the displacement mapping technique, which requires one additional 2D and two additional 3D texture fetches. The second 3D texture fetch is required to get the intensity at the new displaced position. While the other techniques can be applied exactly as known from polygonal rendering, again



(a) (b)

Fig. 7. A texture mapped spherical volume data set without (left) and with (right) applying a modified trilinear filter for the texture coordinate lookup. The adapted filtering results in smoother transitions along the seam.



(a) (b)

Fig. 8. CT scan of a hand using different texturing techniques. Cross-hatching (a) and the integration of a 3D wood texture with a 2D leaf texture (b).

the integration of displacement mapping requires a little extra effort. Based on the displacement texture, displacement mapping results in an additional volume texture fetch at the displaced position. Applying this technique to a thin structure, as for instance to a visible inner layer having a small w coordinate, may result in the disappearance of the structure. Therefore, we modulate the amount of displacement with the w coordinate, resulting in larger offsets for larger w coordinates. In addition to the effects shown in Figure 2, all concepts known from polygonal texture mapping, e. g., texture coordinate transformation, detail texturing and clamping, can be applied easily.

Additionally, since we have a true volumetric parametrization, we can exploit the w coordinate to achieve a unified combination with 3D texturing. Samples close to the surface may be 2D textured using our parametrization. Samples further away might use a conventional 3D texture lookup based on their spatial coordinates. A smooth transition between these samples can be employed to avoid discontinuities. An example is shown in Figure 1 (b), where a brick solid texture is used for the interior of the object (visible on the planar cut) while a surface-aligned 2D texture is employed near surface regions.

Similar to polygonal rendering, where texture coordinates are specified for vertices and are interpolated across a polygon, our volumetric texturing approach requires an interpolation of texture coordinates between voxels. However, using trilinear filtering results in artifacts for sampling points lying on different sides of the seam, as can be seen in Figure 7 (a). These artifacts result from the fact that the seam represents a discontinuity in the $< u, v >$ coordinates, where one coordinate is 0.0 on one side and 1.0 on the other side. When applying texture filtering, this results in interpolated values between 0.0 and 1.0 for the respective coordinate, which become visible as artifacts. We can avoid this by using a modified trilinear filter, which detects discontinuities in the $< u, v >$ space and only takes neighboring voxels on the same side of the seam into account (Figure 7 (b)). Since current graphics hardware enables such custom filtering mechanisms, we are able to implement this approach within a fragment shader.

Figure 8 shows some effects we are able to achieve by applying the introduced concepts to real-world data. Figure 8 (a) shows the application of cross-hatching to the skin layer in combination with a conventional texture applied to the bones, since when depicting surface details with strokes, the spatial comprehension of semi-transparent surfaces can be improved [20], [24]. Figure 8 (b) shows an example of a seamless integration of 2D and 3D textures, where a 3D wood texture is combined with a 2D leaf texture.

Photorealistic rendering. The appearance of organic materials is heavily influenced by light interactions below the surface [16]. To achieve a realistic depiction of these materials, numerous approaches for modeling subsurface scattering have been proposed [12], [17], [21]. An advantage of using a sampled volumetric representation is that the data set contains information about the structure of an object below its surface. In combination with our parametrization, 2D textures can be employed to introduce subtle variations of material properties along the surface of a volumetric structure. A texture transfer-function allows the assignment of different 2D textures to regions in a volume data set based on the measured property (e. g., density in Hounsfield units in case of CT data). For each sample point along a viewing ray, its material properties are determined by a lookup in the assigned 2D texture using the texture coordinates established through our parametrization. Alternatively, in low resolution cases where adjacent layers cannot be distinguished based on the measured properties, the w coordinate can be employed to achieve the layered appearance.

The augmentation of a volumetric illumination model with conventional 2D textures can lead to a more convincing depiction with little additional effort. To demonstrate this, we employ a variation of the direct volume illumination model presented by Schott et al. [51]. This method uses a specialized conical phase function suitable for real-time rendering. Additionally, we combine this approach with the forward-scattering approximation proposed by Kniss et al. [27]. The outcome of this process is shown in Figure 9, where we have applied a photographic skin texture to a CT scan of a human head, while soft tissue below the skin uses a different red-toned texture. We show a comparison of Phong shading without textures (a), Phong shading with textures (b), volumetric illumination using the same texture for soft tissue and skin (c), and volumetric illumination with different textures for soft tissue and skin (d).

Fig. 9. Volume data set rendered using Phong shading without textures (a), Phong shading with textures (b), volumetric illumination using the same texture for soft tissue and skin (c), and volumetric illumination with different textures for soft tissue and skin (d). The insets show the used textures and the underlying parametrization.

Due to the layering of skin and soft tissue in the CT scan, a more convincing appearance of the skin is achieved in Figure 9 (d) without employing a specialized skin shader.

Illustrative rendering. Besides leveraging the degree of realism, the unified texturing approach is also beneficial for achieving illustrative rendering effects. Often in volumetric data, objects of interest may be occluded by less important objects. Cutaway illustrations address this problem by omitting parts of the occluding objects. For polygonal data specialized techniques are required to generate these cutaway illustrations (e. g., Burns and Finkelstein [5]). In contrast to existing approaches presented for illustrative volume rendering, our parametrization can be used to generate voxel-precise cutaways interactively without any specific adaption. This is only possible, because our parametrization achieves a synchronized $< u, v >$ mapping for all present layers, which allows to peel away structures layer-wise in a consistent way. Figure 10 shows two such cutaway illustrations, which have been generated interactively by exploiting alpha mapping. Figure 10 (a) does not incorporate any translucency. In Figure 10 (b) the skin layer has been rendered using translucency to depict the volumetric nature of the data. To achieve a more volumetric effect of the cutting edges, we have modulated the alpha value of the texture with the w texture coordinate. It would be much more difficult to achieve a similar effect using a polygonal representation, especially considering that the cuts can be interactively modified by drawing with the mouse in an intuitive manner.

Volume annotation. In addition to the possibilities presented above regarding volume visualization, the proposed parametrization also facilitates interactive annotation of volumes. These annotations are often required in medical diagnosis [29] or to communicate the findings made in scientific data sets [43]. As described in Section 2, several authors have proposed specific annotation techniques, e. g., for adding textual labels [48]. With our technique, we are able to simulate the behavior of many of the discussed techniques.

For demonstration purposes, we have implemented a 3D labeling extension as well as a color brush. Both techniques directly interact with the parametrization, which is used to position the annotation with respect to the data. To obtain the mapping between the annotation and the volume, we exploit an additional framebuffer object, into which we render the $< u, v, w >$ texture coordinates of the first hit points. Thus, we can read back the $< u, v, w >$ coordinates to be modified based on the current mouse position. To apply a 3D text label, the user has to first set an anchor point by simply clicking on the rendering. Then text can be entered, which is directly rendered into the selected texture layer at the position denoted by the read back $< u, v, w >$ coordinate. Since our parametrization aligns the texture with the visible features, a 3D labeling effect is achieved (see Figure 14 (a)). Painting with a brush within the volume can be done in a similar way. However, since we want to achieve a more volumetric paint effect, we render the brush foot print with different sizes in several adjacent layers. A result of such a painting process is shown in Figure 14 (b).

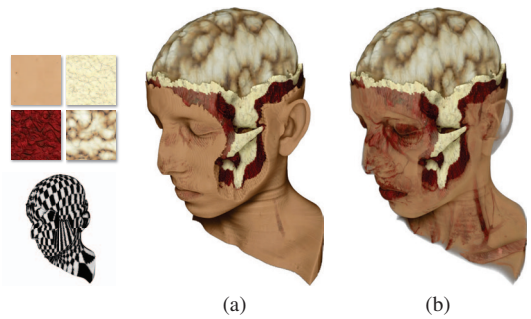
5 Parametrization Analysis

5.1 Skeletonization Parameters

When transforming the level 0 skeleton into a level 1 skeleton by attaching path-lines from points with low divergence, the level of detail of the skeleton can be controlled. To show the impact of this step on the level of detail, we provide a visual comparison of different divergence thresholds as applied to different synthetic datasets with varying shapes in Figure 11. Each row represents one data set, and shows how the centerline and the final cut geometry vary when different divergence thresholds are used. The data sets have been synthesized such that they are continuously changing from spherical to branching structures. From left to right, a value for m and the divergence percentage have been set to $(m = 6, 45\%)$, $(m = 6, 55\%)$, $(m = 6, 60\%)$ and $(m = 10, 45\%)$. As it can be seen, the difference becomes increasingly visible for with more branching structures, though the thresholds need to be adapted to the data set. While for instance $(m = 6, 45\%)$ is appropriate for sphere-like and branching structures, intermediate structures may require a different threshold. However, since the thresholds can be set interactively and direct visual

feedback is provided, the user can choose the values to be optimal for the specific case.

In Figure 12 we show color-coded $< u,v,w >$ parametrizations for various data sets used within this paper. While in some cases as for instance with the head data set, quite simple skeleton-trees are sufficient, depending on the structure of the object more complex skeleton-trees might be needed. In the case with the hand dataset the structural similarity of the skeleton-tree and the discontinuity in the $< u,v,w >$ parametrization along the cut can be seen quite well. However, when dealing with multiple layers of interest the structure formed by these layers may vary. In some cases, the skeleton-tree of the outermost surface may not entirely lie within the innermost layer of interest. In these cases, the skeleton can be computed for the innermost layer of interest instead. To investigate the influence of such a variation on the results of our algorithm, we have generated synthetic data sets reflecting extreme cases. The most difficult scenario would be of course, if the genus of the nested objects was different. Since our algorithm has not been developed for those cases, as the cut-geometry generation does not support circles in the skeleton-tree (see Subsection 3.1), we attempted to simulate this scenario as close as possible. Therefore, the data we apply our technique to is formed by an almost closed torus containing another torus segment (see Figure 13). Due to the mentioned limitations of the skeleton matching, we were not able to entirely close the outer torus. Nevertheless, the data sequence resembles a topology difference as good possible. As it can be seen in Figure 13, the parametrization leads to appropriate results for the first few cases, which is reflected by the fact that the color-coded parametrization spans over the whole data set. However, when the shape differs too much, larger areas are homogeneously colored, which represents similar $< u,v,w >$ parameters. This becomes also present when showing the texture application in the bottom row.



(a) (b)

Fig. 10. Our technique allows to generate voxel-precise cutaways by combining textured volume rendering with alpha painting applied to individual textures (a). To emphasize the volumetric nature of the data set, the skin has been rendered translucently (b). The insets show the used textures and the parametrization.

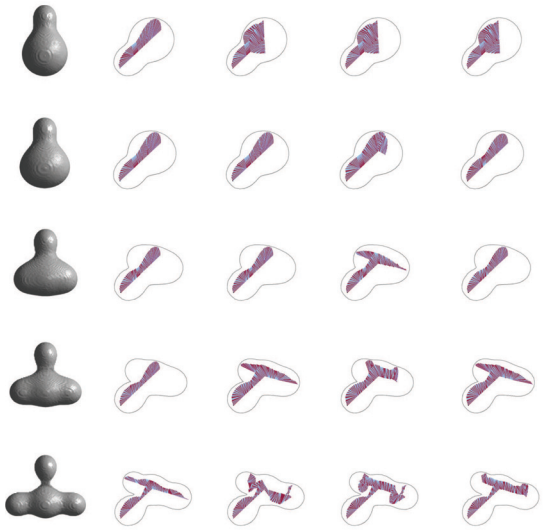


Fig. 11. Comparison of different orders m of the force field as well as the percentage of additional seeds with high divergence (from left to right: $(m = 6, 45\%)$, $(m = 6, 55\%)$, $(m = 6, 60\%)$ and $(m = 10, 45\%)$) and their influence on the resulting cut-geometry for datasets with varying shape properties. The second value can be controlled interactively to obtain the desired results.

5.2 Distortion Analysis

To be able to compare the quality of surface parametrization algorithms, several conformity criteria exist. These criteria usually describe the edge, area as well as angle preservation of the underlying triangles [61]. As no distortion-free parametrization is possible for general surfaces [19], this is also true for volumetric objects. However, since no comparable parametrization technique for volumetric data exists, no measures have been established to quantify the volumetric distortion. Measures for evaluating surface parametrization algorithms are the result of several years of research. Therefore, it would be out of the scope of this paper to build up a full theoretical foundation for evaluating volumetric parametrizations. Instead, we have transferred some of the concepts found

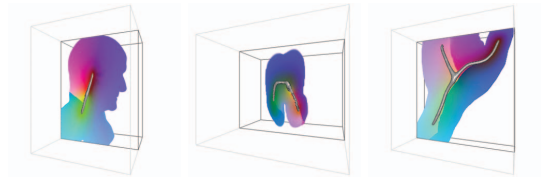


Fig. 12. Color-coded $< u,v,w >$ parametrizations for various data sets used within this paper. From left to right (head, tooth, hand), the structure of the skeleton-tree becomes more complex. The color coding has been achieved by directly mapping $< u,v,w >$ to $< r,g,b >$.

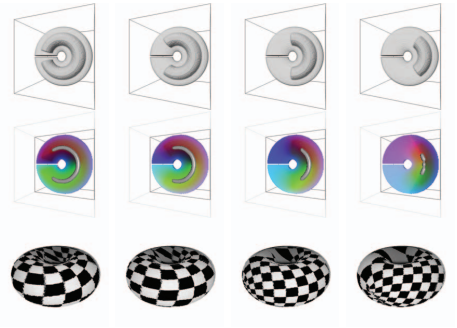


Fig. 13. To demonstrate the limitations of our algorithm, we have applied it to data sets containing nested layer of interest with increasingly varying shape. From left to right, the difference between the shapes of the two considered layers increases. As it can be seen, this results in an increasing distortion. The same color mapping has been used as in Figure 12.

in the surface parametrization literature in order to estimate the quality of our parametrization. First, to be able to estimate the distribution of $< u,v >$ coordinates, we have performed a comparison with a corresponding natural parametrization. Figure 18 shows a comparison of the distribution for the hand data set (see Figure 8) with the distribution for the most similar naturally parametrizable object, a cylinder. For a discrete set of w coordinates over the whole parameter range $[0,1]$, we plot the change of the $< u,v >$ coordinates $\delta = \frac{|uv_i - uv_j|}{|p_i - p_j|}$, where uv_i describes the $< u,v >$ coordinate values at position p_i, and p_j is adjacent to p_i. As Figure 18 shows, our parametrization achieves a mean value and standard deviation comparable to the natural parametrization, which can be interpreted as an indicator for a similar distortion. Although in both cases the standard deviation increases significantly towards the center of the objects, it should be noted that this affects only a very limited fraction of all parametrized voxels. In Figure 18 (b), we have emphasized this fraction with the gray rectangle, it lies below 3.8% of all voxels. We believe that the slight upslope and downslope towards $w = 0$ and $w = 1$ in Figure 18 (b) results from the larger outer surface to skeleton ratio of the hand data set.

To get further insights, we have applied the stretch metric proposed by Sander et al. [49] and compared the achieved results with state-of-the-art surface parametrization techniques.

(a) (b)

Fig. 14. The presented approach supports interactive volume annotation. Volumes can be annotated by adding text labels (a), or by drawing with a paint brush (b).

(a) (b) (c)

Fig. 15. By using our algorithm, meaningful $< u,v >$ parametrizations can be achieved also for surfaces not selected as boundary features. In (a) a parametrized boundary feature is shown, while in (b) and (c) we show the resulting $< u,v >$ parametrization for internal boundaries not considered during the parametrization.

Sander et al. introduce the L^2 surface stretch metric for a mesh M as $L^2(M) = \sqrt{\sum_{T_i \in M} \left(L^2(T_i)\right)^2 A'(T_i) / \sum_{T_i \in M} A'(T_i)}$, where T_i is a triangle of M, $A'(T_i)$ is the surface area of the triangle T_i and $L^2(T_i)$ corresponds to the root-mean-square stretch over all directions.

To be able to apply this metric, we have extracted two iso-surfaces from the volumetric data set shown in Figure 10. To evaluate our technique, we have read back the $< u,v >$ coordinates for each vertex of the isosurface from the parametrization volume, while we used the CGAL library to generate the other parametrizations. Table 1 shows the results which we have achieved for our parametrization as well as for five state-of-the-art surface parametrization algorithms. The table shows that the achieved stretch behavior is comparable to that of the state-of-the-art techniques, although slightly more stretch is involved. While this is partly due to the different parametrization technique, it is also due to the fact that we had to read back the $< u,v >$ coordinates from our parametrization volume. Thus, interpolation introduces additional stretch, since the vertices do not lie directly at grid locations. However, when visually comparing our technique to state-of-the-art surface parametrization algorithms similar stretch behavior is achieved. As seen in Figure 16, when applying a checkerboard texture, independent of the used parametrization method a similar variation in checker size is visible. However, only our parametrization method allows us to apply a single model which supports an easy derivation of parametrizations for layers not considered before the parametrization.

When computing the $< u,v >$ parametrizations by more

TABLE 1
Parametrization comparison of two layers of the parametrization shown in Figure 10. The comparison is based on the L^2 stretch metric [49].

	skin	brain
our technique	354.433	200.178
Floater [15]	302.436	175.727
conformal [13]	303.568	175.935
barycentric [59]	307.715	179.549
authalic [9]	301.953	175.625
LSCM [34]	227.205	106.209

(a) our technique (b) Floater (c) conformal

(d) barycentric (e) authalic (f) LSCM

Fig. 16. Comparison of $< u,v >$ parametrizations on an isosurface derived from our volume parametrization (a), Floater parametrization [15] (b), conformal mapping [13] (c), barycentric mapping [59] (d), authalic mapping [9] (e), and least squares conformal mapping [34] (f).

sophisticated approaches, which is definitely possible, it is still necessary to derive meaningful w coordinates. While there have been some efforts undertaken into this direction [37], a synchronized consideration of multiple $< u,v >$ mappings was not possible, yet. This synchronized consideration is the main benefit of a global volume parametrization, where coherency is desired between several nested layers. Therefore, we have also taken into account how the parametrization behaves for the remaining structures and have analyzed the $< u,v >$ parametrization of different surfaces within the volume, which are implicitly extracted through an appropriate transfer function. Figure 15 shows the comparison of three of these surfaces, as extracted from the nucleon data set. In this data set, we have chosen the surface shown in Figure 15 (a) as boundary feature during step 3. The images in Figure 15 (b) and (c) show the results for two other boundaries not taken into account when computing the parametrization. Even in the presented cases, where the surface geometry is considerably different (torus- vs. sphere-shaped), meaningful results can be achieved when applying the checkerboard texture. By simply interpolating the $< u,v >$ coordinates of adjacent layers, this would not be possible.

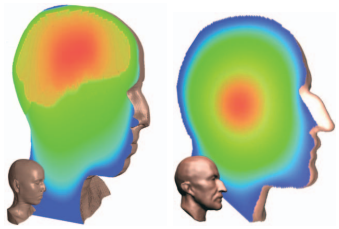


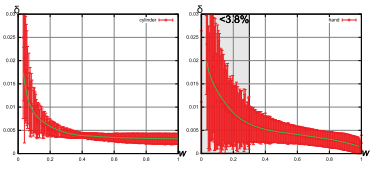
Fig. 17. w coordinate distribution as achieved by our feature-driven approach (a) and harmonic volumetric mapping [36] (b). The color-coding has been adapted according to [36], i. e., applying a rainbow color map, where small w are mapped to red and large w are mapped to blue.

While the previous analyses only allow to draw conclusions regarding the $< u,v >$ mapping, the w coordinate distribution also requires some inspection. Therefore, we have compared our approach to the harmonic volumetric mapping [36] as well as the improved harmonic volumetric mapping technique [37], which also allow to derive w coordinates and thus are the most similar approaches. As described in Section 2, harmonic volumetric mapping is used to propagate a surface parametrization into the interior of a polygonal model. One of the differences to our technique is that no knowledge about the layers of interest can be incorporated. The effect of this difference becomes clear in Figure 17, where different hues are used to depict the w texture coordinates. Since with our approach shown in Figure 17 (a) the brain surface has been incorporated as layer of interest, textures can be applied to it (see Figure 10). This is not possible when using harmonic volumetric mapping as shown in Figure 17 (b).

When working with our results, we did not experience any parameter cracks as discussed by Yoshizawa et al. [61]. This leads to a smooth texture application, as indicated by the checkerboard texturing examples presented throughout this paper.

5.3 Stability Remarks

Finally, we would like to make some stability remarks. We have investigated all steps of our workflow depicted in Figure 4. The first source of potential instability is the skeleton-tree extraction. However, Cornea et al. have evaluated different algorithms and were able to show that the potential-field curve-skeleton algorithm achieves the best results [6]. This complies with our experience, since we were able to obtain a meaningful curve-skeleton for all tested data sets. Once a skeleton-tree had been generated, we could also always



(a) (b)

Fig. 18. The change δ of $< u,v >$ coordinate values along the w axis is similar, when comparing a natural parametrization (a) with a parametrization achieved by using our algorithm (b). As indicated by the gray rectangle, not only the mean values but more importantly the standard deviations are comparable for more than 96% of the data.

identify a seam-tree with the same connectivity and thus were able to perform the volumetric cut. Regarding the stability of step 3 observations from classical mesh parametrization apply, since the mass-spring approach could also be replaced by other surface parametrization techniques. However, step 4 depends on a mass-spring system, and in general mass-spring systems might suffer from instability issues. Since mass-spring systems are frequently used in other areas, i. e., virtual surgery and cloth simulation, sophisticated extensions are available to guarantee stability [2]. While we did not experience any instability effects in step 4, one of the existing extensions could be integrated to deal with this issue.

6 Conclusions and Future Work

We have proposed a novel approach, which allows a unified 2D and 3D texture mapping of volumetric objects. To achieve this, we have generated a volumetric parametrization, exploiting the curve-skeleton of volumetric objects as well as a novel 3D seam generation algorithm. By taking into account relevant boundary layers within the data set, we are able to provide a reasonable parametrization for both volumetric regions as well as material boundaries. This is the first approach allowing a unified application of 2D and 3D textures. The proposed parametrization algorithm is the first global volume parametrization concept for arbitrarily shaped data sets of genus zero. Using this parametrization, we have shown how interactive volume visualization can benefit from texture mapping. We have successfully applied various texturing techniques, e. g., bump mapping and displacement mapping, to volumetric objects. Similar to texture mapping of polygonal objects, this increases the realism of volume graphics. Furthermore, we have applied the presented techniques in the area of interactive illustration by demonstrating the construction of interactive cutaways as well as performing volume annotation.

In the future, we would like to further extend the presented parametrization by exploiting contour trees, which have already been employed in the field of volume rendering [55], [60]. Thus, it might be possible to facilitate a fully automatic extraction of the features of interest. Furthermore, the generation of seamless textures for a given parametrization is an interesting area for further research.

Acknowledgments

This work has been supported by ELLIIT, the Strategic Area for ICT research, funded by the Swedish Government, and by the ViMaL project (FWF, no. P21695).

References

[1] A. Baer, C. Tietjen, R. Bade, and B. Preim. Hardware-accelerated stippling of surfaces derived from medical volume data. In *IEEE/EG EuroVis*, pages 235–242, 2007.

[2] Y. Bhasin and A. Liu. Bounds for damping that guarantee stability in mass-spring systems. In *Medicine Meets Virtual Reality 14*, pages 55–60, 2006.

[3] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007.

[4] K. Bürger, J. Krüger, and R. Westermann. Direct volume editing. *IEEE TVCG*, 14(6):1388–1395, 2008.

[5] M. Burns and A. Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Trans. Graph.*, 27(5):124:1–124:9, 2008.

[6] N. D. Cornea, D. Silver, and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE TVCG*, 13(3):530–548, 2007.

[7] N. D. Cornea, D. Silver, X. Yuan, and R. Balasubramanian. Computing hierarchical curve-skeletons of 3D objects. *Visual Computer*, 21(11):945–955, 2005.

[8] B. Cutler, J. Dorsey, L. McMillan, M. Müller, and R. Jagnow. A procedural approach to authoring solid models. In *ACM SIGGRAPH*, pages 302–311, 2002.

[9] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2002.

[10] F. Dong and G. Clapworthy. Volumetric texture synthesis for non-photorealistic volume rendering of medical data. *Visual Computer*, 21(7):463–473, 2005.

[11] Y. Dong, S. Lefebvre, X. Tong, and G. Drettakis. Lazy solid texture synthesis. *Computer Graphics Forum*, 27(4):1165–1174, 2008.

[12] C. Donner and H. W. Jensen. Light diffusion in multi-layered translucent materials. In *ACM SIGGRAPH*, pages 1032–1039, 2005.

[13] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *ACM SIGGRAPH*, pages 173–182, 1995.

[14] M. S. Floater and K. Hormann. Surface parameterization: A tutorial and survey. In *Advances in Multires. for Geom. Modelling, Mathematics and Visualization*, pages 157–186. 2005.

[15] M. S. Floater, G. Kós, and M. Reimers. Mean value coordinates in 3d. *Comput. Aided Geom. Des.*, 22:623–631, 2005.

[16] P. Hanrahan and W. Krueger. Reflection from layered surfaces due to subsurface scattering. In *ACM SIGGRAPH*, pages 165–174, 1993.

[17] X. Hao, T. Baby, and A. Varshney. Interactive subsurface scattering for translucent meshes. In *ACM I3D*, pages 75–82, 2003.

[18] E. R. S. Hodges, editor. *The Guild Handbook of Scientific Illustration*. John Wiley & Sons, 2nd edition, 2003.

[19] K. Hormann, K. Polthier, and A. Sheffer. Mesh parameterization: Theory and practice. In *ACM SIGGRAPH Asia Courses*, 2008.

[20] V. Interrante, H. Fuchs, and S. M. Pizer. Conveying the 3D shape of smoothly curving transparent surfaces via texture. *IEEE TVCG*, 3(2):98–117, 1997.

[21] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. In *ACM SIGGRAPH*, pages 511–518, 2001.

[22] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. In *ACM SIGGRAPH*, pages 561–566, 2005.

[23] I. Kabul, D. Merck, J. Rosenman, and S. M. Pizer. Model-based solid texture synthesis for anatomic volume illustration. In *EG Workshop on Visual Computing for Biology and Medicine*, pages 133–140, 2010.

[24] S. Kim, H. Hagh-Shenas, and V. Interrante. Conveying three-dimensional shape with texture. In *ACM APGV*, pages 119–122, 2004.

[25] G. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Volume Visualization*, pages 79–86, 1998.

[26] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of IEEE Visualization 2001*, pages 255–262, 2001.

[27] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson. A model for volume lighting and modeling. *IEEE TVCG*, 9(2):150–162, 2003.

[28] J. Kopf, C.-W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, and T.-T. Wong. Solid texture synthesis from 2D exemplars. In *ACM SIGGRAPH*, volume 26, pages 2:1–2:9, 2007.

[29] A. Krüger, C. Kubisch, G. Strauß, and B. Preim. Sinus endoscopy - application of advanced GPU volume rendering for virtual endoscopy. *IEEE TVCG*, 14(6):1491–1498, 2008.

[30] Y. Kurzion, T. Möller, and R. Yagel. Size preserving pattern mapping. In *IEEE Visualization*, pages 367–373, 1998.

[31] Y. Lee, H. Kim, and S. Lee. Mesh parameterization with a virtual boundary. *Computers & Graphics*, 26(5):677–686, 2002.

[32] S. Lefebvre and C. Dachsbacher. Tiletrees. In *ACM I3D*, pages 25–31, 2007.

[33] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[34] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *ACM SIGGRAPH*, pages 173–182, 2002.

[35] X. Li, X. Guo, H. Wang, X. Gu, and H. Qin. Meshless harmonic volumetric mapping using fundamental solution methods. *IEEE Transactions on Automation Science and Engineering*, 6(2):409–422, 2009.

[36] X. Li, X. Guo, H. Wang, Y. He, X. Gu, and H. Qin. Harmonic volumetric mapping for solid modeling applications. In *ACM SPM*, pages 109–120, 2007.

[37] X. Li, H. Xu, S. Wan, Z. Yin, and W. Yu. Feature-aligned harmonic volumetric mapping using MFS. In *SMI '10: Proceedings of Shape Modeling International 2010*, 2010. *(to appear)*.

[38] A. Lu, D. S. Ebert, W. Qiao, M. Kraus, and B. Mora. Volume illustration using wang cubes. *ACM Trans. Graph.*, 26(2):11, 2007.

[39] F. Manke and B. C. Wuensche. Texture-enhanced direct volume rendering. In *GRAPP*, pages 185–190, 2009.

[40] T. Martin, E. Cohen, and M. Kirby. Volumetric parameterization and trivariate b-spline fitting using harmonic functions. In *SPM '08: Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, pages 269–280, 2008.

[41] C. M. Miller and M. W. Jones. Texturing and hypertexturing of volumetric objects. In *IEEE/EG Volume Graphics*, pages 117–125, 2005.

[42] S. Owada, F. Nielsen, M. Okabe, and T. Igarashi. Volumetric illustration: Designing 3D models with internal textures. In *ACM SIGGRAPH*, pages 322–328, 2004.

[43] D. Patel, C. Giertsen, J. Thurmond, J. Gjelberg, and E. Gröller. The seismic analyzer: Interpreting and illustrating 2D seismic data. *IEEE TVCG*, 14(6):1571–1578, 2008.

[44] D. Patel, C. Giertsen, J. Thurmond, and E. Gröller. Illustrative rendering of seismic data. In *Vision, Modeling, and Visualization*, pages 13–22, 2007.

[45] N. Pietroni, M. A. Otaduy, B. Bickel, F. Ganovelli, and M. Gross. Texturing internal surfaces from a few cross sections. *Computer Graphics Forum*, 26(3):637–644, 2007.

[46] S. D. Porumbescu, B. Budge, L. Feng, and K. I. Joy. Shell maps. *ACM Trans. Graph.*, 24(3):626–633, 2005.

[47] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *ACM SIGGRAPH*, pages 581–586, 2001.

[48] T. Ropinski, J.-S. Praßni, J. Roters, and K. Hinrichs. Internal labels as shape cues for medical illustration. In *Vision, Modeling, and Visualization*, pages 203–212, 2007.

[49] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *ACM SIGGRAPH*, pages 1032–1039, 2005.

[50] R. Satherley and M. W. Jones. Hypertexturing complex volume objects. *Visual Computer*, 18(4):226–235, 2002.

[51] M. Schott, V. Pegoraro, C. Hansen, K. Boulanger, and K. Bouatouch. A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum*, 28(3):855–862, 2009.

[52] A. Sheffer and J. Hart. Seamster: inconspicuous low-distortion texture seam layout. In *IEEE Visualization*, pages 291–298, 2002.

[53] P. Shen and P. Willis. Texture for volume character animation. In *ACM GRAPHITE*, pages 255–264, 2005.

[54] P. Shen and P. Willis. Texture mapping volume objects. In *Vision, Video, and Graphics*, pages 45–52, 2005.

[55] S. Takahashi, I. Fujishiro, and Y. Takeshima. Interval volume decomposer: a topological approach to volume traversal. In *Visualization and Data Analysis*, pages 103–114, 2005.

[56] K. Takayama and T. Igarashi. Layered solid texture synthesis from a single 2D exemplar. In *ACM SIGGRAPH Posters*, 2009.

[57] K. Takayama, M. Okabe, T. Ijiri, and T. Igarashi. Lapped solid textures: filling a model with anisotropic textures. *ACM Trans. Graph.*, 27(3):1–9, 2008.

[58] S. M. F. Treavett and M. Chen. Pen-and-ink rendering in volume visualisation. In *IEEE Visualization*, pages 203–210, 2000.

[59] W. T. Tutte. How to draw a graph. *London Mathematical Society*, 13(52):743–768, 1963.

[60] G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE TVCG*, 13(2):330–341, 2007.

[61] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. A fast and simple stretch-minimizing mesh parameterization. In *SMI '04: Proceedings of Shape Modeling International 2004*, pages 200–208, 2004.

[62] K. Zhou, X. Huang, X. Wang, Y. Tong, M. Desbrun, B. Guo, and H.-Y. Shum. Mesh quilting for geometric texture synthesis. *ACM Trans. Graph.*, 25(3):690–697, 2006.

[63] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3D: an interactive system for point-based surface editing. In *ACM SIGGRAPH*, pages 322–329, 2002.

Timo Ropinski is professor in interactive visualization at Linköping University, Sweden, where he is heading the scientific visualization group. His research is focused on volume rendering, biomedical visualization and interactive visual decision making. He is a member of the winning team of the IEEE Visualization contest 2010, initiator of the Voreen software project (www.voreen.org), and he has held tutorials at Eurographics, ACM SIGGRAPH and IEEE Visualization. Furthermore, he is a member of the IEEE Computer Society, ACM, and Eurographics.

Stefan Diepenbrock is a Ph.D. student at the University of Münsters Visualization and Computer Graphics Research Group. He received his master's degree in 2009 from the University of Münster. His research interest include volume rendering and medical visualization with a special focus on interactive techniques. He is a member of the winning team of the IEEE Visualization contest 2010, and he is one of the core developers of the Voreen project.

Stefan Bruckner received his Ph.D. in 2008 from the Vienna University of Technology (VUT). He is currently an Assistant Professor at the Institute of Computer Graphics and Algorithms at VUT. In 2009/2010, he spent one year as a visiting Postdoctoral Research Fellow at Simon Fraser University, Canada. His research interests include biomedical and illustrative visualization, volume rendering, and visual data exploration. In 2011, he was awarded the Eurographics Young Researcher Award. He is a member of the IEEE Computer Society, ACM SIGGRAPH, and Eurographics.

Klaus Hinrichs received the Ph.D. degree in computer science from the Swiss Federal Institute of Technology (ETH) in Zurich in 1985. He is a full professor of computer science and head of the visualization and computer graphics research group at the University of Münster, Germany. His research interests include visualization, computer graphics, algorithms and data structures for geometric computation, and spatial databases. He is a member of the IEEE.

Eduard Gröller is associate professor at the Vienna University of Technology, Austria, and adjunct professor of computer science at the University of Bergen, Norway. His research interests (http://www.cg.tuwien.ac.at/research/vis/) include computer graphics, flow visualization, volume visualization, medical visualization, and information visualization. He co-authored more than 190 scientific publications and acted as a co-chair, IPC member, and reviewer for numerous conferences and journals in the field. Dr. Gröller is currently Co-Chief Editor of the Computer Graphics Forum journal.