DOI: 10.1111/cgf.14296
Eurographics Conference on Visualization (EuroVis) 2021
R. Borgo, G. E. Marai, and T. von Landesberger
(Guest Editors)

Volume 40 (2021), Number 3

Thin-Volume Visualization on Curved Domains

Felix Herter[1], Hans-Christian Hege[1], Markus Hadwiger[2], Verena Lepper[3] and Daniel Baum[1]

[1]Zuse Institute Berlin (ZIB), Department of Visual and Data-Centric Computing, Berlin, Germany
[2]King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
[3]Ägyptisches Museum und Papyrussammlung, Staatliche Museen zu Berlin, Berlin, Germany

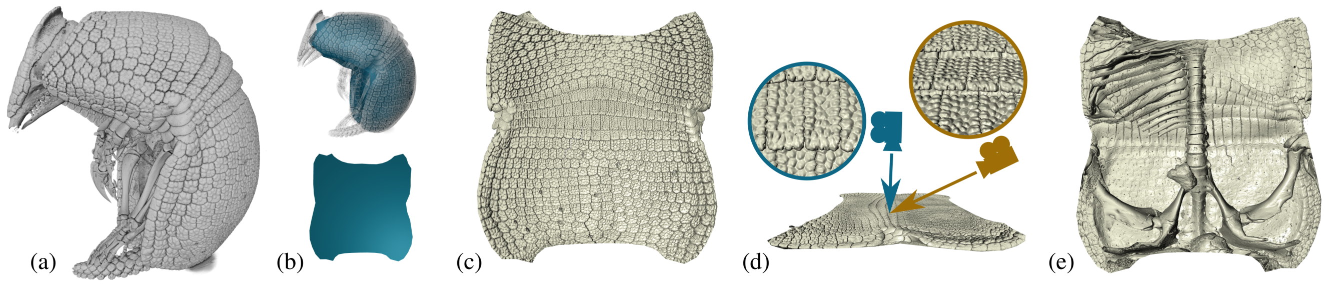

Figure 1: *Thin iso-surface rendering of an armadillo carapace: (a) volume rendering of the dataset; (b) corresponding surface meshes $\mathcal{M}_{\text{curv}}$ and $\mathcal{M}_{\text{flat}}$; (c) thin iso-surface rendering of a view from the outside; (d) a free moving camera enables different views of the curved domain; (e) thin iso-surface rendering of a view from the inside with part of the skeleton interactively removed by shortening the ray lengths.*

Abstract

Thin, curved structures occur in many volumetric datasets. Their analysis using classical volume rendering is difficult because parts of such structures can bend away or hide behind occluding elements. This problem cannot be fully compensated by effective navigation alone, as structure-adapted navigation in the volume is cumbersome and only parts of the structure are visible in each view. We solve this problem by rendering a spatially transformed view of the volume so that an unobstructed visualization of the entire curved structure is obtained. As a result, simple and intuitive navigation becomes possible. The domain of the spatial transform is defined by a triangle mesh that is topologically equivalent to an open disc and that approximates the structure of interest. The rendering is based on ray-casting, in which the rays traverse the original volume. In order to carve out volumes of varying thicknesses, the lengths of the rays as well as the positions of the mesh vertices can be easily modified by interactive painting under view control.

We describe a prototypical implementation and demonstrate the interactive visual inspection of complex structures from digital humanities, biology, medicine, and material sciences. The visual representation of the structure as a whole allows for easy inspection of interesting substructures in their original spatial context. Overall, we show that thin, curved structures in volumetric data can be excellently visualized using ray-casting-based volume rendering of transformed views defined by guiding surface meshes, supplemented by interactive, local modifications of ray lengths and vertex positions.

CCS Concepts

• *Human-centered computing* → *Scientific visualization; Visualization techniques;* • *Computing methodologies* → *Rendering;*

1. Introduction

Direct volume rendering (DVR) is a well-established, versatile tool for visualizing volumetric data. However, if a volume contains very many or large structures that occlude other structures, DVR may not lead to the desired result. Adjusting transfer functions [LKG*16] or utilizing volume editing [BKW08] to remove occluding structures can partially solve this problem, but this process can be cumbersome and time-consuming. A particular problem are structures that are bent and thus occlude other parts of the same structure, so that removal of these structures by means of transfer functions or volume editing is not possible. Some applications also require the entire structure to be visible at once to minimize navigation and allow easy comparison of multiple datasets in a single view.

For certain types of structures, special solutions have been developed to address these issues. For example, for tubular structures like colons or vessels, curved planar reformation [KFW*02] has been developed. Another important, ubiquitous structure type are thin, sheet-like volumetric structures whose thickness may vary spatially. Examples include tomographed documents that are rolled and folded [BLLR15,LRLH17,BLH*17], consisting of writing materials of various kinds, such as papyrus, parchment, lead, or silver.

While materials such as lead, silver and parchment are relatively easy to segment in the volume data, the same is not true for antique papyri, which are often in very poor condition. For such objects, it is very difficult to develop a method that reliably and automatically segments the writing medium. Data of this kind, namely ancient papyri, which are rolled and folded, and sometimes severely damaged, motivated the development of the interactive thin-volume visualization presented here. However, as we will show, the method can be usefully applied to a much broader class of problems and applications.

In this paper, we present a ray-casting-based visualization technique for representing thin, volumetric structures that

- provides complete, unobstructed views;
- is generic in the sense that it can be applied to a large variety of datasets and applications;
- supports the interactive visual analysis, demonstrated by two intuitive editing tools;
- allows for effective, structure-adaptive navigation in the volume.

The technique assumes the existence of two triangular meshes, a curved one, $\mathcal{M}_{\text{curv}}$, and a flat one, $\mathcal{M}_{\text{flat}}$. Both meshes need to be in a one-to-one correspondence. While $\mathcal{M}_{\text{curv}}$ is assumed to trace the structure of interest inside the volume, $\mathcal{M}_{\text{flat}}$ is used to render it. Creating $\mathcal{M}_{\text{curv}}$ can be tedious for some data, but rather easy for others. In many cases, $\mathcal{M}_{\text{curv}}$ only needs to roughly approximate the structure of interest and can be further adapted by the presented user interactions. For the examples given in Sect. 6, we will briefly describe how we created the curved and flattened surface meshes. However, we will not describe how such surfaces can be generated in general, as this is very data-specific. One way to obtain *corresponding* surfaces is to generate $\mathcal{M}_{\text{flat}}$ from $\mathcal{M}_{\text{curv}}$ by flattening. For flattening, we make use of state-of-the-art methods, see for example the works by Floater et al. [FH05], Sheffer et al. [SPR06], or Li and Iyengar [LI14]. Thus, this work does neither make contributions to the field of automatic surface mesh generation from image data nor to the field of mesh flattening.

2. Related Work

The method presented in this paper is related to unrolling, unfolding, or generally flattening of thin volumetric structures embedded in a larger 3D volume. The terms “unrolling” and “unfolding” imply that the structure to be unrolled or unfolded was originally a (nearly) planar, thin volumetric structure, or at least that its medial surface can be mapped almost isometrically onto a plane, i.e., that it is developable. The term “flattening” is more general and usually refers to methods that also work for structures that can only be unrolled or unfolded with some distortion.

In the following, we will review methods that fall into the above categories as well as methods that generate transformed views of the data. Since the goal of the presented method is the visualization of thin volumetric structures, in this section we will focus on methods that work with volumetric data. There are a few survey papers that have reviewed methods and applications related to our method, including the recent report on flattening in medical applications by Kreisler et al. [KMM*18] and the survey on manipulating sampled object representations by Chen et al. [CCI*07]. We will review selected publications from these surveys as well as additional ones not mentioned there.

An application class in which transformed views have long played an important role is the visualization and analysis of vessels and colons in medicine. Here, transformed views are necessary to allow an unobstructed view of elongated structures that bend in the 3D images recorded using CT or MRI. For these structures, curved planar reformations (CPRs) [KFW*02] were developed. A CPR starts from the centerline of an elongate structure and spans a surface by extending the centerline in opposite directions from each point on the centerline. The result is a parametrized surface that is then used for volume reformation. The proposed CPR methods differ slightly in the exact way the surfaces are spanned [KFW*02, Kan04]. While most CPR methods only generate a cut through the 3D image or a projection, Williams et al. [WGC*07] extend the method by combining it with direct volume rendering (DVR), thus allowing the user to obtain a 3D impression of the colon inside. Instead of placing a local plane through the elongated structure, other methods display the neighborhood of the centerline by sending radial rays off from many points on the centerline. Vilanova et al. [VBWKG01] use nonlinear ray-casting for unfolding the entire inner surface of the colon in order to avoid the duplicate rendering of structures in strongly bent regions. In a similar spirit, Lampe et al. [LCMH09] also use inside-out rendering from the centerline and apply it to comparison of vessel structures and stream lines. In contrast, Hong et al. [HGQ*06] use a conformal mapping to parametrize the colon surface. In combination with DVR, they visually unfold the colon surface.

Apart from vessels and colons, other anatomical structures have also been approached using flattening techniques [MPG*17, KST*14]. While Martinke et al. [MPG*17] extend previous approaches by creating shape-adaptive unfoldings to better reflect the cross-sectional shape of the structure of interest, Kretschmer et al. [KST*14] use generic 2D surface meshes that serve to flatten the structures of interest, for example, the ribcage, pelvis, or feet. They apply as-rigid-as-possible volume parametrization and resample the image data into a new, flattened space in which they can then utilize all visualization techniques available for 3D image data.

Unrolling and unfolding techniques have also been applied to data from digital humanities and materials science. In digital humanities, usually written documents need to be unfolded, including parchment rolls [LRLH17], silver packages [BLLR15], and papyrus rolls and packages [BLH*17]. In materials science, unrolling is particularly interesting for the analysis of batteries [ZAF*20]. Due to the small thickness of the structures in all these applications, it is critical that both the position of the mesh used for unfolding and the thickness are well defined so that no important information is overlooked.

Additionally, the fields of interactive volume editing [BKW08] and interactive volume deformation [WRS01, CCI*07] are also related to our work. However, to the best of our knowledge, none of

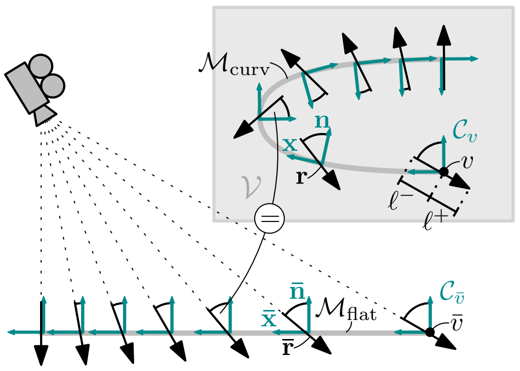

Figure 2: *Correspondence of rays for volume rendering. Each view ray* $\bar{\mathbf{r}}$ *is transformed from a coordinate system* $\mathcal{C}_{\bar{v}} = \{\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{n}}\}$ *for* $\mathcal{M}_{\text{flat}}$*, to the corresponding ray* $\mathbf{r}$ *in* $\mathcal{C}_v = \{\mathbf{x}, \mathbf{y}, \mathbf{n}\}$ *for* $\mathcal{M}_{\text{curv}}$.

these methods has ever been applied to the reformatting volume visualizations reviewed in this section.

Most similar to our approach are the works by Hong et al. [HGQ*06] and Williams et al. [WGC*07], which both provide approaches based on ray-casting to the unfolding of tubular structures. Both use curves to define the structure of interest inside the volume. However, the work by Hong et al. [HGQ*06] only supports fixed cameras, which do not allow to interact with the rendering as one is used to in common volume rendering. Williams et al. [WGC*07], on the other hand, generate a ruled surface by expanding the curve, such that a direction without curvature exists at every point on the surface. Although technically a surface, the supported types of surfaces and their construction are too restrictive for tracing general surface-like structures.

The approach described here allows both to freely move the camera, which eases the 3D understanding, and the use of generic 2D surface meshes similar to the work by Kretschmer et al. [KST*14]. However, unlike the latter work, we provide direct rendering based on ray-casting, instead of resampling the volume. This allows us to interactively modify both the position of the surface mesh and the length of the rays. We consider the resulting flexibility as major advantage of our method.

3. Rendering Approach

In classical volume rendering using standard GPU ray-casting, rays are usually set up by rasterizing some form of *proxy geometry*, for example the bounding box of the volume [EHK*04]. The volume is then sampled along the ray from a start to an end position, e.g., from where the ray enters the bounding box to where it exits again.

This section describes our approach for thin-volume rendering. While we used a perspective camera model, the technique can be readily adapted to implement an orthographic camera model.

3.1. Ray Setup in Thin-Volume Rendering

In contrast to the standard approach, in the thin-volume rendering framework the proxy geometry used for ray-casting consists of a flattened mesh $\mathcal{M}_{\text{flat}}$, as illustrated in Fig. 2. However, while rays $\bar{\mathbf{r}}$ originally correspond to the direction from the camera to points on the mesh $\mathcal{M}_{\text{flat}}$, before the volume $\mathcal{V}$ can be sampled, each ray must first be transformed into the volume space of $\mathcal{V}$, giving the corresponding transformed rays $\mathbf{r}$. This transformation is determined by the curved geometry of the mesh $\mathcal{M}_{\text{curv}}$, which is embedded in $\mathcal{V}$. The volume is then sampled along the transformed rays $\mathbf{r}$, using user-specified ray segment lengths in front and behind the prescribed geometry $\mathcal{M}_{\text{curv}}$, determining the view-dependent thickness of the *thin volume* around the curved domain described by $\mathcal{M}_{\text{curv}}$.

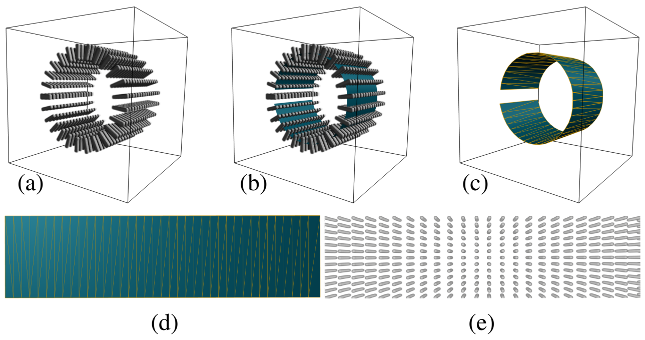

Figure 3: *Volume rendering using transformed rays. (Top) A test volume with the surface* $\mathcal{M}_{\text{curv}}$ *shown embedded. (Bottom) The flattened mesh* $\mathcal{M}_{\text{flat}}$ *comprises the proxy geometry for the volume rendering using transformed view rays on the right.*

The meshes $\mathcal{M}_{\text{flat}}$ and $\mathcal{M}_{\text{curv}}$ are required to be in one-to-one correspondence, i.e., they have to have the same number and ordering of vertices and faces, and the same connectivity. Because the mesh $\mathcal{M}_{\text{flat}}$ is flat, we obtain a view into the volume that is free of occlusion, however guided by the curved geometry of $\mathcal{M}_{\text{curv}}$. Fig. 3 depicts a simple example. The mesh $\mathcal{M}_{\text{flat}}$ can also be used directly for user input and visual feedback. Because $\mathcal{M}_{\text{curv}}$ is embedded in the volume $\mathcal{V}$, it traces the desired structure of interest, and it also provides supporting information about the structure's geometry.

The goal of the ray transformations is to produce volume renderings that feel natural. If the camera looks at $\mathcal{M}_{\text{flat}}$ from a direction close to orthogonal to it (taking into account the diverging rays in a perspective camera), the volume $\mathcal{V}$ will be sampled along rays that are close to orthogonal to $\mathcal{M}_{\text{curv}}$ (i.e., locally close to orthogonal to the tangent plane at each given point on the curved surface approximated by $\mathcal{M}_{\text{curv}}$). Likewise, when the viewing direction onto $\mathcal{M}_{\text{flat}}$ is tilted, the rays along which the volume $\mathcal{V}$ is sampled are tilted the same way, at the same angles, relative to $\mathcal{M}_{\text{curv}}$ (its tangent planes).

3.2. Ray Transformation with Adapted Coordinate Frames

The key to transforming view rays from $\mathcal{M}_{\text{flat}}$ to $\mathcal{M}_{\text{curv}}$, embedded in $\mathcal{V}$, is to set up suitably corresponding coordinate frames for each point $\bar{p} \in \mathcal{M}_{\text{flat}}$ and the corresponding point $p \in \mathcal{M}_{\text{curv}}$. To achieve this in a way that is well adapted to the curved surface deformation represented by the given mesh $\mathcal{M}_{\text{curv}}$, we use a method based on (overcomplete) frames [CK13], as described below. In contrast to standard tangent space normal mapping [AMHH08], our method essentially computes a least-squares fit over each 1-ring vertex neighborhood to capture the deformation of $\mathcal{M}_{\text{curv}}$ more ac-

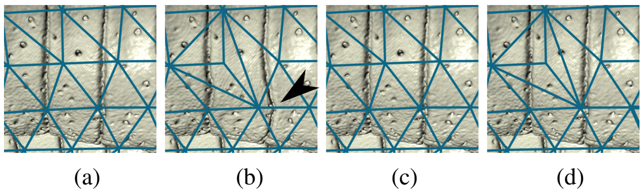

Figure 4: *Comparison of tangent space basis transformations for two different triangulations of $\mathcal{M}_{\text{flat}}$ and $\mathcal{M}_{\text{curv}}$: (a) and (b) show the results when transforming based on a single tangent space basis pair; (c) and (d) show the results using our frame-based approach. While the standard approach shows deviations (arrow), our frame-based approach renders the same result for both triangulations.*

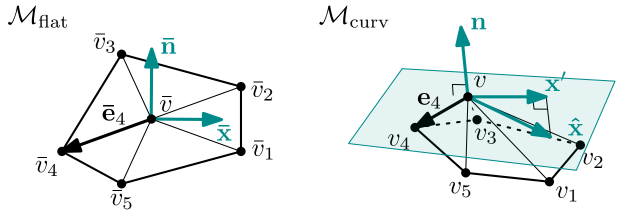

Figure 5: *For the transformation of view rays from $\mathcal{M}_{\text{flat}}$ to $\mathcal{M}_{\text{curv}}$, we compute how the neighborhood of vertex $\bar{v} \in \mathcal{M}_{\text{flat}}$ transfers to the neighborhood of vertex $v \in \mathcal{M}_{\text{curv}}$, by using two corresponding overcomplete frames $\{\bar{\mathbf{e}}_k\}, \{\mathbf{e}_k\}$ of K vectors (here, $K = 5$), computed from the 1-ring vertex neighborhoods of $\bar{v}$ and v, respectively.*

curately than just transforming a single tangent space basis. This is particularly helpful in making the ray transformation less dependent on the triangulation, and it does not require a surface parameterization. A quality comparison illustrating the benefit of our frame-based approach is depicted in Fig. 4.

Proxy Geometry and Correspondence with Curved Domain

The main proxy geometry used for ray-casting is the flat mesh $\mathcal{M}_{\text{flat}}$, and the corresponding curved mesh $\mathcal{M}_{\text{curv}}$ determines the transformation of view rays required to obtain the thin-volume rendering relative to the curved domain described by $\mathcal{M}_{\text{curv}}$.

In our implementation, both $\mathcal{M}_{\text{flat}}$ and $\mathcal{M}_{\text{curv}}$ are triangle meshes. We label the vertex set of $\mathcal{M}_{\text{flat}}$ as $\{\bar{v}_1, \bar{v}_2, \cdots, \bar{v}_n\}$, and that of $\mathcal{M}_{\text{curv}}$ as $\{v_1, v_2, \cdots, v_n\}$. Every vertex $\bar{v}_i$ in $\mathcal{M}_{\text{flat}}$ has a corresponding vertex v_i in $\mathcal{M}_{\text{curv}}$. Furthermore, we require that $\mathcal{M}_{\text{flat}}$ and $\mathcal{M}_{\text{curv}}$ have the same topology. That is, $(\bar{v}_i, \bar{v}_j)$ form an edge in $\mathcal{M}_{\text{flat}}$ if and only if (v_i, v_j) form an edge in $\mathcal{M}_{\text{curv}}$. For points in the interior of triangles, the correspondence between $\mathcal{M}_{\text{flat}}$ and $\mathcal{M}_{\text{curv}}$ is established in a straightforward manner via barycentric interpolation: For a point $\bar{p}$ in a triangle $(\bar{v}_i, \bar{v}_j, \bar{v}_k)$, we can write $\bar{p} = \alpha\bar{v}_i + \beta\bar{v}_j + \gamma\bar{v}_k$, with barycentric coordinates (α, β, γ). The corresponding point p in $\mathcal{M}_{\text{curv}}$, in the corresponding triangle (v_i, v_j, v_k), is simply $p = \alpha v_i + \beta v_j + \gamma v_k$.

Coordinate Frame Transformation to the Curved Domain

To prepare for transforming view rays $\bar{\mathbf{r}}$ from $\mathcal{M}_{\text{flat}}$ to $\mathcal{M}_{\text{curv}}$, embedded in the volume $\mathcal{V}$, we first define an orthonormal basis $\mathcal{C}_{\bar{v}} = \{\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{n}}\}$ for every vertex $\bar{v} \in \mathcal{M}_{\text{flat}}$. The three basis vectors are identical for all vertices, and are chosen such that they correspond to a right-handed coordinate system, where the vector $\bar{\mathbf{n}}$ is normal to the surface $\mathcal{M}_{\text{flat}}$. For each vertex $\bar{v}$, we then transform this basis to a corresponding transformed basis $\mathcal{C}_v = \{\mathbf{x}, \mathbf{y}, \mathbf{n}\}$ at the corresponding vertex $v \in \mathcal{M}_{\text{curv}}$, using the method described below. Fig. 2 depicts this simplified to the 2D case. Furthermore, to obtain a basis at every point $p \in \mathcal{M}_{\text{curv}}$, including the interior of triangles, we interpolate the basis vectors using the barycentric interpolation described above, followed by re-normalization and orthogonalization of the interpolated bases. This results in an interpolated transformed basis $\mathcal{C}_p = \{\mathbf{x}, \mathbf{y}, \mathbf{n}\}$, at any point $p \in \mathcal{M}_{\text{curv}}$, which will afterward be used for the transformation of view rays.

To obtain the basis $\mathcal{C}_v = \{\mathbf{x}, \mathbf{y}, \mathbf{n}\}$ at the vertex $v \in \mathcal{M}_{\text{curv}}$, we first obtain the normal vector $\mathbf{n}$ directly as the normal vector of $\mathcal{M}_{\text{curv}}$. However, the other two basis vectors $\mathbf{x}$ and $\mathbf{y}$ must be computed such that they adapt to the deformation of $\mathcal{M}_{\text{flat}}$ to $\mathcal{M}_{\text{curv}}$. To compute a basis vector $\mathbf{x}$ that is adapted to the transformation of a whole neighborhood on the 2-manifold $\mathcal{M}_{\text{flat}}$, into the corresponding neighborhood on the 2-manifold $\mathcal{M}_{\text{curv}}$, we want to take into account the whole 1-ring neighborhood of the vertex $\bar{v}$, transforming into the corresponding 1-ring neighborhood of v. In order to achieve this, instead of directly transforming a coordinate basis for the tangent plane of $\mathcal{M}_{\text{flat}}$ at the vertex $\bar{v}$, consisting of two linearly independent vectors, we use a *frame* $\{\bar{\mathbf{e}}_k\}_{k=1}^{K}$, which is a spanning set comprising K vectors [CK13], where for 2-manifolds $K \geq 2$.

We compute the frame vectors $\bar{\mathbf{e}}_k$ from the 1-ring vertices $\{\bar{v}_k\}$, around the center vertex $\bar{v}$, as $\bar{\mathbf{e}}_k := \bar{v}_k - \bar{v}$. This is illustrated in Fig. 5 for $K = 5$. Because this frame is neither orthogonal nor linearly independent, for the computation of frame coefficients with respect to $\{\bar{\mathbf{e}}_k\}$, we have to use the corresponding *dual frame* $\{\tilde{\mathbf{e}}_k\}$. We use the *canonical dual frame* [HKLW07, p.157], given by

$$\tilde{\mathbf{e}}_k := \mathrm{S}^{-1}\,\bar{\mathbf{e}}_k. \tag{1}$$

Here we need S^{-1}, the inverse of the frame operator S, given by

$$\mathrm{S}\mathbf{v} := \sum_k \langle \mathbf{v}, \bar{\mathbf{e}}_k \rangle\, \bar{\mathbf{e}}_k, \quad \text{for any vector } \mathbf{v}. \tag{2}$$

In components, we can compute the frame operator S as the matrix $\mathrm{S} = \mathrm{T}^T\mathrm{T}$, with the analysis operator T given by the $K \times 2$ matrix

$$\mathrm{T} := \begin{bmatrix} — & \bar{\mathbf{e}}_1 & — \\ — & \bar{\mathbf{e}}_2 & — \\ & \vdots & \\ — & \bar{\mathbf{e}}_K & — \end{bmatrix}. \tag{3}$$

In our case, the frame operator S is therefore given by the 2×2 matrix $\mathrm{S} = \mathrm{T}^T\mathrm{T}$. This matrix is non-singular, because otherwise $\{\bar{\mathbf{e}}_k\}$ would not be a spanning set. Thus, we can directly compute S^{-1}.

By using the dual frame $\{\tilde{\mathbf{e}}_k\}$, the expansion of $\bar{\mathbf{x}}$ with respect to the frame $\{\bar{\mathbf{e}}_k\}$ can now be written using frame coefficients c_k, as

$$\bar{\mathbf{x}} = \sum_k c_k(\bar{\mathbf{x}})\, \bar{\mathbf{e}}_k, \quad \text{with} \quad c_k(\bar{\mathbf{x}}) := \langle \bar{\mathbf{x}}, \tilde{\mathbf{e}}_k \rangle. \tag{4}$$

We can now compute the transformed vector $\mathbf{x}'$, which is the vector $\bar{\mathbf{x}}$ transformed from $\mathcal{M}_{\text{flat}}$ to $\mathcal{M}_{\text{curv}}$, which we then only have to normalize to obtain the final vector $\mathbf{x}$. We first compute an expansion with the frame coefficients c_k computed for $\mathcal{M}_{\text{flat}}$, however we expand with the *transformed* frame $\{\mathbf{e}_k\}$ at the vertex $v \in \mathcal{M}_{\text{curv}}$, which is given by $\mathbf{e}_k := v_k - v$ (Fig. 5). We then project into the tangent plane orthogonal to the normal vector $\mathbf{n}$. This is given by

$$\mathbf{x}' = \hat{\mathbf{x}} - \langle \hat{\mathbf{x}}, \mathbf{n} \rangle \mathbf{n}, \quad \text{with} \quad \hat{\mathbf{x}} = \sum_k c_k(\bar{\mathbf{x}})\, \mathbf{e}_k, \;\; c_k(\bar{\mathbf{x}}) := \langle \bar{\mathbf{x}}, \tilde{\mathbf{e}}_k \rangle. \quad (5)$$

The *canonical* dual (Eq. (1)) chooses the coefficient vector (c_k) of minimum L_2 norm in Eq. (4). In this sense, (c_k) in Eq. (5) gives a least-squares fit to the tangential component of the deformation around v, while the normal component is captured by the prescribed vector $\mathbf{n}$. The final transformed orthonormal basis $\{\mathbf{x}, \mathbf{y}, \mathbf{n}\}$ is then

$$\mathbf{x} = \mathbf{x}'/\|\mathbf{x}'\|, \;\; \mathbf{y} = \mathbf{n} \times \mathbf{x}, \;\; \mathbf{n}. \quad (6)$$

The basis $\{\mathbf{x}, \mathbf{y}, \mathbf{n}\}$ is simply a rotation of the basis $\{\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{n}}\}$. However, its tangential rotation orthogonal to the surface normal $\mathbf{n}$ at v takes into account the transformation of the whole 1-ring neighborhood of the vertex $\bar{v}$, due to the frame-based computation above.

Ray Transformation

We transform an arbitrary view ray $\bar{\mathbf{r}}$, given relative to $\mathcal{M}_{\text{flat}}$, to the corresponding view ray $\mathbf{r}$, given relative to $\mathcal{M}_{\text{curv}}$, as follows. At any given point $\bar{p} \in \mathcal{M}_{\text{flat}}$, intersected by the view ray described by the vector $\bar{\mathbf{r}}$, we expand $\bar{\mathbf{r}}$ relative to the orthonormal basis given by $\mathcal{C}_{\bar{p}} = \{\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{n}}\}$, i.e., $\bar{\mathbf{r}} = \bar{r}_x \bar{\mathbf{x}} + \bar{r}_y \bar{\mathbf{y}} + \bar{r}_n \bar{\mathbf{n}}$. Given the ray $\bar{\mathbf{r}}$ in world coordinates (r_x, r_y, r_z), we compute the coefficients $(\bar{r}_x, \bar{r}_y, \bar{r}_n)$ as

$$(\bar{r}_x, \bar{r}_y, \bar{r}_n)^T = \mathcal{C}_{\bar{p}}^{-1} (r_x, r_y, r_z)^T. \quad (7)$$

We interchangeably use $\mathcal{C}_{\bar{p}}$ for both the set of three basis vectors as well as for the matrix with column vectors given by these basis vectors. We then use the *same* coefficients $\{\bar{r}_x, \bar{r}_y, \bar{r}_n\}$ to get the corresponding transformed view ray $\mathbf{r}$, relative to the transformed (rotated) orthonormal basis given by $\mathcal{C}_p = \{\mathbf{x}, \mathbf{y}, \mathbf{n}\}$ at the point p, of the mesh $\mathcal{M}_{\text{curv}}$. That is, at p we obtain the ray $\mathbf{r} = \bar{r}_x \mathbf{x} + \bar{r}_y \mathbf{y} + \bar{r}_n \mathbf{n}$.

3.3. Ray-Casting

Given a point $p \in \mathcal{M}_{\text{curv}}$ in the volume space $\mathcal{V}$, on the embedded mesh $\mathcal{M}_{\text{curv}}$, and the corresponding view ray $\mathbf{r}$ intersecting the point p, ray-casting is performed by sampling the ray segment from a signed distance ℓ^- along $\mathbf{r}$, with respect to $\mathcal{M}_{\text{curv}}$, to a signed distance ℓ^+ along $\mathbf{r}$. Thus, sampling is performed along the line segment

$$p(t) := p + \left(\ell^- + t\,(\ell^+ - \ell^-)\right) \mathbf{r}, \quad \text{with } t \in [0,1], \quad (8)$$

where $\mathbf{r}$ is the ray direction, and $p \in \mathcal{M}_{\text{curv}}$ is the point on the curved mesh around which ray casting is performed corresponding to the “thickness” $|\ell^+ - \ell^-|$ of the thin volume around the point p.

A simple illustrative example of thin-volume rendering using the described approach is depicted in Fig. 3.

Determining Ray Segment Lengths

By default, the user can specify the parameters ℓ^- and ℓ^+ as global constants. However, we store these parameters for every vertex of the mesh $\mathcal{M}_{\text{flat}}$, interpolating them to the interior of triangles using barycentric interpolation. This allows us to use adapted ray segment lengths. In particular, we allow the user to paint directly on the flat mesh $\mathcal{M}_{\text{flat}}$ in order to adapt the length parameters as desired, see Sect. 4.1. Since $\mathcal{M}_{\text{curv}}$ traces the structure of interest, using this capability for example allows the user to adjust the sampled segment according to the thickness of the structure of interest.

4. Interaction and Navigation

The surface $\mathcal{M}_{\text{flat}}$ is not only a very simple proxy geometry for ray-casting but it also provides an ideal basis for implementing user interaction. Because it is flat, it is guaranteed that no part of the geometry can be occluded by itself. In this section, we give examples for how we utilize $\mathcal{M}_{\text{flat}}$ for user interaction. For technical details, the reader is referred to Sect. 5.

4.1. Adjustment of Ray Lengths

Our thin-volume rendering allows using locally varying ray lengths, that is, a potentially different ray length per mesh vertex. In particular, it is not always possible to compute ray lengths that result in satisfying visualizations for all parts of the thin volume around the curved surface $\mathcal{M}_{\text{curv}}$. This can happen, for example, if the structure of interest has a variable thickness. Globally modifying the ray length might solve problems in some regions but create new ones in others, for example, by inadvertantly removing parts of the structure of interest or introducing new occluding structures.

We have therefore implemented an interaction technique that, similar to an eraser tool in a 2D image processing software, allows one to increase or decrease the ray length locally using the mouse.

4.2. Adjustment of Vertex Position

If the mesh $\mathcal{M}_{\text{curv}}$ deviates too much from the structure of interest, locally changing the ray lengths might not be sufficient. In order to cope with this situation, we have implemented another interaction technique that, instead of adjusting ray lengths locally, modifies the actual vertex positions of the mesh $\mathcal{M}_{\text{curv}}$ by moving vertices along their surface normals.

4.3. Selection of Regions of Interest

In addition to interactively changing ray lengths and vertex positions, $\mathcal{M}_{\text{flat}}$ can also be used for selecting regions of interest for more detailed inspection. Using the mouse, the user can select a region in screen space, from which all triangles and vertices are computed that project to this region. For these, we can then compute the spatially neighboring vertices and triangles in $\mathcal{M}_{\text{curv}}$, which can then either be highlighted in the thin-volume rendering or be used to crop out a volumetric region from the original volume to visualize it using classical DVR. See for example Fig. 7. Both navigation modes help to overcome the loss of spatial information introduced by unfolding. This is particularly important for structures that are rolled or folded multiple times.

5. Implementation

We have integrated a prototype implementation of the proposed method, written in C++ and using the OpenGL API, into the visualization software Amira [SWH05].

5.1. Rendering Approach

The method described in Sect. 3 fits nicely into the shader-based rendering pipeline of OpenGL, using only vertex and fragment shaders. The 3D volume data $\mathcal{V}$ is passed to the GPU as a 3D texture. Entities defined for each OpenGL vertex are passed to the GPU as attributes. These attributes include:

- The vertex coordinates of the geometry to be rendered, that is, $\{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_n\}$ of $\mathcal{M}_{\text{flat}}$. As we assume $\mathcal{M}_{\text{flat}}$ to be a triangular mesh, it can natively be handled by OpenGL.
- Texture coordinates for computing the anchor points of the segments along which $\mathcal{V}$ will be sampled. For this purpose, we need the vertex coordinates $\{v_1, v_2, \ldots, v_n\}$ of $\mathcal{M}_{\text{curv}}$.
- To map camera ray directions via $\bar{\mathbf{r}}$ to $\mathbf{r}$, the transformations $\mathcal{C}_{\bar{p}}$ and $\mathcal{C}_p$ are required. By embedding $\mathcal{M}_{\text{flat}}$ into the xy-plane and orienting it such that the normal direction points into the positive z direction, we let the coordinate systems $\mathcal{C}_{\bar{p}}$ and the world coordinate system coincide. Thus, we can skip the multiplication with $\mathcal{C}_{\bar{p}}^{-1}$ as described in Eq. (7) and only supply the matrix $\mathcal{C}_v$ from which $\mathcal{C}_p$ can be derived via barycentric interpolation.
- Two scalar values ℓ^+ and ℓ^- that define the length of the sampled segment around each position on $\mathcal{M}_{\text{curv}}$.

The vertex shader essentially just passes attributed values to the fragment shader, where the interesting part of the work is done. In the fragment shader, we compute the ray direction $\bar{\mathbf{r}}$, and map it to $\mathbf{r} = \mathcal{C}_p \bar{\mathbf{r}}$. The sampled segment is determined as given in Eq. (8), using the interpolated attributes ℓ^- and ℓ^+ at the position p corresponding to the fragment. Sampling and compositing are then performed in the standard way, as described, for example, by Engel et al. [EHK*04].

5.2. Interaction and Navigation

Implementation of the interaction and navigation functionality is done in the application, outside of the OpenGL pipeline.

Adjustment of Ray Lengths and Vertex Positions

Both ray length adjustment and vertex movement have the same user interface. The user triggers the effect by dragging the mouse across the screen while pressing the left mouse button, i.e. painting over the area that is to be affected. At each moment, the influence area is a circular region in screen space around the current mouse position. To give some visual feedback on the influence area, we draw a white, inwardly fading circle of the according size (Fig. 6). The user can adjust the radius of the influence area by pressing the middle mouse button and dragging the mouse vertically. Pressing the middle mouse button and dragging the mouse horizontally adjusts the effect strength. As a visual cue on the effect strength, a proportional part of the white, inwardly fading circle is colored in blue or red, depending on the sign of the effect, that is, whether vertices are moved along or against the normal direction, or whether ray lengths are increased or decreased.
For a smoother feeling, we scale the effect strength with a Gaussian centered at the current mouse position, and with standard deviation of half the radius of the influence area.

In the following, we describe the selection of the affected vertices in more detail. The circular influence area in screen space corresponds to a cylindrical volume that spans the whole depth from -1 to $+1$ in normalized device coordinate space, and to a cone frustum with circular cross sections that spans from the near to the far plane in world space which, in our case, coincides with the model space of $\mathcal{M}_{\text{flat}}$. Let p_n and p_f denote the points at which the cone axis intersects the near and far plane, and let r_p be the cone radius at any point p in the segment spanned by p_n and p_f. To test if a vertex $\bar{v}$ is affected, we check if its projection a onto the cylinder axis lies between the near and far plane. If it does, the distance to the cone, $|\bar{v} - a|$ is compared against the cone radius r_a. If $\bar{v}$ lies inside the cone, we use the relative distance $d_{\bar{v}} = |\bar{v} - a| / r_a$ to compute the Gaussian weakening factor for $\bar{v}$ as $\mathcal{N}(d_{\bar{v}}; 0, r_a/2)$. Here, $\mathcal{N}(\cdot\,; \mu, \sigma)$ describes a normal distribution with mean μ and standard deviation σ.

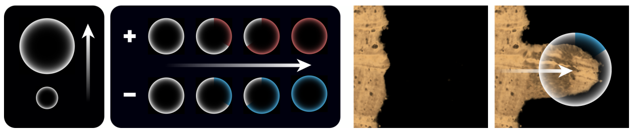

Figure 6: *A white circle gives visual feedback on the influence area of the interaction. The size can be adjusted with vertical mouse movement, the effect strength with horizontal mouse movement. The effect strength is indicated by a colored portion of the circle, red for an increasing, blue for a decreasing effect. The right image pair shows the interaction tool in action.*

Selection of Regions of Interest

Defining regions of interest on $\mathcal{M}_{\text{flat}}$ and locating them in $\mathcal{V}$ or computing close areas to them in $\mathcal{M}_{\text{curv}}$ is straightforward. Using the mouse, the user selects a rectangular region on the screen. For each vertex of $\mathcal{M}_{\text{flat}}$, we test if it is rendered into this region by mapping the x and y coordinates of the selection and of $\mathcal{M}_{\text{curv}}$ in the normalized device coordinate space and comparing the coordinates. The selected vertices are indicated by a color overlay on the rendering of $\mathcal{M}_{\text{curv}}$ (Fig. 7(d,f)).

Having computed the set of vertices that fall into the selected region, the corresponding vertex set in $\mathcal{M}_{\text{curv}}$ can be used to either define an appropriate region of $\mathcal{V}$ or to expand the selection by locating other vertices in $\mathcal{M}_{\text{curv}}$ that are close to the initial selection.

When specifying the region of interest in $\mathcal{V}$, we use the largest and smallest x, y, z components of all selected points to define an axis aligned box. This box can be used to clip the volume during traditional volume rendering (Fig. 7(e,g)).

When expanding the selection to vertices that are close in $\mathcal{V}$, the user can specify the maximum and minimum distance that any vertex is allowed to have to the set of initially selected vertices. Including those vertices into the color overlay of the selection serves as feedback to the user (Fig. 7(d,f)).

6. Results

We have used the prototype implementation of our thin-volume rendering method to visualize and explore six datasets. In defining

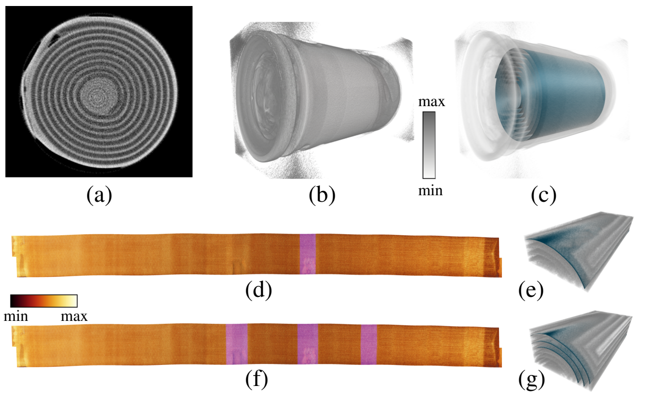

Figure 7: *Neutron tomography image of a battery after discharging: (a) cross-section through the middle; (b) volume rendering using gray-scale colormap; (c) volume rendering with $\mathcal{M}_{\text{curv}}$; (d) thin-volume rendering with selection, also shown in ROI (e); (f) thin-volume rendering with extended selection, showing neighboring regions, also as ROI (g).*

$\mathcal{M}_{\text{curv}}$, we made heavy use of an existing software infrastructure that allowed us, among other things, to generate 1D contours in 2D images, to connect a sequence of 1D contours into a triangulated surface mesh, to remesh and smooth such a surface, and finally to flatten it. All labor or time estimates apply if such infrastructure is in place.

Fine tuning the general rendering parameters required little effort in all cases considered here, as we had sliders available to interactively adjust a) the minimum and maximum values of the transfer function, b) the magnitude of the initially constant ray length, and c) the iso-value (for thin iso-surface rendering). Depending on the desired quality of the result, local interactive ray length adjustments and geometry manipulations can consume significantly more time. The results shown here were obtained with at most a few minutes of manual interaction, indicating a sufficiently close initial tracing of $\mathcal{M}_{\text{curv}}$, and thus little need for excessive interactive correction.

Table 1 lists relevant specifications for each dataset. All results were obtained on a workstation running Ubuntu 18.04 with Intel Xeon(R) (3.20 GHz $\times$ 12) CPU and an Nvidia GeForce GTX 1080 GPU. As can be seen from the fps numbers, all datasets can be rendered with interactive frame rates.

6.1. Battery – Region Selection

Here, we present the results for a battery that was imaged using Neutron tomography [ZAF*20] (data courtesy by Ralf Ziesche, University College London). The dataset is used to illustrate the utility of our method in analyzing adjacent regions. The results are shown in Fig. 7.

The curved surface, $\mathcal{M}_{\text{curv}}$, was generated from three 1D contours that trace the electrodes in three 2D cross-sectional images oriented orthogonally to the winding axis (Fig. 7(a)). The images were chosen at the beginning, middle, and end of the electrode cylinder. The contours were equidistantly resampled to form polylines and the vertices of adjacent polylines were then connected to form the triangular mesh $\mathcal{M}_{\text{curv}}$ (Fig. 7(c)). Near-isometric flattening [AZvT19] was applied to generate $\mathcal{M}_{\text{flat}}$.

Classical volume rendering (Fig. 7(b)) makes it hard to inspect the whole electrode, while thin-volume renderings (Fig. 7(d,f)) show the unrolled electrode in a single view. Additionally, it allows one to easily select regions (Fig. 7(d,e)) for which neighboring regions can then automatically get selected in the unrolled data (Fig. 7(f)) and the original image (Fig. 7(g)). Here, the highlighted regions show an irregularity that occurs at neighboring windings. It can be seen that the dark region in the left selection and the bright region in the middle selection are located close to each other on adjacent layers.

This example illustrates that spotting an irregularity and precisely locating it in the thin-volume rendering is simple, whereas it would be very difficult in classical DVR. The combination of thin- and classical volume rendering, using the earlier one for interaction, allows the analysis in both the original and the unrolled state.

6.2. Silver Scroll – Interactive Ray Adjustment

The μCT dataset of a silver scroll [BLLR15, LR20] found at Jerash, Jordan, was analyzed to showcase the use of the interactive ray adjustment to remove self-occluding parts from the unrolled visualization.

Similar to the battery dataset, $\mathcal{M}_{\text{curv}}$ was obtained by first tracing 1D contours, which in turn were equidistantly sampled, and the resulting polylines were connected to a surface mesh. Due to the much greater complexity of the dataset, we traced contours in 64 out of 1834 cross-sections. The initial curved surface mesh was then refined, smoothed, and finally flattened using quasi-isometric flattening [AZvT19].

The result of applying thin-volume rendering to the silver scroll is shown in Fig. 8(d). Instead of DVR, for this dataset we applied iso-surface ray-casting since only the surface of the silver scroll is of interest. In Fig. 8(e), the points on the iso-surface are color-coded according to the distance (along the ray) from the curved surface, where blue denotes the distance away from and red closer to the viewer. As can be seen, in most regions the writing is clearly visible, but there are some regions, where selecting a single ray length does not provide satisfying results. An example is illustrated in Fig. 8(f). Using the interactive ray-length adjustment (Sect. 4.1), we locally modified the ray length (Fig. 8(f)) and thereby created an unobstructed view of the writing (Fig. 8(h)).

6.3. Papyrus – Interactive Geometry Manipulation

In this section, we present the results for a μCT dataset [MAB*20] of an ancient papyrus package (L/El227b/4-pG Greek papyrus; data courtesy by Eve Menei and Marc Etienne, Musée du Louvre, Paris). The package is folded twice in orthogonal directions and it is too fragile to be physically opened. The visual results for this dataset are shown in Fig. 9.

In this case, the surface generation started from the flattened mesh, $\mathcal{M}_{\text{flat}}$, where the vertices were arranged in a regular grid.

Dataset	Rendering type	Volume dim.	#Faces	#Samples	fps	Generation of $\mathcal{M}_{\text{curv}}$
Battery	DVR	$704 \times 664 \times 1024$	2540	30	65	$\approx$ 1 hour
Silver scroll	iso-surface	$400 \times 408 \times 1834$	884,352	80	66	$\approx$ 2 days
Papyrus	DVR	$1812 \times 424 \times 1492$	210,000	50	42	$\approx$ 2 hours
Armadillo	iso-surface	$624 \times 830 \times 997$	780	1400	66	$\approx$ 1 hour
Ribcage	DVR	$512 \times 512 \times 264$	990	60	64	$\approx$ 1 hour
Knee	DVR	$160 \times 384 \times 384$	6752	80	65	$\approx$ 1 hour

Table 1: *Type of thin-volume rendering, volume dimensions, number of faces in $\mathcal{M}_{\text{curv}}$ and $\mathcal{M}_{\text{flat}}$, maximum number of samples per ray for image qualities as depicted in this article, corresponding frame rates, and the time it took to generate $\mathcal{M}_{\text{curv}}$ per dataset. In the prototype implementation the number of samples was kept constant per dataset (that is, shorter ray segments were sampled with higher frequency). The higher number of samples for the Armadillo is due to the much greater ray lengths with which the dataset was rendered. The times to generate $\mathcal{M}_{\text{curv}}$ presuppose the existence of a software infrastructure as described in Sect. 6.*

This mesh was then virtually folded along two orthogonal folding lines. The resulting idealized folded mesh was then fitted to the image data using landmark-based thin-plate spline warping [Boo89].

The final curved mesh, $\mathcal{M}_{\text{curv}}$, only coarsely traces the papyrus structure (Fig. 9(b)), while the dense layering restricts the lengths of the rays if rendering neighbouring layers should be avoided. Accordingly, the initial rendering of the papyrus shown in Fig. 9(c) contains more holes than can be explained by the poor condition of the papyrus. When rendering an iso-surface with color-coding the distance of the iso-surface points from $\mathcal{M}_{\text{curv}}$, we get hints about how to move the mesh vertices to fill the holes. The color (blue or red) indicates on which side of $\mathcal{M}_{\text{curv}}$ the rendered point lies, the saturation indicates its distance to $\mathcal{M}_{\text{curv}}$. For example, the blue color surrounding the hole in the attached subfigure of Fig. 9(c) indicates to which side the papyrus submerges. Shifting the vertices according to the surrounding color closes the hole as the modified $\mathcal{M}_{\text{curv}}$ more closely traces the papyrus (Fig. 9(d)).

Even though the papyrus package was only folded twice, volume rendering of the "folded" data makes it very hard to see any letters. This problem becomes even more severe when more complicated, multi-folded packages are to be unfolded.

The presented example shows that the thin-volume renderings allow to inspect the whole surface of the unfolded papyrus without any occlusions of the papyrus itself (Fig. 9(c,d)). The task of interactively fitting an initial geometry to a thin structure image data in 3D is not trivial. Already the evaluation of the quality of a fit can become intricate. Using the proposed method of painting over clearly identifiable misaligned regions on a flat mesh while getting immediate feedback about the quality of the fit, turns this task into a very intuitive and even pleasant one. Future work will explore more elaborate geometry manipulation techniques to increase the flexibility of the manipulation and further relax the requirements for an initial fit of $\mathcal{M}_{\text{curv}}$.

6.4. Armadillo – Interactive Ray Adjustment

In this section, we present the results of applying thin iso-surface rendering to the carapace of a southern three-banded armadillo (*Tolypeutes matacus*), imaged using computed tomography (data courtesy by Ramon Nagesan and Cody Thompson from the Museum of Zoology, University of Michigan). The dataset was created as part of the oVert project [oVe17, WCLR*18]. Here, we show the applicability of our method to a structure for which no isometric mapping into the plane exists. We also show the usability of interactive ray adjustment to remove parts of the skeleton when looking at the carapace from the inside.

The results of applying thin iso-surface rendering to this dataset are shown in Fig. 1. For generating the curved surface, $\mathcal{M}_{\text{curv}}$, we manually drew 18 contours, which were equidistantly resampled and connected to a surface mesh. Again, we applied quasi-isometric flattening [AZvT19] to create the flattened mesh, $\mathcal{M}_{\text{flat}}$.

As can be seen in Fig. 1(a), volume rendering of the unflattened dataset does not allow one to see the whole carapace at once. However, this is often required for data comparison. We applied iso-surface ray-casting to render the flattened carapace. Since the outside is not occluded, no interaction was necessary to obtain the desired result (Fig. 1(c)). However, the inside of the carapace is heavily occluded by the skeleton, because we chose a long ray length. This example shows that if $\mathcal{M}_{\text{curv}}$ has a low curvature, even rather thick volume slabs can be rendered with reasonable results. By applying the interactive ray-length adjustment, we removed the upper right part of the skeleton (Fig. 1(d)) to create an unobstructed view onto the inside of the carapace.

The flattened view from the inside allows the user to relate nearby structures to the position of the carapace. Interactive modification of the ray lengths enables the reduction of occluding structures to the desired degree, which might even mean the complete removal of inner structures.

6.5. Ribcage – Flattening

Here, we present the result for the CT scan of the Visible Human Male dataset [SASW96]. In particular, we look at the ribcage and unfold the ribs (Fig. 10).

The curved mesh, $\mathcal{M}_{\text{curv}}$, was generated from 16 manually drawn polylines, each with the same number of points so that they roughly trace the ribcage. This surface was then flattened using a relaxation-based approach [BLH*17] by center-aligning the polylines before starting the relaxation.

The resulting thin-volume visualization shows a high similarity to the resampling-based results presented by Kretschmer et al. [KST*14]. Compared to the classical DVR of the unflattened dataset, the thin DVR does not require to rotate the dataset in order to completely inspect all ribs. However, our approach still allows rotating and zooming in order to obtain better views or to

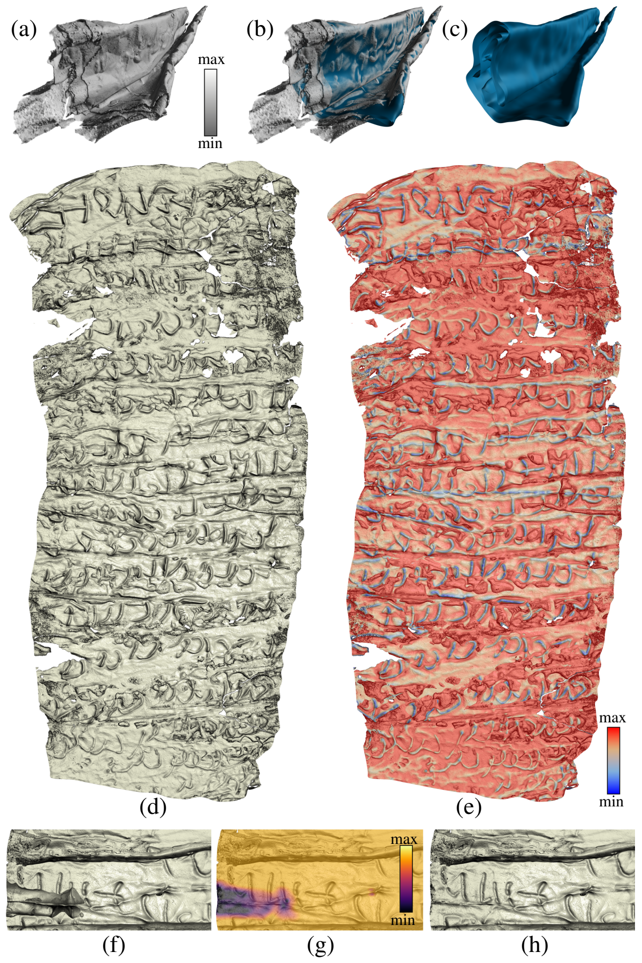

Figure 8: *Silver scroll: (a) volume rendering of μCT image; (b) same as (a) with $\mathcal{M}_{\text{curv}}$; (c) $\mathcal{M}_{\text{curv}}$; (d) thin iso-surface rendering; (e) thin iso-surface rendering with distance to $\mathcal{M}_{\text{curv}}$ color-coded (blue=behind, red=in front of $\mathcal{M}_{\text{curv}}$); (f) zoom-in with occluding structure; (g) modified ray length color-coded; (h) occluding structure partially removed.*

more closely inspect anomalies. Note also that information about the spatial context, such as muscle tissue, is preserved.

6.6. Knee – Interactive Geometry Manipulation

The data visualized in this section is from the OAI database [OAI, PGE*06]. It shows a knee imaged with MRI that has a cartilage lesion located at the medial femoral condyle.

The curved surface (Fig. 11(a)) represents the articular cartilage region of the femur (data courtesy by Alexander Tack and Felix Ambellan) [ATEZ19]. The flattened surface (Fig. 11(b)) was generated using near-isometric flattening based on the shape representation described by Ambellan et al. [AZvT19].

The automatically generated surface does not immediately allow one to see all the interesting structures (Fig. 11(c)), since part of the surface lies inside the bone instead of on its boundary. Interactively adjusting the mesh vertices brings out a bright structure next to the medial meniscus which represents accumulated joint fluid as an indicator of a pathology (Fig. 11(d)).

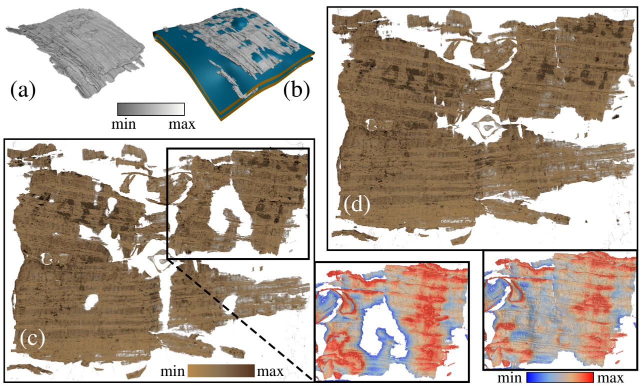

Figure 9: *Greek papyrus package: (a) volume rendering; (b) volume rendering with $\mathcal{M}_{\text{curv}}$; (c) thin-volume rendering for badly fitting $\mathcal{M}_{\text{curv}}$; (d) improved fit of $\mathcal{M}_{\text{curv}}$ revealing some Greek letters.*

For such data, using classical DVR for diagnostic purposes is difficult, since the structure of interest is both curved and very thin. Adjusting the visualization using DVR by changing the transfer function or volume editing will be difficult and time-consuming. Thin-volume rendering might be an effective alternative for this and similar diagnoses that frequently occur in clinical practice.

7. Discussion & Conclusion

We have presented an interactive ray-casting-based method for the rendering of thin volumetric structures with a curved domain. The presented results suggest that the method is generic in the sense that it can be easily applied to a large variety of datasets even though it was originally developed for the analysis of rolled and folded writing material, in particular papyrus. The achieved frame rates (Table 1) indicate the suitability of the method for interactive exploration. All datasets, despite their range of volume dimensions, mesh sizes and sampling rates, could be rendered with frame rates larger than 40 fps. The method works also surprisingly well even for quite thick structures (see, for example, the Armadillo dataset in Fig. 1).

The major bottleneck for applying the tool is the generation of curved surfaces approximating the thin volumes to be rendered. To speed-up this process, specifically tailored application-dependent tools are needed. Presenting these tools was outside the scope of this paper.

A limitation of the presented method is that linear rays transformed using curved meshes as described, will eventually cross. How quickly this happens depends on both ray length and local curvature of $\mathcal{M}_{\text{curv}}$. Crossing rays can lead to duplicated renderings of the same feature at different locations of $\mathcal{M}_{\text{flat}}$ and to an inversion of their order. This was not an issue for the mostly thin volumes or, in case of the Armadillo, for volumes with low curvature investigated here. In order to support the rendering of thicker volumes

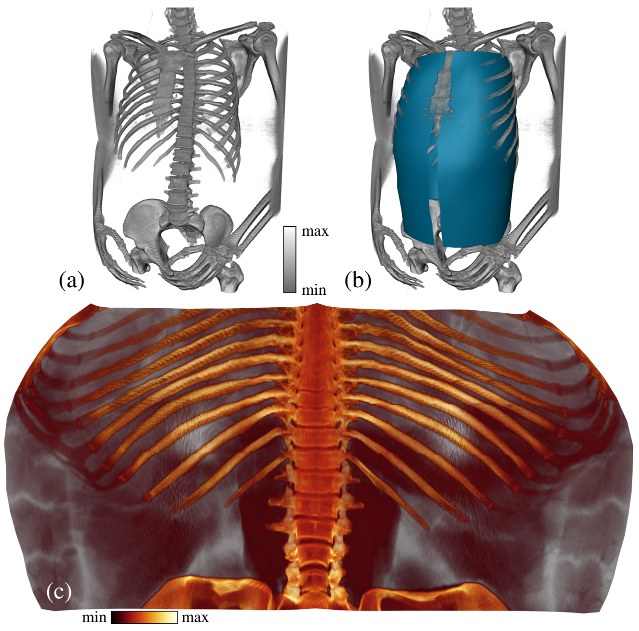

Figure 10: *Ribcage of the Visable Human Male: (a) DVR; (b) DVR+$\mathcal{M}_{\text{curv}}$; (c) thin DVR.*

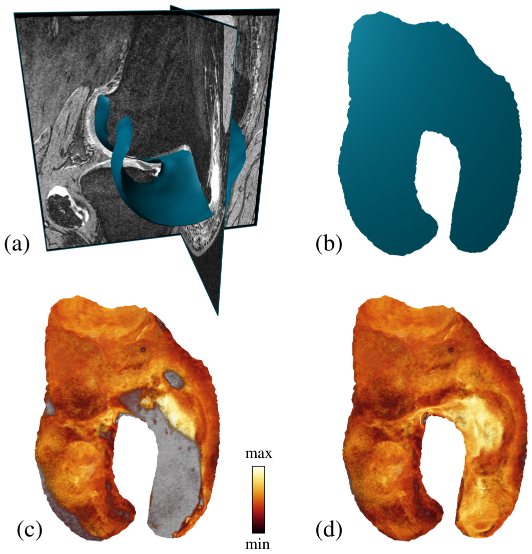

Figure 11: *Femur with cartilage lesion in medial compartment: (a) cross-sections of MRI image with $\mathcal{M}_{\text{curv}}$; (b) $\mathcal{M}_{\text{flat}}$; (c) initial thin-volume rendering; (d) thin DVR after vertex adjustment.*

around surface meshes with higher curvature than that used for the Armadillo carapace, we plan to explore the use of non-linear ray-casting as described by Gröller [Grö95] and Vilanova et al. [VBWKG01]. However, an extension for non-orthogonal rays will be needed, at the least.

Even though the thin-volume rendering is most interesting when the camera looks orthogonally onto $\mathcal{M}_{\text{flat}}$, we deliberately allow the camera to be changed arbitrarily to give the user the same look and feel as with a classical volume rendering. For the same reason, we implemented perspective camera mode, which results in a change of the camera orientation across the screen space, see for example, Fig. 3. Since we currently use view-independent ray lengths, the thickness of the rendered volume changes with the camera orientation, which is another limitation of the presented approach. If the angle at which we look at the flat surface is around 90°, the thickness change is negligible, but the smaller the angle gets the more noticeable does the thickness change become. This is another point we want to look at in the future.

Pseudo-coloring the iso-surface rendering with the signed distance of the iso-surface points to the curved surface provides very important visual cues for moving the mesh vertices. This has currently only been implemented for iso-surface rendering but would also be versatile for the volume rendering mode.

The thin-volume rendering approach was originally developed to support the visual analysis of rolled and folded papyri, which are particularly difficult to analyze due to the inhomogeneous structure of papyrus that is further complicated by the often poor state of preservation of the ancient documents. Prototypically, we have implemented a few tools for interactive editing that demonstrate that intuitive data exploration is feasible using the suggested framework. While the developed interaction tools provide a first step into analyzing such complicated objects, the development of more sophisticated mesh deformation tools is planned.

Acknowledgments

Felix Herter was partially funded by the European Research Council (Project "ELEPHANTINE", ID 637692). Hans-Christian Hege's work was supported by Deutsche Forschungsgemeinschaft (DFG) through grant CRC 1114 “Scaling Cascades in Complex Systems”, Project (C06) “Multi-scale structure of atmospheric vortices”. Markus Hadwiger was supported by King Abdullah University of Science and Technology (KAUST). We thank Ralf Ziesche (University College London) and Tobias Arlt (Helmholtz-Zentrum Berlin (HZB)) for providing the battery dataset and Marc Etienne, Eve Menei (Musée du Louvre, Paris), Tobias Arlt and Heinz-Eberhard Mahnke (HZB) for providing the papyrus dataset. Ramon Nagesan and Cody Thompson (Museum of Zoology, University of Michigan) are gratefully acknowledged for providing the Armadillo dataset. We also thank Alexander Tack and Felix Ambellan (Zuse Institute Berlin) for their help with the knee dataset and Felix Ambellan for providing the flattening module.

References

[AMHH08] Akenine-Möller T., Haines E., Hoffman N.: *Real-Time Rendering*, 3rd ed. AK Peters/CRC Press, 2008. doi:10.1201/9781315365459. 3

[ATEZ19] Ambellan F., Tack A., Ehlke M., Zachow S.: Automated segmentation of knee bone and cartilage combining statistical shape knowledge and convolutional neural networks: Data from the osteoarthritis initiative. *Med. Image Anal. 52*, 2 (2019), 109 – 118. doi:10.1016/j.media.2018.11.009. 9

[AZvT19] Ambellan F., Zachow S., von Tycowicz C.: A surface-theoretic approach for statistical shape modeling. In *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI), Part IV*

(2019), pp. 21–29. doi:10.1007/978-3-030-32251-9_3. 7, 8, 9

[BKW08] Bürger K., Krüger J., Westermann R.: Direct volume editing. *IEEE Trans. Vis. Comput. Graph. 14*, 6 (2008), 1388–1395. doi:10.1109/TVCG.2008.120. 1, 2

[BLH*17] Baum D., Lindow N., Hege H.-C., Lepper V., Siopi T., Kutz F., Mahlow K., Mahnke H.-E.: Revealing hidden text in rolled and folded papyri. *Appl. Phys. A 123*, 3 (2017), 171. doi:10.1007/s00339-017-0808-6. 2, 8

[BLLR15] Barfod G. H., Larsen J. M., Lichtenberger A., Raja R.: Revealing text in a complexly rolled silver scroll from Jerash with computed tomography and advanced imaging software. *Sci. Rep. 5* (2015), 17765. doi:10.1038/srep17765. 2, 7

[Boo89] Bookstein F. L.: Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 11*, 6 (1989), 567–585. doi:10.1109/34.24792. 8

[CCI*07] Chen M., Correa C., Islam S., Jones M. W., Shen P.-Y., Silver D., Walton S. J., Willis P. J.: Manipulating, deforming and animating sampled object representations. *Comput. Graph. Forum 26*, 4 (2007), 824–852. doi:10.1111/j.1467-8659.2007.01102.x. 2

[CK13] Casazza P., Kutyniok G.: *Finite Frames—Theory and Applications*. Springer, 2013. doi:10.1007/978-0-8176-8373-3. 3, 4

[EHK*04] Engel K., Hadwiger M., Kniss J. M., Lefohn A. E., Salama C. R., Weiskopf D.: Real-time volume graphics. In *ACM Siggraph 2004 Course Notes*. ACM, 2004, pp. 29–es. doi:https://doi.org/10.1201/b10629. 3, 6

[FH05] Floater M. S., Hormann K.: Surface parameterization: a tutorial and survey. In *Advances in multiresolution for geometric modelling*. Springer, 2005, pp. 157–186. doi:10.1007/3-540-26808-1_9. 2

[Grö95] Gröller E.: Nonlinear ray tracing: Visualizing strange worlds. *Vis. Comput. 11*, 5 (1995), 263–274. doi:10.1007/BF01901044. 10

[HGQ*06] Hong W., Gu X., Qiu F., Jin M., Kaufman A.: Conformal virtual colon flattening. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling* (2006), pp. 85–93. doi:10.1145/1128888.1128901. 2, 3

[HKLW07] Han D., Kornelson K., Larson D., Weber E.: *Frames for Undergraduates*. American Mathematical Society, 2007. 4

[Kan04] Kanitsar A. M.: Curved planar reformation for vessel visualization. http://diglib.eg.org/handle/10.2312/8168, 2004. *PhD Thesis,* TU Vienna. 2

[KFW*02] Kanitsar A., Fleischmann D., Wegenkittl R., Felkel P., Gröller M. E.: CPR - Curved Planar Reformation. In *IEEE Visualization 2002* (Oct. 2002), pp. 37–44. doi:10.1109/VISUAL.2002.1183754. 1, 2

[KMM*18] Kreiser J., Meuschke M., Mistelbauer G., Preim B., Ropinski T.: A survey of flattening-based medical visualization techniques. *Comput. Graph. Forum 37*, 3 (2018), 597–624. doi:10.1111/cgf.13445. 2

[KST*14] Kretschmer J., Soza G., Tietjen C., Suehling M., Preim B., Stamminger M.: ADR – anatomy-driven reformation. *IEEE Trans. Vis. Comput. Graph. 20*, 12 (2014), 2496–2505. doi:10.1109/TVCG.2014.2346405. 2, 3, 8

[LCMH09] Lampe O. D., Correa C., Ma K.-L., Hauser H.: Curve-centric volume reformation for comparative visualization. *IEEE Trans. Vis. Comput. Graph. 15*, 6 (2009), 1235–1242. doi:10.1109/TVCG.2009.136. 2

[LI14] Li X., Iyengar S.: On computing mapping of 3d objects: A survey. *ACM Comput. Surv. (CSUR) 47*, 2 (2014), 1–45. doi:10.1145/2668020. 2

[LKG*16] Ljung P., Krüger J., Gröller E., Hadwiger M., Hansen C. D., Ynnerman A.: State of the art in transfer functions for direct volume rendering. *Comput. Graph. Forum 35*, 3 (2016), 669–691. doi:10.1111/cgf.12934. 1

[LR20] Lichtenberger A., Raja R.: Jerash silver scroll: Computed tomography data, 2020. figshare. doi:https://doi.org/10.6084/m9.figshare.12136380.v1. 7

[LRLH17] Liu C., Rosin P. L., Lai Y.-K., Hu W.: Robust virtual unrolling of historical parchment XMT images. *IEEE Trans. Image Process. 27*, 4 (2017), 1914–1926. doi:10.1109/TIP.2017.2783626. 2

[MAB*20] Mahnke H.-E., Arlt T., Baum D., Hege H.-C., Herter F., Lindow N., Manke I., Siopia T., Menei E., Etienne M., Lepper V.: Virtual unfolding of folded papyri. *J. Cult. Herit. 41* (2020), 264–269. doi:10.1016/j.culher.2019.07.007. 7

[MPG*17] Martinke H., Petry C., Grosskopf S., Suehling M., Soza G., Preim B., Mistelbauer G.: Bone fracture and lesion assessment using shape-adaptive unfolding. In *VCBM 17: Eurographics Workshop on Visual Computing for Biology and Medicine* (2017), Eurographics Association, pp. 149–158. doi:10.2312/vcbm.20171249. 2

[OAI] OAI: OAI: The Osteoarthritis Initiative. http://nda.nih.gov/oai. 9

[oVe17] oVert: OVert: Open Exploration of Vertebrate Diversity in 3D. http://www.idigbio.org/wiki/index.php/OVert:_Open_Exploration_of_Vertebrate_Diversity_in_3D, 2017. 8

[PGE*06] Peterfy C., Gold G., Eckstein F., Cicuttini F., Dardzinski B., Stevens R.: MRI protocols for whole-organ assessment of the knee in osteoarthritis. *Osteoarthr. Cartil. 14* (2006), 95–111. doi:10.1016/j.joca.2006.02.029. 9

[SASW96] Spitzer V., Ackerman M. J., Scherzinger A. L., Whitlock D.: The visible human male: a technical report. *J. Am. Med. Inform. Assoc. 3*, 2 (1996), 118–130. doi:10.1136/jamia.1996.96236280. 8

[SPR06] Sheffer A., Praun E., Rose K.: Mesh parameterization methods and their applications. *Found. Trends Comput. Graph. Vis. 2*, 2 (2006), 105–171. doi:10.1561/0600000011. 2

[SWH05] Stalling D., Westerhoff M., Hege H.-C.: Amira: a highly interactive system for visual data analysis. In *The Visualization Handbook*. Elsevier, 2005, pp. 749–767. doi:10.1016/B978-012387582-2/50040-X. 5

[VBWKG01] Vilanova Bartroli A., Wegenkittl R., König A., Gröller E.: Nonlinear virtual colon unfolding. In *Proceedings Visualization, 2001. VIS'01.* (2001), IEEE, pp. 411–579. doi:10.1109/VISUAL.2001.964540. 2, 10

[WCLR*18] Watkins-Colwell G., Love K., Randall Z., Boyer D., Winchester J., Stanley E., Blackburn D.: The walking dead: status report, data workflow and best practices of the overt thematic collections network. *Biodivers. Inf. Sci. Stand.* (2018). doi:10.3897/biss.2.26078. 8

[WGC*07] Williams D., Grimm S., Coto E., Roudsari A., Hatzakis H.: Volumetric curved planar reformation for virtual endoscopy. *IEEE Trans. Vis. Comput. Graph. 14*, 1 (2007), 109–119. doi:10.1109/TVCG.2007.1068. 2, 3

[WRS01] Westermann R., Rezk-Salama C.: Real-time volume deformations. *Comput. Graph. Forum 20*, 3 (2001), 443–451. doi:10.1111/1467-8659.00537. 2

[ZAF*20] Ziesche R. F., Arlt T., Finegan D. P., Heenan T. M., Tengattini A., Baum D., Kardjilov N., Markötter H., Manke I., Kockelmann W., et al.: 4d imaging of lithium-batteries using correlative neutron and X-ray tomography with a virtual unrolling technique. *Nat. Commun. 11*, 1 (2020), 1–11. doi:10.1038/s41467-019-13943-3. 2, 7