Unified Boundary-Aware Texturing for Interactive Volume Rendering

Timo Ropinski, *Member*, *IEEE*, Stefan Diepenbrock,
Stefan Bruckner, *Member*, *IEEE Computer Society*,
Klaus Hinrichs, *Member*, *IEEE Computer Society*, and
Eduard Gröller, *Member*, *IEEE Computer Society*

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 Fig. 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 (Fig. 1a), while 2D textures can be assigned to surfaces in order to display information on material boundaries (Fig. 1b). 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 Fig. 2). To do so, we had to 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 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

- *T. Ropinski is with the University of Linköping, Kungsgatan 54, Norrköping 60233, Sweden. E-mail: timo.ropinski@liu.se.*
- *S. Diepenbrock and K. Hinrichs are with the University of Münster, Einsteinstr. 62, Muenster 48149, Germany. E-mail: {diepenbrock, khh}@uni-muenster.de.*
- *S. Bruckner and E. Gröller are with the Vienna University of Technology, Favoritenstrasse 9-11/E186, Wien A-1040, Austria. E-mail: {bruckner, groeller}@cg.tuwien.ac.at.*

Manuscript received 15 Feb. 2011; revised 18 Nov. 2011; accepted 2 Dec. 2011; published online 7 Dec. 2011.
Recommended for acceptance by P. Cignoni.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-02-0035.
Digital Object Identifier no. 10.1109/TVCG.2012.285.

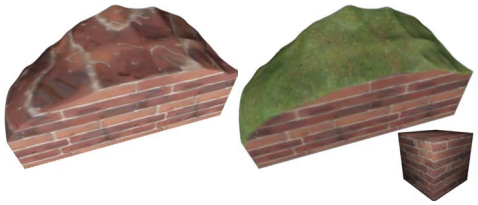


Fig. 1. A volume data set rendered with a 3D brick texture (a), and with an additional 2D grass texture (b). While 3D texturing results in a carved-out effect, 2D texturing can be used to depict surface details.

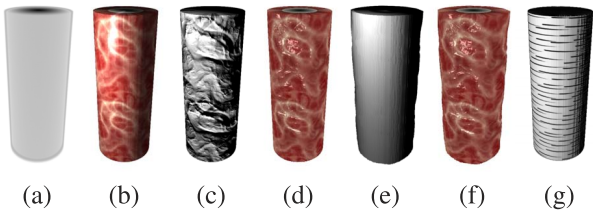


Fig. 2. Application of different texture mapping techniques to a volumetric cylinder data set (a). Color mapping in combination with Phong shading (b), bump mapping (c), bump mapping and color mapping in combination with specular lighting (d), displacement mapping with diffuse shading (e), displacement 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).

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].

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 triplanar 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 toward 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 Fig. 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

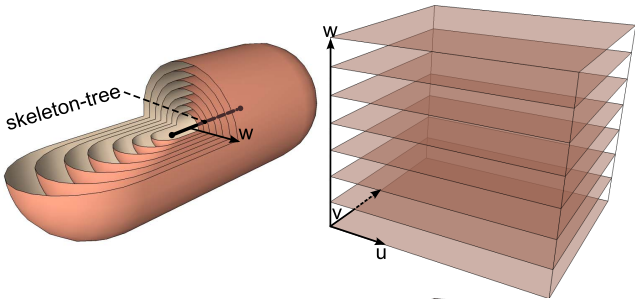


Fig. 3. Our parametrization model is characterized by nested layers, which are aligned around its skeleton-tree (left). In the cube-shaped parameter space, layers are stacked (right).

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 Fig. 3, the nested layers of an object of genus 0 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,

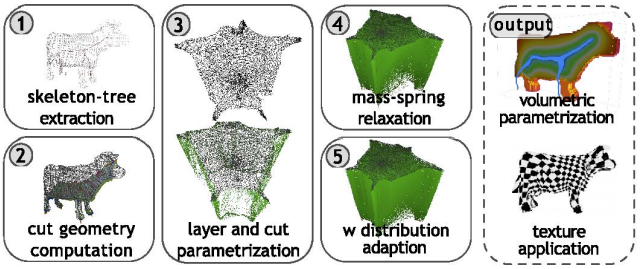


Fig. 4. The workflow of our approach is divided into five subsequent steps. First, a skeleton-tree is computed, which serves as the basis for the innermost parametrization layer (1). Based on the skeleton-tree, we compute the cut geometry (2), used to enable unfolding. After the volume has been cut, 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 toward the outer surface (5). This processing results in a parametrization volume, which can be for instance exploited to achieve texturing effects.

e.g., the seamster algorithm [52], but no algorithms have been proposed yet for generating volumetric cuts.

Fig. 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 Fig. 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 centerline. 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 Fig. 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 Fig. 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 sections, we will discuss all steps shown in Fig. 4 in greater detail.

3.1 Skeleton-Tree Extraction

As shown in Fig. 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, componentwise 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 preprocessing 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 pathlines 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 pathlines. In the third step, the level 0 skeleton is transformed into a level 1 skeleton by attaching pathlines 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 Section 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

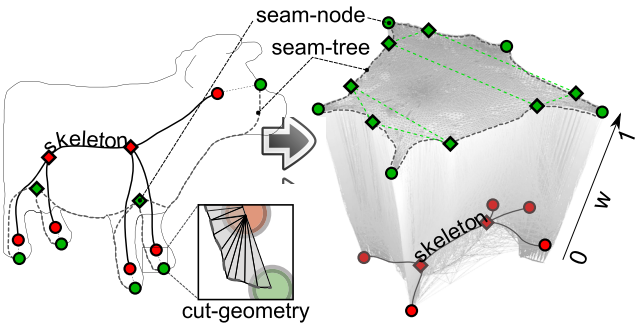


Fig. 5. An illustration of the volumetric cutting. Based on the skeleton-tree (red nodes), we compute a seam-tree on the surface (green nodes). The volume is cut based on the dotted cutting lines derived from the two tree structures (left) and unfolded into the parameter cube (right).

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 section. Furthermore, since we work on voxel data for rendering, subvoxel 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 pathlines, 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

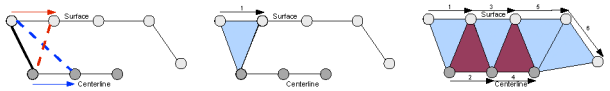
When dealing with surface parametrizations, cutting is required to be able to embed an arbitrary surface into the plane. Similarly, a volume has to be cut in order to embed it into the cube-shaped parameter space as shown in Fig. 3. To generate the required volumetric cut in step 2 of our algorithm, 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 Fig. 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 these 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, 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 Fig. 5(left)), we extend the adjacent skeleton-tree segment toward 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 Fig. 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 an 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,



(a) Initial positions with movement alternatives (red, blue). (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 by marching along the seam-tree and the skeleton-tree in a synchronized manner.

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 Fig. 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. Fig. 6a 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. Fig. 6b 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 Fig. 6c, the skeleton-tree cursor has finished and steps 5 and 6 create a triangle fan around this end of the centerline.

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 extendable 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 Fig. 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 Fig. 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 Fig. 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 Fig. 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 Fig. 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 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 toward 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 0. Our w distribution adaption is based on the observation that for objects of genus 0 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 Fig. 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. Fig. 2 shows the application of different texturing effects as commonly used for polygonal rendering. While classical direct volume rendering (DVR) has been applied in Fig. 2a, Fig. 2b shows the application of a color texture, which modifies the diffuse color as fed into the Phong illumination model. As shown in Fig. 2c, bump mapping now also becomes possible in the context of volume rendering, which has been combined with color mapping and specular highlights in Fig. 2d. Finally, Figs. 2e and 2f 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 Fig. 2g).

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 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 Fig. 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

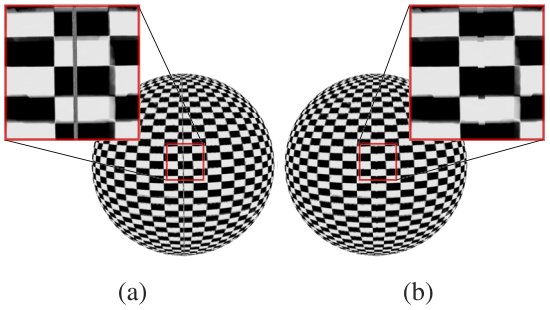


Fig. 7. A texture mapped spherical volume data set without (a) and with (b) applying a modified trilinear filter for the texture coordinate lookup. The adapted filtering results in smoother transitions along the seam.

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 Fig. 1b, 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 Fig. 7a. 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 (Fig. 7b). Since current graphics hardware enables such custom filtering mechanisms, we are able to implement this approach within a fragment shader.

Fig. 8 shows some effects we are able to achieve by applying the introduced concepts to real-world data. Fig. 8a 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 semitransparent surfaces can be improved [20], [24]. Fig. 8b 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

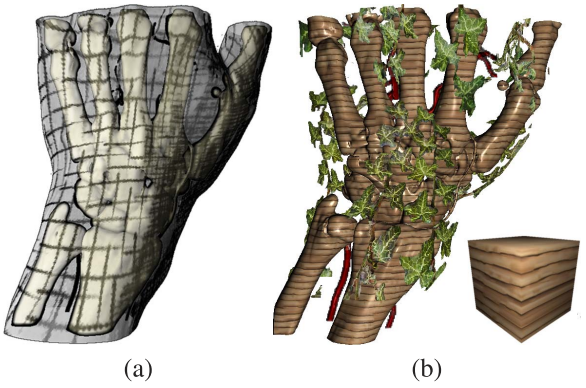


Fig. 8. CT scan of a hand using different texturing techniques. Cross-hatching (a) and the integration of a 3D wood texture with a 2D leaf texture (b).

the 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 Fig. 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). Due to the layering of skin and soft tissue in the CT scan, a more convincing appearance of the skin is achieved in Fig. 9d 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 layerwise in a consistent way. Fig. 10 shows two such cutaway illustrations,

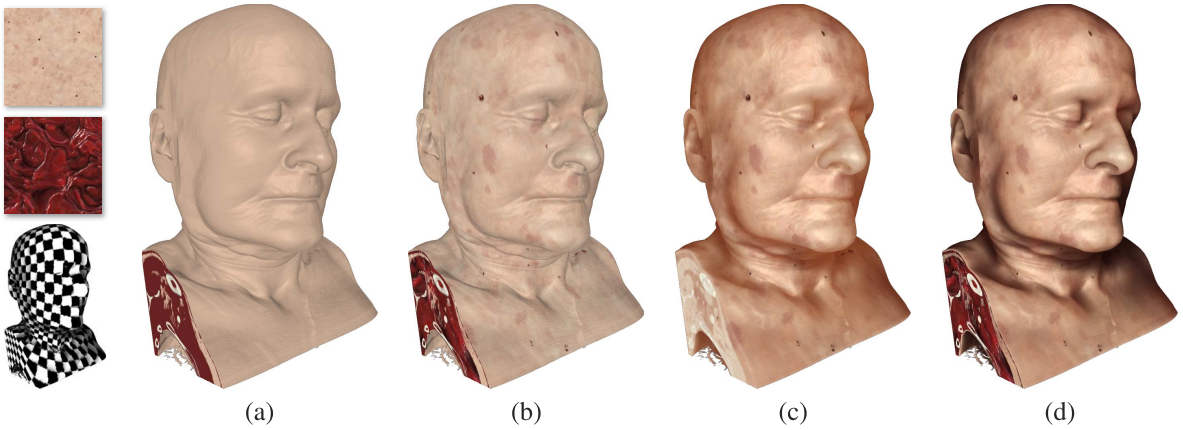


Fig. 9. Volume data set rendered using Phong shading without textures (a), Phong shading with textures (b), volumetric illumination using the same texture for soft tissue and skin (c), and volumetric illumination with different textures for soft tissue and skin (d). The insets show the used textures and the underlying parametrization.

which have been generated interactively by exploiting alpha mapping. Fig. 10a does not incorporate any translucency. In Fig. 10b, 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 Fig. 14a). 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 Fig. 14b.

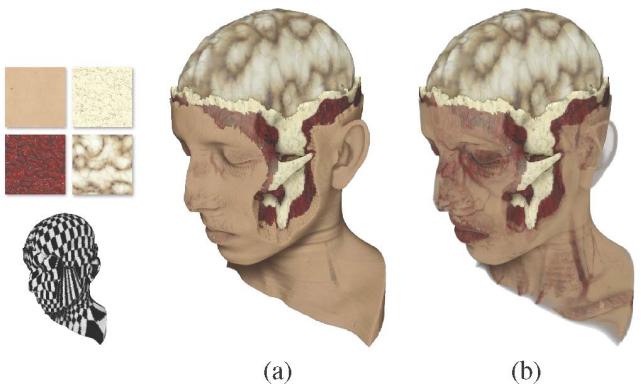


Fig. 10. Our technique allows to generate voxel-precise cutaways by combining textured volume rendering with alpha painting applied to individual textures (a). To emphasize the volumetric nature of the data set, the skin has been rendered translucently (b). The insets show the used textures and the parametrization.

5 Parametrization Analysis

5.1 Skeletonization Parameters

When transforming the level 0 skeleton into a level 1 skeleton by attaching pathlines 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 data sets with varying shapes in Fig. 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 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.

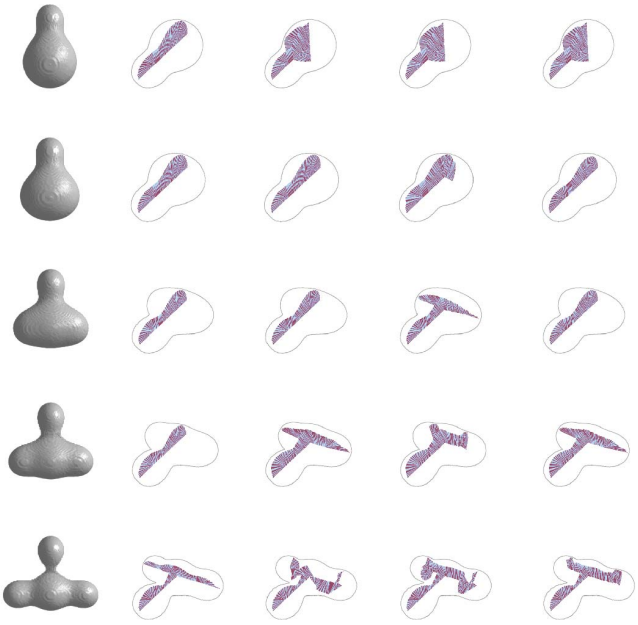


Fig. 11. Comparison of different orders m of the force field as well as the percentage of additional seeds with high divergence (from left to right: $(m = 6, 45\%)$, $(m = 6, 55\%)$, $(m = 6, 60\%)$, and $(m = 10, 45\%)$) and their influence on the resulting cut geometry for data sets with varying shape properties. The second value can be controlled interactively to obtain the desired results.

In Fig. 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 data set, 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 Section 3.1), we attempted to simulate this scenario as

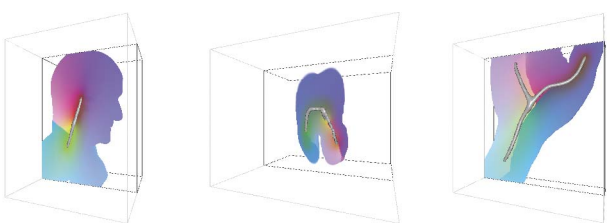


Fig. 12. Color-coded $<u, v, w>$ parametrizations for various data sets used within this paper. From left to right (head, tooth, hand), the structure of the skeleton-tree becomes more complex. The color coding has been achieved by directly mapping $<u, v, w>$ to $<r, g, b>$.

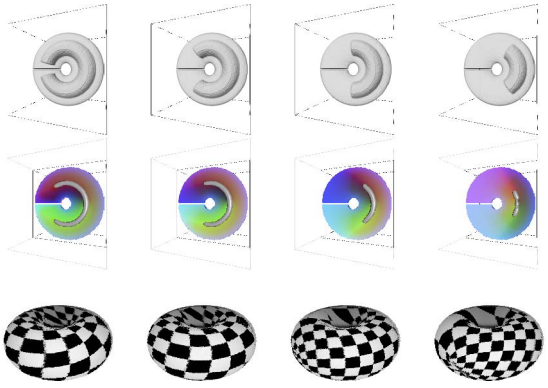


Fig. 13. To demonstrate the limitations of our algorithm, we have applied it to data sets containing nested layers of interest with increasingly varying shape. From left to right, the difference between the shapes of the two considered layers increases. As it can be seen, this results in an increasing distortion. The same color mapping has been used as in Fig. 12.

close as possible. Therefore, the data we apply our technique to are formed by an almost closed torus containing another torus segment (see Fig. 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 as possible. As it can be seen in Fig. 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.

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

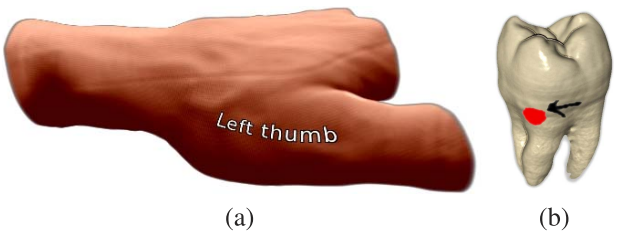


Fig. 14. The presented approach supports interactive volume annotation. Volumes can be annotated by adding text labels (a), or by drawing with a paint brush (b).

TABLE 1
Parametrization Comparison of Two Layers of the Parametrization Shown in Fig. 10

	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

The comparison is based on the L^2 stretch metric [49].

some of the concepts found 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. Fig. 18 shows a comparison of the distribution for the hand data set (see Fig. 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 Fig. 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 toward the center of the objects, it should be noted that this affects only a very limited fraction of all parametrized voxels. In Fig. 18b, we have emphasized this fraction with the gray rectangle, it lies below 3.8 percent of all voxels. We believe that the slight upslope and downslope toward $w = 0$ and $w = 1$ in Fig. 18b 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. Sander et al. introduce the L^2 surface stretch metric for a mesh M as $L^2(M) = \sqrt{\sum_{T_i \in M} (L^2(T_i))^2 A'(T_i) / \sum_{T_i \in M} A'(T_i)}$, 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 isosurfaces from the volumetric data set shown in Fig. 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 Fig. 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.

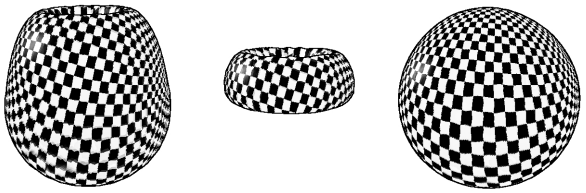
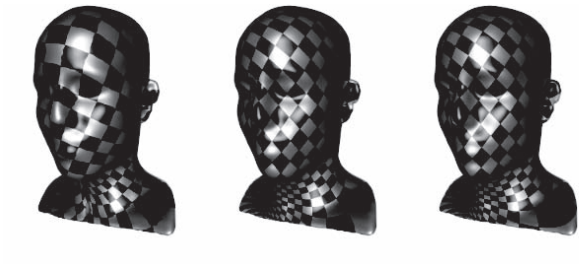


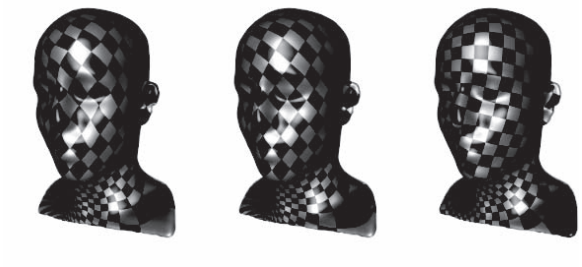
Fig. 15. By using our algorithm, meaningful $<u, v>$ parametrizations can be achieved also for surfaces not selected as boundary features. In (a), a parametrized boundary feature is shown, while in (b) and (c) we show the resulting $<u, v>$ parametrization for internal boundaries not considered during the parametrization.

When computing the $<u, v>$ parametrizations by more 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. Fig. 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 Fig. 15a as boundary feature during step 3. The images in Figs. 15b and 15c 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- versus 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.

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 Fig. 17, where different hues are used to depict the w texture coordinates. Since with our approach shown in Fig. 17a the brain surface has been incorporated as layer of interest, textures can be applied to it (see Fig. 10). This is not possible when using harmonic volumetric mapping as shown in Fig. 17b.



(a) our technique (b) Floater (c) conformal



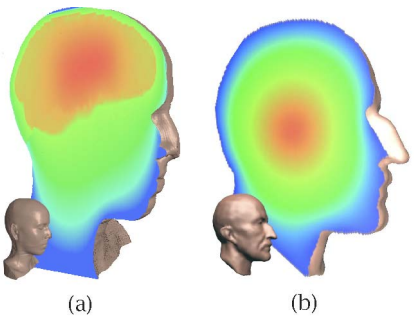
(d) barycentric (e) authalic (f) LSCM

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).

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.

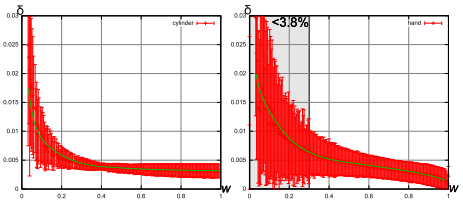
5.3 Stability Remarks

Finally, we would like to make some stability remarks. We have investigated all steps of our workflow depicted in Fig. 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 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



(a) (b)

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.



(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 percent of the data.

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 0. 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 SFB 656 MoBil (project Z1) funded by the Deutsche Forschungsgemeinschaft (DFG), by ELLIIT, the Strategic Area for ICT research, funded by the Swedish Government, and by the ViMaL project (FWF, no. P21695). The presented concepts have been integrated into the Voreen volume rendering engine (www.voreen.org).

REFERENCES

[1] A. Baer, C. Tietjen, R. Bade, and B. Preim, "Hardware-Accelerated Stippling of Surfaces Derived from Medical Volume Data," *Proc. IEEE/EG Symp. Visualization,* pp. 235-242, 2007.

[2] Y. Bhasin and A. Liu, "Bounds for Damping that Guarantee Stability in Mass-Spring Systems," *Medicine Meets Virtual Reality 14,* pp. 55-60, IOS Press, 2006.

[3] S. Bruckner and M.E. Gröller, "Style Transfer Functions for Illustrative Volume Rendering," *Computer Graphics Forum,* vol. 26, no. 3, pp. 715-724, 2007.

[4] K. Bürger, J. Krüger, and R. Westermann, "Direct Volume Editing," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1388-1395, Nov./Dec. 2008.

[5] M. Burns and A. Finkelstein, "Adaptive Cutaways for Comprehensible Rendering of Polygonal Scenes," *ACM Trans. Graphics,* vol. 27, no. 5, pp. 124:1-124:9, 2008.

[6] N.D. Cornea, D. Silver, and P. Min, "Curve-Skeleton Properties, Applications, and Algorithms," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 3, pp. 530-548, May/June 2007.

[7] N.D. Cornea, D. Silver, X. Yuan, and R. Balasubramanian, "Computing Hierarchical Curve-Skeletons of 3D Objects," *Visual Computer,* vol. 21, no. 11, pp. 945-955, 2005.

[8] B. Cutler, J. Dorsey, L. McMillan, M. Müller, and R. Jagnow, "A Procedural Approach to Authoring Solid Models," *Proc. ACM SIGGRAPH,* pp. 302-311, 2002.

[9] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," *Computer Graphics Forum,* vol. 21, no. 3, pp. 209-218, 2002.

[10] F. Dong and G. Clapworthy, "Volumetric Texture Synthesis for Non-Photorealistic Volume Rendering of Medical Data," *Visual Computer,* vol. 21, no. 7, pp. 463-473, 2005.

[11] Y. Dong, S. Lefebvre, X. Tong, and G. Drettakis, "Lazy Solid Texture Synthesis," *Computer Graphics Forum,* vol. 27, no. 4, pp. 1165-1174, 2008.

[12] C. Donner and H.W. Jensen, "Light Diffusion in Multi-Layered Translucent Materials," *Proc. ACM SIGGRAPH,* pp. 1032-1039, 2005.

[13] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. ACM SIGGRAPH,* pp. 173-182, 1995.

[14] M.S. Floater and K. Hormann, "Surface Parameterization: A Tutorial and Survey," *Advances in Multiresolution for Geometric Modelling Mathematics and Visualization,* pp. 157-186, Springer, 2005.

[15] M.S. Floater, G. Kós, and M. Reimers, "Mean Value Coordinates in 3D," *Computer Aided Geometric Design,* vol. 22, pp. 623-631, 2005.

[16] P. Hanrahan and W. Krueger, "Reflection from Layered Surfaces due to Subsurface Scattering," *Proc. ACM SIGGRAPH,* pp. 165-174, 1993.

[17] X. Hao, T. Baby, and A. Varshney, "Interactive Subsurface Scattering for Translucent Meshes," *Proc. ACM Symp. Interactive 3D Graphics (I3D),* pp. 75-82, 2003.

[18] *The Guild Handbook of Scientific Illustration,* E.R.S. Hodges, ed., second ed. John Wiley & Sons, 2003.

[19] K. Hormann, K. Polthier, and A. Sheffer, "Mesh Parameterization: Theory and Practice," *Proc. 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 Trans. Visualization and Computer Graphics,* vol. 3, no. 2, pp. 98-117, Apr.-June 1997.

[21] H.W. Jensen, S.R. Marschner, M. Levoy, and P. Hanrahan, "A Practical Model for Subsurface Light Transport," *Proc. ACM SIGGRAPH,* pp. 511-518, 2001.

[22] T. Ju, S. Schaefer, and J. Warren, "Mean Value Coordinates for Closed Triangular Meshes," *Proc. ACM SIGGRAPH,* pp. 561-566, 2005.

[23] I. Kabul, D. Merck, J. Rosenman, and S.M. Pizer, "Model-Based Solid Texture Synthesis for Anatomic Volume Illustration," *Proc. EG Workshop Visual Computing for Biology and Medicine,* pp. 133-140, 2010.

[24] S. Kim, H. Hagh-Shenas, and V. Interrante, "Conveying Three-Dimensional Shape with Texture," *Proc. ACM Symp. Applied Perception in Graphics and Visualization (APGV),* pp. 119-122, 2004.

[25] G. Kindlmann and J.W. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," *Proc. IEEE Symp. Vol. Visualization,* pp. 79-86, 1998.

[26] J. Kniss, G. Kindlmann, and C. Hansen, "Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets," *Proc. IEEE Visualization (VIS '01),* pp. 255-262, 2001.

[27] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson, "A Model for Volume Lighting and Modeling," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 2, pp. 150-162, Apr.-June 2003.

[28] J. Kopf, C.-W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, and T.-T. Wong, "Solid Texture Synthesis from 2D Exemplars," *Proc. ACM Trans. Graphics,* vol. 26, no. 3, pp. 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 Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1491-1498, Nov./Dec. 2008.

[30] Y. Kurzion, T. Möller, and R. Yagel, "Size Preserving Pattern Mapping," *Proc. IEEE Visualization,* pp. 367-373, 1998.

[31] Y. Lee, H. Kim, and S. Lee, "Mesh Parameterization with a Virtual Boundary," *Computers & Graphics,* vol. 26, no. 5, pp. 677-686, 2002.

[32] S. Lefebvre and C. Dachsbacher, "Tiletrees," *Proc. ACM Symp. Interactive 3D Graphics and Games (I3D),* pp. 25-31, 2007.

[33] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications,* vol. 8, no. 3, pp. 29-37, May 1988.

[34] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *Proc. ACM SIGGRAPH,* pp. 173-182, 2002.

[35] X. Li, X. Guo, H. Wang, X. Gu, and H. Qin, "Meshless Harmonic Volumetric Mapping Using Fundamental Solution Methods," *IEEE Trans. Automation Science and Eng.,* vol. 6, no. 3, pp. 409-422, July 2009.

[36] X. Li, X. Guo, H. Wang, Y. He, X. Gu, and H. Qin, "Harmonic Volumetric Mapping for Solid Modeling Applications," *Proc. ACM Symp. Solid and Physical Modeling (SPM),* pp. 109-120, 2007.

[37] X. Li, H. Xu, S. Wan, Z. Yin, and W. Yu, "Feature-Aligned Harmonic Volumetric Mapping Using Mfs," *SMI '10: Proc. Int'l Conf. Shape Modeling,* 2010.

[38] A. Lu, D.S. Ebert, W. Qiao, M. Kraus, and B. Mora, "Volume Illustration Using Wang Cubes," *ACM Trans. Graphics,* vol. 26, no. 2, article 11, 2007.

[39] F. Manke and B.C. Wuensche, "Texture-Enhanced Direct Volume Rendering," *Proc. Int'l Conf. Computer Graphics Theory and Applications (GRAPP),* pp. 185-190, 2009.

[40] T. Martin, E. Cohen, and M. Kirby, "Volumetric Parameterization and Trivariate b-Spline Fitting Using Harmonic Functions," *SPM '08: Proc. ACM Symp. Solid and Physical Modeling,* pp. 269-280, 2008.

[41] C.M. Miller and M.W. Jones, "Texturing and Hypertexturing of Volumetric Objects," *Proc. IEEE/EG Symp. Vol. Graphics,* pp. 117-125, 2005.

[42] S. Owada, F. Nielsen, M. Okabe, and T. Igarashi, "Volumetric Illustration: Designing 3D Models with Internal Textures," *Proc. ACM SIGGRAPH,* pp. 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 Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1571-1578, Nov./Dec. 2008.

[44] D. Patel, C. Giertsen, J. Thurmond, and E. Gröller, "Illustrative Rendering of Seismic Data," *Proc. Vision, Modeling, and Visualization,* pp. 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,* vol. 26, no. 3, pp. 637-644, 2007.

[46] S.D. Porumbescu, B. Budge, L. Feng, and K.I. Joy, "Shell Maps," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 626-633, 2005.

[47] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-Time Hatching," *Proc. ACM SIGGRAPH,* pp. 581-586, 2001.

[48] T. Ropinski, J.-S. Praßni, J. Roters, and K. Hinrichs, "Internal Labels as Shape Cues for Medical Illustration," *Proc. Vision, Modeling, and Visualization,* pp. 203-212, 2007.

[49] P.V. Sander, J. Snyder, S.J. Gortler, and H. Hoppe, "Texture Mapping Progressive Meshes," *Proc. ACM SIGGRAPH,* pp. 1032-1039, 2005.

[50] R. Satherley and M.W. Jones, "Hypertexturing Complex Volume Objects," *Visual Computer,* vol. 18, no. 4, pp. 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,* vol. 28, no. 3, pp. 855-862, 2009.

[52] A. Sheffer and J. Hart, "Seamster: Inconspicuous Low-Distortion Texture Seam Layout," *Proc. IEEE Conf. Visualization,* pp. 291-298, 2002.
[53] P. Shen and P. Willis, "Texture for Volume Character Animation," *Proc. ACM Int'l Conf. Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE),* pp. 255-264, 2005.
[54] P. Shen and P. Willis, "Texture Mapping Volume Objects," *Proc. Int'l Conf. Vision, Video, and Graphics,* pp. 45-52, 2005.
[55] S. Takahashi, I. Fujishiro, and Y. Takeshima, "Interval Volume Decomposer: A Topological Approach to Volume Traversal," *Proc. SPIE Conf. Visualization and Data Analysis,* pp. 103-114, 2005.
[56] K. Takayama and T. Igarashi, "Layered Solid Texture Synthesis from a Single 2D Exemplar," *Proc. 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. Graphics,* vol. 27, no. 3, pp. 1-9, 2008.
[58] S.M.F. Treavett and M. Chen, "Pen-and-Ink Rendering in Volume Visualisation," *Proc. IEEE Visualization,* pp. 203-210, 2000.
[59] W.T. Tutte, "How to Draw a Graph," *London Math. Soc.,* vol. 13, no. 52, pp. 743-768, 1963.
[60] G.H. Weber, S.E. Dillard, H. Carr, V. Pascucci, and B. Hamann, "Topology-Controlled Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 2, pp. 330-341, Mar.-Apr. 2007.
[61] S. Yoshizawa, A. Belyaev, and H.-P. Seidel, "A Fast and Simple Stretch-Minimizing Mesh Parameterization," *SMI '04: Proc. Int'l Conf. Shape Modeling,* pp. 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. Graphics,* vol. 25, no. 3, pp. 690-697, 2006.
[63] M. Zwicker, M. Pauly, O. Knoll, and M. Gross, "Pointshop 3D: An Interactive System for Point-Based Surface Editing," *Proc. ACM SIGGRAPH,* pp. 322-329, 2002.

Timo Ropinski is a 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, IEEE Computer Society, ACM, and Eurographics.

Stefan Diepenbrock received the master's degree in 2009 from the University of Münster. He is currently working toward the PhD degree at the University of Münsters Visualization and Computer Graphics Research Group. 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 the PhD degree 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 PhD 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 coauthored more than 190 scientific publications and acted as a cochair, IPC member, and reviewer for numerous conferences and journals in the field. He is currently a cochief editor of the *Computer Graphics Forum* journal. He is a member of the IEEE Computer Society.