This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3133592, IEEE Transactions on Visualization and Computer Graphics

Nanotilus: Generator of Immersive Guided-Tours in Crowded 3D Environments

Ruwayda Alharbi, Ondřej Strnad, Laura R. Luidolt, Manuela Waldner, David Kouřil, Ciril Bohak, Tobias Klein, Eduard Gröller, Ivan Viola

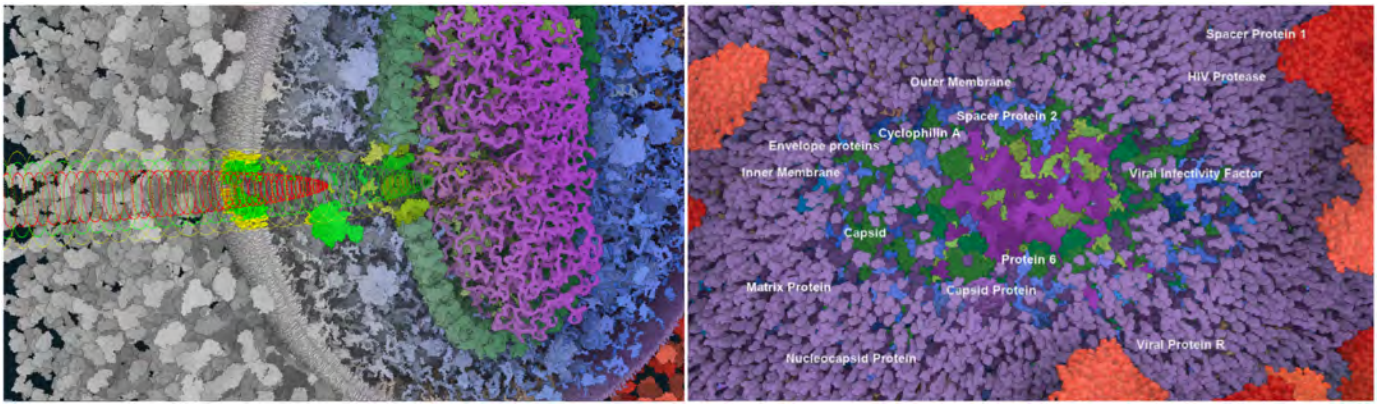


Fig. 1: Nanotilus entering the center of the HIV model. Several protein instances are sparsified to reveal the internal part of the model. Left: A grayscale model with highlighted proteins inside the Nanotilus hull is combined with the model's final depiction. The model is dissected by a cutting plane perpendicular to the Nanotilus hull. Right: Resulting view from the Nanotilus perspective.

Abstract—Immersive virtual reality environments are gaining popularity for studying and exploring crowded three-dimensional structures. When reaching very high structural densities, the natural depiction of the scene produces impenetrable clutter and requires visibility and occlusion management strategies for exploration and orientation. Strategies developed to address the crowdedness in desktop applications, however, inhibit the feeling of immersion. They result in nonimmersive, desktop-style outside-in viewing in virtual reality. This paper proposes *Nanotilus*—a new visibility and guidance approach for very dense environments that generates an *endoscopic* inside-out experience instead of outside-in viewing, preserving the immersive aspect of virtual reality. The approach consists of two novel, tightly coupled mechanisms that control scene sparsification simultaneously with camera path planning. The sparsification strategy is localized around the camera and is realized as a multi-scale, multi-shell, variety-preserving technique. When Nanotilus dives into the structures to capture internal details residing on multiple scales, it guides the camera using depth-based path planning. In addition to sparsification and path planning, we complete the tour generation with an animation controller, textual annotation, and text-to-visualization conversion. We demonstrate the generated guided tours on mesoscopic biological models – SARS-CoV-2 and HIV. We evaluate the Nanotilus experience with a baseline outside-in sparsification and navigational technique in a formal user study with 29 participants. While users can maintain a better overview using the outside-in sparsification, the study confirms our hypothesis that Nanotilus leads to stronger engagement and immersion.

Index Terms—VR immersive, Visibility management, Path planning, Storytelling, Visualization

1 Introduction

In the wake of the pandemic outbreak, the ultrastructure of biological entities, such as the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is no longer the exclusive knowledge of a small group of structural biologists. The broader population has become familiar with the viral architecture and its elementary building blocks. Today, the public has a better understanding of what a vaccine consists of, what different types exist, and how it contributes to acquiring immunity. Overall, public knowledge of molecular assemblies and the interest in the topic are much higher than in prepandemic times. Simultaneously, biologists now have better ways to depict biological structures, which can be specified using cellPack [1], ChimeraX [2], or Mesoscope [3] by applying a recently introduced three-dimensional (3D) rapid modeling approach [4]. These models of the *mesoscale* ultrastructure (i.e., the details of molecular assemblies) are essential for scientific publications and molecular dynamics simulations. Thus far, these scientifically relevant models are

- *R. Alharbi, O. Strnad, C. Bohak and I. Viola are with King Abdullah University of Science and Technology (KAUST), Saudi Arabia. E-mails: {ruwayda.alharbi | ondrej.strnad | ciril.bohak | ivan.viola }@kaust.edu.sa. R. Alharbi and O. Strnad are co-first authors.*
- *L. Luidolt, M. Waldner, D. Kouřil and E. Gröller are with TU Wien. E-mails: {laura | waldner | dvdkouril | groeller }@cg.tuwien.ac.at.*
- *T. Klein is with Nanographics. E-mail: tobias@nanographics.at*

Manuscript received Month day, 202X; revised Month day, 202X.

not used for dissemination to the broader public. Biologists neither have the training nor the time to create exciting scientific content directly from their models for a broad audience. Instead, illustrative (and often misleading) 3D models or 2D illustrations are created for public awareness. The pipeline for science dissemination to a broad audience is performed separately, generating substantial overhead and sometimes leading to unfortunate conceptual miscommunications.

However, science-generated molecular models of biological entities could be used in science dissemination directly if the dissemination process were automatized so that biologists would not require in-depth science communication training. Such dissemination would require an automatized system, in which the biologist would describe the model content with plain text, which would be used with the model to generate an exciting 3D animation or an interactive 3D exploration. Our vision is to establish such a technology.

The essential building elements would automatically guide a camera according to the biologist's text, and the described structures would be visually presented to the viewer. With molecular models, this becomes a nontrivial problem, as the scene is very crowded with densely packed molecular structures, which are isotropically distributed in all three spatial dimensions. An advanced visibility management strategy is necessary to *sparsify* the environment. Ideally, such a sparsification process preserves the scene's fundamental properties, among others, the notion of crowdedness, while securing the visibility of the selected structures. Our prior work has made it possible to automatically generate a guided tour for desktop viewing [5]. However, the perception of crowdedness has been traded for clear visibility of the discussed structures. In prior work, the viewer is placed outside the scene and peers into it. In this work, we investigate a new automatic guided-tour generation that preserves the notion of crowdedness and allows the audience to become deeply immersed into the space with nanoscale details.

A well-suited type of science communication employs an immersive environment with a large 360° dome stereo projection or virtual reality (VR) headsets. These display technologies convey complex 3D structural arrangements much better than standard 2D displays. They are very engaging means for science dissemination, further propelling knowledge transfer. When using a previously developed desktop-centric guided-tour generation [5] in such environments, the experience can be described as *watching a movie* in an immersive setting. We call this type of egocentric view an *outside-in* view, where the users view the entire scene in front of them.

The immersion did not encompass the actual space or model being communicated, which is the challenge we address in this paper. We propose *Nanotilus*, a guided-tour generator for immersive display environments to communicate multi-scale, crowded, scientifically accurate 3D models, representing the structure of complex biological entities. In addition, the generator preserves the immersive, overwhelming feeling of the environment so that, in VR, the viewer is surrounded by all structures that form the model. We call this type of egocentric view an *inside-out* view, where the user is placed in the middle of the scene, and the scene elements are located around and behind the user.

The Nanotilus guided-tour generation consists of several constituents, where two components represent technical novelties. Additional components complete the necessary functionality and are taken from prior research. Technically novel is the *sparsification* technique, which is tightly coupled with *journey planning*. These two parts are coupled with a text-to-itinerary conversion and labeling from prior research. Together, all these components constitute the Nanotilus guided-tour generator as a novel scientific contribution on a system level. The components have been designed to match the following set of requirements:

- R1 (Immersion): Preserve the immersion to create a feeling that one is part of the scene.
- R2 (Realism): Preserve the realism of the model by minimizing the sparsification effect on the scene.
- R3 (Variety): Maximize the variety of information by prioritizing the removal of redundant structures and avoid removing unique structures.
- R4 (Multi-scale): Convey the multi-scale hierarchical architecture of the biological entities.
- R5 (Smoothness): Preserve the smoothness of the visual experience by avoiding the sudden disappearance of the scene's elements.
- R6 (Engagement): Maximize the engagement that increases the audience's eagerness to learn.

2 Related Work

Nanotilus employs a novel path planning and visibility management technique to generate a guided tour in an immersive environment. Our work is related to several areas, where we discuss the most relevant previous studies in each of those.

Smart Visibility: The impediment to observing internal features due to occlusion from other objects affects many 3D data visualizations. Several methods have been proposed for occlusion management. Elmqvist and Tsigas [6] surveyed common approaches and identified five major design patterns. Viola and Gröller [7] compiled *smart visibility* techniques that either adjust optical attributes through cutaway, section, or ghosted views [8] or alter spatial layouts, using exploded views or deformations [9].

Removing parts of a 3D model may lower occlusion, but it can eliminate salient features or obscure the overall characteristics of the data. A trade-off is often determined based on the importance of the depicted features. Bruckner et al. [10] detected homogeneous regions in volumetric models and reduced their opacity to reveal high-frequency objects that they assume have higher importance. Viola and Gröller [11] determined visibility based on an importance function derived from the model features. Krüger et al. [12] applied a *Focus+Context* concept as often employed in visualizations. They modulated the transparency to combine ray-casted focus and context layers separately. The approach by Li et al. [13] automatically creates cutaway illustrations for surface meshes after an initial manual shape categorization step.

Several authors have formulated visibility management as an optimization problem. Sigg et al. [14] proposed a

Monte Carlo technique to automatically detect the best position and parameterization of a cut-away primitive. Further, Ament et al. [15] optimize light attenuation with a single scattering in direct volume rendering to selectively illuminate and uncover essential structures. Birkeland et al. [16] determined clipping regions by fitting an elastic membrane in a force field defined by the model features. Similarly, the method by Díaz et al. [17] permits the extrusion of segmented surfaces (e.g., bone structures) from the clipping plane.

Molecular models often feature a high density of information similarly as in volumetric data. Researchers most often deal with occlusion by removing some of the density (i.e., they *sparsify* the crowded model). Kouřil et al. [18] applied two sparsification strategies to navigate the hierarchy of a 3D molecular model. A limitation of their work is that when the user navigates deeper into the hierarchy, more of the model is removed, leading to a reduced understanding of the crowdedness. Le Muzic et al. [19] proposed a technique for authoring cutaway illustrations of mesoscopic biological models consisting of many elementary instances. Sparsification consists of clipping objects and visibility equalization. Visibility equalization comprises a series of visualization control bars overriding the clipping state of instances according to type. We consider our sparsification approach to be an *automatic visibility equalizer*, where the viewer does not have to control the visibility manually. Nanotilus combines both the clipping objects and visibility equalizer into a single entity and uses heuristics to guide the sparsification process.

A significant drawback of cutaway views is that they inherently eliminate portions of the data from a visualization. Occlusion management methods that alter spatial characteristics (e.g., exploded views or peel-away views) circumvent this problem. Li et al. [20] create interactive exploded views of an elaborate 3D model. The 3D model is structured in an explosion graph encoding the displacement of parts in relation to others. Birkeland et al. [21] described the automatic creation of view-dependent peel-away views for volumetric data. Sorger et al. [22] proposed a system for producing reusable animated transitions tailored to molecular datasets. Exploded views were used to examine the layering of hierarchical structures. Finally, Elmqvist [23] distorted the space to manage occlusion, where a spherical force field repels objects close to the 3D cursor. A follow-up user study [24] determined that the approach has high precision and is quite time-consuming. Elmqvist's approach is similar to Nanotilus in terms of employing an influence zone around a 3D position and modifying the characteristics of the objects in the zone. Our work additionally suggests several extensions. For example, to account for multi-scale scenarios, multiple levels of influence zones and different geometries of the zones (i.e., shells) surrounding the camera are used instead of a single-level spherical influence zone.

Virtual Camera Control: Christie and Olivier [25] comprehensively reviewed camera control in the context of computer graphics. In addition, Mindek et al. [26] introduced a data-sensitive navigation model to enhance medical visualization interaction. In the case of large molecular scenes, the interaction techniques must consider the multi-scale characteristics. Mackinlay et al. [27] determined camera movement speed as a percentage of the distance to the target, resulting in fast movement far from the object but slow and controlled movement close to it. Moreover, McCrae et al. [28] described a technique for navigating multi-scale datasets employing a cubemap as an image-based representation of the nearby environment. These studies address navigation in a multi-scale environment; however, they avoid collisions, which is inapplicable in molecular biology. In molecular biology the models are densely packed, and it is crucial to apply a visibility technique that creates a space for navigation. Trellet et al. [29] suggested a navigation strategy aimed at crowded molecular biology. They employed transparency and exploded views to deliver a clear view of the target objects.

Another significant component in controlling a virtual camera is its trajectory. Several studies have investigated path planning algorithms used for road maps [30], [31], [32], [33], [34], [35]. The main idea is to automatically precompute a probabilistic collision-free road map and use a search algorithm at runtime to determine the trajectories. Hsu et al. [33] used this approach for producing visually pleasing camera animations from volumetric data. Path planning is also used to address different problems in VR. An example is stereoscopic adjustments for group presentations in a cave automatic virtual environment (CAVE), as an alternative to the commonly used single-person head tracking [36]. Research in robotics is highly relevant to path planning. Depth cameras have emerged as appropriate sensors for robotic navigation in complex indoor environments [37]. For example, Galvane et al. [38] described techniques to follow targets in a dynamic environment with camera-equipped quadrotor drones. In addition, Nägeli et al. [39] allowed aerial cinematography movement using multiple drones in real-time with multiple on-screen subjects. In our work, we propose a path planning that creates a *weighted roadmap*. Instead of constructing a graph based on potential locations, we use the depth buffer to estimate these. Edges connecting locations are evaluated. Then, we use Dijkstra's algorithm to determine the optimal path.

Storytelling in Visualization: Animated visualizations can effectively present fascinating stories about scientific data. The survey by Chao et al. [40] focuses on storytelling literature in visualization and describes its essential elements. Ma et al. [41] presented examples of effective storytelling specifically with scientific data, using such systems as AniViz [42]. Furthermore, Wohlfart and Hauser [43] utilised interactive volume visualization for guided story creation. Thöny et al. [44] described the design and requirements for interactive storytelling within 3D geographic visualizations. Additionally, Lidal et al. [45] proposed a graphical approach to gathering and visualizing the reasoning process for generating geological sketches. Metamorphers by Sorger et al. [22] define animation through reusable storytelling templates for molecular models.

Our work builds on the *Molecumentary* method by Kouřil et al. [5]. This method comprises a framework for developing documentary-style content employing scientific visualization. The approach automatically generates fly-throughs of hierarchical molecular models such that leverages previously written expert explanations to supply an accompanying verbal commentary using text-to-speech technology.

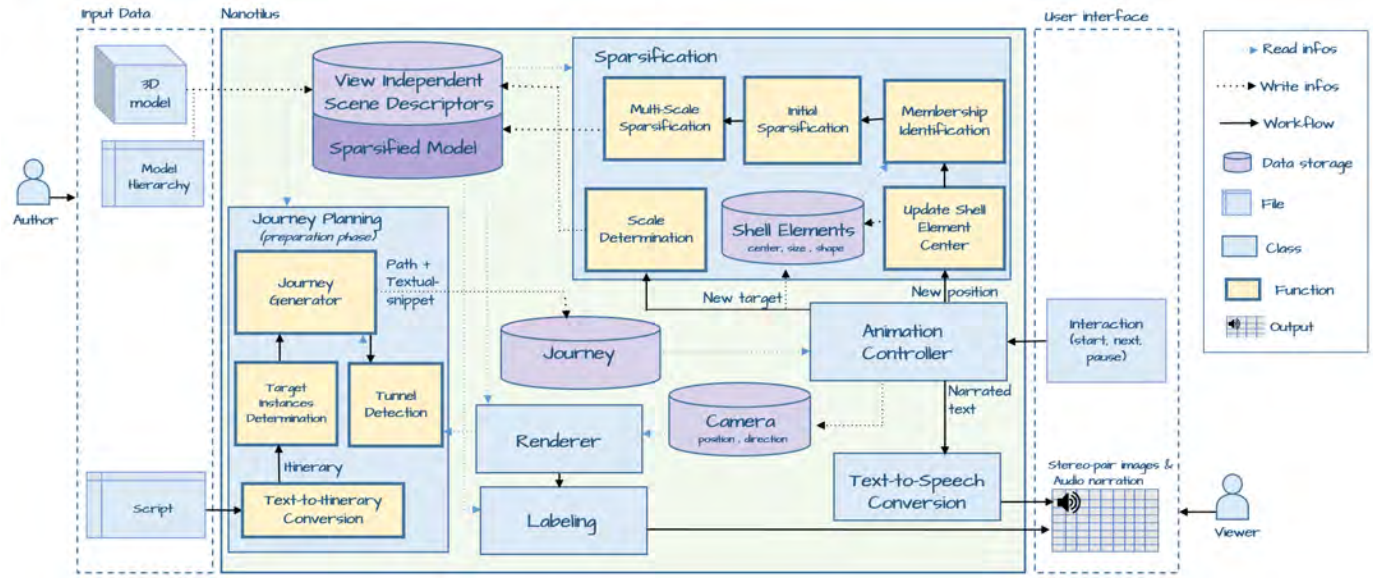


Fig. 2: High-level schematic overview of Nanotilus. The author of the guided provides the system with a 3D model, the hierarchy, and a script that contains the itinerary (the names of types that the tour should visit). Journey planning receives the script, extracts the itinerary, and builds the journey. Once the journey is created, it can control the tour by interacting with the animation controller, which provides information to the camera, sparsification, and text-to-speech conversion components based on input from the viewer and the journey.

A drawback of molecumentaries, which we address in our work, is the occlusion management that removes most of the scene when presenting the internal features.

Storytelling in Virtual Reality: Immersive and interactive VR visualizations offer significant benefits to both research and education. Slater et al. [46] have evaluated VR developments since the 1980s, when the concepts were first conceived. Keiriz et al. [47] used VR to provide a more immersive way to explore brain data. Boges et al. [48] presented an immersive system for interactively exploring and analyzing 3D cellular nanoscale models of the brain. Cremer and Kearney [49] described a method for creating complex traffic scenarios to examine human driving performance. Further, Ponder et al. [50] used interactive storytelling to create an immersive VR decision training system.

Immersive movies can also play a critical role in learning. For example, Parong and Mayer [51] compared instructional effectiveness between immersive VR and a desktop slideshow. Students evaluated VR lessons as more motivating and enjoyable compared to a PowerPoint slideshow. Dooley [52] pointed out that the freedom of users to change the view in VR generates additional challenges for the authoring of VR narratives. Zhang et al. [53] examined the influence of interactivity in an educational VR experience concerning immunology learning. Although interactivity is useful in improving content engagement, it does not apply to all actions. Thus, a balance should be maintained between user control and automation.

Several researchers have investigated the immersive effect of VR in data visualization. In addition, VR enables the user to gaze *into* the data, rather than *onto* the data. For example, Yang et al. [54] studied room-scale immersive networks, establishing the need for a trade-off between user engagement, physical and mental demand, and efficiency. Further, Kraus et al. [55] evaluated the effect of immersion by comparing immersive scatterplot visualizations. Sorger et al. [56] discussed the advantages of an outside overview and inside detailed perspectives in exploring medical data networks. Our work complements these findings. We visualize inherently spatial data and employ novel visibility management specifically for immersive storytelling.

3 Nanotilus - Technical Overview

The goal of our pipeline is to automatically generate engaging guided tours through nanoscale detail as a novel, attractive form of disseminative visualization for a broad audience, from the bench *directly* to outreach. Soon, many 3D models of biological structures at the mesoscale can be produced by scientists for research and publication purposes. With minimal additional effort, such an outcome should be repurposed for public outreach. The minimal effort requires to formulate a plain-text script that will define the guided tour content through the model. The script has to use consistent terminology with names associated to the model elements. The generated outcome can be immediately deployed in immersive dome stereo projections or guided VR tours.

In this section, we clarify the terminology of the individual components involved in a guided-tour generation (Figure 2). Nanotilus expects input of two types of users: **authors** and **viewers**. The **author** is the writer or director of the story who provides the system with a **3D model** together with a **script** (i.e., textual story) which are then used to generate the immersive guided tour. The **viewer** is immersed in VR to explore structural details described in the story. The role of the viewer is similar to an audience in a cinema or theatre.

A guided tour is a sequence of **stereo-pair images** coupled with an **audio narration** from **a text-to-speech**

conversion. The stereo-pair is **rendered** based on **camera** settings, **sparsification** settings, a **model**, and the associated **labels**. An **animation controller** provides information to the camera and sparsification components based on input from the **viewer** and the **journey**. A journey is the **path** between several points of interest with associated textual narration. These points of interest are selected structural instances of the model that are highlighted and discussed in the narration of the guided tour.

The input **3D model** represents the molecular biological structure of mesoscale organisms, such as viruses, at the atomistic resolution. The model may consist of several thousands of **instances** each formed by millions of atoms densely packed to communicate the crowded situation in real organisms. It does not explicitly contain the water molecules that sometimes represent 50% to 90% of the volume like in the blood plasma. Including water molecules, the scene would be completely packed. Without water molecules there are **void spaces** scattered inside the model. Every model is coupled with its hierarchy (see Figures 11 and 12 in the supplementary material, Appendix C). **Leaf instances** at the bottom of the hierarchy comprise the geometries of individual molecules. The leaf instances assemble into more complex geometries of **intermediate instances**. The **ancestors** of an instance are all the intermediate instances on the path from that instance up to the root. Each instance is of a certain molecular **type** that has a particular name. A **script** by the **author** contains the names of the described types. We extract the sequence of types from the script and build an **itinerary**.

Journey planning takes the sequence of the types in the itinerary and transforms it into a **path** (i.e., a sequence of 3D positions corresponding to the **target instances** of the itinerary types). The instances are selected based on their spatial position so that the camera can easily navigate between them. The path of points with a text snippet from the script associated with each **itinerary** element form a **journey**. If only these points are visited, the resulting camera path would likely not be as engaging. For example, after visiting one target instance along the path, a natural path for the camera would be to navigate through the model's void spaces and deviate slightly from the intended direction so that the camera does not need to travel directly through all instances obstructing the way.

Navigation through void spaces minimizes the number of instances to sparsify, which subsequently preserves the model *realism* (R2). We can observe through anecdotal evidence that passing through void spaces of the scene increases the participants' immersion in the virtual environment. If everything is cut in front of them, the coexistence with the scene will not be as strong. Therefore, we argue that navigation through void spaces increases the *immersive* experience (R1). In addition, we have observed that many participants were much more engaged in the virtual environment if they were immersed in it. As a metaphorical example, using the natural void spaces to cross a forest would be a more engaging adventure than using the main road. So, navigation through void spaces increases the *engagement* (R6). Therefore, journey planning additionally searches for the model's void spaces and considers these void spaces for traversal during the journey. The path is subdivided into line segments, ensuring that the journey takes a specific, more detailed route through the model between instances and uses the available void space. Therefore, some points along the path are associated with a target instance and text narration, and some points are included to facilitate natural corridors in the model for the camera path.

At the journey planning stage, the path is just a sequence of points. The path is not yet the specific smooth **trajectory** the camera will be moving along. A smooth camera trajectory is generated by fitting a higher-order curve through the path's points. The absence of the water molecules assures to a large degree the existence of natural void spaces that are then used by the camera to pass through. If the scene in a particular region is too dense for a given magnification level, an artificial tunnel will be generated by **sparsification**. We apply **sparsification** to the model around the camera so that interesting and essential instances remain visible, while superfluous instances of a particular type become temporarily **hidden**. Sparsification is localized around the camera, which minimizes the sparsification effect on the model, preserving the model *realism* (R2) and scene *immersion* (R1). However, filtering the instances and prioritizing the removal of redundant structures, increase the *variety* (R3) of information displayed in the scene. In the **sparsified model** the camera is still enclosed by structures to preserve the immersive experience, some local void space always exists around the viewer. Sparsification is staged through multiple concentric **shell elements** surrounding the camera to avoid instances abruptly disappearing in front of the viewer and to preserve the *smoothness* (R5) of the presented information. Sparsification increases from the outermost shell toward the center. The innermost shell hides all instances that would otherwise hit the camera. Long genetic molecules and other fiber structures are exempt from hiding. They are pushed away from the camera trajectory to reinforce the sense of immersion further. Pushing the important structures away also minimizes the number of instances to sparsify, which preserves the model's *realism* (R2) and scene *immersion* (R1). It evokes the experience of displacing objects to cross a crowded area, which increases *engagement* (R6). During a journey, a user may encounter instances at different scales that represent intermediate or leaf instances in the hierarchy. To convey *multiple scales* (R4) to the user, **multi-scale sparsification** is required.

4 Journey Planning

An immersive guided tour requires a path that visits exciting places within the 3D model. Moreover, the path should run between the instances in the model so that the camera traverses the model's natural void spaces and unnecessary sparsification is reduced. The journey planning starts by identifying a set of target instances that journey should visit. This process generates a path that connects selected instances, while traversing through the model's void spaces. The journey planning is completed in a preprocessing stage and the outcome is used during runtime with scene traversal. The detail explanation of each step of journey planning can be found in the following subsection.

4.1 Target Instances Determination

The first step in journey planning is to determine representative instances from the itinerary. The itinerary is extracted from the input script. The script comprises individual sentences, and type names are extracted from these sentences. Every type in the itinerary has an associated textual snippet $text(T_i)$ in the narration using text-to-speech synthesis. This process is described with further detail in our previous work, where it is called the text-to-vis method [5]. The guided tour *itinerary* is a sequence of types $\langle T_1, ..., T_n \rangle$ used to determine the sequence of target instances whose 3D positions define the initial path. The initial position P_0 of the camera is outside of the model, so the whole model is inside the camera frustum. For each type T_i, we select one particular instance S_i among the many by taking the one closest to the camera position P_{i-1}. From the camera position P_i, which is set to view the instance S_i, the closest instance S_{i+1} of type T_{i+1} is found. Then, camera position P_{i+1} is determined, and so on, until we find a corresponding instance S_i for every type T_i of the itinerary. This leads to sequences of target instances $S = \langle S_1, ..., S_n \rangle$ and camera positions $P = \langle P_0, ..., P_n \rangle$ that form the initial path. The initial path is further refined by analyzing the possible ways through the crowded model, leading to a path with two-level indexing, where all points between P_i and P_{i+1} are indexed with $\langle P_{i,0}..P_{i,m} \rangle$. Level-one point P_i corresponds to level-two point $P_{i,0}$, and $P_{i,m}$ is the last level-two point that connects to P_{i+1}.

4.2 Tunnel Detection

Although the 3D model is densely packed, the camera can typically traverse through several existing holes. We employ a vision-based approach to determine the path through the holes in the model from target instance S_i to target instance S_{i+1}. We render the scene to obtain a depth image from the camera position P_i, with the *look-at* vector pointing to the target instance S_{i+1}. Then we analyze the depth buffer and identify **tunnels** that end farthest from the camera, and their opening is sufficiently large on the rendered image. Next, we move the camera towards the tunnel, adjust the camera look-at vector to the target instance S_{i+1} again and repeat the entire process, until the target instance S_{i+1} has become sufficiently visible.

The approach analyses the depth and ID buffers. Both are attachments of a frame buffer object. Therefore, every time the scene is rendered, both buffers are updated immediately within a single draw call. For each pixel, the depth buffer contains the distance to the instance closest to the camera. The ID buffer contains the identifier of the rendered instance or a default value. While the background is not considered, the tunnels are detected using GPU-based connected-component labeling called *Label-equivalence* [57], [58]. The algorithm detects the regions where the voids are projected in the depth map. We refer to these regions as *void depth projection (VDP)*. The existence of *VDPs* is assured as these are 2D regions detected in the depth map, which is updated in every frame based on the camera position and rotation. As the tunnels or *VDPs* lead to instances, their existence is assured if at least one instance is visible in the scene. In this case, the tunnel leads to that instance.

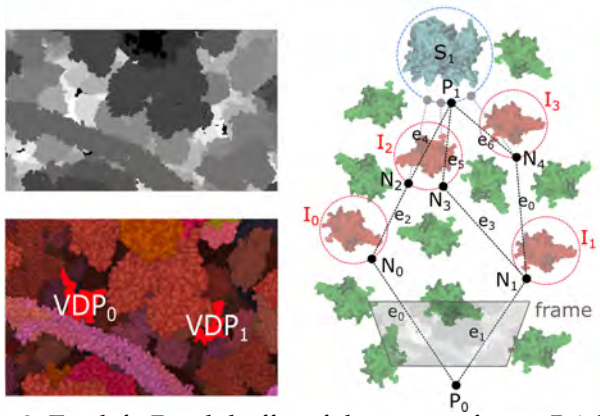


Fig. 3: Top left: Depth buffer of the current frame. Brighter colors indicate instances farther from the camera. Bottom left: The two farthest void depth projections, VDP_0 and VDP_1, are used for detecting I_0 and I_1 and computing N_0 and N_1. Right: Illustration of depth-based journey planning building the graph G. Intermediate points N_j represent nodes in the graph that are computed.

Therefore, as the viewer is enclosed in the model, tunnels, as we defined them, are presented during the guided tour.

The *Label-equivalence* algorithm identifies which pixels in the depth buffer belong to the same connected cluster or *VDP*. The algorithm relies on three GPU functions: initialization, scan, and analysis. The initialization function uses the pixel's linear address as a unique label for that pixel. The scan function compares each pixel's depth value with its eight neighbors to determine whether it is connected to one of them (i.e., has a similar depth value). If so, it updates the pixel label with the smallest label. Finally, the analysis function checks the label of each pixel to determine whether it refers in turn to an even smaller label. This process continues until the root of a chain of labels is found. The root label becomes the label of that pixel. The scan and analysis functions are repeated until label stability is achieved (i.e., labels are no longer updated in a pass). We also collected size and 3D position information about the detected *VDPs*. The analysis of the depth buffer typically provides several good tunnel candidates.

4.3 Journey Generator

Journey planning computes and evaluates several candidate paths for the camera to move from one 3D position to the next. Once we reach the target instance, one of the candidate paths is selected and used by the animation controller.

Candidate paths, are stored in a directed acyclic graph (DAG) to allow efficient searching through them. Every path is given as a poly-line connecting 3D points. An edge in the DAG represents a segment of the poly-line. Both endpoints of a segment have corresponding nodes in the graph. Edges are weighted, which allows us to search for an optimal path. For example, edges are assigned a higher weight if they correspond to broader and longer tunnels.

Journey planning processes instances S_i sequentially in steps. In every step, a DAG G containing alternative paths between the camera position P_{i-1} and a 3D point close to instance S_i is created. Intermediate nodes and edges are inserted into G while analyzing the depth buffers in recursive

calls. An example of graph G is depicted in Figure 3. A detailed exemplary description for the first instance S_1 of the sequence follows. The remaining instances from S are computed analogously.

In the beginning, G is created and initialized with a single root node representing the 3D camera position P_0. The scene is rendered from the camera position P_0 pointing to the instance S_0. By rendering the scene, a *frame* is obtained. Then, the depth buffer of the frame is analyzed and *VDPs* are found as described in Section 4.2. In the next step, we evaluate each VDP_j. The $weight(VDP_j)$ is calculated as the sum of depths of all pixels belonging to VDP_j. This calculation assigns high weight to *VDPs* corresponding to larger and longer tunnels. For each *VDP*, a set of instances $I = \{I_1, ..I_j..I_o\}$ from the ID buffer is detected, based on the pixel coordinates of the center of the void depth projection. Point N_j is calculated as the intersection of the line from camera P_0 to instance I_j and the bounding sphere of I_j. The distance of the point N_j from the camera P_0 can be optionally limited by a $maxEdgeLength$ to prevent moving too far through the model. A new $node(N_j)$ is inserted into graph G, referencing the intersection point N_j. A new edge e_j connecting $node(P_0)$ with $node(N_j)$ is also inserted. The weight of the edge e_j is set to $weight(VDP_j)$.

In the following step, the camera is placed at positions of points N_j, oriented toward S_1. If instance I_j is a leaf instance and not an intermediate instance, I_j is hidden because it will likely block the camera view in the next iteration of the algorithm. The scene is rendered again, and the process is repeated until the target instance S_1 (or at least one of its leaf instances meaning S_1 is visible partially) is detected in the ID buffer.

In the final path-refinement step, a set of leaf nodes L from G end up very near S_1, and their average position L_{avg} is computed. The point P_1 is set to the position of the closest point on the bounding sphere of S_1 to L_{avg}, and edges previously ending in L are retargeted to P_1. In addition a reference to S_1 for P_1 is stored. Once G with multiple path-segments is computed, the highest weighted path-segment connecting P_0 with P_1 from G is determined. Points P_0 and P_1 are taken to define the level-two points $P_{0,0} \dots P_{0,m}$ of the refined path. As the edges in the graph have numerical weights, we used Dijkstra's algorithm to obtain an optimal path. Finally, the temporary DAG G structure is cleared for the next path-processing step. The remaining instances S_i, with $i \in [2, \dots, n]$, are processed analogously. The pseudocode of Algorithm 1 can be found in the supplementary material, Appendix A.

5 SPARSIFICATION

When the viewer enters an immersive world of a packed mesoscale molecular biology, the scene density evokes an experience like cutting through a jungle, even if there are tunnels to pass through. Therefore we designed a local, camera-centric sparsification procedure, reminiscent of the Nautilus submarine from science fiction because of the involved geometries and the user experience. The sparsification is controlled by three nested and concentric shells surrounding the camera. These three shells approximate the change in visibility function that varies from 0 to 1, where 0 means nothing is visible, and 1 means everything is visible. We use three shells for illustration, but this can be generalized to an arbitrary number. Based on the overlap of the model instances with the shells, the visibility of all but a few selected instances is modulated, or the instances are entirely hidden. Nanotilus generates endoscopic views where the scene elements surround the viewer. The shells are designed as an ellipsoidal shape that is anisotropic along the tangent of the journey path , as a result, the sparsification of the model is stronger in the forward direction and weaker orthogonal to the tangent, which provides the viewer with necessary space to observe the environment. In addition, the sparsification reduces structural occlusion and provides the user with endoscopic views that convey the crowdedness in the model, while offering an unobstructed journey through the model.

Nanotilus performs a multi-scale sparsification to cope with the multi-scale hierarchy of the 3D mesoscale model. It detects the appropriate scale of the target instance and sparsifies the model accordingly. For example, if the target instance in the HIV model is a particular virion as an intermediate instance, the granularity of sparsification should affect other instances on the same scale (i.e., other virions). Sparsification, in this case, should not just hide some leaf instances belonging to neighboring virions. Instead, entire intermediate instances should be hidden. If a HIV capsid protein is a target instance, a natural behavior would be to sparsify neighboring virions on the intermediate instance scale. However, in the case of a target instance's parent, lower-level instances should be sparsified to open the virus and reveal the capsid protein. Thus the sparsification granularity is modulated based on the hierarchical vicinity of an instance to the target instance.

The sparsification process is performed during the guided tour execution. The camera motion frequently invokes the sparsification which is scale-aware (i.e., based on the target instance characteristics). In the Nanotilus framework, this information is passed on to the sparsification component by the animation controller. When a new target instance is selected, the sparsification component defines the scale on which the occluding instances should be sparsified. This has been illustrated in the above examples with HIV as the target instance on the intermediate scale and the capsid protein as the target instance on the leaf scale. Spatially-large target instances require stronger sparsification than spatially small instances. Therefore, Nanotilus changes the sparsification region based on the target instance. It gradually increases or decreases the dimensions of the shell geometries to match the bounding-sphere size of the target instance. The scale is determined in Algorithm 2 in the supplementary material, Appendix A as described in Section 5.2.

Changing the camera position leads to a sparsification update (i.e., deciding what is shown or hidden) in three steps. In the first step, as the camera serves as a pivot point for the shells, these are transformed to a new position with respect to it. In the second step, instances are associated through overlap with one of the shells. In the third step, multi-scale sparsification is performed. In this step, all instances are assigned a weight representing their priority of being hidden, based on their abundance and distance from

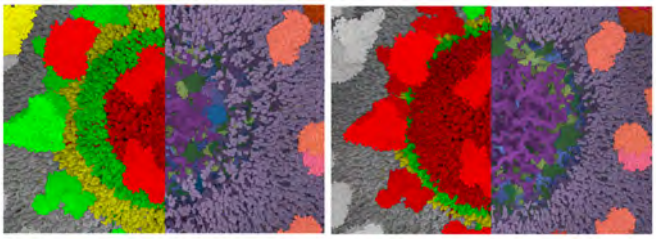


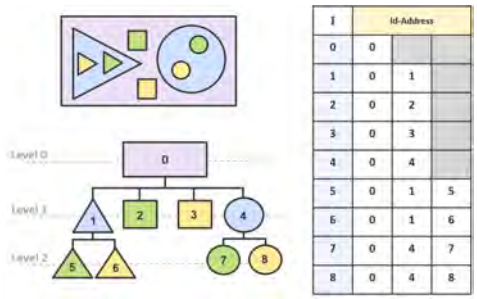
Fig. 4: Illustration of the sparsification process with different sizes of the innermost shell (the right image has a larger innermost shell). The right sides of both images display the resulting sparsification whereas the left sides illustrate the shell membership in colors, the red color represents the instances that belong to most inner shell, the green and yellow colors represent outer shells' instances. A larger inner shell produces a sharper transition between sparsified and non-sparsified regions.

the camera. The visibility decision considers the multi-scale scene hierarchy and results in a sparsified model. These sparsification steps are described in detail in Sections 5.1, 5.3, and 5.4 (see also Figure 2).

5.1 Shell Elements

The three shells define concentric bounding geometries around the camera and are responsible for preserving the viewer's comfort zone. Visible collisions of the camera with the innermost shell's instances should be avoided. In contrast, structures farther away from the camera should be perceivable, which requires sparsification in the middle and outermost shells at varying degrees. This method somewhat invokes the experience of using a vehicle lights at night. Nanotilus sparsification realizes gradual visibility management through three nested shells surrounding the camera to avoid the sudden disappearance of instances in front of the viewer. The outermost shell performs the initial stage of sparsification. The middle shell performs a further sparsification, and the innermost shell hides all instances obstructing the camera movement. Each shell has a different *visibility percentage* (i.e., the visibility percentage of the innermost shell is set to 0.0, which means all instances are entirely hidden to avoid collision with the camera). The visibility percentage increases with each larger shell. In the middle shell, only the scarcest and vital instances remain visible. In the outermost shell more instances populate the shell space. Outside the outermost shell visibility reaches 1.0, meaning that all instances remain fully visible and are shown. Altogether, the number of nested shells, the gradual visibility change between them, and the animated opacity transition between visible and hidden instances result in a smoothly perceived sparsification.

In principle, the shells can take on any 3D geometric shape where a collision query can be solved. We tested several shell geometries, such as nested ellipsoids, nested cubes, and nested egg shapes. The latter geometries are loosely similar to a perspective-viewing frustum but with a limited sparsification zone. We concluded that nested ellipsoids provide the most immersive experience, especially when opening compartmental boundary structures. We used the standard ellipsoid equation [59] to mathematically define the shell geometry of Nanotilus, where (p_x, p_y, p_z) is



I	Id-Address		
0	0		
1	0	1	
2	0	2	
3	0	3	
4	0	4	
5	0	1	5
6	0	1	6
7	0	4	7
8	0	4	8

Fig. 5: Left-top: Illustration of a scene containing a multi-scale model. Left-bottom: Hierarchy of the multi-scale model. Each tree node is associated with a unique ID. Right: ID-addresses of instances.

a point on the surface, (e_x, e_y, e_z) is the center point of the ellipsoid, and a, b, and c are the scaling parameters for each coordinate axis.

$$\frac{(p_x - e_x)^2}{a^2} + \frac{(p_y - e_y)^2}{b^2} + \frac{(p_z - e_z)^2}{c^2} = 1 \qquad (1)$$

The shell size specifies the volume of the sparsification region. Nanotilus updates the shell size based on the bounding sphere of the target instance when a new target is selected. Nanotilus interpolates between the current shell size and a newly estimated size based on the bounding sphere of the new target instance during the path traversal. To compute the final ellipsoid size from the target bounding sphere, we assigned its radius to the smallest parameter of the inner-most ellipsoid and updated other ellipsoid parameters to preserve the original ratio. Accordingly, we update the parameters of other two outer ellipsoids to preserve the scale ratio between them. Another possibility is to select the innermost shell with an overproportioned size, as it is the most important one that represents the comfort zone. The illustration in Figure 4 depicts the results of two different inner shell sizes. A larger inner shell produces a faster sparsification leading to a sharper transition between sparsified and non-sparsified regions of the model. The middle and outermost shells in this comparison have the same size.

5.2 Scale Determination

The proposed multi-scale sparsification scheme is unequal in terms of how the instances are sparsified. In the framework, the instances are hierarchically organized in a tree, similar to the commonly known scene graph structure. When a particular instance is close in the hierarchy to the target instance (e.g., example, being its parent or grandparent), sparsification is performed on a lower granularity level (i.e., leaf instance level). When the target instance is more distant from the instance, it will be sparsified with higher granularity. The children of a target instance are not sparsified as they constitute the target instance under inspection. To automate multi-scale sparsification, we define the following instance addressing scheme. Each instance is assigned a unique **ID** and a hierarchically composed **ID-address**. We use the ID-address of the *target instance* to determine the scale of sparsification for a particular instance. The similarity between the instance ID-address to be sparsified and the target instance ID-address determines the sparsification scale.

We explain the concept with a simple example in Figure 5. A schematic multi-scale model is depicted at the top left. Below is the corresponding hierarchical tree, where each level in the tree represents a distinct scale. The instance's depth is the number of edges from that instance up to the tree's root. The address definitions for each instance I_i based on the IDs of its own, its parents, and its ancestors are on the right. The sequence of IDs along the path from the root to an instance defines the **ID-address** of that instance inside the 3D model. For example, the ID-address of the green circle is $\{0.4.7\}$. This address indicates us that to observe instance I_7, we must sparsify instance I_0 first, followed by instance I_4. Based on the above address, there is no need to sparsify the blue triangle. If we decide to hide it, we must consider the entire subtree below the blue triangle to be a single element and hide it all at once. We can realize this if we compare the ID-address of the target instance (i.e., $\{0.4.7\}$) with the ID-addresses of the blue, green, and yellow triangles, which are $\{0.1\}$, $\{0.1.5\}$, and $\{0.1.6\}$, respectively. The **scale-based-ID** of an instance is the first ID in its ID-address that does not match the corresponding ID in the target ID-address. Thus, for the blue, green, and yellow triangles, the scale-based-ID is 1. Therefore, if sparsification is used, it hides Level 1 instances (where the root is Level 0). The final visibility decision concerning every instance relies on its scale-based-ID. In the example, the visibility of instance I_1 overrides the visibility values of the green and yellow triangles. Therefore, if we decide to remove the blue triangle, we also eliminate all of its children (i.e., the yellow and green triangles).

The sparsification has to consider the *previous target instance* along the journey, too. For example, in Figure 5, let's assume the previous target instance is the green circle, and we select the green rectangle as the new target instance. Then, the scale-based-ID of the yellow, green, and blue circle is 4. When we are inside the model closely observing target I_7, and we switch to the new target instance, the whole I_4 subtree disappears as a result of multi-scale sparsification. More natural behavior is to sparsify the previous target on a lower granularity level. Therefore, we calculate the scale-based-ID of an instance by comparing it with the current and the previous target ID-addresses. This procedure is called *scale determination* and is listed in Algorithm 2 in the supplementary material, Appendix A. Because the model is static, the ID-addresses of model's instances are computed once after the model and hierarchy are loaded. Figure 6 shows an example of using the scale-determination.

5.3 Membership Identification

Nanotilus generates endoscopic views where the scene elements surround the viewer. It only sparsifies instances lying at least partially inside the shells. The mathematical description of the shell geometry is used to determine whether the bounding sphere of an instance intersects one of the shells. All shells are tested for intersection. If an intersection exists, the computation ends, and the instance is considered a member of the smallest shell it intersects. If the bounding sphere of the instance does not intersect with any shell, it has membership to the region outside the outermost shell. We compile additional information, such as the number of visible and invisible instances in each shell

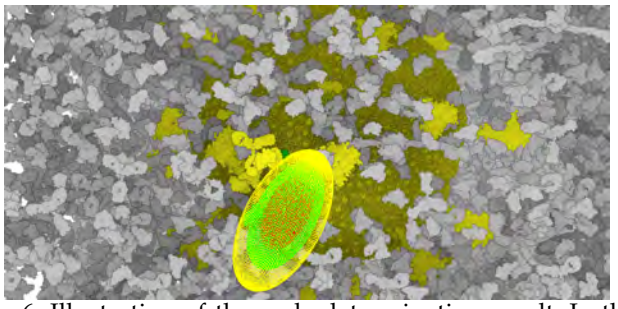


Fig. 6: Illustration of the scale-determination result. In this example, the target is one of the plasma proteins, thus, sparsification is on the virus scale. The whole virion has the yellow membership which means it belongs to the most outer shell because this virion partially intersects with that shell.

and their distances from the camera, with the membership information during one compute shader call. Furthermore, we determine the number of visible instances per type, which is helpful for evaluating the importance of the types.

5.4 Sparsification Update

Sparsification is realized in two consecutive phases. The first phase does not consider the scale, where some instances that intersect the shells are selected for sparsification. The second phase considers the scale, where the sparsification decision from the previous step may be overridden by the sparsification value of one of the instance's ancestors in the model hierarchy.

Phase 1: Initial Sparsification: After membership identification, we know which instances are inside the shells. The shells are associated with the visibility percentage determining the limit of how many instances inside the respective shell can remain visible. This initial sparsification selects instances that should be hidden and updates their visibility values accordingly.

Every instance that is located outside the shells should not be sparsified. If the instance was hidden previously due to a prior intersection with the shells, it should now be visible again. Otherwise, a previously hidden instance remains hidden to maintain visual coherence as long as it is inside one of the shells. On the other hand, if the instance is visible and inside one of the shells, the algorithm checks every instance inside the shell for each shell to determine whether the desired number of invisible instances inside the shell has already been reached. For a shell element that has N instances and visibility percentage p, this *threshold* is computed as $N - pN$. If the number of invisible instances is still not achieved, each visible instance is a *candidate* to be hidden. A weight is associated with each candidate instance that represents the instance priority to be hidden. Abundant instances and those located closer to the camera are assigned a higher priority of being hidden, while important instances are assigned a higher priority to remain visible. The weight is affected by two values: the distance from the instance to the camera and the importance of the type, computed as the complement of the ratio of instances of that type that remain visible inside the sparsification region. This is depicted in Figure 7, where the importance of the red

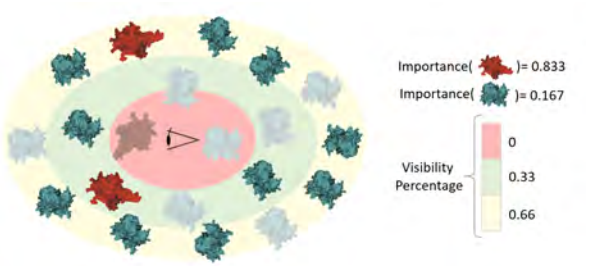


Fig. 7: Illustration of the sparsification process. The instances are sparsified based on the importance of the type (cyan or red) and membership to a shell. Low saturation colors represent hidden instances. All instances in the innermost shell have been removed regardless of their importance to avoid the block view. Some abundant instances have been removed from the second and third shells.

molecule is calculated as $[1 - (2/(2 + 10))]$. The weight is calculated as follows:

$$w = ||importance||_{[0,1]} \cdot ||dist||_{[0,1]}, \quad (2)$$

where $||importance||$ represents the importance of the type, normalized to the range $[0, 1]$. Zero corresponds to the lowest importance, and one corresponds to the highest importance. Moreover, $||dist||$ represents distance to camera, also normalized to the range $[0, 1]$, where 0 is the closest, and 1 is the farthest distance inside the sparsification region. A smaller instance weight w indicates a stronger candidate for being hidden. The algorithm sorts the candidate instances based on weight. Finally, the instances with the smallest weight are hidden until the threshold is achieved for a given shell. The pseudocode of *Initial Sparsification* can be found in Algorithm 3, in the supplementary material, Appendix A.

The leaf instances are the basic building blocks of intermediate instances. Thus, selecting a leaf instance means its ancestors are also indirectly selected. Algorithm 3 applies only to the leaf instances, and once an instance is selected, the instance and some of its ancestors are hidden. The algorithm propagates the hidden status from the leaf instance through the ancestors until it reaches the ancestor that has an ID which matches the leaf instance's scale-based-ID.

Phase 2: Multi-scale Sparsification: To convey the model's multiple scales to the user, multi-scale sparsification should be applied. In the previous step, specific instances have been selected to be sparsified. However, based on the scale-based-ID, that sparsification may be overridden by the value of one of that instance's ancestors. The scale-based-ID is given by the scale determination algorithm described in Section 5.2. In the Multi-scale Sparsification stage, the Nanotilus uses the scale-based-ID to determine the final visibility of every leaf instance, depending on the hidden/unhidden status of the ancestor on the level of the current scale value. For every leaf instance, we fetch the sparsification status of the instance that has an ID corresponding to the scale-based-ID and use it to update the sparsification status of that leaf instance. The *Multiscale Sparsification* is formally described in Algorithm 3, in the supplementary material, Appendix A.

Some instances in the model are so valuable that they should remain unhidden for the entire journey. For example, nonlocal genetic molecules, such as RNA, occupy a large space. However, their strand-like structure is sparse enough; therefore, these nonlocal structures are never hidden. Instead, they are pushed away from the innermost shell. The displacement direction is computed based on the strand's control points positions and the shell's center position. The pushed control points return to their original positions after they leave the sparsification region.

Instance Fading Effect: When instances become suddenly hidden or unhidden in the sparsification phase, it creates a visual popping artifact, which is inconvenient during the guided tour experience. Viewers must involuntarily redirect their gaze toward the visual popping position, which disturbs the experience. Therefore, instead of suddenly hiding or showing instances, we introduce the fading effect of the instances that change sparsification status. In other words, once the status of an instance changes from hidden to unhidden, or vise versa, as a result of the sparsification stage, the transparency of that instance is gradually increased/decreased with every rendering call until it becomes fully transparent/opaque. With that, the user perceives sparsification as a smooth process rather than a discrete event.

The fading effect is achieved by the rendering of the scene in two independent passes. Only opaque instances are rendered in the first pass, whereas transparent instances with the corresponding alpha transparency are drawn in the second pass. The transparency change is animated over several frames. This change is performed as an off-screen rendering into two framebuffers objects (FBOs) which are composited using alpha blending.

6 Remaining Elements of Nanotilus

In journey planning in Section 4, Nanotilus computes a path connecting target instances, which is primarily collision-free. The animation controller interpolates the path into a high-order curve and positions the camera along that curve based on the journey's progress. Every time the camera position is updated, sparsification is also updated. The shell elements are oriented so that their forward direction are aligned to the tangent on the curve, which results in the perception that the user is inside the *Nautilus* submarine. As our shell element is ellipsoid, the sparsification of the model is stronger in the forward direction and weaker orthogonal to it.

Furthermore, the animation controller is responsible for triggering text-to-speech events. As the path connects $node(P_i)$, the sequence of instances S can be obtained as $S = \langle P_1.ref, ..., P_n.ref \rangle$. From journey planning, every type T_i is associated with a text snippet $text(T_i)$.

The animation controller synchronizes the camera with the text-to-speech synthesizer. Before the next step along the journey, it fetches the target instance S_i. Then, the camera traverses to target instance S_i, and the narration snippet $text(T_i)$ is played in the form of audio. After both the camera traversal and narration are finished, the user can freely explore the current part of the model. Once the exploration is finished, the animation controller continues to target instance S_{i+1}.

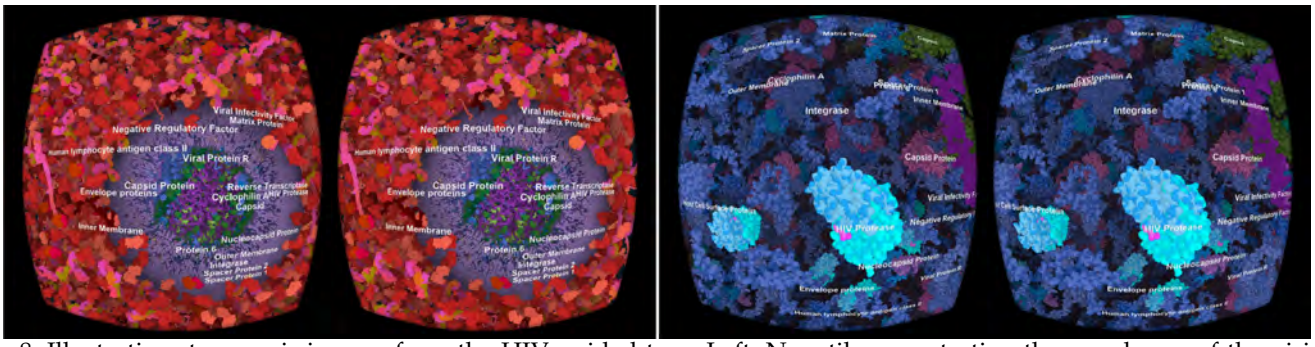


Fig. 8: Illustrative stereo-pair images from the HIV guided tour. Left: Nanotilus penetrating the membrane of the virion. Right: Nanotilus focused on an instance inside the virion.

During the guided tour, the structural information annotating the model is augmented with audio narration and labels. Labeling provides the user with additional descriptive information about the model. Moreover, labels help the user associate the information from the narration to the particular instances. We integrated the labeling approach by Kouřil et al. [18], [60], which identifies the types of instances currently visible and places labels for the representative instances for all types in the current viewpoint. However, restricting the labeling process to only once the camera rotation and movement is completely stopped is infeasible for the VR environment. If the user observes the model through a VR head-mounted display (HMD), the camera orientation and position are tracked and continuously updated by the HMD. The camera never stops; there is slight movement at all times. Therefore, we extended the original method to label the model several times, transiting from one instance to another. In other words, if the camera navigates from structure S_i to structure S_{i+1}, the arclength of the path is calculated. As the camera progresses along the path, after it reaches 25%, 50%, 75%, and 95% of its arc length, labeling is performed based on the current view direction. This labeling provides the viewers an actual description of the model even if they rotate their heads 180° along the journey.

7 Technical Implementation

We implemented Nanotilus using the Marion framework [61], employing an impostor rendering pipeline based on the work by Le Muzic et al. [62]. The entire framework is implemented in C++, OpenGL, and GLSL and relies on the Qt library. Nanotilus is implemented using GLSL compute shaders with only small data transfer between the CPU and GPU. As the target environment for the presentation of the resulting guided tours, we used the HTC Vive Cosmos. Therefore, we implemented OpenVR support in the framework.

The target model typically consists of several thousands of instances (all formed by millions of atoms) with a hierarchy that defines the logical structure of the model. The illustrations of both hierarchies of the HIV and SARS-CoV-2 models are presented in Figures 11 and 12 in the supplementary material, Appendix C. The HIV model was generated by Scripps Research [1], [63]. The SARS-CoV-2 model was created using our recently introduced modeling approach [4]. The virion model was enclosed in the model of blood plasma to create conditions similar to the HIV model.

We automatically generated guided tours for these two molecular models, HIV and SARS-CoV-2, in blood plasma, with comparable numbers of molecules (18,515 for HIV and 14,583 for SARS-CoV-2) and lipids (approx. 200,000 in HIV, 140,000 in SARS-CoV-2). Both models have RNA strands. We authored a script for each model (340 words with 11 mentioned types for HIV and 354 words with 10 mentioned types for SARS-CoV-2). The sample screenshots from the HIV tour are presented in Figure 8 and Figure 1.

After the performance optimizations described in the supplementary material, Appendix B, we set the resolution to 1064×1256 pixels per eye. The application ran on an Intel Xeon Gold 6242, GeForce 3090, Windows 10 Education, Qt 5.14.2 with 25 FPS per eye, on average. While the FPS recommendations for VR are typically higher, the framerate was sufficient for conducting the user study to assess whether Nanotilus preserves an immersive experience.

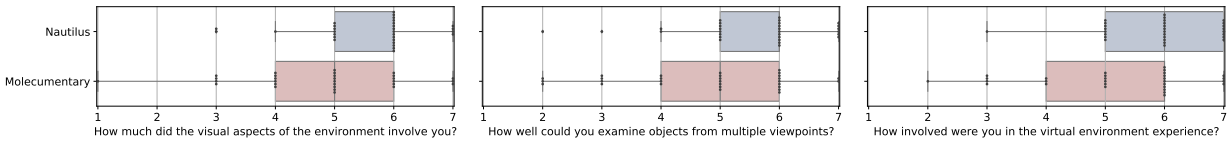
8 User Study

We invited 29 bioscience students to participate in a VR user study. In the study, participants conducted two guided tours through 3D models of viruses (HIV and SARS-CoV-2) using a VR HMD. The study took around 20 to 30 minutes, and participants were compensated with gift vouchers (50 USD each). The Institutional Biosafety and Bioethics Committee at KAUST approved the study.

The focus is on the formal comparison of the outside-in and inside-out views (Nanotilus) to explore the main contributions of the approach in a user study. We chose Molecumentary [5], which uses a cutting plane as an occlusion management technique to guide the user through an automated story, as a state-of-the-art baseline condition representing the outside-in view. Thus, the two conditions primarily differ from each other in how they visually present the current focus of the story to the user. In contrast to Molecumentary, Nanotilus allows users to become fully immersed in the instances and feel as if they were inside the model. Increasing the sense of presence is also likely to increase learning and understanding [64]. Therefore, we hypothesize that users would report a deeper understanding of the 3D structure of the presented molecules despite the density of the model due to our smart visibility handling using the inside-out view.

8.1 Stimuli and Apparatus

The textual story was split into separate textual snippets for each target instance. These snippets were presented using



(a) $Z = 2.562, p = .010$ (b) $Z = 2.480, p = .013$ (c) $Z = 2.163, p = .031$

Fig. 9: Items from the sense of presence questionnaire displaying significant results of the Wilcoxon signed rank tests.

text-to-speech synthesis. As the focus of the study was on visual scene perception and not navigation, we used the Wizard of Oz method to guide the users through both scenes. Upon verbal request, the experimenter advanced the story so that the user was transitioned to the next target instance, and the next section in the synthesized audio narration was played. At each target instance, the tour was paused, and the users could look around freely to inspect their surroundings.

The users stood while viewing both scenes using an HTC Vive Cosmos. As we employed the Wizard of Oz method with speech commands, users did not receive any pointing or teleportation controllers. This way, inexperienced VR users also could navigate the scene efficiently, without introducing a potential confounding effect.

8.2 Study Design and Procedure

We employed a within-subject design with one experimental factor (interface: Nanotilus or Molecumentary). Figure 13 depicts the SARS-CoV-2 scene for both study conditions. To compensate for learning effects, we counterbalanced the order of appearance of the two interfaces and the assignment of the two data sets to the two interfaces.

To measure the users' sense of presence and overall impression, we collected the following dependent measures. First, after each condition, users completed a questionnaire related to the sense of presence. We adapted a widely adopted presence questionnaire [64] by selecting only questionnaire items referring to the visual input and overall user experience. Second, we issued a final preference questionnaire, in which users rated their overall impressions of the two conditions and provided free written feedback on each condition after exposure to both conditions. The complete questionnaires, responses, and procedures are presented in the User study document in the supplementary material.

Of the 29 students, 17 were female and 12 male, ages 22 to 39, with a mean age of 27.1. All students were master's degree or Ph.D. students of bioscience. Their self-reported knowledge of SARS-CoV-2 and HIV was an average of 4.5 and 3.8, respectively, on a 7-point Likert scale. With the bioscience student sample, we assumed that all users were generally interested in the topic, yet not experts with deep prior knowledge of the viral structures.

8.3 Study Results

Presence was measured using the results from the sense of presence questionnaire. For the 15 items in the adapted presence questionnaire, three responses presented in Figure 9 yielded significant results. In all three cases, the responses were significantly higher for Nanotilus than Molecumentary (Figure 9). Participants were more involved (visually and overall) in the Nanotilus condition and could more easily inspect the structures from multiple viewpoints. The remaining questionnaire results are provided in the User study document in the supplementary material.

The overall average preference rating on the 7-point Likert scale was slightly higher for Nanotilus (4.25) than for Molecumentary (4.07), but this difference is not statistically significant ($Z = -0.787, p = 0.431$). We performed open coding to qualitatively analyze the users' subjective textual feedback of both conditions to understand the perceived strengths and weaknesses better. Two independent coders established four feedback categories: experience (engagement and overall user experience), spatial cognition (understanding of scales and spatial relations), locomotion (feedback concerning movement and scene interaction), and guidance (orientation and ability to follow the story). All coded utterances were categorized into positive and negative feedback (Figure 10). The feedback confirms that Nanotilus leads to a more positive user experience overall. For example, people commented that they *"liked how close I was to the structures"*, *"felt [they were] inside the virus"*, and found it *"very immersive"*. In contrast, Molecumentary seems to provide an easier way to follow the story. Users commented that this outside-in view *"gives you an overview"*, *"is very clear"*, and uses *"good highlighting"*, whereas participants commented that they sometimes did not know where to focus using Nanotilus. Users criticized the *"lack of guide arrows"* or other types of annotation for the current objects of interest.

For both interfaces, some users expressed the wish to have more control over their location inside the scene. For example, one user stated *"wished I could walk closer inside or walk backward away from the structures"* using Nanotilus. Two users mentioned that they would like to observe selected structures from different angles (two for Nanotilus, one for Molecumentary). Using Molecumentary, users wished to *"go inside the virus and see what is there"* and *"see the structures [more] closely"*. Even though steering locomotion is known to induce cybersickness symptoms [65], only two users reported slight feelings of dizziness (both for Nanotilus). Three users explicitly mentioned that they did not feel sick or dizzy (one for Nanotilus and two for Molecumentary). One strength of Nanotilus' inside-out view for molecular visualization is its ability to enhance spatial cognition. According to the users, it *"intuitively showed the size of the molecules [and] their relation to one another"* so that *"it is nice to see [...] how small some of the constructs are compared to others; also cells are quite densely packed"*. In contrast, one user commented that *"the scale using the Molecumentary tour was harder to comprehend as the virus took up most of the visual field*

in front of the view, making it seem larger in comparison to the virus in the Nanotilus tour". The comprehensive coded list of user comments is presented in the User study document in the supplementary material.

9 Discussion

The sense of presence questionnaire results and user feedback indicate that Nanotilus' immersive inside-out view increases engagement and involvement. However, this increased engagement does not automatically lead to a higher acceptance of Nanotilus as compared to a state-of-the-art outside-in view using a clipping plane to resolve occlusions. In the latter case, users reported a clearer overview and could therefore more easily follow the narration. Studies comparing looking *into* abstract data, such as scatterplots and graphs, as opposed to looking *onto* the data, have reported similar results. Users are more engaged inside a data set [54] and tend to feel more present [55]. Moreover, they can gain new, insightful perspectives on the data [56]. However, they also experience a higher task load [54], more loss of overview [55], and an increased tendency toward cybersickness [56].

In summary, immersing *into* the structures seems to trade the ability to maintain an overview against a more pronounced sense of presence (i.e., a sense of *being in* the environment [64]). Results from the subjective user feedback indicate that this increased sense of presence can improve users' spatial understanding of the visualized structures. It is known from VR research that immersion can improve spatial understanding [66], yet there has been no scientific evidence so far whether an inside-out view can improve the spatial understanding of a molecular scene compared with an outside-in view. Future work must consider ways to fluidly transition between a clear outside-in overview of the narration to an inside-out view for a detailed and more engaging inspection of structures to combine the advantages.

10 Conclusion and Future Work

We developed a novel guided-tour generator that can transform a 3D model and associated script into an immersive guided tour about the biological nanoworld. On top of multiple existing technologies, we developed a new journey planning technique and a new sparsification method, which enable an engaging, immersive experience when coupled together. There are two steps to improve the current software prototype: performance acceleration and a new sparsification technique that strengthens the sense of orientation while preserving the immersion. The Nanotilus guided tour generator is designed for a linear narrative; however, in principle, it is easily expandable to support non-linear storytelling. In such a scenario, the story has several paths. It branches and forms a story tree or even a story graph. With such a representation, the interaction can be as follows: the user visits a particular node and is offered several ways to proceed, or the user engages in conversation with the system, and the most related edge in this story structure would be followed. Conversational visualization is an important topic to further investigate in this particular setting or visualization in general. On the author side, this means not only one story needs to be authored, but the entire guided structure is authored with various degrees of automation. In this paper, we focused on assisted forms of interaction, as it is frequently the choice in explanatory visualization scenarios. Based on the user study, we consider the proposed sparsification strategy feasible and conceive a combination of sparsification geometry for Molecumentary and Nanotilus. In principle, we can smoothly combine these two sparsification geometries, provided both are formulated as implicit objects. With such a combination, we can occasionally enable a general oversight view using Molecumentary sparsification, which is blended back into the immersive experience of Nanotilus. Thus, we do not require proxy elements, such as a miniaturized world overview, which requires divided attention from the users. The degree of sparsification was the control factor in our study. However, the navigation through the void spaces could also have a strong influence on the perception. There are indications in prior work that support this consideration. For example, in VR training simulations, the amount of visual details had an influence on the training success [67], [68]. In VR crowd simulations, the density influences where participants look [69]. We can therefore expect that the number of visible instances, how they are presented, and how the user can navigate through them, also have a considerable influence on the users' experiences. We consider this as an important point for future research.

In the Nanotilus guided-tour generator, we provide the itinerary from the author's script. Although this itinerary results in an engaging visual experience, the viewers passively consume the provided information. Another method could be to provide the viewers with the option to create their own stories throughout the model, potentially leading to even stronger engagement. Nanotilus currently follows a predefined tour plan. In an interactive scenario, multiple options exist regarding where to go and what to explore. Another challenge is to create an explorative environment, which still guarantees a specific learning outcome. For this purpose, subtle gaze direction and flicker-guiding techniques [70] can unperceivably guide the viewers to explore specific structures, with the illusion of free will, where viewers would feel they discovered the intended target instances by choice.

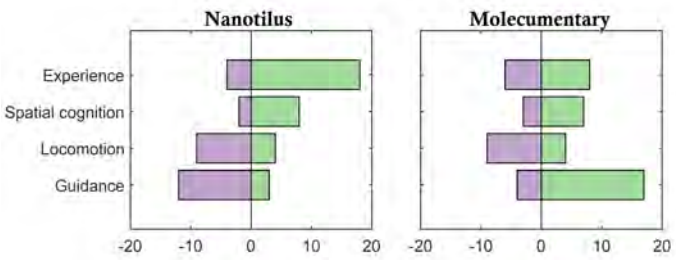


Fig. 10: User feedback for Nanotilus (left) and Molecumentary (right) coded into four categories. Bars indicate the amount of positive and negative feedback for each category and condition.

Acknowledgments

The research was supported by the King Abdullah University of Science and Technology (BAS/1/1680-01-01) and

KAUST Visualization Core Lab. We thank nanographics.at for providing the Marion software, Alex Kouyoumdjian for insightful comments, and Jules Verne for inspiration.

REFERENCES

[1] G. T. Johnson, L. Autin, M. Al-Alusi, D. S. Goodsell, M. F. Sanner, and A. J. Olson, "cellPACK: a virtual mesoscope to model and visualize structural systems biology," *Nature methods*, vol. 12, no. 1, pp. 85–91, 2015.

[2] E. F. Pettersen, T. D. Goddard, C. C. Huang, E. C. Meng, G. S. Couch, T. I. Croll, J. H. Morris, and T. E. Ferrin, "UCSF ChimeraX: Structure visualization for researchers, educators, and developers," *Protein Science*, vol. 30, no. 1, pp. 70–82, 2021.

[3] L. Autin, M. Maritan, B. A. Barbaro, A. Gardner, A. J. Olson, M. Sanner, and D. Goodsell, "Mesoscope: A Web-based Tool for Mesoscale Data Integration and Curation," in *Workshop on Molecular Graphics and Visual Analysis of Molecular Data*, 2020.

[4] N. Nguyen, O. Strnad, T. Klein, D. Luo, R. Alharbi, P. Wonka, M. Maritan, P. Mindek, L. Autin, D. Goodsell, and I. Viola, "Modeling in the Time of COVID-19: Statistical and Rule-based Mesoscale Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 722–732, 2020.

[5] D. Kouřil, O. Strnad, P. Mindek, S. Halladjian, T. Isenberg, M. E. Gröller, and I. Viola, "Molecumentary: Scalable Narrated Documentaries Using Molecular Visualization," *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[6] N. Elmqvist and P. Tsigas, "A Taxonomy of 3D Occlusion Management for Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 5, pp. 1095–1109, 2008.

[7] I. Viola and M. E. Gröller, "Smart Visibility in Visualization," in *Proc. of Computational Aesthetics in Graphics, Visualization and Imaging*, 2005, pp. 209–216.

[8] S. K. Feiner and D. D. Seligmann, "Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations," *The Visual Computer*, vol. 8, no. 5, pp. 292–302, 1992.

[9] M. Agrawala, D. Phan, J. Heiser, J. Haymaker, J. Klingner, P. Hanrahan, and B. Tversky, "Designing Effective Step-by-Step Assembly Instructions," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 828—837, 2003.

[10] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller, "Illustrative Context-Preserving Exploration of Volume Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1559–1569, 2006.

[11] I. Viola, M. Feixas, M. Sbert, and M. E. Gröller, "Importance-Driven Focus of Attention," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 933–940, 2006.

[12] J. Kruger, J. Schneider, and R. Westermann, "ClearView: An Interactive Context Preserving Hotspot Visualization Technique," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 941–948, 2006.

[13] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin, "Interactive Cutaway Illustrations of Complex 3D Models," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 31—42, 2007.

[14] S. Sigg, R. Fuchs, R. Carnecky, and R. Peikert, "Intelligent Cutaway Illustrations," in *Proceedings of the 2012 IEEE Pacific Visualization Symposium*, 2012, pp. 185—192.

[15] M. Ament, T. Zirr, and C. Dachsbacher, "Extinction-Optimized Volume Illumination," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 7, pp. 1767–1781, 2017.

[16] Å. Birkeland, S. Bruckner, A. Brambilla, and I. Viola, "Illustrative Membrane Clipping," *Computer Graphics Forum*, vol. 31, no. 3pt1, pp. 905–914, 2012.

[17] J. Díaz, E. Monclús, I. Navazo, and P. Vázquez, "Adaptive Cross-Sections of Anatomical Models," *Comput. Graph. Forum*, vol. 31, no. 7pt2, pp. 2155—-2164, 2012.

[18] D. Kouřil, T. Isenberg, B. Kozlíková, M. Meyer, M. E. Gröller, and I. Viola, "HyperLabels: Browsing of Dense and Hierarchical Molecular 3D Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 8, pp. 3493–3504, 2021.

[19] M. Le Muzic, P. Mindek, J. Sorger, L. Autin, D. S. Goodsell, and I. Viola, "Visibility Equalizer Cutaway Visualization of Mesoscopic Biological Models," *Computer Graphics Forum*, vol. 35, no. 3, pp. 161–170, 2016.

[20] W. Li, M. Agrawala, B. Curless, and D. Salesin, "Automated Generation of Interactive 3D Exploded View Diagrams," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1—7, 2008.

[21] Å. Birkeland and I. Viola, "View-Dependent Peel-Away Visualization for Volumetric Data," in *Proceedings of the 25th Spring Conference on Computer Graphics*, 2009, pp. 121—128.

[22] J. Sorger, P. Mindek, P. Rautek, M. E. Gröller, G. Johnson, and I. Viola, "Metamorphers: Storytelling Templates For Illustrative Animated Transitions in Molecular Visualization," in *Proceedings of the Spring Conference on Computer Graphics 2017*, 2017, pp. 27–36.

[23] N. Elmqvist, "BalloonProbe: Reducing Occlusion in 3D Using Interactive Space Distortion," in *Proc. VRST*, 2005, pp. 134–137.

[24] N. Elmqvist and M. Eduard Tudoreanu, "Occlusion Management in Immersive and Desktop 3D Virtual Environments: Theory and Evaluation," *International Journal of Virtual Reality*, vol. 6, no. 2, pp. 21–32, 2007.

[25] M. Christie and P. Olivier, "Camera Control in Computer Graphics: Models, Techniques and Applications," in *ACM SIGGRAPH ASIA 2009 Courses*, ser. SIGGRAPH ASIA '09, 2009.

[26] P. Mindek, G. Mistelbauer, M. E. Gröller, and S. Bruckner, "Data-Sensitive Visual Navigation," *Computers & Graphics*, vol. 67, no. C, pp. 77–85, 2017.

[27] J. D. Mackinlay, S. K. Card, and G. G. Robertson, "Rapid Controlled Movement through a Virtual 3D Workspace," *SIGGRAPH Comput. Graph.*, vol. 24, no. 4, pp. 171—-176, 1990.

[28] J. McCrae, I. Mordatch, M. Glueck, and A. Khan, "Multiscale 3D Navigation," in *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, 2009, pp. 7—14.

[29] M. Trellet, N. Ferey, M. Baaden, and P. Bourdot, "Content and task based navigation for structural biology in 3D environments," in *2015 IEEE 1st International Workshop on Virtual and Augmented Reality for Molecular Science (VARMS@IEEEVR)*, 2015, pp. 31–36.

[30] B. Salomon, M. Garber, M. C. Lin, and D. Manocha, "Interactive Navigation in Complex Environments Using Path Planning," in *Proceedings of the 2003 Symposium on Interactive 3D Graphics*, 2003, pp. 41—50.

[31] A. Calomeni and W. Celes, "Assisted and Automatic Navigation in Black Oil Reservoir Models Based on Probabilistic Roadmaps," in *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, 2006, pp. 175—-182.

[32] F. Yan, Y.-S. Liu, and J.-Z. Xiao, "Path Planning in Complex 3D Environments Using a Probabilistic Roadmap Method," *Int. J. Autom. Comput.*, vol. 10, no. 6, pp. 525—-533, 2013.

[33] W. Hsu, Y. Zhang, and K. Ma, "A Multi-Criteria Approach to Camera Motion Design for Volume Data Animation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2792–2801, 2013.

[34] P. Knöbelreiter, R. Berndt, T. Ullrich, and W.-D. Fellner, "Automatic Fly-through Camera Animations for 3D Architectural Repositories," in *GRAPP 2014 - Proceedings of the 9th International Conference on Computer Graphics Theory and Applications*, 2014, pp. 335–341.

[35] T. Oskam, R. W. Sumner, N. Thuerey, and M. Gross, "Visibility transition planning for dynamic camera control," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009, pp. 55–65.

[36] B. Sommer, A. Hamacher, O. Kaluza, T. Czauderna, M. Klapperstück, N. Biere, M. Civico, B. Thomas, D. Barnes, and F. Schreiber, "Stereoscopic Space Map – Semi-immersive Configuration of 3D-stereoscopic Tours in Multi-display Environments," 2016.

[37] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 2012, pp. 692–697.

[38] Q. Galvane, C. Lino, M. Christie, J. Fleureau, F. Servant, F.-L. Tariolle, and P. Guillotel, "Directing Cinematographic Drones," *ACM Transactions on Graphics*, vol. 37, no. 3, pp. 1–18, 2018.

[39] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-Time Planning for Automated Multi-View Drone Cinematography," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[40] C. Tong, R. Roberts, R. Borgo, S. Walton, R. S. Laramee, K. Wegba, A. Lu, Y. Wang, H. Qu, Q. Luo, and X. Ma, "Storytelling and Visualization: An Extended Survey," *Information*, vol. 9, no. 3, 2018.

[41] K. Ma, I. Liao, J. Frazier, H. Hauser, and H. Kostis, "Scientific Storytelling Using Visualization," *IEEE Computer Graphics and Applications*, vol. 32, no. 1, pp. 12–19, 2012.

[42] H. Akiba, C. Wang, and K. Ma, "AniViz: A Template-Based Animation Tool for Volume Visualization," *IEEE Computer Graphics and Applications*, vol. 30, no. 5, pp. 61–71, 2010.

[43] M. Wohlfart and H. Hauser, "Story Telling for Presentation in Volume Visualization," in *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, 2007, pp. 91—98.
[44] M. Thöny, R. Schnürer, R. Sieber, L. Hurni, and R. Pajarola, "Storytelling in Interactive 3D Geographic Visualization Systems," *ISPRS International Journal of Geo-Information*, vol. 7, no. 3, 2018.
[45] E. M. Lidal, H. Hauser, and I. Viola, "Geological Storytelling - Graphically Exploring and Communicating Geological Sketches," in *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, K. Singh and L. B. Kara, Eds., 2012.
[46] M. Slater and M. V. Sanchez-Vives, "Enhancing our lives with immersive virtual reality," *Frontiers in Robotics and AI*, vol. 3, p. 74, 2016.
[47] J. Keiriz, L. Zhan, M. Chukhman, O. Ajilore, A. Leow, and A. Forbes, "Exploring the Human Connectome Topology in Group Studies," 2017. [Online]. Available: https://arxiv.org/abs/1706.10297
[48] D. Boges, M. Agus, R. Sicat, P. J. Magistretti, M. Hadwiger, and C. Calì, "Virtual reality framework for editing and exploring medial axis representations of nanometric scale neural structures," *Computers & Graphics*, vol. 91, pp. 12–24, 2020.
[49] J. Cremer and J. K. Kearney, "Scenario Authoring for Virtual Environments," in *Proceedings of the IMAGE VII Conference*, 1994, pp. 141–149.
[50] M. Ponder, B. Herbelin, T. Moler, S. Schertenlieb, B. Ulicny, G. Papagiannakis, N. Thalmann, and D. Thalmann, "Immersive VR decision training: Telling interactive stories featuring advanced virtual human simulation technologies," Jan 2003, p. 97–106.
[51] J. Parong and R. E. Mayer, "Learning science in immersive virtual reality," *Journal of Educational Psychology*, vol. 110, no. 6, p. 785, 2018.
[52] K. Dooley, "Storytelling with virtual reality in 360-degrees: a new screen grammar," *Studies in Australasian Cinema*, vol. 11, no. 3, pp. 161–171, 2017.
[53] L. Zhang, D. A. Bowman, and C. N. Jones, "Exploring Effects of Interactivity on Learning with Interactive Storytelling in Immersive Virtual Reality," in *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, 2019, pp. 1–8.
[54] Y. Yang, M. Cordeil, J. Beyer, T. Dwyer, K. Marriott, and H. Pfister, "Embodied Navigation in Immersive Abstract Data Visualization: Is Overview+Detail or Zooming Better for 3D Scatterplots?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, p. 1214–1224, Feb 2021.
[55] M. Kraus, N. Weiler, D. Oelke, J. Kehrer, D. A. Keim, and J. Fuchs, "The Impact of Immersion on Cluster Identification Tasks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 525–535, Jan 2020.
[56] J. Sorger, M. Waldner, W. Knecht, and A. Arleo, "Immersive Analytics of Large Dynamic Networks via Overview and Detail Navigation," in *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2019, pp. 144–1447.
[57] K. A. Hawick, A. Leist, and D. P. Playne, "Parallel Graph Component Labelling with GPUs and CUDA," *Parallel Computing*, vol. 36, no. 12, pp. 655–678, 2010.
[58] D. P. Playne and K. Hawick, "A New Algorithm for Parallel Connected-Component Labelling on GPUs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 6, pp. 1217–1230, 2018.
[59] E. Lengyel, *Mathematics for 3D game programming and computer graphics*. Nelson Education, 2012.
[60] D. Kouřil, L. Čmolík, B. Kozlíková, H. Y. Wu, G. Johnson, D. S. Goodsell, A. Olson, M. E. Gröller, and I. Viola, "Labels on Levels: Labeling of Multi-Scale Multi-Instance and Crowded 3D Biological Environments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 977–986, 2019.
[61] P. Mindek, D. Kouřil, J. Sorger, D. Toloudis, B. Lyons, G. Johnson, M. E. Gröller, and I. Viola, "Visualization Multi-Pipeline for Communicating Biology," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 883–892, 2018.
[62] M. L. Muzic, L. Autin, J. Parulek, and I. Viola, "cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets," in *Eurographics Workshop on Visual Computing for Biology and Medicine*, K. Bühler, L. Linsen, and N. W. John, Eds., 2015, pp. 61–70.
[63] G. T. Johnson, D. S. Goodsell, L. Autin, S. Forli, M. F. Sanner, and A. J. Olson, "3D molecular models of whole HIV-1 virions generated with cellPACK," *Faraday Discussions*, vol. 169, pp. 23–44, 2014.
[64] B. G. Witmer and M. J. Singer, "Measuring presence in virtual environments: A presence questionnaire," *Presence*, vol. 7, no. 3, pp. 225–240, 1998.
[65] J. Clifton and S. Palmisano, "Effects of steering locomotion and teleporting on cybersickness and presence in HMD-based virtual reality," *Virtual Reality*, vol. 24, no. 3, pp. 453–468, 2020.
[66] D. A. Bowman and R. P. McMahan, "Virtual reality: how much immersion is enough?" *Computer*, vol. 40, no. 7, pp. 36–43, 2007.
[67] E. D. Ragan, D. A. Bowman, R. Kopper, C. Stinson, S. Scerbo, and R. P. McMahan, "Effects of Field of View and Visual Complexity on Virtual Reality Training Effectiveness for a Visual Scanning Task," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 7, p. 794–807, Jul 2015.
[68] S. Lessels and R. A. Ruddle, "Movement Around Real and Virtual Cluttered Environments," *Presence: Teleoperators and Virtual Environments*, vol. 14, no. 5, pp. 580–596, Oct 2005.
[69] F. Berton, L. Hoyet, A.-H. Olivier, J. Bruneau, O. Le Meur, and J. Pettre, "Eye-Gaze Activity in Crowds: Impact of Virtual Reality and Density," in *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2020, pp. 322–331.
[70] S. Rothe, D. Buschek, and H. Hußmann, "Guidance in cinematic virtual reality-taxonomy, research status and challenges," *Multimodal Technologies and Interaction*, vol. 3, no. 1, p. 19, 2019.

Ruwayda Alharbi is a Ph.D. student at King Abdullah University of Science and Technology (KAUST), Saudi Arabia. She received her Master's degree in 2016 from King Saud University, Saudi Arabia. Her research interest lies in scientific visualization, where she focuses on designing novel visualization methods that support exploration and understanding of 3D mesoscale biological models.

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

Laura R. Luidolt has been a Ph.D. student at TU Wien working on XR, in particular perception in VR.

Manuela Waldner is an Assistant Professor at the Institute of Visual Computing and Human-Centered Technology at TU Wien. Her research interests cover human-computer interaction and visualization, with a special focus on perception of visualizations and intelligent visual interfaces for the interactive exploration and analysis of complex data. Waldner has a PhD in computer science from the Graz University of Technology.

David Kouřil is a post-doctoral researcher at the Masaryk University in Brno, Czech Republic. He received his doctorate in 2021 from TU Wien in Vienna, Austria. His research topic lies in scientific visualization, where he focuses on three-dimensional data coming from structural biology. He designs novel visualization and interaction methods to support exploration and understanding of the environments that this data represents.

Ciril Bohak is a postdoctoral research fellow at King Abdullah University of Science and Technology (KAUST), Saudi Arabia, and an assistant professor at the Faculty of Computer and Information Science, University of Ljubljana, Slovenia. He received his B.Sc., M.Sc., and Ph.D. degree from the University of Ljubljana. His research covers computer graphics, scientific visualization, and human-computer interaction.

Tobias Klein is CEO and co-founder of Nanographics, which creates biomedical animations and software solutions. He received his doctoral degree from TU Wien, Austria, where he was working on biological mesoscale visualization and model generation. In his work, he tries to build a bridge between research and industry covering topics ranging from scientific visualization to parallel computing.

Eduard Gröller is a Professor at the Institute of Visual Computing and Human-Centered Technology at TU Wien, and adjunct professor of computer science at the University of Bergen, Norway. His research interests include computer graphics, visualization, and visual computing.

Ivan Viola is a Professor at King Abdullah University of Science and Technology (KAUST), Saudi Arabia. He graduated from TU Wien, Austria, in 2005 he took a postdoc position at the University of Bergen, Norway, where he was gradually promoted to the professor rank. In 2013 he received a WWTF grant to establish a research group at TU Wien. At KAUST, he continues developing new technologies that make visual, in-silico life at nanoscale possible. Viola co-founded the Nanographics startup to commercialize nanovisualization technologies.