The Visual Computer
https://doi.org/10.1007/s00371-023-02828-8

ORIGINAL ARTICLE

Volume conductor: interactive visibility management for crowded volumes

Žiga Lesar[1] · Ruwayda Alharbi[2] · Ciril Bohak[1,2] · Ondřej Strnad[2] · Christoph Heinzl[3] · Matija Marolt[1] · Ivan Viola[2]

Accepted: 26 February 2023

Abstract
We present a novel smart visibility system for visualizing crowded volumetric data containing many object instances. The presented approach allows users to form groups of objects through membership predicates and to individually control the visibility of the instances in each group. Unlike previous smart visibility approaches, our approach controls the visibility on a per-instance basis and decides which instances are displayed or hidden based on the membership predicates and the current view. Thus, cluttered and dense volumes that are notoriously difficult to explore effectively are automatically sparsified so that the essential information is extracted and presented to the user. The proposed system is generic and can be easily integrated into existing volume rendering applications and applied to many different domains. We demonstrate the use of the volume conductor for visualizing fiber-reinforced polymers and intracellular organelle structures.

Keywords Volume visualization · Visibility management · Crowded volumes

1 Introduction

Surface rendering methods used in modern volume rendering applications typically focus on object boundaries and thus provide insufficient insight into volumetric datasets found in many branches of science. Such volumetric datasets can be visualized with direct volume rendering (DVR) methods, which are often coupled with *visibility management* techniques to reveal or emphasize specific structures or regions of interest. Examples include transfer function specification, planar reformation, clipping geometry, cutaway views, exploded views, and spatial deformations. Such techniques are said to be *smart* as they emphasize the most relevant data through dynamic changes in visual representations, deformations, or spatial modifications of parts of the data [26].

In many cases, a volume is densely populated with numerous instances of structures that occlude each other, and the absence of visibility management results in an uninterpretable and cluttered image. We call such volumes *crowded volumes*. Examples include scans of polycrystalline materials, fiber-reinforced polymers, and intracellular biological structures. For such data, existing volume rendering methods are not suitable because the amount and density of the instances are so high that occlusion impedes the spatial perception of patterns and distributions in the data. Usually, it is not one specific instance that is of interest, but rather the distribution of instances throughout the volume; therefore, a suitable visibility management strategy is to *sparsify* the volume by removing or fading some instances. The instances are often characterized by additional attributes, such as volume, surface area, orientation, position, and others, and the users are typically interested in investigating the spatial distribution of the instances with specific attributes.

For the visualization of such volumes, we propose the *volume conductor*, a novel interactive exploration and visibility management system for crowded volumes. With the volume conductor, the user directs the color and visibility of groups of instances to obtain the desired visualization of a crowded volume. The user can rapidly achieve the desired visualization by organizing instances into a hierarchy

✉ Žiga Lesar
ziga.lesar@fri.uni-lj.si

1 Faculty of Computer and Information Science, University of Ljubljana, Večna pot 113, 1000 Ljubljana, Slovenia

2 Visual Computing Center, King Abdullah University of Science and Technology (KAUST), 23955-6900 Thuwal, Kingdom of Saudi Arabia

3 University of Passau, Innstraße 41, 94032 Passau, Germany

Published online: 29 March 2023

of groups and managing their visibility through an easy-to-use user interface. The sparsification of each group can be directly adjusted through manipulation of scented sliders [29]. The sparsification routine automatically determines which instances are visually suppressed and which remain visible, which is a significantly different process compared to transfer function-based visibility management, where the visibility is implicitly controlled. Having said that, the volume conductor is not simply an instance filtering system. It works on volumes and poses no limits on transparency. The instances can thus also be smoothly faded away. Not only can the transparency be modified due to sparsification, but the transfer function can also be augmented per instance, interactively. We also present a technique for combining the resulting sparsified volume with the raw data volume, which enables the integration of the volume conductor into existing DVR techniques. An overview of these features is presented in Fig. 1. The system is generic and can be applied to many domains, which we demonstrate in three use cases.

We emphasize the following original contributions of our work:

- rendering method-independent interactive visibility management system for visualizing crowded volumes;
- combined rendering of sparsified segmented volumes and raw data volume for providing the context of displayed instances within the raw data;
- procedural generation of GPU shader code based on user-defined hierarchically organized instance attributes; and
- domain expert-defined use cases showcasing the benefits of the proposed system.

2 Related Work

A crowded volume is a dense representation exhibiting a high degree of mutual occlusion among the contained structures. Visibility in volumes is usually directed with transfer functions [18]. Traditionally, transfer functions map the intensities and their gradients to optical properties without awareness of the individual instances and as such do not necessarily reduce occlusion or expose relevant data. To prevent the occlusion of important information and provide enough context for better spatial comprehension, *smart visibility management* techniques are indispensable when rendering structures that exhibit large amounts of occlusion. Standard techniques include clipping planes, cutaway views, and automatic or interactive transfer function specification. Early approaches addressed the problem of the automatic generation of see-through technical illustrations [6,7] with view-dependent transparency techniques. Image-space depth sorting is used to generate semi-transparent visualizations, combined with the existing rendering methods. Viola et al. [26] discussed how to visually expose essential parts of the data in volumetric renderings with an importance-driven feature enhancement technique enabling the automatic generation of cutaway and ghost views. Ament et al. [1] used selective illumination to automatically highlight important structures in a volume and make them visible from the camera. Viega et al. [25] presented volumetric lenses for interactively enhancing interesting regions. Their approach was later extended to volumetric data by Ropinski et al. [20]. Kubisch et al. [15] used breakaway views and ghost views, on a practical use case of tumor surgery planning. In addition, Chan et al. [4] explored spatial relations between the structures in a volume, such as overlaps or touching points, and suggested different techniques for visualizing them. Correa and Ma [5] introduced visibility histograms to automatically adjust the transfer function based on user-defined opacity mapping.

While these techniques allow users to expose parts of the data in different ways, they do not offer precise control over which data should be exposed and to what degree. Moreover, none of them explicitly target crowded volumes and thus are not particularly useful for visualizing such data. In the best case, extensive preprocessing is necessary to prepare the data for visualization using a specific method. In contrast, our proposed approach offers interactive sparsification with

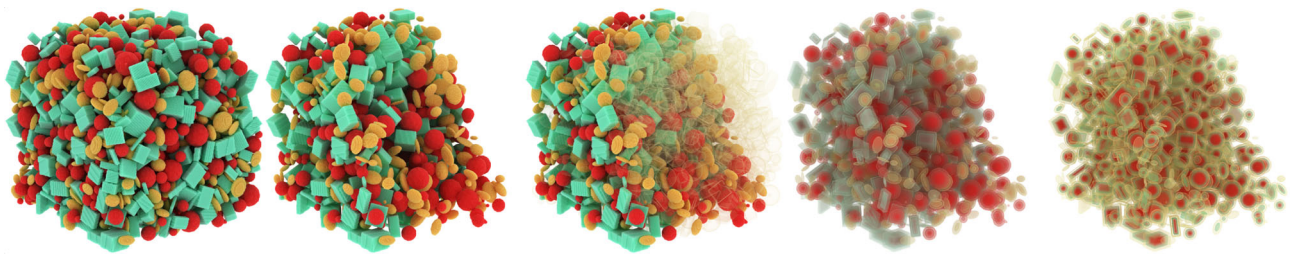

Fig. 1 Features of the volume conductor. From left to right: instance grouping and colorization, sparsification, ghosting, blending with raw data, and opacity transfer from sparsification to raw data. Such visualizations are easily achieved with the volume conductor, while standard volume rendering would be unusable since the values and gradients are shared by the background and the instances

minimal preprocessing and is independent of the rendering method.

Context-preserving methods consider critical aspects of the visualization and retain them regardless of the view and data orientation. The magic volume lens [27] retains the context by deformation, instead of removal of structures, where the user selects the magnifying part. Krüger et al. [14] presented a distance-based importance mapping of transparency to preserve the context surrounding the focused part of the data. A context-preserving method by Bruckner et al. [3] introduces an easily controlled context-preserving approach considering the shading intensity, gradient magnitude, eye distance, and previously accumulated opacity to reduce the opacity of less essential regions of the volume. This method is also adapted for one of the proposed sparsification functions, where it is used to control the importance of entire instances rather than individual voxels.

Context-preserving approaches work well on uncrowded data. However, for crowded volumes, their use alone is not enough to present and retain a global context around specific instances in a crowded environment. We complement the existing contributions by extending the voxel-based view of the data to an instance-based view through aggregation over the instances.

Streamline visualization aims toward automatic selection and rendering of the most representative instance in a crowded environment. Günther et al. [8] addressed the problem by optimizing the streamline opacity, extending it for visualizing surfaces [10], sets of streamlines [9], and a joint dataset with points, lines, and surfaces [11]. Kanzler et al. [13] estimated view-dependent visibility of the streamlines in screen space on a GPU roughly based on screen-space occupancy maps by Marchesin et al. [19] to handle line density control. Streamlines typically represent the properties of a continuous vector field, which is not the case for the data presented in this paper, meaning we cannot use the same sparsification approach. Moreover, the above methods do not offer the user explicit control over the grouping, sparsification, and emphasis of instances for which the volume conductor was explicitly designed.

Le Muzic et al. [16] introduced *visibility equalizers* as a tool to interactively sparsify mesoscopic biological data. Their data consist of molecular instances organized into a hierarchical representation. The visibility of these hierarchical groups is estimated in real time and can be adjusted through sliders. Sparsification is hard-bound to the hierarchical scene arrangement, and no additional properties can be used for sparsification. Instances are hidden either randomly or based on the distance from a clipping primitive. Only structurally identical hierarchically organized instances are supported. The scene is rendered through an instancing-enabled graphics pipeline.

Visibility equalizers form the basis for a more general concept of the volume conductor presented in this paper. In the volume conductor, the sparsification procedure is generalized through the use of *sparsification functions*. Furthermore, the hierarchy in the volume conductor is not fixed but can be constructed and interactively adjusted by the user. A significant difference is that the volume conductor operates on instance properties where any property can be selected for sparsification, thus allowing complete flexibility for the users to analyze the data according to any property of interest. This is enabled through the authoring environment, where properties are selected and organized into hierarchies. The reason for this is that the volume conductor additionally targets exploration and analytical scenarios, whereas visibility equalizers were primarily concerned with the communicative visualization intent.

In summary, although related to crowded environments, the above-listed related work fails to address crowded volumes and interactive exploration. Our contribution provides the user with complete control over instance grouping, sparsification, and rendering. While other approaches assume specific structures (e.g., streamlines or molecules), the volume conductor is domain-agnostic and does not pose any constraints on the structures and their properties. We support our claims with three use cases.

3 Volume Conductor

The volume conductor is an explicit visibility management technique for crowded volumetric data, where the user forms groups of instances of structures based on their attributes and controls the visibility of the instances with scented sliders [29] to communicate how many of them should be visible under the current transfer function and camera settings. The architecture of our solution is displayed in Fig. 2 and explained in the following paragraphs.

Two volumes are required at the input: the *raw data volume* and *segmentation volume*. In the latter, each voxel is assigned a numerical identifier corresponding to an individual instance or the background. As the segmentation process is not part of our pipeline, we do not propose any particular segmentation technique, and a suitable one can be chosen based on the given domain. Every instance may hold a set of additional attributes, such as length, volume, surface area, and orientation, supplied with the two volumes. Attributes are not necessarily inherent to each instance, as they can, for example, express local statistics such as the distance to the nearest instance or the number of instances in a given radius.

The user first forms groups of instances by specifying a set of *group membership predicates*, which assigns each instance to a group. Depending on the user's visualization goals, the groups may contain instances of interest with spe-

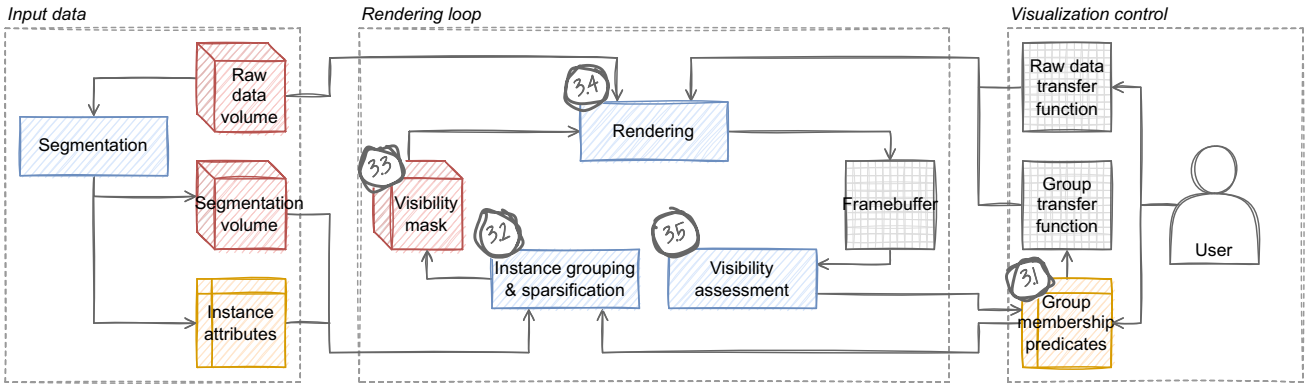

Fig. 2 High-level overview of the volume conductor. The segmentation volume passes through the grouping and sparsification procedure to generate the visibility mask, which is then rendered along with the raw data. Visibility is assessed from the rendered image and used to update the user interface. Circled numbers indicate section numbers, where that part of the volume conductor is described

cific attributes, instances that are less relevant and are to be sparsified, or even those that represent unwanted information, such as segmentation errors or noise. To facilitate this task, we developed an easy-to-use user interface for the manipulation of group membership predicates (we discuss it in Sect. 3.1).

After the groups are formed, the user assigns a color, opacity, and visibility ratio to each group. The colors may include transparency if the user wants a partially transparent group of instances. Our solution readily supports a separate transfer function for each group, but we opted for a single color to keep the user interface simple. The group's visibility ratio is used in the sparsification procedure when determining the number of shown and hidden instances in that group. The user can choose between various sparsification functions to control which instances are affected. The sparsification procedure and various *sparsification functions* are discussed in Sect. 3.2.

Based on the group membership predicates, the *visibility mask* (a volume that encodes the visibility and group membership of every voxel) and a corresponding transfer function are generated for rendering. Their format and the generation process are described in Sect. 3.3. The visibility mask can be visualized with any existing volume rendering method, and we demonstrate this by integrating the volume conductor into directional occlusion shading (DOS) [21] and path tracing renderers.

Finally, the user can *blend the raw data* into the visualization to display information lost during segmentation or to enhance the raw data rendering with the capabilities of the volume conductor. The rendering process and blending are covered in Sect. 3.4.

As the sparsification procedure does not consider occlusion between the instances when determining their visibility, we perform *visibility assessment* by computing the actual visibility ratio of each group as observed from the camera. The ratio is calculated by rendering the segmentation volume into an ID frame buffer and counting different IDs in each group. This information is fed back into the user interface, forming a feedback loop with the user and ensuring that the user settings are accurately reflected in the rendered image. This process is explained in more detail in Sect. 3.5.

3.1 Instance grouping

A group membership predicate may be any Boolean expression that signals instance membership in a group based on its attributes. The user creates the groups by specifying their membership predicates. There is an additional background group with no membership predicates. When computing the visibility mask, the groups are traversed and their predicates evaluated in order to assign the instances to the groups. If an instance satisfies the predicates of several different groups, it is assigned to the first such group; therefore, each instance belongs to exactly one group. If an instance does not satisfy the predicates of any group, it is assigned to the background group.

Group membership predicates are defined in run time by the user. To simplify the user interface, we decided to allow only predicates of a specific format easily represented in the user interface. Every predicate includes the following information that the user can adjust:

- the instance attribute that this predicate is based on, and
- a series of ranges that the attribute value may fall into.

We provide two ways of structuring the group membership predicates: sequential and hierarchical. A sequential set of predicates can be directly evaluated, whereas a hierarchical set must first be linearized. In this context, every path from

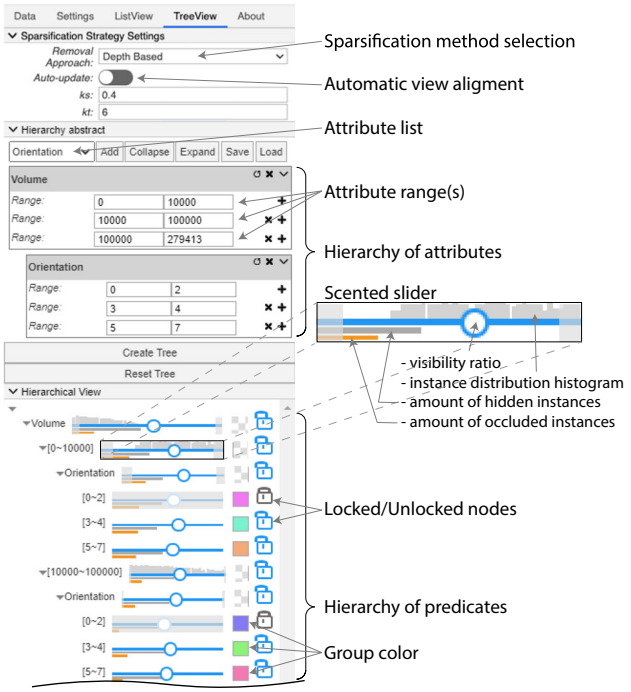

Fig. 3 User interface for instance grouping with an example hierarchy. The user has built an abstract hierarchy from two attributes: volume and orientation (three value ranges each). The hierarchy with the sparsification and coloring controls for each group is shown in the bottom half

the root predicate to a leaf predicate defines one group. The child predicates of a single parent predicate form a disjunctive relation between themselves, whereas the nesting of the predicates signifies a conjunctive relation between the parent and child predicates. The evaluation of child predicates is short-circuited so that every child predicate is only evaluated if an instance also satisfies the corresponding parent predicate. Finding the first available group amounts to finding a path in the hierarchy from a root predicate to a leaf predicate, where all predicates on that path are satisfied. This form of manipulation complies with a typical workflow to a large degree, where the user first broadly divides the instances into several groups and refines them with further subdivisions. For example, in the hierarchy in Fig. 3 the user has defined 3 ranges for the volume of an instance and 3 ranges for the orientation of an instance, resulting in 9 groups.

The user interface for group membership predicate manipulation is illustrated in Fig. 3. First, the user builds a hierarchy of attributes and defines a set of value ranges for each attribute. This form is expanded to obtain a copy of the value ranges of the children for every value range of the parent. After expansion, each path from the root to a leaf node represents a single group, where a single value range of a single attribute is evaluated on each level of the hierarchy.

Groups hold the following additional information used for colorization and sparsification:

- the color of the group used when generating the group transfer function, and
- the visibility ratio between the number of shown and hidden instances in the group.

Once the hierarchy is built, the colors and visibility ratios can be applied to the resulting groups. On creation, each group is assigned a color with a random hue based on the golden ratio sequence [22]. The user can interactively sparsify the groups by dragging their corresponding visibility ratio sliders. Sliders are assigned to all nodes of the hierarchy so that the user can change the visibility ratios of several groups simultaneously. When the user changes the visibility ratio of a group, the visibility ratios of the subgroups are updated accordingly, and vice versa, so that the visibility ratio of the parent group remains the weighted average of the visibility ratios of the subgroups based on the number of member instances. A group can also be locked so that it is not affected by such cascaded updates. Following the scented widget paradigm [29], the scented slider for every group also depicts a histogram of the values of the corresponding attribute so that the user has a rough idea of the distribution of the attribute values of visible instances. The slider contains two additional tracks for the proportion of hidden and occluded instances. These are calculated in the visibility assessment step described in Sect. 3.5.

3.2 Instance sparsification

The sparsification procedure determines whether an instance should be visible or hidden. We only consider binary visibility to keep the user interface simple, although our implementation readily supports partial transparency via the generated transfer function. The sparsification procedure is designed as an extension of the voxel-based methods (e.g., [3,26]) so that the importance of an instance is the average importance of its constituent voxels. First, we randomly shuffle the instances to prevent any correlations between their initial order and spatial distribution. When the sparsification begins, every voxel is assigned an importance, which is averaged over the instances. Afterward, the instances in each group are sorted based on importance, and those with the lowest importance are hidden. The number of instances to be hidden is determined by the visibility ratio of the group, set by the user as described in Sect. 3.1. The function that assigns importance to a voxel is called the *sparsification function*. We use three sparsification functions operating on voxel positions $\mathbf{x}$, each serving a different purpose in the visualization:

- *uniform*, defined as follows:

$$p_u(\mathbf{x}) = 1,$$

which assigns uniform importance and is used to sparsify the volume without changing the data distribution pattern;
- *depth-based*, defined as follows:

$$p_d(\mathbf{x}) = \|\mathbf{x} - \mathbf{e}\|,$$

which assigns importance based on the distance from the camera $\mathbf{e}$ and is used to create a peeling effect;
- *context preserving*, based on the context-preserving model [3] defined as follows:

$$p_c(\mathbf{x}) = \|\nabla v_{\text{raw}}(\mathbf{x})\|^{(\kappa_t \cdot s(\mathbf{x}) \cdot p_d(\mathbf{x}))^{\kappa_s}}.$$

See below for a detailed explanation of different quantities.

Uniform sparsification keeps the spatial distribution of the instances unchanged, whereas depth-based sparsification reveals the internals of the volume similar to a cutaway plane, but without cutting through the instances (see Fig. 7 and the online supplemental material[1]). To balance the two effects, we adapted the context-preserving model [3], which exhibits a similar cutaway plane functionality while allowing us to adjust the sharpness and depth of the cutaway. In [3], the context-preserving method was used to reduce the opacity of the less critical samples, whereas we instead use it to compute the importance of an entire instance. Conceptually, the model places a virtual light into the scene and assigns less importance to instances that receive a considerable amount of light, are located closer to the camera, and are internally more homogeneous. The virtual light acts as a melting source that more strongly affects the instances with a smaller projected area toward the light (refer to [3] for details). This outcome is a direct consequence of the shading factor $s(\mathbf{x})$, for which we use the Blinn–Phong shading model. To keep the user interface simple, we place the virtual light in the same position as the camera, though it could be placed anywhere. Furthermore, the parameter κ_t controls the depth of the cutaway plane, where higher values correspond to cut-offs further away from camera, while the parameter κ_s controls how gradual the cutoff is, where higher values result in a sharper cut-off. When κ_t is zero, this function reduces to uniform sparsification. In our use cases, values from 0 to 10 for κ_s and κ_t worked well. The gradient magnitude $\|\nabla v_{\text{raw}}(\mathbf{x})\|$ of the raw volume value v_{raw} acts as an indicator of the homogeneity of the data, so a more heterogeneous instance is regarded as more important.

Each of the presented sparsification functions has specific uses, and the power of this approach comes from the ability to combine them. When the density of the instances is extremely high, the user might choose to first bring it down to a reasonable level with uniform sparsification and then use a more sophisticated approach, such as depth-based or context-preserving sparsification, as these are view dependent. To achieve this, our method tracks which instances have already been hidden and prioritizes them during sorting. Consequently, the user can change the sparsification function on the fly and layer the results without any additional controls in the user interface. We refer the reader to the online supplemental material for a video demonstrating this feature.

3.3 Visibility mask

To render the groups of instances, we introduce an intermediate representation called the *visibility mask* that encodes the visibility of instances, their group membership, and the color and transparency for rendering. Formally, the visibility mask is a map from $\mathbb{R}^3$ to $\mathbb{R}^2$, constructed so that the voxels belonging to different groups (including the background) map to different 2D tuples. We store the visibility mask as a volume on the GPU. We generate a corresponding transfer function and use the 2D tuples from the visibility mask as texture coordinates for the transfer function, matching traditional post-classification volume rendering. Consequently, we can use any existing volume rendering algorithm to render the visibility mask. In general, the 2D tuples in the visibility mask may be arbitrary, as long as different groups of instances map to different 2D tuples. To make the best possible use of the available transfer function space, we decided to arrange the visibility mask values in a circular pattern, as illustrated in Fig. 4. The background is mapped to the center of the transfer function, whereas the individual groups are spread uniformly around the circle inscribed in the texture space. These locations are denoted as maskValue and are defined as follows:

$$\text{maskValue}(k) = \begin{cases} \left(\frac{1}{2}, \frac{1}{2}\right), & k = 0, \\ \left(\frac{1}{2}, \frac{1}{2}\right) + \left(\frac{1}{2}\cos\phi, \frac{1}{2}\sin\phi\right), & k > 0, \end{cases}$$
$$\phi = \frac{2\pi(k-1)}{N-1},$$

where k is an index of a group (with zero being the background) and N is the total number of groups. This arrangement ensures an uninterrupted interpolation path between the mask values of each group and the background, preventing any classification-related artifacts at instance boundaries, which would not be possible with a 1D transfer function. Intergroup boundaries may still cause slight rendering artifacts; however, they were barely noticeable in the use cases, and correct treatment would necessitate a far more complex solution, such as the one by Al-Thelaya et al. [24].

[1] https://github.com/UL-FRI-LGM/vpt-conductor/raw/master/supplemental.pdf.

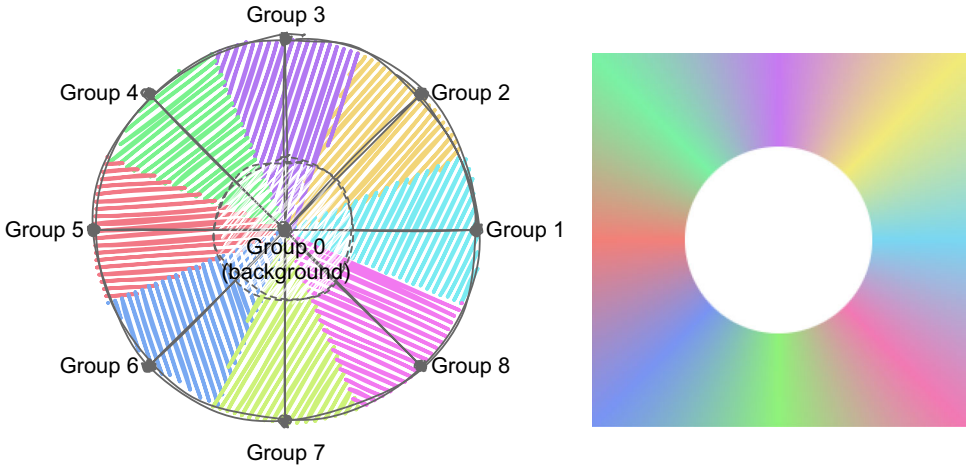

Fig. 4 Visibility mask values are mapped to the transfer function in a circular pattern. Straight lines represent interpolation paths. Paths between the background and individual groups do not intersect with each other

We compute the visibility mask on the GPU with a procedurally generated compute shader, which makes our approach fast, general, and easily extensible. The list of group membership predicates translates into a sequence of if–else statements that takes an instance with its attributes as input and outputs the mask value of the corresponding group (see Fig. 6 and the algorithm in the online supplemental material[1]). The if statements are augmented so that sparsification is included by assigning the mask value of the background to the hidden instances. The compute shader is run for all voxels to generate the visibility mask. The corresponding transfer function is generated by wrapping a $1 \times N$ strip of pixels around the transparent center of the transfer function (see Fig. 4).

The motivation behind this design lies in the inability to interpolate the integer labels of the segmentation volume. One strategy to overcome this is to use nearest neighbor sampling, but this would result in a blocky volume and low-quality image. This problem was also recognized by Al-Thelaya et al. in a recent contribution [24], but their solution involves considerable processing time and a specialized rendering algorithm, as the solution is primarily targeted at data analysis, not rendering. By contrast, the visibility mask is much simpler and orders of magnitude faster to compute, and does not necessitate a specialized rendering algorithm. Additionally, we can leverage hardware interpolation of the 2D tuples from the visibility mask and use post-classification during rendering. Thus, we preserve the high-frequency details in the volume and retain sufficient rendering quality.

Since all voxels from a single instance are assigned the same mask value, instance boundaries may appear blocky despite the ability to interpolate the mask volume. However, with the ability of interpolation, we can filter the mask volume with a low-pass filter, effectively smoothing out the instance boundaries without removing the high-frequency details, which are stored in the raw data volume. The effect of filtering is shown in Fig. 5.

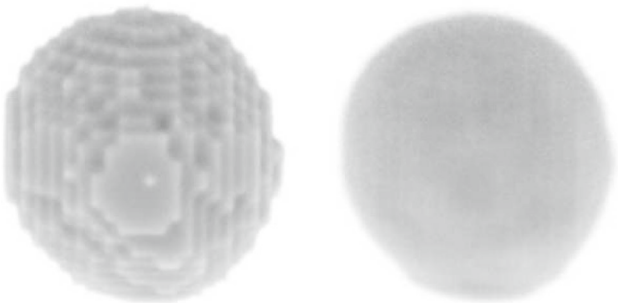

Fig. 5 A single instance rendered without (left) and with (right) visibility mask filtering. A $3 \times 3 \times 3$ box filter was used

3.4 Rendering and blending

The volume conductor is independent of the rendering method. We chose to demonstrate its functionality using two rendering methods: directional occlusion shading [21] because it can simulate direct illumination at interactive frame rates and path tracing for its ability to produce physically realistic results (refer to the online supplemental material[1] for a visual comparison). We augmented the rendering procedure with blending between the visibility mask and raw data volume to allow the user to see the internals of the instances while using the sparsification functionality. In addition to visibility mask v_{mask} and its corresponding transfer function t_{mask}, we have raw data volume v_{raw} and its transfer function t_{raw} defined by the user. Both are sampled to obtain color C and opacity a from position $\mathbf{x}$ within the volume:

$$(C_{\text{mask}},\ a_{\text{mask}}) = t_{\text{mask}}(v_{\text{mask}}(\mathbf{x})),$$
$$(C_{\text{raw}},\ a_{\text{raw}}) = t_{\text{raw}}(v_{\text{raw}}(\mathbf{x})).$$

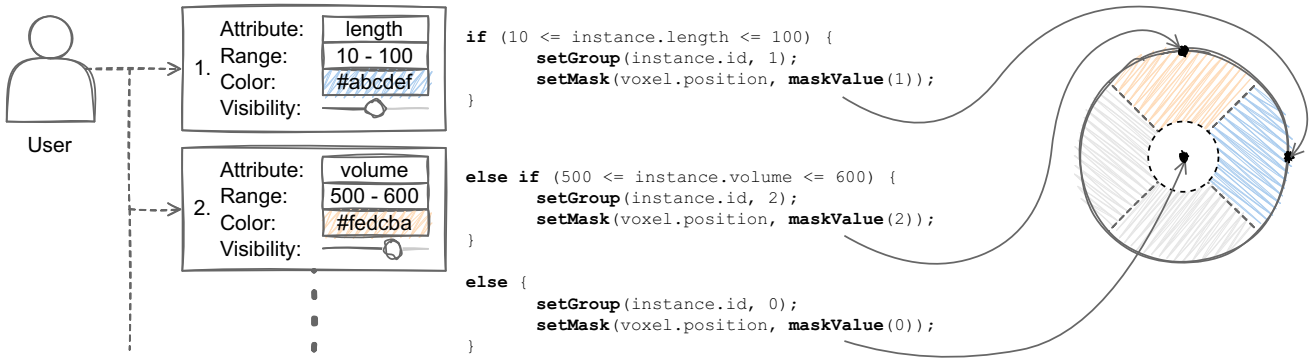

Fig. 6 Shader generation and mask transfer function generation from group membership predicates

We linearly blend the colors with the interpolation weight w_{color}:

$$C_{\text{final}} = (1 - w_{\text{color}})C_{\text{mask}} + w_{\text{color}}C_{\text{raw}}.$$

We also subject the opacity of the raw data to sparsification; therefore, it is treated slightly differently than the colors. First, we transfer the opacity from the visibility mask to the raw data volume with the interpolation weight w_{transfer}:

$$a_{\text{transfer}} = (1 - w_{\text{transfer}})a_{\text{mask}} + w_{\text{transfer}}a_{\text{mask}}a_{\text{raw}}.$$

Then, we compute the final opacity as a linear interpolation between the transferred opacity and raw data opacity with interpolation weight w_{alpha}:

$$a_{\text{final}} = (1 - w_{\text{alpha}})a_{\text{transfer}} + w_{\text{alpha}}a_{\text{raw}}.$$

The user sets all interpolation weights in the user interface. When w_{color} and w_{alpha} are both 1, this reduces to raw data volume rendering. When both weights and w_{transfer} are 0, only the visibility mask defines the output. Example images produced with the values between these extremes can be seen in the supplemental video.

While t_{mask} is generated automatically by the volume conductor, t_{raw} is constructed by the user. Both t_{raw} and t_{mask} have a large impact on the final rendering: t_{mask} is responsible for sparsification and colorization, whereas t_{raw} is responsible for the volumetric visualization of the internals of the instances. If t_{raw} sets the opacity of certain structures to zero, those structures would not be visible even if the sparsification procedure marks them as visible. In a crowded environment, t_{raw} is practically useless by itself, while t_{mask} can be used effectively without t_{raw}. There exist many different methods for constructing a transfer function, which can be manual, semi-automatic, or automatic. We opted for a manual one, which was an arbitrary choice, independent of the volume conductor, and easy to implement. Since the main contributions of the volume conductor are interactive sparsification and colorization, we decided to focus on those aspects rather than the construction of t_{raw}. Nevertheless, t_{mask} could very well be involved in the construction of t_{raw}, for example, by enhancing the instance boundaries, or by fine-tuning the t_{raw} toward a specific instance or group of instances. We leave such enhancements for future research.

3.5 Visibility assessment

Due to the occlusion and projection, the density of the instances after sparsification often does not precisely reflect the visibility ratio set by the user. We measure the actual visibility ratio of the instances as observed from the camera and inform the user about the number of occluded instances. We update an ID frame buffer during rendering, which holds the ID of the nearest visible instance for every pixel and the group to which it belongs. We measure the number of visible instances in each group by counting the unique instance IDs from the group present in the ID frame buffer. We know the total number of instances in each group and the number of hidden instances; thus, we also know how many instances from that group are occluded. This information is presented to the user through an additional track under the scented slider (see Fig. 3).

4 Implementation Details

With the focus on making the method interactive, we implemented the computationally intensive parts on the GPU and integrated them into the VPT framework [17] using WebGL 2.0 Compute.[2] We also retained the ability to perform hardware sampling and interpolation to ensure the method remains independent of the rendering technique. We store the segmentation volume on the GPU as a 3D texture of 32-bit unsigned integers (format `R32UI`) and their respective attributes in a shader storage buffer object (SSBO).

[2] https://github.com/UL-FRI-LGM/vpt-conductor.

The layout and format of the data in the SSBO are such that it directly maps to an array of `structs` in a shader, where each `struct` holds the attributes of a single instance. Thus, each integer value from the segmentation volume acts as an index for accessing the attributes of the corresponding instance. As the layout of the `struct` varies between datasets, it must be supplied along with the volumes and instance attributes. We use a simple JSON file with the attribute names and types, which is enough to generate the struct definition in the OpenGL shading language (GLSL). We access the instance attributes in the procedurally generated compute shader when generating the visibility mask. As displayed in Fig. 6, the group membership predicates translate directly into a sequence of GLSL if-else statements, which assign a mask value to every voxel in the visibility mask, similar to the approach by Schulte zu Berge et al. [23], although our predicates are not used during rendering but instead are employed to generate the visibility mask. We store the resulting visibility mask on the GPU as a 3D texture of 2D 8-bit normalized integer tuples. We use the format `RGBA8` because it is one of the few that can be written to using a WebGL 2.0 compute shader (in contrast to `RG8`, which would be more appropriate), and it allows hardware interpolation. The compute shader is regenerated after every change in the group membership predicates. The shader execution is independent between voxels, so no communication is needed between the workgroups, and the size of the workgroups may be arbitrary. In our implementation, we chose $16 \times 16 \times 1$ because it maps reasonably well to most modern hardware.

5 Results

5.1 Use cases

We tested the volume conductor on two domains: fiber-reinforced polymers in the field of material science and intracellular organelles in the field of microbiology. Volumetric images from both domains are crowded and, therefore, a perfect fit for the volume conductor. We collaborated with experts from these two fields and instructed them to use the volume conductor in their daily workflow. After a month of everyday use, we gathered their feedback and asked them to state the advantages and disadvantages of the method compared to their usual tools and workflow.

5.1.1 Material Science

Fiber-reinforced polymers are in high industrial demand due to their strength, durability, elasticity, and low weight, and the demand is steadily growing [12]. These physical properties are directly related to the distribution and density of internal structures, such as fibers, inclusions, and pores, and their properties, such as length, volume, and orientation. Therefore, a tool that can isolate and visualize structures with specific properties is crucial for scientists to analyze and improve the materials. In this domain, a typical imaging technique is the 3D X-ray CT. DVR renderings of the resulting volumes are often difficult or even impossible to interpret due to crowdedness. Specialized analysis tools are available, such as Feature Scout [28].

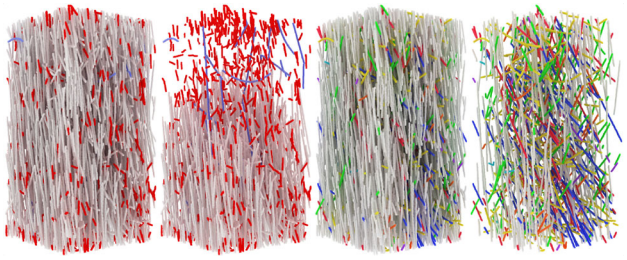

Fig. 7 Fiber use case. From left to right: (1) colorization of short (red) and bent (blue) fibers, (2) the same as (1) but with 50 % of the remaining fibers hidden by the depth from above, (3) colorization by orientation, and (4) the same as (3) but with 80 % of the vertical fibers hidden with the context-preserving model

We considered two datasets: (1) a $400 \times 401 \times 800$ volume containing 3828 glass fiber instances with 18 attributes each, and (2) a $512 \times 512 \times 512$ volume containing 6888 instances of pores between carbon fibers with 41 attributes each. Both volumes were acquired with 3D X-ray CT. The use cases were provided by the University of Applied Sciences, Upper Austria, a research associate with three years of experience and a senior researcher with more than 15 years of experience in material science data visualization. The experts compared the use of Feature Scout and volume conductor on two use cases commonly encountered in their everyday work. They were also involved in the design process for both systems and had good insight into their functionalities.

Use Case 1—Fiber analysis In glass fiber-reinforced materials, fiber characteristics must be analyzed, and the spatial distribution of the fibers with specific properties must be determined. In a strong material, the fibers are uniformly oriented and uniformly distributed across the volume. Short and bent fibers that do not contribute to the strength of the material can be quickly identified by colorization (Fig. 7, two left images). The interactive sparsification feature of the volume conductor can be used to obtain a general overview of the directional distribution of the fibers (Fig. 7, right two images).

Use Case 2—Pore analysis In carbon fiber-reinforced polymers, the existing pores inside the material must be examined, especially their shape and spatial distribution. For example, needle-shaped pores have a higher potential for crack initiation than elliptical or spherical pores. With both the volume conductor and Feature Scout, an appropriate categorization

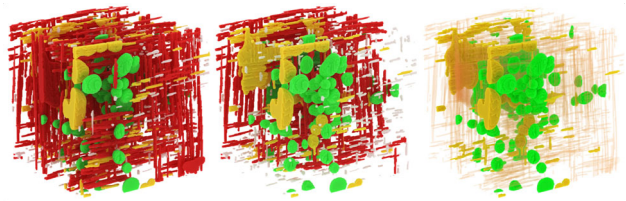

Fig. 8 Use case with pores. From left to right: (1) colorization by roundness (needle-shaped pores are red), (2) the same as (1) but with 50 % of the needle-shaped pores hidden by depth, and (3) the same as (2) but with a higher transparency of needle-shaped pores to emphasize the round ones

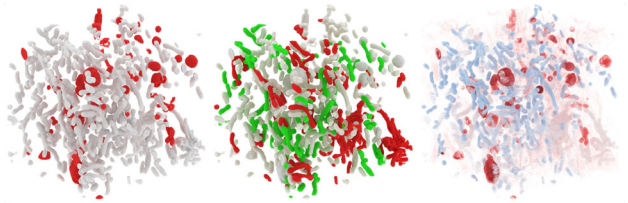

Fig. 9 Mitochondria use case, showing colorization by organelle type, where mitochondria are colored red (left image), colorization of branched (red) and thinned (green) mitochondria (middle image), and blending with raw data (right image). The volume has been sparsified to only show mitochondria and endolysosomes, whereas other organelles, such as Golgi apparatuses and fusiform vesicles, have been hidden. A more crowded example of the same dataset can be seen in [2,30]

can be constructed to obtain a visual overview of the needle-shaped pores by categorizing the instances according to shape (Fig. 8, two left images). In contrast with Feature Scout, the volume conductor provides interactive sparsification for a better overview of the pore distribution.

After using the volume conductor, the material science experts came to the following conclusions:

- Uniform sparsification primarily helps when sparsifying the crowded volume of fibers while maintaining the distribution of the instances throughout the volume.
- View-aligned sparsification functions (depth-based and context-preserving functions) are beneficial during the exploration of the volume internals (Fig. 8, right two images). Compared to hard culling, such sparsification does not deform or cut away parts of instances and therefore does not alter their visual appearance. The ability to layer view-aligned sparsification on top of uniform sparsification is a powerful feature when determining the distribution properties of internal instances.
- On-the-fly changes to the membership predicates accelerate the exploration process because the user can add additional predicates and refine the visualization at will, which is much more tedious in Feature Scout.
- The ability to select individual instances directly in the rendering and inspect their attributes is missing.

The experts claim that sparsification is an invaluable tool for data exploration in both use cases because it allows them to immediately observe the distribution of fibers and pores with specific properties. The experts stated that a featureful tool, such as Feature Scout, could benefit from the volume conductor and help them in their everyday workflow.

5.1.2 Cell biology

We tested the volume conductor on a 3D microscopy sample of a cell inside a mouse bladder. The volume was provided by the Institute of Cell Biology of the University of Ljubljana. Microbiologists at the institute study the distribution, density, and shape of intracellular organelles to explore and understand various cellular processes. In one specific case, they were interested in the mitochondria and endolysosomes regarding their size, shape, curvature, and possible branchings and narrowings (Fig. 9, middle image). Considering that thousands of such organelles may exist even in a small subsection of a cell, scientists need a tool, such as the volume conductor, to visualize and analyze organelles and their properties.

The 1280 × 1024 × 1024 microscopy sample was segmented, and the features were extracted (see [30] for details), yielding 3051 instances with 21 attributes each. The use case was given by the Institute of Cell Biology at the Faculty of Medicine of the University of Ljubljana: an assistant professor with 16 years of experience and a professor with more than 30 years of experience. They are both familiar with the software used in their fields, such as ImageJ and 3D Slicer. They tested the volume conductor in a one-day test run under the supervision of one of the authors. The professors compared the volume conductor with the software stack they use in their regular work.

Use Case 3—Mitochondria analysis For analysis and exploration of the volumetric microbiological data obtained using modern microscopy techniques (e.g., focused ion beam scanning electron microscopy), researchers still primarily employ slice-based visualization tools (e.g., ImageJ) or regular DVR tools (e.g., 3D Slicer). More advanced analysis tools exist such as Imaris Essentials, Invivo, and Osirix, but, to the best of our knowledge, these tools only provide transfer function-based volume rendering and surface-based segmentation visualization. The tasks, such as examining the distribution of intracellular organelles, are very time-consuming and demanding with existing tools. The volume conductor facilitates these tasks considerably and enables users to perform complex analyses. For example, the shapes, quantity, orientation, and distribution of mitochondria can be studied using the grouping and sparsification functionality of the volume conductor. Additionally, microbiologists can view the segmented intracellular structures with raw volumetric data, allowing

them to visualize the internals of the organelles in a sparsified environment (Fig. 9, right image).

After using the volume conductor, microbiology experts came to the following conclusions:

- The ability to sparsify the volume and colorize the instances with specific properties is beneficial to the exploration process. Compared to the existing tools, we can benefit from the volume conductor when getting an overview of the data, as the uninteresting instances can be hidden, shifting the focus toward the relevant ones.
- The set of sparsification functions enables more accessible volume exploration and provides more refined control over what to display and what to hide. The layering functionality can be used to quickly sparsify one type of organelles while opening up the internals of the volume for further analysis.
- A joint visualization of raw and segmented data offers the user a view of the interior of the segmented structures without the surrounding clutter. This is especially helpful when the images have low contrast or the raw values of the instances are similar to those of the background.
- A disadvantage is that the user must provide the segmentation volume and instance attributes, which are not always available or easily computed or obtained.
- It would be beneficial if such a visualization tool was coupled with a specialized analysis tool. Specifically, selecting individual instances by clicking on them in the rendering and observing their properties would be a very powerful feature.

5.2 Performance evaluation

We evaluated the volume conductor on three computers: a laptop with Intel HD Graphics 530 integrated graphics with 20 GB of shared RAM, a desktop computer with an Nvidia GeForce GTX 1060 graphics card with 6 GB of video RAM, and a professional workstation with an Nvidia Quadro RTX 8000 graphics card with 48 GB of video RAM. Three datasets were used: fibers, pores, and mitochondria, presented in Sect. 5.1. We measured the time of execution of the following three steps:

1. linearization of the predicate hierarchy and shader recompilation,
2. visibility mask computation, and
3. rendering with directional occlusion shading.

The measurements were executed 10 times (10 linearizations, 10 visibility mask computations, and 10 rendered frames), and the average time per step was recorded. We used the Google Chrome web browser version 87. The evaluation was performed for a nontrivial predicate hierarchy with two layers and five ranges for each layer. The camera was configured so that the volume filled the screen. This configuration is essential because the rendering time is highly view-dependent. All measurements are gathered in the supplemental material[1], and the measurements for Nvidia Quadro RTX 8000 are presented in Fig. 11.

Additionally, we measured the time for visibility mask computation with respect to volume size and the number of instances. We used 4 synthetic datasets with 2000, 2258, 3658, and 22670 instances, at various resolutions from $8 \times 8 \times 8$ to $768 \times 768 \times 768$. The measurements, plotted in Fig. 10, show that even if the visibility mask is recalculated every frame (which is an unrealistic stress test), we can achieve interactive framerates for volumes up to $512 \times 512 \times 512$ on commodity GPUs, and even more, if visibility mask filtering is disabled. Of course, this is true only if a sufficiently fast rendering method is also used. Figure 10 shows that the visibility mask computation time is linear in the number of voxels. Since the number of instances is small in relation to the number of voxels, the corresponding overhead is asymptotically negligible.

The performance evaluation demonstrates that the presented approach remains interactive on platforms with varying capabilities. Figure 11 and the related graphs in the supplemental material[1] show much better performance of the dedicated graphics hardware compared to integrated graphics. The visibility mask computational time is proportional to the volume size, and the rendering time is proportional to the frame buffer resolution, both of which conform to expectations. The measurements reveal one peculiarity in the shader rebuild time, which is substantially smaller for integrated graphics than dedicated graphics hardware. This discrepancy is likely caused by the additional communication time between the CPU and dedicated GPU. Due to the nonoptimized implementation, the most computationally intensive part of the volume conductor is the rendering process, which should not be an issue as the rendering method can be easily swapped. The visibility mask computation time is more relevant to the evaluation of the method than rendering time and is low enough for interactive use even on commodity hardware. In fact, in a realistic scenario, the visibility mask is recomputed only when the user modifies the instance groups or the visibility ratio of a group, but not during rendering.

6 Discussion

The feedback from both expert groups reveals that the interactive sparsification functionality is the most valuable feature of the volume conductor. The volume conductor provides a means for acquiring a holistic view of the data before detailed analysis. It shifts the focus from a low-level view of the raw data to a high-level view of the structures and their proper-

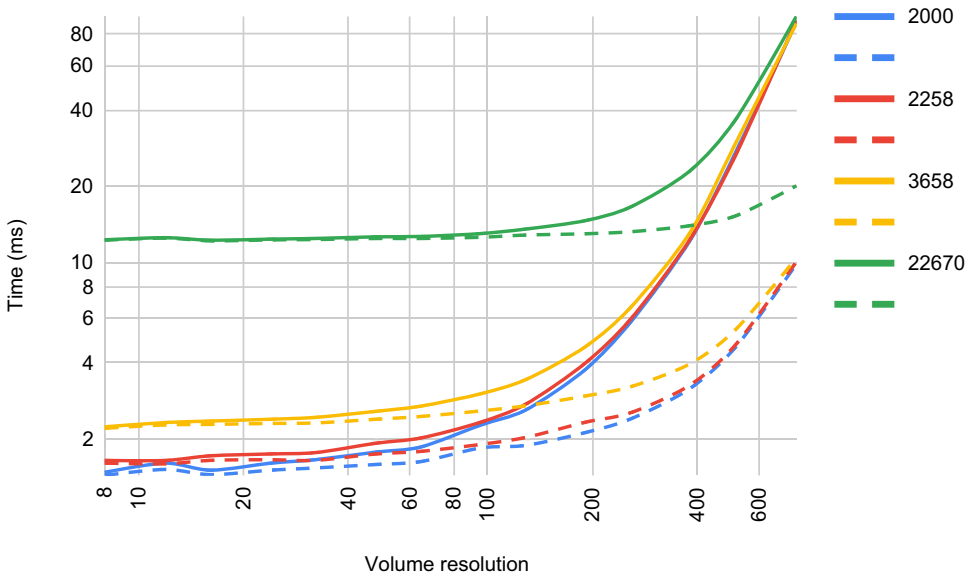

Fig. 10 Visibility mask computation time with respect to volume size and the number of instances, measured on consumer-grade hardware, with (solid lines) and without (dashed lines) visibility mask filtering

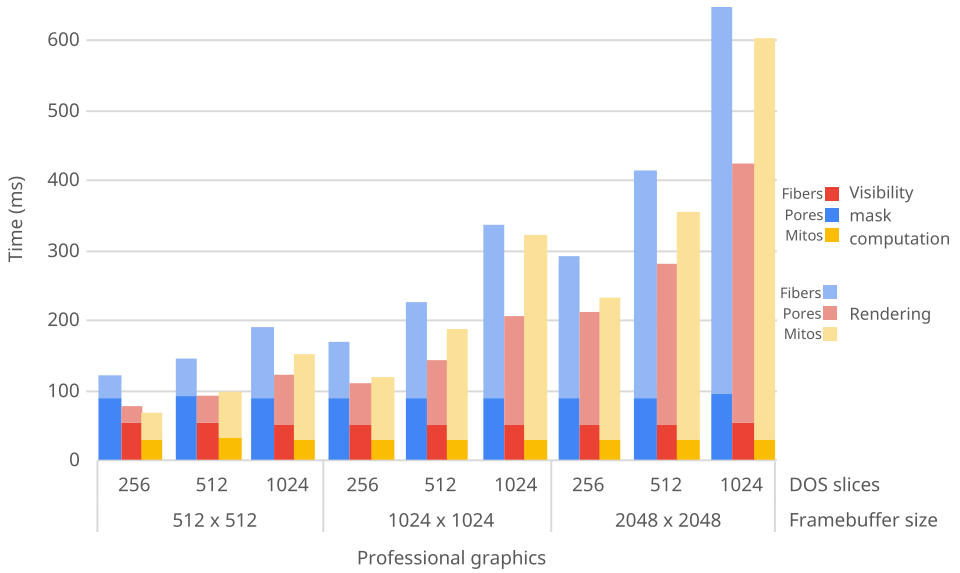

Fig. 11 Performance evaluation on a professional workstation with an Nvidia Quadro RTX 8000 graphics card. Linearization and shader recompilation times are negligible and not depicted in the graph

ties, and the user interface is designed to support this view. For the material science experts, the main benefit of the volume conductor is sparsification and its independence from the rendering method, which makes the volume conductor particularly easy to integrate into existing tools. Specifically, the volume conductor could be easily integrated into Feature Scout, enhancing it with sparsification and colorization capabilities, while benefiting from its analytics.

The volume conductor can enhance visualization tools with sparsification capabilities, as long as at least a crude instance segmentation is available. According to the microbiology experts, the reliance on segmentation availability is one of the main disadvantages of the volume conductor. Obtaining an accurate instance segmentation is a tedious process if automatic tools are not available, and without it, the volume conductor is inoperative. However, a crude segmentation is often enough to reap the benefits of the volume conductor. In fact, it is even possible to identify segmentation errors when the visualization is combined with raw data. For example, the intracellular organelle dataset shown in Fig. 9 indeed contained numerous segmentation errors. On several occasions, multiple organelles have been merged into a single instance and consequently marked as being branched. Such errors are clearly visible in the rendering and are even more apparent during interactive grouping and sparsification.

In contrast with fiber-reinforced polymers, the microbiological use case revealed a strong need for a joint display of segmented and raw data. According to the microbiology experts, the internal structures inside specific organelles can easily be identified, which was not possible with existing

methods that could not hide the surrounding clutter. In the volume conductor, though, all of this is possible out of the box, without any data conversions.

In many cases, the users work with layered materials, where the main challenge is visualizing the interior of the dataset. The volume conductor works equally well for layered materials. Layers can be designated in the segmentation and may be hidden during visualization in the same way as other instances. If layers can be further broken down into individual instances, an appropriate sparsification function can be used to achieve the peeling effect, or the distance from the interior can be exposed as an attribute to form groups layer by layer.

Experts from both groups pointed out the ease of use of the volume conductor due to the user interface designed specifically for this purpose. However, when using a joint display of segmented and raw data, the user must still design an appropriate transfer function for the raw data. This necessity results in a usability gap between the two rendering modes. The user can become lost even in the easy-to-use user interface when presented with a wide array of possibilities for the attribute hierarchy, and the confusion only intensifies with a long list of instance attributes. It is not unusual to compute more than 20 different features for a single instance. With machine learning, this number of features is easily surpassed by many orders of magnitude. Machine learning could eventually be used for automatic instance grouping, colorization, and sparsification to shield the user from the numerous resulting possibilities for visualization. However, this is out of the scope of this paper and will be considered for future work.

Another important challenge lies in the attribute selection and grouping, and the possibility of the sparsification to skew or distort the perceived instance distribution. At the start of data exploration, the experts usually have a rough idea of what attributes and values to select for grouping. After their initial selection, they will get visual feedback through the visualization and scented sliders on whether their selection was suitable. First, the scented sliders show a histogram of attribute values, which help the experts confirm or reject their expectations about the distribution of the attribute values. Second, the contradicting examples and outliers will be visible in the rendering. In the worst case, such important instances will be hidden by the sparsification, but this can easily be avoided by instructing the experts to always examine the visualization of each group without sparsification in addition to their usual workflow.

In addition to sparsification, different types of projections can also affect spatial perception. In particular, perspective projection tends to distort objects, which might negatively affect distribution and shape analysis. Any projection can be used in the volume conductor so long as the same projection is used for both the rendering and visibility assessment.

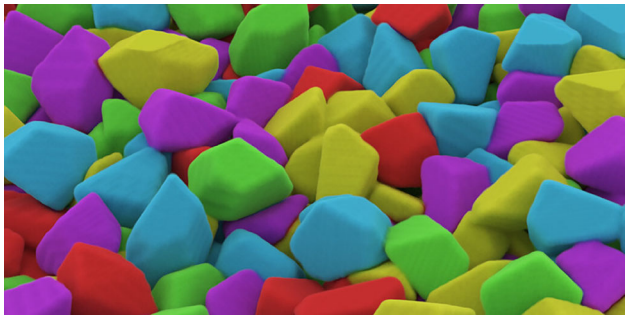

Fig. 12 A zoomed-in view of a synthetic $512 \times 512 \times 512$ volume with 22670 instances in 5 groups

The most apparent technical drawback of the method is inter-group boundaries, which are not accounted for and may cause slight rendering artifacts as the interpolation path between the mask values of two different groups may cross mask values from other groups. This is a valid concern, and such situations can often occur in practice. Fortunately, in our use cases, they cannot be seen in the final renders. To be sure that this problem is insignificant in practice, we stress-tested our method on a synthetic $512 \times 512 \times 512$ volume tightly packed with 22670 instances, which we equally and randomly divided into 5 groups. Surprisingly, there were no noticeable artifacts in the final rendering (see Fig. 12), no matter what kind of sparsification we chose. We, therefore, conclude that it is safe to disregard this concern, especially in the pursuit of speed and simplicity of the method.

7 Conclusion

In the paper, we presented the volume conductor—a smart visibility management technique for visualization and sparsification of crowded volumetric data. The technique uses an instance segmentation volume and user-defined group membership predicates to generate the visibility mask, which encodes the visibility of instances and instance groups. The user controls the sparsification through sparsification functions, which assign importance to each voxel. The instances are sorted based on the average importance of their constituent voxels and shown or hidden based on the user-defined visibility ratio. Finally, a visibility assessment forms a feedback loop with the user. The system is interactive on consumer-grade hardware.

We demonstrated how the volume conductor can be beneficial for exploring crowded volumes compared to regular DVR. With the proposed method, the user can group the instances based on simple predicates or a hierarchy of predicates and interactively adjust their density to reveal more information about the structure of the volume and the distribution of instances. We achieved this by separating instance

grouping, visibility mask computation, transfer function generation, and rendering. The resulting method is general and easily extensible due to the procedurally generated compute shader. We applied the technique to two domains, in which the experts reported improvements in their workflow and exploration process.

In the future, we intend to improve the volume conductor with single or multiple instance selections in the cases where the user aims to make fine adjustments to the visibility of specific instances. We also plan a method for automatic or user-guided instance grouping to make the exploration process even faster and simpler. We believe the technique will prove invaluable for interactive data exploration in many research fields where crowded volumetric environments are in use.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00371-023-02828-8.

Acknowledgements The authors would like to thank Julia Maurer from the University of Applied Sciences, Upper Austria, for providing carbon-fiber-reinforced polymer data and the presented use cases; Samo Hudoklin and Rok Romih from the University of Ljubljana for providing cell structure data and the presented use cases.

Funding Open access publishing supported by the Slovenian Research Agency and Central Technical Library in Ljubljana. The research was partially supported by the King Abdullah University of Science and Technology—award number BAS/1/1680-01-01, partially by funding from the Austrian Research Promotion Agency (FFG) within the program line "TAKE OFF," FFG Grant No. 874540 "BeyondInspection," and by research subsidies granted by the government of Upper Austria during the "X-Pro" project.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors have no financial or proprietary interests in any material discussed in this article.

References

1. Ament, M., Zirr, T., Dachsbacher, C.: Extinction-optimized volume illumination. IEEE Trans. Visual Comput. Graph. **23**(7), 1767–1781 (2017)
2. Bohak, C.: Intracellular Compartments in Urothelial Cells of Mouse Bladder. In: KAUST Cell Visualization Summit. King Abdullah University of Science and Technology (KAUST), Thuwal, Kingdom of Saudi Arabia (2019)
3. Bruckner, S., Grimm, S., Kanitsar, A., Groller, E.: Illustrative context-preserving exploration of volume data. IEEE Trans. Visual Comput. Graph. **12**(6), 1559–1569 (2006)
4. Chan, M.Y., Qu, H., Chung, K.K., Mak, W.H., Wu, Y.: Relation-aware volume exploration pipeline. IEEE Trans. Visual Comput. Graph. **14**(6), 1683–1690 (2008)
5. Correa, C.D., Ma, K.L.: Visibility histograms and visibility-driven transfer functions. IEEE Trans. Visual Comput. Graph. **17**(2), 192–204 (2011)
6. Diepstraten, J., Weiskopf, D., Ertl, T.: Transparency in interactive technical illustrations. Comput. Graph. Forum **21**(3), 317–325 (2002)
7. Diepstraten, J., Weiskopf, D., Ertl, T.: Interactive cutaway illustrations. Comput. Graph. Forum **22**(3), 523–532 (2003)
8. Günther, T., Rössl, C., Theisel, H.: Opacity optimization for 3D line fields. ACM Trans. Graph. **32**(4), 1–8 (2013)
9. Günther, T., Rössl, C., Theisel, H.: Hierarchical opacity optimization for sets of 3D line fields. Comput. Graph. Forum **33**(2), 507–516 (2014)
10. Günther, T., Schulze, M., Esturo, J.M., Rössl, C., Theisel, H.: Opacity optimization for surfaces. Comput. Graph. Forum **33**(3), 11–20 (2014)
11. Günther, T., Theisel, H., Gross, M.: Decoupled opacity optimization for points, lines and surfaces. Comput. Graph. Forum **36**(2), 153–162 (2017)
12. Heinzl, C., Stappen, S.: STAR: visual computing in materials science. Comput. Graph. Forum **36**(3), 647–666 (2017)
13. Kanzler, M., Ferstl, F., Westermann, R.: Line density control in screen-space via balanced line hierarchies. Comput. Graph. **61**, 29–39 (2016)
14. Kruger, J., Schneider, J., Westermann, R.: ClearView: an interactive context preserving hotspot visualization technique. IEEE Trans. Visual Comput. Graph. **12**(5), 941–948 (2006)
15. Kubisch, C., Tietjen, C., Preim, B.: GPU-based smart visibility techniques for tumor surgery planning. Int. J. Comput. Assist. Radiol. Surg. **5**(6), 667–678 (2010)
16. Le Muzic, M., Mindek, P., Sorger, J., Autin, L., Goodsell, D.S., Viola, I.: Visibility equalizer cutaway visualization of mesoscopic biological models. Comput. Graph. Forum **35**(3), 161–170 (2016)
17. Lesar, Ž., Bohak, C., Marolt, M.: Real-time interactive platform-agnostic volumetric path tracing in WebGL 2.0. In: Proceedings of the 23rd International ACM Conference on 3D Web Technology—Web3D '18, pp. 1–7. ACM Press, New York (2018)
18. 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), 669–691 (2016)
19. Marchesin, S., Chen, C.K., Ho, C., Ma, K.L.: View-dependent streamlines for 3D vector fields. IEEE Trans. Visual Comput. Graph. **16**(6), 1578–1586 (2010)
20. Ropinski, T., Steinicke, F., Hinrichs, K.: Interactive importance-driven visualization techniques for medical volume data. In: Proceedings of the International Fall Workshop on Vision, Modeling, and Visualization (VMV), pp. 273–280 (2005)
21. Schott, M., Pegoraro, V., Hansen, C., Boulanger, K., Bouatouch, K.: A directional occlusion shading model for interactive direct volume rendering. Comput. Graph. Forum **28**(3), 855–862 (2009)

22. Schretter, C., Kobbelt, L., Dehaye, P.O.: Golden ratio sequences for low-discrepancy sampling. J. Graph. Tools **16**(2), 95–104 (2012)
23. Schulte zu Berge, C., Baust, M., Kapoor, A., Navab, N.: Predicate-based focus-and-context visualization for 3D ultrasound. IEEE Trans. Visual Comput. Graph. **20**(12), 2379–2387 (2014)
24. Thelaya, K.A., Agus, M., Schneider, J.: The mixture graph-a data structure for compressing, rendering, and querying segmentation histograms. IEEE Trans. Visual Comput. Graph. **27**(2), 645–655 (2021)
25. Viega, J., Conway, M.J., Williams, G., Pausch, R.: 3D magic lenses. In: Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology, UIST '96, pp. 51–58. Association for Computing Machinery, New York (1996)
26. Viola, I., Gröller, E.: Smart visibility in visualization. In: Neumann, L., Sbert, M., Gooch, B., Purgathofer, W. (eds.) Computational Aesthetics in Graphics, Visualization and Imaging, pp. 209–216. The Eurographics Association (2005)
27. Wang, L., Zhao, Y., Mueller, K., Kaufman, A.: The magic volume lens: An interactive focus+context technique for volume rendering. In: IEEE Visualization, pp. 367–374. IEEE (2005)
28. Weissenbock, J., Amirkhanov, A., Li, W., Reh, A., Amirkhanov, A., Groller, E., Kastner, J., Heinzl, C.: FiberScout: An interactive tool for exploring and analyzing fiber reinforced polymers. In: 2014 IEEE Pacific Visualization Symposium, pp. 153–160. IEEE (2014)
29. Willett, W., Heer, J., Agrawala, M.: Scented widgets: improving navigation cues with embedded visualizations. IEEE Trans. Visual Comput. Graph. **13**(6), 1129–1136 (2007)
30. Žerovnik Mekuč, M., Bohak, C., Hudoklin, S., Kim, B.H., Romih, R., Kim, M.Y., Marolt, M.: Automatic segmentation of mitochondria and endolysosomes in volumetric electron microscopy data. Comput. Biol. Med. 103693 (2020)

Žiga Lesar is a Ph.D. student, a researcher, and a teaching assistant at the Faculty of Computer and Information Science, University of Ljubljana, Slovenia. He received his B.Sc. in 2014 and M.Sc. in 2018 for his work on interactive volume rendering with web technologies. His research is focused primarily on interactive computer graphics, especially volume rendering and visualization.

Ruwayda Alharbi is a Ph.D. student at KAUST, Saudi Arabia. She received her master's degree in 2016 from King Saud University, Saudi Arabia. Her research interests lie in scientific visualization, where she focuses on designing novel visualization methods that support exploration and understanding of 3D biological models.

Ciril Bohak is a researcher and a teaching assistant at the Faculty of Computer and Information Science, University of Ljubljana, Slovenia. He received B.Sc., M.Sc. and Ph.D. from University of Ljubljana. His research covers computer graphics, scientific visualization, and human–computer interaction.

Ondřej Strnad is a research scientist at KAUST, Saudi Arabia. He received his doctoral degree from Masaryk University in Brno, Czech Republic in 2014. His research interests stretch over scientific visualization, geometry algorithms, and computer graphics. Recently, he joined NANOVIS group at KAUST to work on technologies that deliver new visualizations and techniques regarding mesoscale biological models.

Christoph Heinzl received his PhD degree in computer science from TU Wien in the field of visualization and analysis of industrial XCT data. He is currently a professor at the University of Passau and a group leader at Fraunhofer IIS/EZRT. His current research covers visual analysis and visualization in nondestructive testing and materials science, a research domain in which he acquired various applied and basic research grants on national and European level.

Matija Marolt is an associate professor at the University of Ljubljana, Faculty of Computer and Information Science. He received his Ph.D. in 2002 and is currently head of the Laboratory for Computer Graphics and Multimedia and is the chair for Multimedia. His research interests are in multimedia information retrieval and visualization.

Ivan Viola is an associate professor at King Abdullah University of Science and Technology, Saudi Arabia. Viola has graduated from TU Wien, Austria, in 2005 and moved for a postdoctoral fellowship to the University of Bergen, Norway, where he was gradually promoted to the rank of Professor. In 2013, he has been awarded a Vienna Science and Technology Fund grant to establish research group at TU Wien. Viola has co-founded the startup Nanographics, to commercialize nanovisualization technologies.