Eurographics Conference on Visualization (EuroVis) 2021
R. Borgo, G. E. Marai, and T. von Landesberger
(Guest Editors)

Volume 40 (2021), Number 3

Accelerating Visualization of Dense Molecular Scenes through Adaptive Occlusion Estimation

Matúš Talčík[1], Barbora Kozlíková[1], Peter Mindek[2,3]

[1]Department of Visual Computing, Faculty of Informatics, Masaryk University, Brno, Czech Republic
[2]Institute of Visual Computing & Human-Centered Technology, Faculty of Informatics, TU Wien, Vienna, Austria
[3]Nanographics GmbH

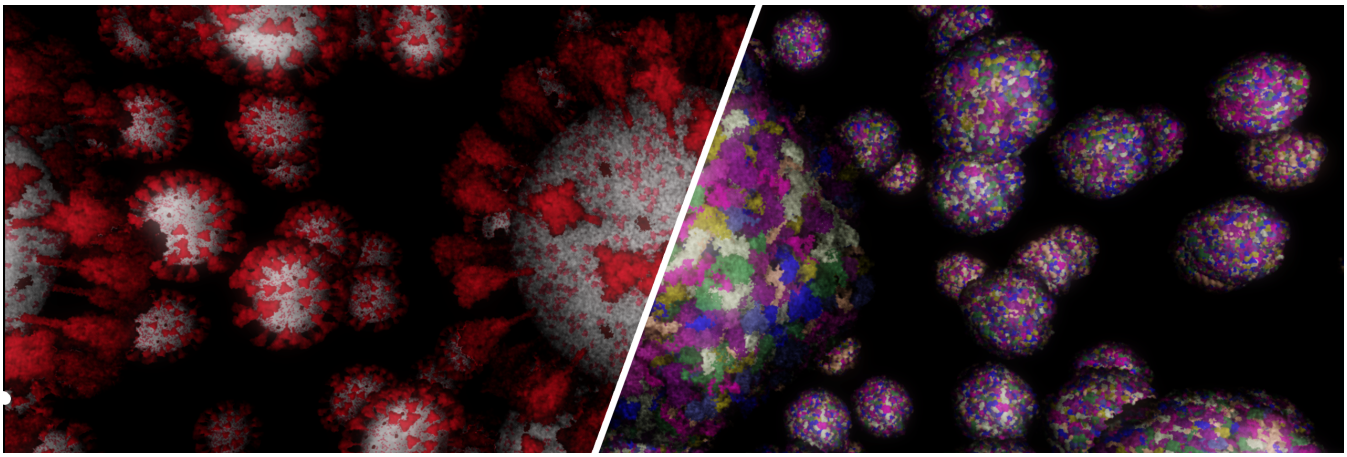


Figure 1: Demonstration of our algorithm. We showcase scenes consisting of 1000 structures that together have around 80 million molecules (left) and 120 billion atoms (right). The scenes run in real-time at between 10 and 30 frames per second.

Abstract

Visual communication of complex phenomena taking place in living organisms is substantial in many areas, namely in education, interdisciplinary exchange of knowledge, or dissemination of scientific findings to a general audience. Communicating molecular processes to a lay audience requires a thorough depiction of the context, which consists of molecular landscapes of immense size. Depicting them in real-time as means for realistic and engaging interactive visualization poses significant challenges to the rendering performance. However, achieving interactive framerates for complex scenes, operating with structures on all scales spanning from atoms up to almost entire cells, still cannot be reached by the existing algorithms. Therefore, we present a novel approach that significantly reduces the GPU workload for large and dense molecular scenes. It allows us to scale up the interactive visualization of molecular scenes consisting of 10^{11} atoms, which is on the scale of an entire bacterial cell. As the models for such large structures are still not available up to their atomistic details, we tested the performance of our approach on synthetic models. Compared to the current state-of-the-art approaches, we achieved the 3x to 7x speedup.

CCS Concepts
• ***Computing methodologies*** → *Rasterization;* ***Visibility;*** • ***Human-centered computing*** → ***Scientific visualization;***

1. Introduction

Molecular visualization has been traditionally used in biology and biochemistry to understand and analyze large molecular structures and interactions between them. Historically, the molecular models were assembled manually, and with the advent of computer graphics, first virtual models of individual molecules started to emerge. Nanoscale structures, such as protein molecules consisting of hun-

dreds or thousands of atoms, could not be studied in detail without creating their representative 3D models, as the imaging techniques back then did not offer sufficient resolution.

Even with today's techniques, which allow the scientists to observe nanoscale structures near the atomic resolution, such as x-ray crystallography or cryogenic electron microscopy, the data has to be reconstructed into representative 3D models in order to be visualized. Computer graphics remains the only viable means for interactive analysis and visualization of molecular structures at atomic resolution.

Nowadays, the dedicated programmable graphics hardware enables us to visualize structures that are orders of magnitude larger than a single protein. As the computational power of graphics hardware rose, we are now able to interactively visualize structures in so-called mesoscale, i.e., structures of the size of whole virus particles (so-called virions), in their atomic resolution. The ability to display structures of such sizes made it plausible to use molecular visualization to create complex molecular animations and interactive experiences. Such visualizations are invaluable in conveying detailed explanations of molecular processes that consist of large numbers of interacting molecules. The primary purpose of these explanatory visualizations is to be utilized in knowledge dissemination when the goal is to communicate certain aspects of the molecular realm to a general audience. The limited background knowledge of laypersons implies that the context has to be properly visualized together with the main concept.

Biomolecular processes are inherently multi-scale, so they have to be explained on different levels of detail. To convey them in a clear way to the lay audience, they have to be visualized as such. Generally, the processes start as interactions between individual molecules depending on their atomic configurations, which in turn has many implications on larger, mesoscale molecular structures, such as cell organelles or even the entire cells. Displaying all of the necessary context for such a process would require rendering of an overwhelmingly large number of molecules.

With the currently available methods, the only plausible way to render such complex scenes is to simplify them to a large extent. This can be done, for example, by abstracting molecular details into simpler geometric forms. However, there are several problems associated with such an approach. When the goal is to illustrate the process to a layperson, who does not have any prior knowledge about the conveyed phenomenon or molecular visualization as such, biological structures cannot be arbitrarily simplified to the point which would require any additional background knowledge in order to understand the story. Representations of varying detail within a single story could introduce ambiguities that could lead to miscommunication of the intended message. This is the main motivation for developing methods that are able to display large mesoscale models, which serve for communicating molecular processes, entirely in the atomic resolution.

In general, when creating stories using molecular visualization, it is necessary to consider that the introduced abstraction is a separate visual channel that can unintentionally communicate information that may be misinterpreted by a lay audience. When conveying processes, such as the mechanisms of vaccines, there is a clear need to create the sense of trust in what is being shown. The implied realism inherent to models in atomic resolution might be a decisive aspect in the acceptance of the shown visualization to the lay audience as truth, rather than an uncertain, general concept. With processes, such as the mechanism of a vaccine, this is particularly important, as understanding of them by general audience might be crucial for their acceptance by people and thus influencing the public health.

There already exist methods that are able to interactively display large molecular scenes consisting of up to 15 billion atoms [FKE13, LBH12, LMAPV15]. However, these methods rely on heavy level-of-detail simplification, which does not scale beyond individual molecules, and there is no general solution that can create simplification of every biological structure. Moreover, even with such an approach, they still cannot reach the scales necessary for interactive exploration of very large structures, such as bacteria or even cells.

With these requirements and limitations in mind, we propose a method for interactive visualization of large molecular scenes, where the preservation of the atomic resolution is maintained for all visible molecules. Instead of further simplification of biological structure, our method evaluates the potential visibility of their molecules and removes those which will be occluded from all available viewpoints. It does so iteratively during rendering, thus significantly reducing the preprocessing step. In consequence, only the visible molecules are rendered, which presents a significant simplification in dense molecular scenes. Subsequently, we are able to render the individual atoms of the visible molecules at interactive frame rates even for scenes containing more than 10^{11} atoms.

2. Related Work

In this section, we provide the readers with the summary of the research most relevant to our solution. The section is divided into three parts. First, we present the closely related existing algorithms, and we explain the insufficiency of these related methods in comparison to our situation. In the second part, we present the tools that are used for creation of the molecular models in the scale we are targeting and how they are used in storytelling.

2.1. Algorithms

There are two main categories of algorithms aiming to improve the rendering time of complex and densely populated scenes such as those that we are dealing with. Algorithms belonging to the first category try to remove occluded objects from a scene using hierarchical Z-buffer (Hi-Z) [GKM93]. The second category is based on the replacement of objects in the scene by billboards, an approximate representation of the original objects.

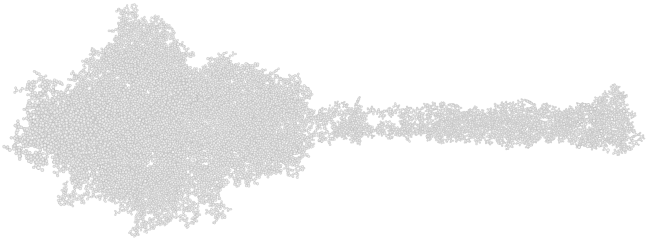
The Hi-Z approach can be improved by selecting a subset of a 3D model, in our case a molecule, as an occluder [ZMHH97]. The occlusion culling can be done either on CPU [Kua] or GPU [ANA]. On GPU, the occlusion can be calculated either using occlusion queries [BWPP04] or vertex and fragment shaders [HC11]. It is possible to create and compact render calls on GPU with modern APIs [HA] to avoid the synchronization between CPU and GPU. Further optimization can be done by clustering polygons

into so-called meshlets and culling them based on the visibility cone [HA15]. Furthermore, temporal coherency can be used by re-projecting the depth buffer from a previous frame [Gre95].

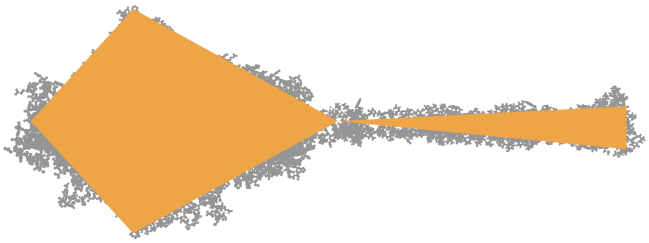
The Hi-Z method uses occluders for the efficient generation of a depth map. Occluders should satisfy two conditions: they should be significantly simpler than their original model so that their rendering time is low, and they should occlude as large space as possible so that they occlude the maximum possible number of objects. These occluders can be either created manually or automatically generated [NAA, SSLL14].

The problems of Hi-Z based methods for our case are two-fold. First, they require the creation of efficient occluders. The Protein Data Bank [BWF*00] contains hundreds of thousands of molecules, making it impossible to expect users to create those occluders manually, as it is done for computer games. However, automatic methods fail to generate satisfactory occluders due to the nature of molecular data, as demonstrated in Figure 2. Second, the Hi-Z method needs to process each object at least twice, first to render occluders and then to sample the depth buffer. In-between, it needs to run a reduction operation to remove null draw calls for efficiency [HA]. In our scenes containing hundreds of millions of occludees, the computation of such operation would be prohibitively expensive. A similar problem is described in the work of Grottel et al. [GRDE10] where they use a grid of cells and thousands of occlusion queries for every single cell. They can hide the performance hit of occlusion queries on a single object consisting of atoms only, but their solution does not scale up to hundreds to thousands of objects that we require in our scenes.

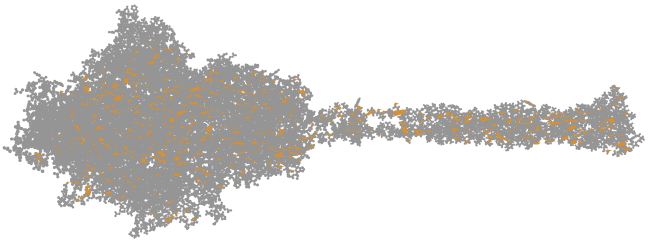
The algorithms of the second category replace the objects, in our case the biological structures, with billboards. Billboards can be 2D [DDSD03] or 3D [DN09, Ney98]. According to Cook et al. [CHPR07], voxel-based methods exhibit the best quality. The state-of-the-art method is based on volumetric billboards [DN09]. However, Cook et al. [CHPR07] note that even for a low-resolution rendering, volumes take around 18MB of memory. This amount of memory would be needed for every single biological structure in our scenes. In addition, the memory would have to be multiplied for every color scheme used in the visualization.



(a) Stick model of the SARS-COV-2 spike protein, consisting of 32,126 atoms.



(b) Example of the desired ideal occluder consisting of only three triangles. Compared to the naive approach when we have one triangle for each atom, this represents a significant reduction.



(c) Real occluder generated by [SSLL14]. It consists of many small triangles that are covering only a small subset of the atoms.

Figure 2: Example demonstrating the difficulty with automatically generated occluders for molecular data. The occupation of the viewpoint by the molecule (in grey) should be overlapped by occluders (yellow) as much as possible and ideally with a small amount of triangles. The existing automatic solutions are producing many occluders of poor quality of coverage.

2.2. Modeling and Storytelling

As already stated, large molecular scenes with atomic detail are impossible to image with a microscope due to physical limitations. Because of that, such scenes are created in various dedicated modeling tools. Several methods and tools have been developed to assist with the process of designing such mesoscale scenes.

Plugins, such as MolecularMaya [Mol], ePMV 23 [JAG*11], or BioBlender [Bio], assist with the import, modeling, and animation of molecular structures within popular 3D animation packages. The GraphiteLifeExplorer 24 [HLLF13] streamlines the construction of complex assemblies with proteins and nucleic acids, allowing users to define 3D paths for DNA and other flexible chains, and populate the scene with associated protein structures. CellPAINT [GAB*18] is a fully manual tool that allows non-expert users to create interactive 2D mesoscale models with an illustrative design imitating hand drawings of David Goodsell [Goo91]. Nguyen et al. [NSK*20] present a rule-based tool to construct mesoscale models and demonstrate it by creating a scientifically accurate, atomistic model of SARS-CoV-2 virion.

These models can be used for interactive storytelling. Le Muzic et al. [LMPSV14] replaced simulation with omniscient intelligence and passive agents to provide real-time animation of molecular processes. Klein et al. [KVGM20] used fully procedural animation to demonstrate microtubules' behavior. Kouřil et al. [KIK*20] created HyperLabels, labels that users can interact with, and combined them with cutaway views to provide the users with insight into the inner structure of the HIV virion.

So far, the largest models for which the interactive frame rates for their visual exploration was reached were capturing a single virus. Therefore, interactive visualizations are nowadays focusing on similar scales. With better algorithms, interactive visualization

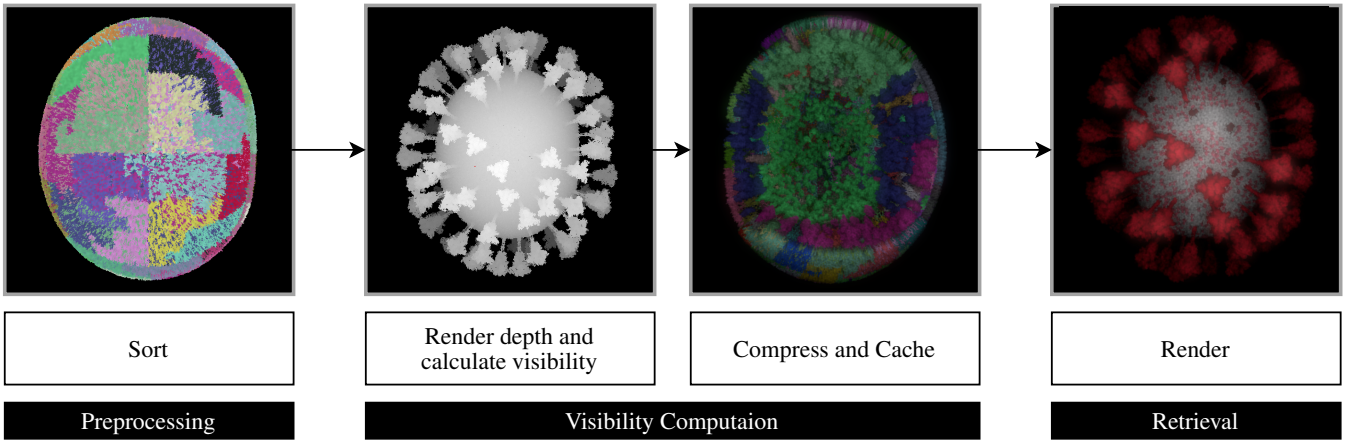


Figure 3: The stages of our method: 1) The preprocessing stage, which is executed during the data loading, sorts molecules of biological structures according to the Hilbert curve mapped onto a sphere. Continuous sorted patches are shown with different colors (some are omitted for clarity). Note that they may overlap as different types of molecules are sorted separately. 2) Visibility computation renders the depth map of a structure from a single viewpoint twice to retrieve visible molecules, which are then cached with our efficient compression based on the Hilbert curve and gap-filling. This stage is performed on-demand in real-time. 3) Retrieval stage retrieves the cached occlusion in $O(1)$ on a CPU and renders the object.

and storytelling could scale to much larger models, such as those of bacteria or even entire human cells.

3. Algorithm

Based on the problems and drawbacks of available solutions, we designed an algorithm that enables a real-time rendering of large biological scenes by efficient caching of visible molecules and their instant retrieval. We can assume that most of the mesoscale biological structures, such as the structure of SARS-CoV-2 virion, have a spherical or oval shape and they consist of densely packed molecules. Based on this assumption, we can expect that most of the molecules of a biological structure are not visible from a single camera viewpoint since they are inside the membrane or occluding each other. This leads to our idea of caching the visibility information of the molecules of the individual biological structures. The visibility information is cached for a given camera viewpoint. It would be impossible to cache every possible camera viewpoint, so we cache only a subset and always pick the one closest to the user's camera viewpoint.

Storing the visibility information for each molecule from each viewpoint could take too much memory, making this approach infeasible. That is why we created an approach to store this information efficiently. To further decrease the memory requirements, we also came up with a lossy compression method that can reduce the memory to an arbitrary limit.

The algorithm consists of the following three stages (see Figure 3):

- In the first stage, we load a scene into the memory. For each biological structure, we sort their molecules in memory so that in the later stages we can store and subsequently retrieve the information about their visibility from a set of predefined viewpoints, whose positions are also precalculated here.
- In the second stage, during the rendering of a frame, we compute visibility information for the predefined viewpoints of biological structures selected based on the users' camera viewpoint. We compute only the viewpoints that have not been computed up to the frame. Subsequently, we compress and store this information.
- In the last stage, based on the user's camera viewpoint, we look up the stored visibility information generated in the second stage and render the final scene.

3.1. Stage 1 – Preprocessing

The general idea behind our preprocessing stage is to have biological structures prepared in a way that allows us to efficiently store their visibility information in the second stage and retrieve it in the last stage. First, we sort molecules in memory in such a way that when they are close to each other in space, they are also close to each other in memory. For this criterion, the Hilbert curve was shown to be effective [MJFS01]. For a single camera viewpoint, this allows us to efficiently store visible molecules as they can be represented as a single continuous memory range.

Most of the visible molecules are on the membrane of biological structures. This situation is demonstrated in Figure 4. To improve the locality criterion, we use the Hilbert curve mapped onto a sphere as in Purser et al. [Pur09] instead of its spatial variant. Hilbert unit sphere is a unit sphere whose origin is aligned to the centroid of the biological structure. To get the Hilbert position of a molecule on a sphere, we map its local position onto Hilbert unit sphere as depicted in Figure 5.

Figure 6 illustrates how the sorted visible molecules can be clustered. We can see that instead of storing visibility information for each molecule separately, we can store an entire cluster as a single range.

Figure 4: Depiction of a single camera viewpoint for dense biological structure. Visible molecules are in blue, occluded ones in gray. Only few molecules, mostly those forming a part of the shell, are visible.

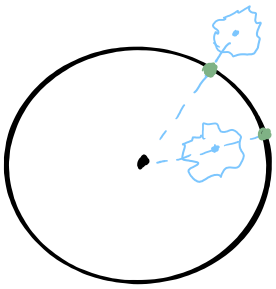


Figure 5: Mapping of molecules of a biological structure on a unit sphere. The unit sphere is black, molecules are blue. Mapped positions are depicted in green.

Second, as it would be impossible to store all possible views for each biological structure, we have to discretize camera viewpoints as part of this preprocessing stage. We calculate a bounding sphere for the biological structure and distribute viewpoints uniformly at every n degrees along the azimuth and polar angle of every biological structure's bounding sphere.

Large n may cause a so-called popping effect when two discretized viewpoints are too far from each other. Moving a camera between them at the time of the switch from one to another may introduce sudden pop of visible molecules. To overcome this, n must be set to a sufficiently small value. By experimenting on several biological structures, we observed that any angle below five degrees is sufficient to avoid visible popping.

3.2. Stage 2 – Visibility Computation

Within the rendering loop, in each frame, we iterate over each biological structure in the viewing frustum. We pick its predefined camera viewpoint that is closest to the camera position. If the camera viewpoint already contains the visibility information in the cache, this stage is terminated for this biological structure. Otherwise, for each camera viewpoint, we find molecules from the corresponding biological structure that are visible and store this information.

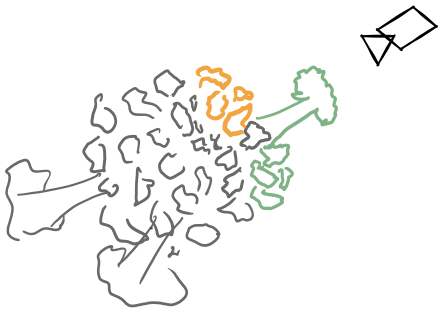


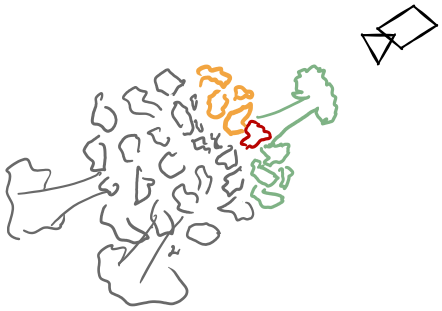
Figure 6: Depiction of visible molecules that can be stored as ranges. Different colors show separate ranges on the Hilbert curve.

To obtain visible molecules for a given camera viewpoint, we render a biological structure twice from that viewpoint. In the first pass, we render a depth buffer. In the second pass, for those fragments of the molecules that passed the depth test, we write a *true* value to a *visibility buffer* that stores the visibility information for each molecule separately. In the next step, we compress the visibility buffer into a *range buffer*, as depicted in Figure 7b. We find every interval of consecutive *true* values and store it to the range buffer as the range of indices where the interval starts and ends. This range will be pointing to the *content buffer* containing molecules of the biological structure.

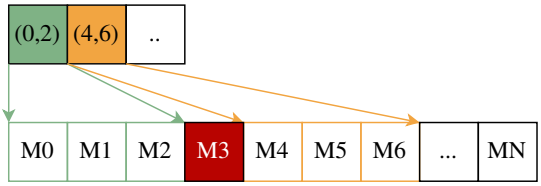
In the compression phase, better results could be reached when we are operating on large ranges. However, due to some overlapping molecules, the Hilbert curve can become disjoint, resulting in a large range buffer. We demonstrate this scenario in Figure 7a, where the red molecule is occluded by a bigger one in a viewing direction and disrupts the Hilbert curve. Instead of a single molecule, this problem can be caused by an entire cluster of molecules. We denote these clusters as *gaps*. To remedy the problem of having too many gaps, we designed the following algorithm that removes them.

The algorithm assigns each gap a *cost* that is defined as a number of molecules of the gap multiplied by the number of their atoms. The cost represents how many additional occluded objects we render if we mark them as visible. Subsequently, gaps with the lowest cost are iteratively removed by joining the neighbouring ranges and with the molecules in the gap. The process is depicted in Figure 7b and Figure 7c. The gap removal creates a lossy compression for range buffers, exchanging lost information about occluded molecules for the saved memory. We discuss the efficiency of this tradeoff in Section 4.

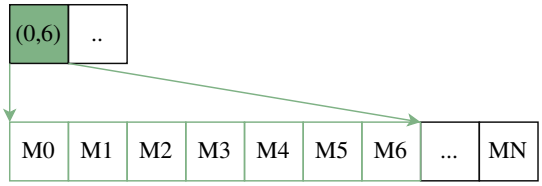
The described iterative removal procedure stops when a user-defined limit is reached. This limit of the number of ranges is based on the available memory, and on the limit of draw calls that the system can process. Discussion regarding the limit can be found in Section 4.



(a) Depiction of how some occluded molecules cause disruption in the Hilbert curve ordering of visible molecules. Grey molecules are occluded in the depicted viewpoint. Yellow and green are visible disjoint clusters in memory. Red molecules form occluded clusters that cause disruption in the Hilbert curve.



(b) The visibility information in Figure 7a in memory. One buffer contains ranges that point to clusters of molecules laid consecutively in memory in the second buffer. The molecule buffer is sorted according to the Hilbert curve.



(c) The visibility information in Figure 7a and 7b after closing two ranges by incorporating the gap between them.

Figure 7: Memory layout of molecules of a biological structure and the associated visibility information. Data associated with each molecule are stored linearly, sorted according to the Hilbert curve. The information which molecules are visible is stored efficiently as ranges of indices pointing to this array.

3.3. Stage 3 – Retrieval of Cached Visibility and Final Rendering

The last stage renders the biological structures from their visible viewpoints. With a fixed number of camera viewpoints established in Section 3.1 we first have to snap users' camera view to the closest camera viewpoint for each biological structure, as depicted in Figure 8. Afterwards, we map the spherical position of all selected viewpoints to their indices pointing to range buffers, as outlined in Algorithm 1. Atoms of molecules are then rendered as 2D sphere impostors.

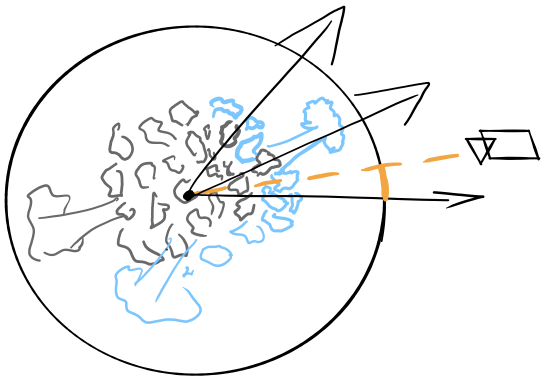


Figure 8: Camera viewpoints and snapping of users' camera to the closest viewpoint. Visible molecules from a snapped viewpoint are shown in blue. Yellow is the actual viewpoint that is being snapped to the closest one.

Algorithm 1 Spherical coordinates to index

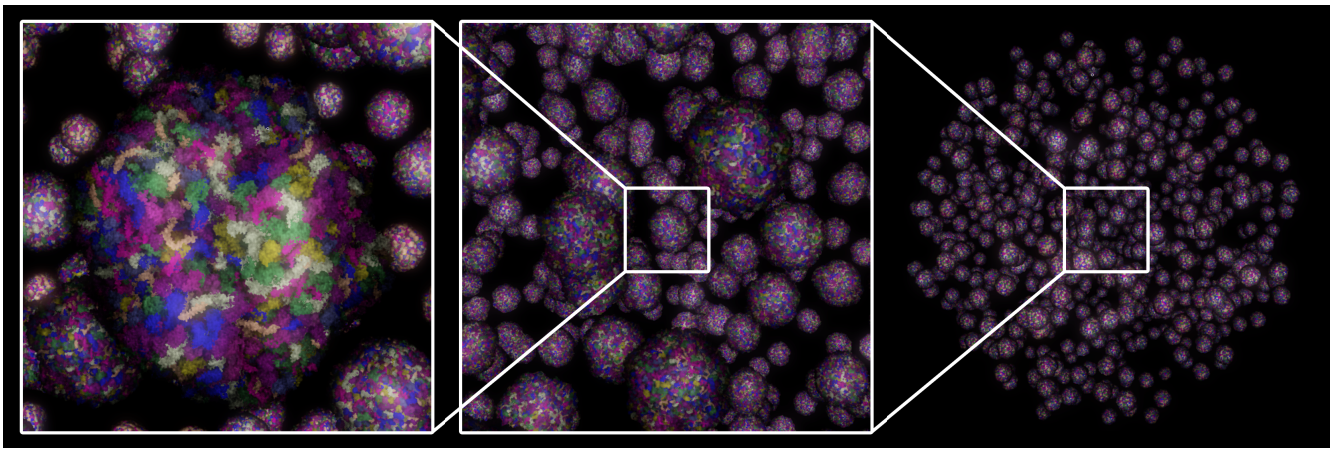
n, number of degrees between two views
$azimuth$, $zenith$, spherical coordinates of the snapped view
$steps \leftarrow 360/n$
$index \leftarrow (azimuth/n) * steps + zenith/n$

4. Results and Discussion

In this section, we summarize the benefits of the proposed algorithm and discuss how we managed to fulfill our initial requirements given the postulated assumptions. The demonstration of the usefulness, scalability, and performance of our approach is showcased on exemplary benchmarking scenes.

The first benefit, which is in line with our initial requirements, is that with our method, we can preserve all the detail down to the atomic level while reaching interactive frame rates even for massive scenes. Moreover, with our approach, we can perform operations that are impossible or cumbersome to reach when using a billboarding method. Among such operations is recoloring every single molecule or even individual atoms at any time. To a certain extent, our method also supports the simulation of molecules' dynamic movements, which gives the scene a sense of realism, as molecules are naturally undergoing constant movements. We can animate these movements, i.e., support a jiggling of molecules and their atoms that resembles the Brownian motion. However, this can be allowed just to the degree when the translation of molecules does not change the information in the visibility buffer storing which molecules are occluded. From our observation, limiting the movements with respect to this restriction is sufficient for such a temperature-dependent random walk capturing features of the diffusive motion that drives the mesoscale dynamics [MWPV15]. Compared to the Hi-Z-based methods, our main benefit lies in the fact that we do not perform any additional computation on the GPU after caching the occlusion results.

To demonstrate the applicability and performance of our method, we focus our evaluation on measuring the gain in the rendering speed for the molecular scenes and the memory consumption. For



(a) Generated biological structure. (b) Partially zoomed-out view. (c) Entire dataset

Figure 9: Benchmark scene.

more fine-grained performance measurement, we also separately tested the effectiveness of our gap removal algorithm.

As it is difficult to obtain molecular models of sizes our method is aiming for, we had to generate a synthesized benchmarking dataset. The basic building block of this dataset is a synthetic biological structure depicted in Figure 9a, consisting of 80 million atoms. Using this model, we generated a molecular scene depicted in Figure 9c with 120 billion atoms.

As it is not meaningful to render molecules in far distance from the camera in atomistic resolution, we employed a level of detail technique described by Le Muzic et al. [LMPSV14] that is targeting to render very large molecular scenes. We implemented the method in the programming language Rust and graphics API WebGPU. We chose WebGPU for its capability to handle large amounts of draw calls that our method can produce. The testing was performed with the implementation in the WebGPU API to handle large amounts of draw calls that can be produced. All benchmark tests were performed on Intel Core i9-98880H CPU and NVIDIA RTX 2070 Max-Q GPU in FullHD resolution.

First, we looked at whether our gap removal algorithm is effective in lossy compression. To observe its efficiency we measured what percentage of molecules is marked as visible using different limit variable. Figure 10 presents data for two benchmark datasets we tested: the SARS-CoV-2 virion and our synthetic dataset. It can be observed that going from 1024 to 32 ranges, which is a reduction factor of 32, the compression marks only a small percentage of molecules as visible. To show how the reduction translates to the actual memory usage of range buffers, the following equation can be used for such calculation:

$$Bytes = Steps^2 * Limit * 8$$

Here $Steps$ equals to $360/N$ and N represents the distance between two camera viewpoints. For example, with a limit of 32 and N set to 5, we use 1.2 MB of memory for a single biological structure.

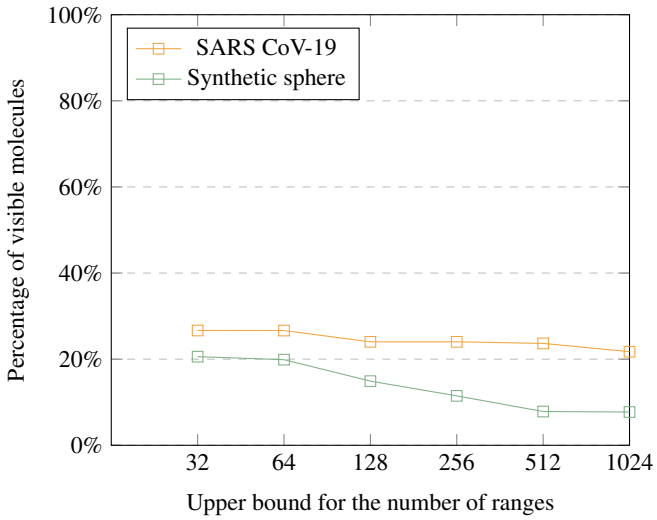


Figure 10: Percentage of visible molecules at different range limits for our two benchmarking datasets.

Second, we look at the benchmarks of rendering a single biological structure. Timings can be seen in Figure 11. As can be observed, for a single structure, there is a small benefit. A significant result is that the structure's visibility computation and rendering are only slightly slower than rendering the structure without occlusion computation.

The last benchmark operates on a synthetic scene shown in Figure 9. The scene contains 1,000 synthetic biological structures that together consist of 18 million molecules and 120 billion atoms. For every single frame, we calculate the visibility information for several camera viewpoints that have not been computed yet. Benchmark results can be seen in Figure 12. We observed that a short time after the application's start, the framerate stabilizes at a much higher value than without using our technique. However, few per-

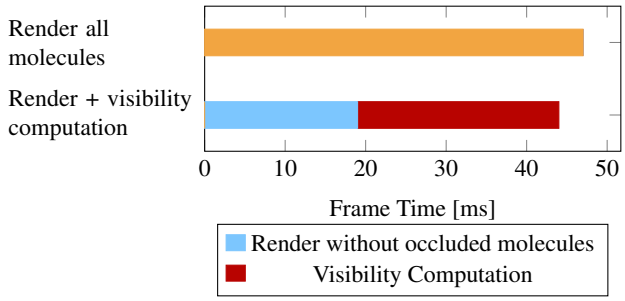


Figure 11: Performance test for camera zoomed-in to a single structure that is covering the entire screen, as depicted in Figure 9a.

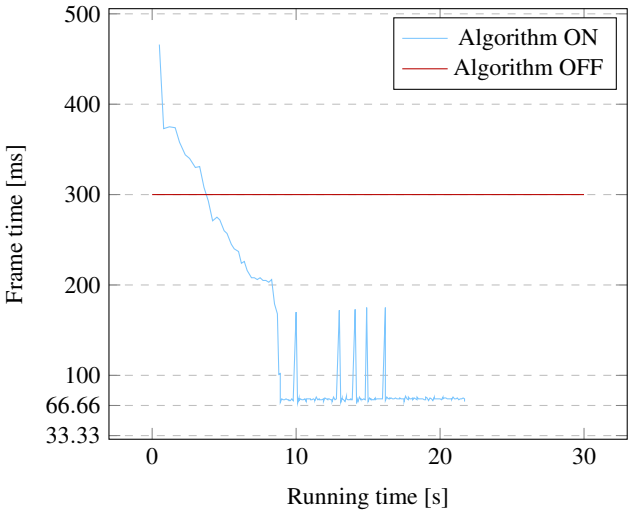


Figure 12: Benchmark of a synthetic scene with caching of the visibility information. Timings across the running time of the application are in blue. The baseline without using our algorithm is in red. The camera is positioned in the middle of the scene.

formance drops can be observed that disappear after the next few seconds of running time, when either most or all of the occlusion is already computed and cached. We also show the improvements in rendering time across different zoom levels of the scene in Figure 13.

With our method, we can scale molecular visualization even further than just using reduction at molecular level as in cellVIEW [LMAPV15]. Our limitation is that motion at the molecular level beyond Brownian cannot be applied, and clipping planes can not use cached visibility.

5. Conclusion and Future Work

In this paper, we presented a method to accelerate interactive visualizations of dense molecular scenes that enables us to render scenes where the number of atoms is ten times of magnitude larger than the existing state-of-the-art approaches. We demonstrated the usability of our proposed algorithm by its application on a synthetically generated scene that simulates the potential use case.

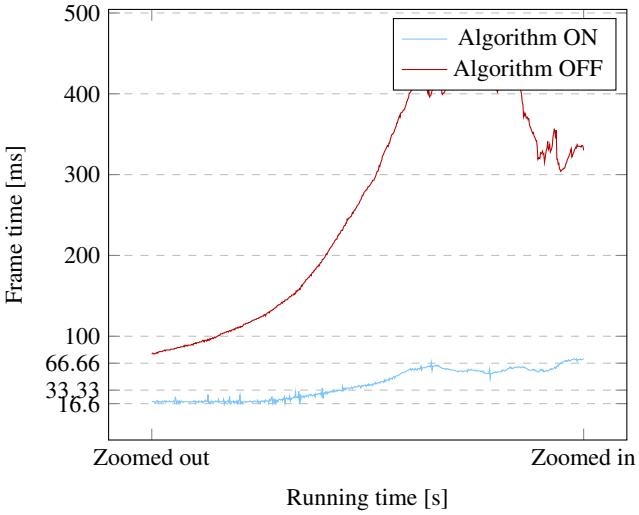


Figure 13: Benchmark of a synthetic scene with and without using our algorithm. Timings are across the whole scene, from a completely zoomed-out view capturing the whole scene to zoomed onto a single structure in the middle of the scene. Timings were performed after completing the computation of all visibility information.

Using our method, we successfully reached the reduction of the rendering time by a factor ranging from 3 to 7, depending on the camera's position. Our method provides a competitive alternative to billboard-based methods while still operating with the geometric representations of objects in a scene. This has many advantages, such as the ability to recolor molecules and atoms, the option to use already established simplification level-of-detail approaches at the molecular level, and to simulate the Brownian motion. All these options are adding the necessary sense of realism and are significantly enhancing the perception and exploration process.

Our conducted benchmarking testing proved that using our approach, we are reaching significant improvements of the rendering process for large molecular scenes. Although the size of the currently available molecular models is not yet reaching those that our approach is able to render, the advancements in generating this biological knowledge and capturing capabilities are rapid and we expect models of such size to be available soon. When the models appear in the future, our approach is ready for their interactive rendering and exploration.

Our method comes with some limitations that we are aiming to overcome in the future. From our observations, at this scale, the rendering performance is limited by the number of viewport culling units. In relation to this, we see two potential future research directions. The first one is to expand the abstraction and simplification beyond the molecular level and thus make our solution more robust and universal with respect to the content of the scene. This may be quite difficult as it is impossible to say whether there can be found an automatic solution to preparing the levels of detail for each biological structure that has to be currently created manually. The second potential future extension is to replace rasterization with ray

tracing. With too many occluded hierarchical structures in a frame, it might be faster to trace a ray with the simultaneous usage of an efficient acceleration structure.

Acknowledgements

The presented work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic within the INTER-COST project no.LTC20033. The work was partially supported by the ILLVISATION grant by WWTF (VRG11-010).

References

[ANA] ANAGNOSTOU K.: Experiments in gpu-based occlusion culling. https://interplayoflight.wordpress.com/2017/11/15/experiments-in-gpu-based-occlusion-culling/. Accessed: 2020-09-27. 2

[Bio] BioBlender. URL: http://www.bioblender.org/. 3

[BWF*00] BERMAN H. M., WESTBROOK J., FENG Z., GILLILAND G., BHAT T. N., WEISSIG H., SHINDYALOV I. N., BOURNE P. E.: The Protein Data Bank. *Nucleic Acids Research 28*, 1 (1 2000), 235–242. doi:10.1093/nar/28.1.235. 3

[BWPP04] BITTNER J., WIMMER M., PIRINGER H., PURGATHOFER W.: Coherent hierarchical culling: Hardware occlusion queries made useful. *Computer Graphics Forum 23* (09 2004), 615 – 624. doi:10.1111/j.1467-8659.2004.00793.x. 2

[CHPR07] COOK R. L., HALSTEAD J., PLANCK M., RYU D.: Stochastic simplification of aggregate detail. *ACM Transactions on Graphics 26*, 3 (7 2007), 79–es. doi:10.1145/1276377.1276476. 3

[DDSD03] DÉCORET X., DURAND F., SILLION F. X., DORSEY J.: Billboard clouds for extreme model simplification. *ACM Trans. Graph. 22*, 3 (July 2003), 689–696. doi:10.1145/882262.882326. 3

[DN09] DECAUDIN P., NEYRET F.: Volumetric billboards. *Computer Graphics Forum 28*, 8 (2009), 2079–2089. doi:https://doi.org/10.1111/j.1467-8659.2009.01354.x. 3

[FKE13] FALK M., KRONE M., ERTL T.: Atomistic visualization of mesoscopic whole-cell simulations using ray-casted instancing. *Computer Graphics Forum 32*, 8 (2013), 195–206. doi:https://doi.org/10.1111/cgf.12197. 2

[GAB*18] GARDNER A., AUTIN L., BARBARO B., OLSON A. J., GOODSELL D. S.: Cellpaint: Interactive illustration of dynamic mesoscale cellular environments. *IEEE Computer Graphics and Applications 38*, 6 (11 2018), 51–66. doi:10.1109/MCG.2018.2877076. 3

[GKM93] GREENE N., KASS M., MILLER G.: Hierarchical Z-Buffer Visibility. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), SIGGRAPH '93, Association for Computing Machinery, p. 231–238. doi:10.1145/166117.166147. 2

[Goo91] GOODSELL D. S.: Inside a living cell. *Trends in Biochemical Sciences 16*, 6 (1991), 203–206. doi:https://doi.org/10.1016/0968-0004(91)90083-8. 3

[GRDE10] GROTTEL S., REINA G., DACHSBACHER C., ERTL T.: Coherent culling and shading for large molecular dynamics visualization. In *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization* (Chichester, GBR, 2010), EuroVis'10, The Eurographs Association & John Wiley & Sons, Ltd., p. 953–962. doi:10.1111/j.1467-8659.2009.01698.x. 3

[Gre95] GREENE N.: *Hierarchical Rendering of Complex Environments*. PhD thesis, University of California, 1995. AAI9539493. 3

[HA] HAAR U., AALTONEN S.: Gpu driven rendering pipelines. http://advances.realtimerendering.com/s2015/aaltonenhaar_siggraph2015_combined_final_footer_220dpi.pdf. Accessed: 2020-09-27. 2, 3

[HA15] HAAR U., AALTONEN S.: GPU-Driven Rendering Pipelines. Presentation at SIGGRAPH 2015: Advances in Real-Time Rendering in Games, 2015. 3

[HC11] HILL S., COLLIN D.: Practical, dynamic visibility for games. In *GPU Pro 2*, Engel W., (Ed.). A K Peters, 2011, pp. 329–347. 2

[HLLF13] HORNUS S., LÉVY B., LARIVIÈRE D., FOURMENTIN E.: Easy DNA modeling and more with GraphiteLifeExplorer. *PloS one 8*, 1 (2013), e53609–e53609. Edition: 2013/01/07 Publisher: Public Library of Science. doi:10.1371/journal.pone.0053609. 3

[JAG*11] JOHNSON G. T., AUTIN L., GOODSELL D. S., SANNER M. F., OLSON A. J.: ePMV Embeds Molecular Modeling into Professional Animation Software Environments. *Structure 19*, 3 (Mar. 2011), 293–303. Publisher: Elsevier. doi:10.1016/j.str.2010.12.023. 3

[KIK*20] KOUŘIL D., ISENBERG T., KOZLIKOVA B., MEYER M., GROELLER E., VIOLA I.: HyperLabels: Browsing of Dense and Hierarchical Molecular 3D Models. *IEEE Transactions on Visualization and Computer Graphics* (2020), 1–1. doi:10.1109/TVCG.2020.2975583. 3

[Kua] KUAH K.: Software occlusion culling. https://software.intel.com/content/www/us/en/develop/articles/software-occlusion-culling.html. Accessed: 2020-09-27. 2

[KVGM20] KLEIN T., VIOLA I., GRÖLLER E., MINDEK P.: Multi-scale procedural animations of microtubule dynamics based on measured data. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 622–632. 3

[LBH12] LINDOW N., BAUM D., HEGE H.-C.: Interactive rendering of materials and biological structures on atomic and nanoscopic scale. *Computer Graphics Forum 31*, 3pt4 (2012), 1325–1334. doi:https://doi.org/10.1111/j.1467-8659.2012.03128.x. 2

[LMAPV15] LE MUZIC M., AUTIN L., PARULEK J., VIOLA I.: cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets. *Eurographics Workshop on Visual Computing for Biomedicine 2015* (2015), 61–70. doi:10.2312/vcbm.20151209. 2, 8

[LMPSV14] LE MUZIC M., PARULEK J., STAVRUM A. K., VIOLA I.: Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. EuroVis '14, Eurographics Association, p. 141–150. 3, 7

[MJFS01] MOON B., JAGADISH H. V., FALOUTSOS C., SALTZ J. H.: Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering 13*, 1 (2001), 124–141. doi:10.1109/69.908985. 4

[Mol] MolecularMaya. URL: https://clarafi.com/tools/mmaya/. 3

[MWPV15] MUZIC M. L., WALDNER M., PARULEK J., VIOLA I.: Illustrative timelapse: A technique for illustrative visualization of particle-based simulations. *2015 IEEE Pacific Visualization Symposium (PacificVis)* (2015), 247–254. 6

[NAA] NAAJI K.: Aabb occluders. http://karim.naaji.fr/blog/2019/15.11.19.html. Accessed: 2020-09-27. 3

[Ney98] NEYRET F.: Modeling, animating, and rendering complex scenes using volumetric textures. 55–70. doi:10.1109/2945.675652. 3

[NSK*20] NGUYEN N., STRNAD O., KLEIN T., LUO D., ALHARBI R., WONKA P., MARITAN M., MINDEK P., AUTIN L., GOODSELL D. S., VIOLA I.: Modeling in the time of covid-19: Statistical and rule-based mesoscale models, 2020. arXiv:2005.01804. 3

[Pur09] PURSER R. J.: Construction of a Hilbert curve on the sphere with an isometric parameterization of area. URL: https://repository.library.noaa.gov/view/noaa/11402. 4

[SSLL14] SILVENNOINEN A., SARANSAARI H., LAINE S., LEHTINEN J.: Occluder simplification using planar sections. *Computer Graphics Forum 33*, 1 (2014), 235–245. doi:10.1111/cgf.12271. 3

[ZMHH97] ZHANG H., MANOCHA D., HUDSON T., HOFF K. E.: Visibility culling using hierarchical occlusion maps. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., p. 77–88. doi:10.1145/258734.258781. 2